

FERNUNIVERSITÄT IN HAGEN

SCHÄTZUNG DER LEISTUNGS-AUFNAHME
DER CPU ANHAND VON TASK-TYPEN

Masterarbeit

Vorgelegt der Fakultät für Mathematik und Informatik (M+I)
der FernUniversität in Hagen
Lehrgebiet
Parallelität und VLSI

Von: Allan Kaufmann

Feldstrasse 20

31275 Lehrte

Matrikelnummer: 9502998

Gutachter: Dr. Sebastian Litzinger
ZWEITGUTACHTER: PROF. DR. JÖRG KELLER
BETREUER: DR. SEBASTIAN LITZINGER

27. Juni 2024

Abstract

Zur Reduzierung des Ressourceneinsatzes benötigen einige statische Scheduler Informationen über die Leistungsaufnahme der CPU, bei Ausführung einer Task, die mittels Benchmarking zu erheben sind. Um den Aufwand zu reduzieren wurde ein Vorgehen zur Schätzung der Leistungsaufnahme der CPU, anhand von Task-Typen evaluiert und in drei Artefakten implementiert: Der Tasktypeanalyser erstellt die Suchkriterien von Task-Typen. Der Apptaskanalyser erstellt Kriterien aus Tasks einer Beispielanwendung. Der Tasktypeestimator besteht aus den Modulen Taskmapper und Estimator. Der Taskmapper verwendet zwei Verfahren, um Tasks einer Anwendung auf Task-Typen abzubilden. Der Estimator schätzt die Leistungsaufnahme der einzelnen Anwendungstasks auf Grundlage der Task-Typen. Der Vergleich dieser Schätzungen mit der tatsächlichen Leistungsaufnahme hat gezeigt, dass insbesondere die Leistungsaufnahme umfangreicher Tasks mit einer Präzision im einstelligen Prozentbereich abzuschätzen sind. Der Aufwand zur Erhebung der Leistungsaufnahme lässt sich durch dieses Verfahren daher deutlich reduzieren.

Inhaltsverzeichnis

I	Grundlagen	1
1	Einleitung	2
1.1	Forschungsthema	2
1.2	Forschungsproblem	2
1.2.1	Forschungsfragen	3
1.2.2	Einordnung des Forschungsproblems	3
1.2.3	Wissenschaftliche Methoden	3
1.3	Hardware und Betriebssystem	4
1.4	Repository	4
2	Forschungsstand und Theoretische Grundlage	5
2.1	Grundbegriffe	5
2.1.1	Statische Scheduler und Scheduling-Entscheidungen	5
2.1.2	Leistungsaufnahme der CPU	5
2.1.3	Benchmarkprogramme	6
2.1.4	Parallelitätsgrad	6
2.1.5	Tasks, Threads und taskbasierte Anwendungen	6
2.1.6	Prototypische Tasks/Task-Typen	7
2.1.7	CPU Instruktionen	7
2.2	weitere relevante Arbeiten	8
2.2.1	epEBench	8
2.2.2	Frequenzskalierung	8
2.2.3	Ressourceneinsatz beim Scheduling	8
2.2.4	edgedetection	8
II	Eruierung und Implementierung des Vorgehens	9
3	Übersicht des Experiments	10
3.1	Übersicht der Implementierung	11

4	Phase 1: Prototypische Tasks	13
4.1	Task-Typen eruieren	13
4.1.1	Task-Typen aus Benchmark-Programmen	13
4.1.2	Benchmark epEBench	14
4.2	Kriterien ermitteln	14
4.2.1	Überprüfung des Modells dadd	14
4.3	Implementierung	15
4.3.1	Vorbereitung	16
4.3.2	Tasktypeanalyzer	16
4.3.3	run_all_tasks.sh	17
4.4	Zusammenfassung der ersten Phase des Vorgehens	18
5	Phase 2: Tasks einer Anwendung	19
5.1	Ermittlung einer Beispielanwendung	19
5.1.1	Kriterien zur Ermittlung geeigneter Anwendungen	19
5.1.2	geeignete Anwendungen	19
5.1.3	edgedetection	20
5.1.4	Zusammenfassung	20
5.2	Implementierung	20
5.2.1	Apptaskanalyzer	21
5.2.1.1	main-Methode	21
5.2.1.2	run_all_tasks.sh	22
5.2.1.3	Beschränkung der Sequenzdateien	22
5.3	Zusammenfassung der zweiten Phase des Vorgehens	23
6	Phase 3: Abbildung der Anwendung auf Task-Typen	25
6.1	Abbildung von Tasks	25
6.1.1	Vergleiche zwischen Anwendungstask und prototypischen Tasks	25
6.2	Vergleich über CPU-Instruktionen	26
6.2.1	CPU-Instruktionen	26
6.2.2	Komplettsuche	27
6.2.3	Ähnlichkeitssuche	27
6.3	Beziehungen zwischen Anwendungs- und prototypischen Task	27
6.3.1	1:1 Zuordnung	27
6.3.2	1:N Zuordnung	28
6.4	Implementierung	29
6.4.1	Tasktypeestimator	30
6.4.1.1	Taskmapper	30
6.4.2	main.cpp	30
6.4.2.1	Testdateien für Abbildungsverfahren	31
6.4.3	taskmapperOneToOne.cpp	31
6.4.4	taskmapperOneToMany.cpp	33
6.4.5	Transfer to epEBench	34

6.5	Eruierung von alternativen Abbildungen	34
6.5.1	Diffsuche	34
6.5.2	Paarsuche	36
6.5.3	Drillingsuche	37
6.6	Zusammenfassung der dritten Phase des Vorgehens	38
7	Phase 4: Schätzung der Leistungsaufnahme	39
7.1	Leistungsaufnahme der CPU messen	39
7.1.1	Auswertung des Energiezählers	39
7.1.2	Parallelitätsgrad	40
7.1.3	CPU-Frequenz	40
7.1.3.1	Änderung der CPU-Frequenz zur Laufzeit	41
7.1.4	Leistungsaufnahme von Task-Typen messen	41
7.1.5	Leistungsaufnahme der Anwendungs-Tasks messen	42
7.2	Schätzung der Leistungsaufnahme	42
7.3	Vergleich Schätzung und Messung der Leistungsaufnahme	42
7.4	Implementierung	42
7.4.1	Dateistruktur des Artefakts	44
7.4.2	config.cpp	44
7.4.3	measure.cpp	45
7.4.3.1	Messung der Leistungsaufnahme von Task-Typen	45
7.4.3.2	Parallelitätsgrad von Task-Typen bei Messung	45
7.4.3.3	Messung der Leistungsaufnahme von Anwendungstasks	46
7.4.4	Vorbereitung des Experiments in edgedetection	47
7.4.4.1	Parametrisierung der Anwendungstasks	47
7.4.4.2	Messung der Leistungsaufnahme in edgedetection	48
7.4.4.3	Parallelisierung mit OMP	48
7.4.4.4	Parallelisierung mit PThread	48
7.4.5	estimator.cpp	49
7.5	Zusammenfassung der vierten Phase des Vorgehens	52
III	Evaluierung des Vorgehens	53
8	Ergebnisse und Analyse	54
8.1	Phase 1: Prototypische Tasks	54
8.2	Phase 2: Tasks einer Anwendung	55
8.3	Phase 3: Abbildung der Anwendung auf Task-Typen	56
8.4	Phase 4: Schätzung der Leistungsaufnahme	58
8.4.1	greyscale	59
8.4.2	sharpencontrast	60
8.4.3	combineimgs	61
8.4.4	copyimage	62

8.4.5	sobelh	63
8.4.6	sobelv	64
8.4.7	writeimage	65
8.4.8	checkcontrast	66
8.4.9	loadimage	67
9	Diskussion und Interpretation	69
9.1	Diskussion	69
9.1.1	Abweichungen im niedrigen Prozentbereich	69
9.1.2	Abweichungen bei Ausführung auf mehreren Kernen	69
9.1.3	Abweichungen bei Tasks mit geringerer Sequenzgröße	70
9.1.4	Reduzierung des Aufwands	70
10	Fazit	71
10.1	Beantwortung der Forschungsfragen	71
10.1.1	Wie werden prototypische Tasks definiert?	71
10.1.2	Wie wird die Leistungsaufnahme dieser prototypischen Tasks ermittelt?	71
10.1.3	Wie werden Tasks einer konkreten Anwendung auf prototypischen Tasks abgebildet?	71
10.1.4	Wie wird die Leistungsaufnahme der CPU anhand von Task-Typen abge- schätzt?	71
10.1.5	Wie weit weichen die Schätzungen von der tatsächlichen Leistungsaufnah- me konkreter Anwendungstasks ab?	72
10.2	Zusammenfassung	72
	Anhang	76
A	Ergänzungen	76
A.1	Abbildungen	76
A.1.1	Menü des Tasktypeestimators	76
A.1.2	Auszug einer Sequenzdatei zum prototypischen Task bitbyte	77
A.1.3	Dateistruktur des Taskmappers	79
A.1.4	Dateistruktur des Tasktypeestimators	79
A.2	Listings	81
A.2.1	Prototypische Tasks aus epEBench	81
A.2.2	Task-Typ dadd im Benchmark epEBench	81
A.2.3	Energiezähler über RAPL-Schnittstelle	82
A.2.4	Assemblercode von dadd	82
A.2.5	Beispiel einer taskbasierte Anwendung in C	83
A.2.6	run_all_tasks.sh aus apptaskanalyzer	84
A.2.7	main.cpp des Taskmappers	84
A.2.8	Implementierung des 1:1-Abbildungsverfahrens	85

A.2.9	Implementierung des 1:N-Abbildungsverfahrens	86
A.2.10	Implementierung eines alternativen Abbildungsverfahrens: Diff-Suche . . .	86
A.2.11	Logdatei für alternative Abbildung: Diffsuche	87
A.2.12	Implementierung eines alternativen Abbildungsverfahrens: Paar-Suche . .	88
A.2.13	Implementierung eines alternativen Abbildungsverfahrens: Drilling-Suche .	90
A.2.14	Transfer Taskmap nach epEBench	91
A.2.15	experiment.config	92
A.2.16	Generierung von Skripten im Tasktypeestimator	93
A.2.17	Messung der Leistungsaufnahme von Task-Typen	93
A.2.18	Messung der Leistungsaufnahme von Anwendungstasks	95
A.2.19	Aufruf von Anwendungstasks in edgedetection	95
A.2.20	Messung des Energieverbrauchs in edgedetection mit Funktionszeiger . . .	97
A.2.21	Parallelisierung von edgedetection mit PThread	98
A.2.22	Schätzung der Leistungsaufnahme im Tasktypeestimator	98
B	Ergänzungen zu den Ergebnissen	101
B.1	Prototypische Tasks	101
B.2	Tasks einer Anwendung	101
B.3	Abbildung der Anwendung auf Task-Typen	102
B.3.1	taskmap.txt	102
B.4	Leistungsaufnahme der CPU anhand von Task-Typen	103
B.4.1	Ergebnisdatei zur Schätzung der Leistungsaufnahme vom 24.06.2024 . . .	103
C	Ergänzungen zu den Implementierungen	124
C.1	Dateistruktur	124
C.2	Init-Skript	124
C.3	Vorbedingungen für Experiment prüfen	126
D	Exposé für Masterarbeit	128
D.1	Zielsetzung und Erkenntnisinteresse	128
D.2	Visualisierung	128
D.3	Phasen	129
D.3.1	Ableitung prototypischer Tasks	129
D.3.2	Leistungsaufnahme prototypischer Tasks in einem Testsytsem	132
D.3.3	Ermittlung der Task einer konkreten Anwendung	132
D.3.4	Berechnung und Ergebnisbestimmung	133
D.4	erwartete Ergebnisse	134
D.5	vorläufige Gliederung	134
D.6	Zeitplan	135
E	In eigener Sache	136
E.1	Aufwandsstatistik	136
E.2	Danksagungen	137

Abbildungsverzeichnis

1.1	Agile Design Research[12]	4
3.1	Visualisierung	11
3.2	Komponentendiagramm für Artefakte aller vier Phasen	12
4.1	Komponentendiagramm für Artefakte der ersten Phase	15
5.1	Komponentendiagramm für Artefakte der zweiten Phase	21
5.2	Sequenzdateien für Tasks von edgedetection	24
6.1	Komponentendiagramm für Artefakte der dritten Phase	29
7.1	Komponentendiagramm für Artefakte der vierten Phase	43
A.1	Tasktypeestimator Menü	77
A.2	Sequenzdateien der prototypischen Tasks	78
A.3	Dateistruktur für Tasktypeestimator	80
C.1	Dateistruktur von PP8	125
D.1	Visualisierung	129

Tabellenverzeichnis

6.1	Logdateien für Paarsuche	37
8.1	Ergebnisse - Prototypische Tasks	55
8.2	Ergebnisse - Tasks einer Anwendung	56
8.3	Abbildung der Anwendungstask auf Task-Typen	57
8.4	Logdateien für Taskmapper	58
8.5	Logdateien für Tasktypeestimator	58
8.6	Ergebnisse - Task greyscale	60
8.7	Ergebnisse - Task Sharpencontrast	61
8.8	Ergebnisse - Task combineimgs	62
8.9	Ergebnisse - Task copyimage	63
8.10	Ergebnisse - Task sobelh	64
8.11	Ergebnisse - Task sobelv	65
8.12	Ergebnisse - Task writeimage	66
8.13	Ergebnisse - Task checkcontrast	67
8.14	Ergebnisse - Task loadimage	68
D.1	Zeitplan	135

Algorithmenliste

1	1:1 Abbildung - vereinfachter Algorithmus	32
2	1:N Abbildung - vereinfachter Algorithmus	35
3	Schätzung der Leistungsaufnahme anhand von Task-Typen	51

Listings

4.1	ebloads.cpp[7]	14
4.2	main.cpp aus Tasktypeanalyzer	16
4.3	run_all_tasks.sh aus Tasktypeanalyzer	17
5.1	Auszug aus main-Methode apptask_analyzer	21
5.2	greyscale aus edgedetection	22
6.1	Assembleranweisung in dadd	26
7.1	Frequenzlevel für i5-8365U	40
7.2	Setzen des CPU-Frequenzlevels	41
7.3	Messung Anwendungstask mit Parallelitätsgrad	46
A.1	Prototypische Tasks aus epEBench ebloads.h[17]	81
A.2	Ausführung des Tasks dadd	81
A.3	Ergebnis Leistungsaufnahme von dadd	81
A.4	Leistungsaufnahme der CPU über RAPL[25]	82
A.5	Assemblercode von dadd[5]	82
A.6	Thread-Programm zur Realisierung einer Matrixmultiplikation[21][s.S. 291	83
A.7	run_all_tasks.sh aus apptasks	84
A.8	Auszug aus main.cpp	84
A.9	Auszug aus taskmapperOneToOne.cpp	85
A.10	Auszug aus taskmapperOneToMany.cpp	86
A.11	Implementierung der Diff-Suche	86
A.12	Logausgabe Diffsuche	88
A.13	Implementierung der Paar-Suche	88
A.14	BildschirmAusgabe bei Paar-Suche	89
A.15	Implementierung der Drilling-Suche	90
A.16	Transfer nach epEBench	91
A.17	Konfigurationsdatei experiment.config	92
A.18	Generierung von Skripten in config.cpp	93
A.19	Messung	93
A.20	Messung der Anwendungstasks	95
A.21	Aufruf von Anwendungstasks in edgedetection	95
A.22	Messung der Leistungsaufnahme mit Funktionszeiger	97
A.23	Parallelisierung mit PThread	98
A.24	Schätzung der Leistungsaufnahme	98

B.1	Umfang der Sequenzen von Task-Typen	101
B.2	Umfang der Sequenzen von Anwendungstasks	101
B.3	Inhalt der Ergebnisdatei taskmap.txt	102
B.4	Ergebnisdatei vom 24.06.2024	103
C.1	init.sh	126
D.1	Prototypische Tasks aus epEBench eblloads.h[17]	129
D.2	Ausführung des Tasks dadd	130
D.3	Ergebnis Leistungsaufnahme von dadd	130
D.4	eblloads.cpp[7]	130
D.5	Assemblercode von dadd[5]	131
D.7	Leistungsaufnahme der CPU über RAPL[25]	133

Teil I

Grundlagen

Kapitel 1

Einleitung

1.1 Forschungsthema

Das Forschungsthema dieser Masterarbeit lautet 'Schätzung der Leistungsaufnahme der CPU anhand von Task-Typen'. Das Thema umfasst demnach die Leistungsaufnahme, also der Energiebedarf [...] pro Zeiteinheit[13], von Prozessoren. Dabei wird jedoch die Leistungsaufnahme der CPU auf Grundlage von Tasks-Typen abgeschätzt und mit der tatsächlichen Leistungsaufnahme der CPU verglichen. Das Ziel dieser Masterarbeit war ein Erkenntnisgewinn darüber, ob und wie präzise die Leistungsaufnahme der CPU anhand von Task-Typen abgeschätzt werden konnte.

1.2 Forschungsproblem

Mit dem Forschungsthema wurde der Untersuchungsgegenstand benannt. Das Forschungsproblem ("research problem") kennzeichnet, welche Erkenntnisse zu welchen Aspekten des Untersuchungsgegenstandes auf welcher theoretischen, empirischen oder methodischen Basis gewonnen werden sollen[18][s.S. 148]. Das Forschungsproblem wurde wie folgt formuliert:

Einige statische Scheduler benötigen Informationen über die Leistungsaufnahme der CPU bei Ausführung einer Task, um Scheduling-Entscheidungen zu treffen. Liegen diese Informationen für eine gegebene Anwendung nicht vor, müssen sie mittels Benchmarking erhoben werden.

Zur Reduzierung des Aufwands ist es wünschenswert, die Leistungsaufnahme einer CPU lediglich einmalig für prototypische Tasks zu erheben. Für eine konkrete Anwendung wäre dann zu bestimmen, inwieweit ihre Tasks diesen Prototypen entsprechen, um eine Schätzung der Leistungsaufnahme durchführen zu können.

In dieser Arbeit soll ein solches Vorgehen implementiert und evaluiert werden.

1.2.1 Forschungsfragen

Zur Untersuchung des Forschungsproblems wurde eine ganze Reihe relevanter Arbeiten geprüft, die den Erkenntnisgewinn zum Forschungsthema zwar stützen, aus denen sich jedoch keine Forschungshypothesen aufstellen lassen. Zur weiteren Untersuchung wurden daher Forschungsfragen formuliert. Die Forschungsfrage („research question“) basiert auf dem bisherigen Forschungsstand und zielt v. a. auf Forschungslücken [18][s.S. 150].

1. Wie werden prototypische Tasks definiert?
2. Wie wird die Leistungsaufnahme dieser prototypischen Tasks ermittelt?
3. Wie werden Tasks einer konkreten Anwendung auf prototypischen Tasks abgebildet?
4. Wie wird die Leistungsaufnahme der CPU anhand von Task-Typen abgeschätzt?
5. Wie weit weichen die Schätzungen von der tatsächlichen Leistungsaufnahme konkreter Anwendungstasks ab?

1.2.2 Einordnung des Forschungsproblems

Zur Beantwortung der Forschungsfragen wurden Daten aus Experimenten erhoben und untersucht. Dabei wurde der Untersuchungsgegenstand in unterschiedlichen Situationen untersucht und in Bezug auf alle Faktoren kontrolliert [3]. Der Untersuchungsgegenstand ist die Leistungsaufnahme der CPU und zu den Faktoren gehören die Taktfrequenz der CPU, sowie der Parallelitätsgrad bei Ausführung.

1.2.3 Wissenschaftliche Methoden

Zur Bearbeitung des Problems wurde zunächst die Design Science Research Methode[1] angewendet. Bei dieser Methode wird ein Artefakt, etwa ein Modell oder ein Vorgehen entwickelt. Dabei werden, im Rahmen von Experimenten, die für das Artefakt benötigten Daten, etwa die Leistungsaufnahme der CPU, empirisch erhoben und ausgewertet. Das geplante Vorgehen wurde in einem Exposé schriftlich fixiert, welches im Kapitel D im Anhang ab Seite 128 angehängt wurde.

Während der Bearbeitung des Forschungsproblems haben sich weitere Anforderungen ergeben, die für die Experimente von Nutzen waren und umgesetzt wurden. Dazu gehören Anpassungen an der Beispielanwendung, die Entwicklung weiterer Artefakte und weiter Abbildungsverfahren. Diese Anforderungen hatten den Zweck, den Erkenntnisgewinn, unter Einhaltung des Terminplans, zu steigern. Aus diesem Grund wurde im weiteren Verlauf die Agile Design Research Methode angewendet [12]. Diese Methode ist an das Manifest für agile Softwareentwicklung[4] angelehnt und erklärt sich daher, dass eine regelmäßige Abstimmung stattgefunden hat und Änderungsbedarfe festgestellt wurden. Bei diesen Abstimmungen wurden neuere Erkenntnisse besprochen und für das weitere Vorgehen priorisiert. Die Erkenntnisse konnten daher in die nächsten

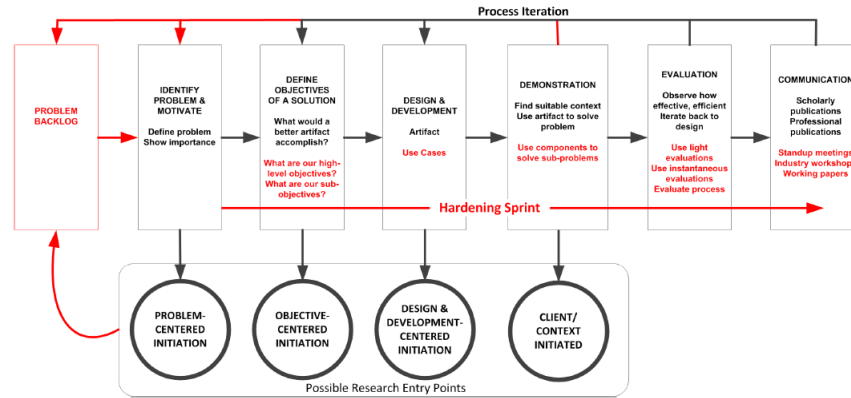


Abbildung 1.1: Agile Design Research[12]

Iterationen einfließen, während andere mögliche Aufwände, mit weniger Auswirkungen auf den Erkenntnisgewinn, herunter priorisiert werden konnten.

1.3 Hardware und Betriebssystem

Die Experimente wurden auf einem Dell Latitude 5500 Notebook durchgeführt. Bei der verwendete CPU handelt es sich um einen Intel CPU i5-8365U, welche über 4 Kerne und 8 Threads verfügt. Die CPU hat eine Grundtaktfrequenz von 1.60 GHz und kann auf bis zu 4.10 GHz getaktet werden kann. Das Notebook ist mit 16 GB DDR4-2400 Arbeitsspeicher und 512 GB Festplattenspeicher ausgestattet. Bei dem Betriebssystem handelt es sich um Ubuntu 64-Bit (Version 22.04.4 LTS).

Näheres zur verwendeten CPU ist auf der Herstellerseite zu entnehmen: <https://ark.intel.com/content/www/de/de/ark/products/193555/intel-core-i5-8365u-processor-6m-cache-up-to-4-10-ghz.html>

1.4 Repository

Der Quelltext für die Artefakte, sowie die Protokolle zur Ausführung des Experiments, stehen in einem Github-Repository zur Verfügung:

<https://github.com/allankaufmann/pp8>

Kapitel 2

Forschungsstand und Theoretische Grundlage

2.1 Grundbegriffe

2.1.1 Statische Scheduler und Scheduling-Entscheidungen

Die meisten der heute verwendeten und entwickelten Prozessoren sind superskalare Prozessoren oder VLIW-Prozessoren, die mehrere Instruktionen gleichzeitig absetzen und unabhängig voneinander verarbeiten können. Dazu stehen mehrere Funktionseinheiten zur Verfügung, die unabhängige Instruktionen parallel zueinander bearbeiten können. [...] Ein Maschinenprogramm für superskalare Prozessoren besteht aus einer sequentiellen Folge von Instruktionen, die per Hardware auf die zur Verfügung stehenden Funktionseinheiten verteilt werden[...]. Dabei wird ein dynamisches, d. h. zur Laufzeit des Programmes arbeitendes Scheduling der Instruktionen verwendet[...]. Im Unterschied dazu wird für VLIW-Prozessoren ein statisches Scheduling verwendet, bei dem die Zuordnung von Instruktionen an Funktionseinheiten bereits vor dem Start des Programms festgelegt wird. Dazu erzeugt ein spezieller Übersetzer Maschinenprogramme mit Instruktionsworten, die für jede Funktionseinheit angeben, welche Instruktionen zum entsprechenden Zeitpunkt ausgeführt wird [21][s. S. 14].

2.1.2 Leistungsaufnahme der CPU

Um die Leistungsaufnahme der CPU auf einer Intelarchitektur zu messen, bietet es sich an, die RAPL-Schnittstelle auf einem Linux-Betriebssystem zu verwenden, um den Energieverbrauch auslesen. Laut Khan kann diese Schnittstelle genutzt werden, um den Energieverbrauch mithilfe von Zählern zu messen[11]. Khan verweist außerdem auf Hähnel, welcher gezeigt hat, wie der Energieverbrauch einer kleinen Anwendung über diese Schnittstelle ausgewertet werden kann [16]. Einen Überblick zu den vorhandenen Zählern ist der Dokumentation zum PowerCAP-Framework zu entnehmen, demnach wird das Attribut `energy_uj` ausgewertet, um den Stand des Energiezählers in Mikrojoule zu erhalten [24].

Laut Strempel berechnet sich die Leistungsaufnahme daraus, dass der Energieverbrauch durch Zeit geteilt wird [23][s.S. 4]. Um die Leistungsaufnahme der CPU P , basierend auf den Energiezählerwerten und der Dauer zwischen den Messungen, zu berechnen, wurde folgende Formel angewandt:

$$P_{\text{mW}} = \frac{\Delta E_{\mu J}}{\Delta t_{\text{ms}}}$$

wobei:

- P_{mW} die Leistung in Milliwatt (mW) ist,
- $\Delta E_{\mu J}$ die Differenz der Energiezählerwerte in Mikro-Joule (μJ) ist,
- Δt_{ms} die Zeitdifferenz in Millisekunden (ms) ist.

2.1.3 Benchmarkprogramme

Die Leistung eines Rechners kann stark vom betrachteten Programm abhängen. Für zwei Programme A und B kann der Fall auftreten, dass Programm A auf einem Rechner X schneller läuft als auf Rechner Y[...]. Da die auszuführenden Programme oft nicht a priori bekannt sind, wurden Benchmarkprogramme entwickelt. Dies sind Testprogramme mit speziellen Charakteristika, deren Ausführungszeit auf verschiedenen Rechnersystemen gemessen werden können, um so eine standardisierte Leistungsbewertung zu ermöglichen [21][s.S.172].

2.1.4 Parallelitätsgrad

Wie im Kapitel 1.2.2 genannt, wurde auch der Parallelitätsgrad bei Ausführung der Anwendung untersucht. Der Parallelitätsgrad beschreibt, auf wie vielen Prozessorkernen die Tasks einer Anwendung gleichzeitig ausgeführt wurden. Dabei ist zu beachten, ob der Task auch tatsächlich auf unterschiedlichen physikalischen Prozessorkernen ausgeführt werden soll, anstatt auf virtuellen Prozessoren. Wenn der Task auf virtuellen Prozessoren ausgeführt werden würde, dann würde es sich um Multithreading [21][s.S.31] handeln. Zum Multithreading gehört auch der Ansatz, mehrere Kontrollflüsse [...] auf einem Prozessor oder Prozessorkern auszuführen, dass der Prozessor je nach Bedarf per Hardware zwischen den Kontrollfüßen umgeschaltet. Dies wird als simultanes Multithreading (SMT) oder Hyperthreading (HT) bezeichnet [20][s. S. 5]. Um zu überprüfen, ob der Parallelitätsgrad einen Einfluss auf die Leistungsaufnahme der CPU hat, wird Multithreading bzw. simultanes Multithreading nicht betrachtet, sondern die Ausführung der Tasks einer Anwendung auf mehreren CPU-Kernen.

2.1.5 Tasks, Threads und taskbasierte Anwendungen

Tasks sind die kleinsten Einheiten der Parallelität, die ausgenutzt werden sollen, und können je nach Zielrechner auf verschiedenen Ebenen der Ausführung identifiziert werden (Instruktionsebe-

ne, Datenparallelität, Funktionsparallelität[...] Eine Task ist eine beliebige Folge von Berechnungen, die von einem einzelnen Prozessor ausgeführt wird.[...]. Die Identifikation der Tasks hängt stark von dem zu parallelisierenden Programm ab. [...] Das Ziel der Zerlegungsphase besteht zum einen darin, genügend Potential für eine parallele Abarbeitung zu schaffen, zum anderen sollte die Granularität der Tasks, d. h. die Anzahl der von einer Task durchgeführten Berechnungen, an das Kommunikationsverhalten der Zielmaschine angepasst werden [21][s.S.122].

Ein Prozess oder Thread ist ein abstrakter Begriff für einen Kontrollfluss, der von einem physikalischen Prozessor ausgeführt wird und der nacheinander verschiedene Tasks ausführen kann [21][s.S.122].

Taskbasierte Anwendungen sind somit Anwendungen, deren Berechnungen unterschiedlichen Prozessen zugeordnet werden.

2.1.6 Prototypische Tasks/Task-Typen

Bei prototypischen Tasks handelt es sich um Task-Kategorien oder Task-Typen, die auf Prozessebene hinsichtlich der Art und der Menge der verwendeten Befehle mit Tasks konkreter Anwendungen vergleichbar sind. Wenn ein prototypischer Task ausgeführt wird, dann sollte die Leistungsaufnahme der CPU annähernd dem Wert entsprechen, als wenn ein äquivalenter Task einer Anwendung ausgeführt werden würde. Da die Tasks von Anwendungen hinsichtlich der verwendeten Befehle und der Leistungsaufnahme sehr unterschiedlich sein können, werden auch unterschiedliche Task-Typen benötigt, damit eine äquivalenter Abbildung und folglich eine präzisere Abschätzung der Leistungsaufnahme möglich ist.

2.1.7 CPU Instruktionen

Das sog. Programmiermodell eines Prozessors stellt die oberste Ebene der Rechnerarchitektur dar. Das Programmiermodell beschreibt die Sicht eines Systemprogrammierers auf den Prozessor und hat daher entscheidenden Einfluss auf die Leistung und insbesondere die Einsatzmöglichkeiten eines Rechners. Dazu beinhaltet das Programmiermodell alle notwendigen Details, um ablauffähige Maschinenprogramme für den betreffenden Prozessor zu erstellen[2][s.S.11, KE2].

Im Rahmen dieser Arbeit wurden Task-Typen untersucht und mit Anwendungstasks verglichen. Es ist natürlich, dass sich die Tasks hinsichtlich des Quelltextes unterscheiden und einen objektiven Vergleich folglich erschweren. Eine Möglichkeit um Anwendungstask mit Task-Typen zu vergleichen ist auf Grundlage des Programmiermodells eines Prozessors. Dabei werden die Befehle, die für den Prozessor erstellt wurden, miteinander verglichen um festzustellen, welchen Task-Typen ein Anwendungstask am ähnlichsten ist.

2.2 weitere relevante Arbeiten

2.2.1 epEBench

Im Konferenzartikel 'epEBench: True Energy Benchmark' haben Holmbacka und Müller mit epEBench einen Benchmark vorgestellt, welcher eine Bewertung der Energie-Effizienz von Systemen durch Modelle ermöglicht. Dabei wurden die Modelle auf Grundlage der CPU-Instruktionen aus zwei Anwendungen abgeleitet[7].

Im Konferenzartikel 'Workload Type-Aware Scheduling on big.LITTLE Platforms' haben Holmbacka und Keller diesen Benchmark auf einer heterogenen Plattform verwendet und gezeigt, dass mithilfe von Task-Typen eine energieeffizientere Ausführung möglich ist, wenn diese beim Scheduling berücksichtigt werden würden [6].

2.2.2 Frequenzskalierung

Im Konferenzartikel 'Scheduling Moldable Parallel Streaming Tasks on Heterogeneous Platforms with Frequency Scaling' haben Litzinger, Keller und Kessler gezeigt, wie anhand von Task-Typen und Frequenzskalierung, eine Verbesserung des Energiebedarfs von 33% erreicht werden könnte[15].

2.2.3 Ressourceneinsatz beim Scheduling

In der Dissertation 'Raising Energy Efficiency and Fault Tolerance with Parallel Streaming Application Scheduling on Multicore Systems' hat Litzinger zwei Ansätze entwickelt, welche darauf abzielen, den Ressourceneinsatz beim Scheduling zu reduzieren, ohne erhebliche Einbußen bei der Lösungsqualität hinnehmen zu müssen[14].

2.2.4 edgedetection

Litzinger und Keller haben 2020 in der Veröffentlichung 'Code generation for energy-efficient execution of dynamic streaming task graphs on parallel and heterogeneous platforms' gezeigt, wie statische Scheduler verwendet werden, um Tasks einer Anwendung auf einer parallelen Plattform energieeffizient abzubilden [22]. In diesem Zusammenhang wurde mit einer taskbasierten Anwendung experimentiert, die eine Kantenerkennung an Bildern durchführt.

Teil II

Eruierung und Implementierung des Vorgehens

Kapitel 3

Übersicht des Experiments

Im Rahmen dieser Arbeit wurde ein Vorgehen implementiert, um die Leistungsaufnahme der CPU anhand von Task-Typen abzuschätzen. Zur Evaluierung wurden Daten aus Experimenten erhoben und untersucht. Um die einzelnen Schritte zu systematisieren und übersichtlich darzustellen, wurde das Vorgehen in vier Phasen aufgeteilt:

1. Phase: Prototypische Tasks
2. Phase: Tasks einer Anwendung
3. Phase: Abbildung Prototypische Tasks auf Tasks der Anwendung
4. Phase: Schätzung der Leistungsaufnahme und Vergleich

Die Abbildung 3.1 zeigt die vier Phasen sowie die enthaltenen Unteraufgaben.

In der ersten Phase wurden geeignete prototypische Tasks evaluiert, die einerseits zur Abschätzung der Leistungsaufnahme geeignet sind, andererseits den Tasks einer konkreten Anwendung zugeordnet werden können.

In der zweiten Phase wurde eine geeignete Beispielanwendung evaluiert, um aus den Tasks Kriterien für die Schätzung der Leistungsaufnahme zu ermitteln.

In der dritten Phase wurde auf Grundlage einer Ähnlichkeitssuche eine Zuordnung der Anwendungstasks zu den prototypischen Tasks hergeleitet.

In der vierten Phase wurde die Leistungsaufnahme der CPU anhand von Task-Typen abgeschätzt. Um das Vorgehen zu überprüfen wurden die Leistungsaufnahme der Anwendungstasks gemessen und mit der Abschätzung verglichen.

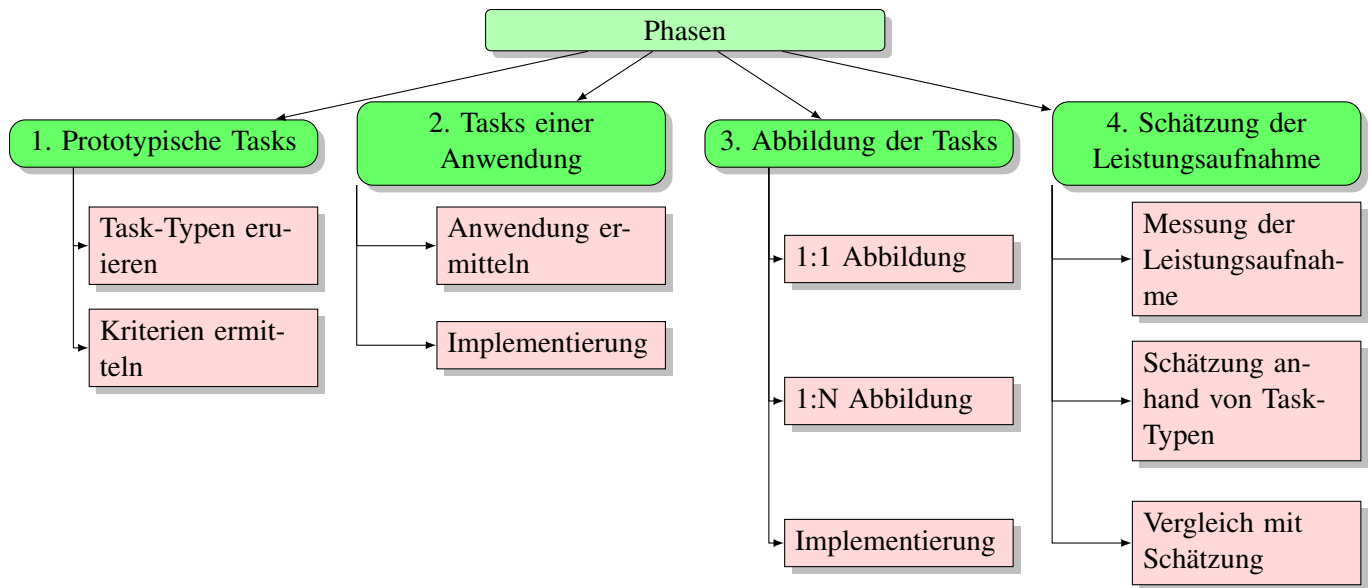


Abbildung 3.1: Visualisierung

3.1 Übersicht der Implementierung

Die Abbildung 3.2 dient zur Übersicht der implementierten Artefakte. Diese werden nachfolgend kurz vorgestellt:

1. Tasktypeanalyzer - dient zur Erhebung der Kriterien von prototypischen Tasks.
2. Apptaskanalyzer - dient zur Erhebung der Kriterien von Tasks einer Beispielanwendung.
3. Tasktypeestimator - besteht aus zwei Modulen - dem Taskmapper und dem Estimator. Der Taskmapper bildet Anwendungstasks auf prototypische Tasks ab. Der Estimator schätzt die Leistungsaufnahme der CPU anhand von Task-Typen ab und vergleicht diese Schätzung mit der Leistungsaufnahme bei Ausführung der Anwendungstasks.

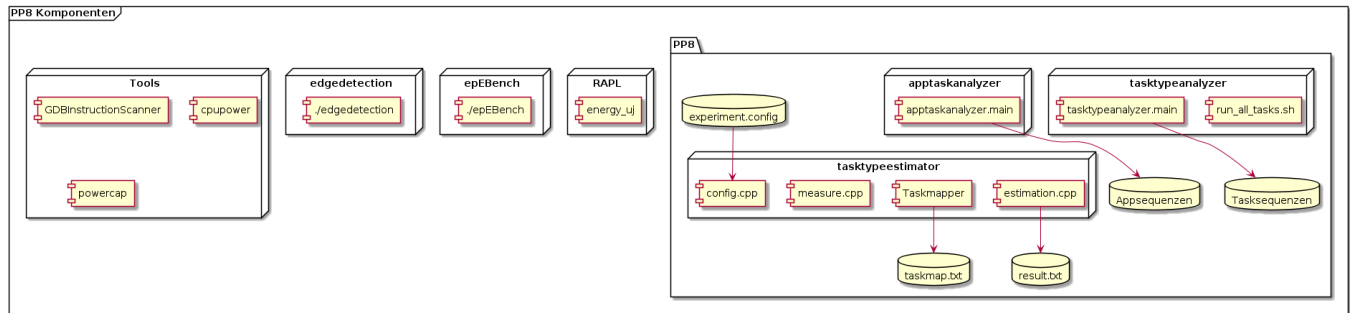


Abbildung 3.2: Komponentendiagramm für Artefakte aller vier Phasen

Die nachfolgenden Kapitel beschreiben zu jeder dieser Phasen die Eruiierung sowie die Implementierung des Vorgehens.

Kapitel 4

Phase 1: Prototypische Tasks

4.1 Task-Typen eruieren

4.1.1 Task-Typen aus Benchmark-Programmen

Um die Leistungsaufnahme der CPU abschätzen zu können, sind zunächst geeignete Tasks zu ermitteln. Die prototypischen Tasks sollten auf Instruktionsebene ausreichend variieren, damit das Spektrum möglicher Anwendungstasks möglichst weit abgedeckt ist. Da zur standardisierten Leistungsbewertung von Rechnersystemen Benchmark-Programme verwendet werden, bietet es sich an, die zur Bewertung verwendeten Testprogramme auf Eignung als prototypische Tasks zu untersuchen. Da Benchmark-Programme zur Leistungsbewertung also Testprogramme ausführen, die aus realen Anwendungen abgeleitet wurden, sollte bei Ausführung dieser Testprogramme eine vergleichbare Leistungsaufnahme festzustellen sein.

Um die Leistungsaufnahme unter verschiedenen Faktoren des Experiments untersuchen zu können, muss ein geeignetes Benchmark-Programm über Parameter verfügen, die eine Steuerung der prototypischen Tasks ermöglichen. Zu diesen Parameter gehören die Dauer der Ausführung, sowie der Parallelitätsgrad. Ein weiterer Faktor für die CPU-Taktfrequenz. Das Benchmark-Programm sollte also die Ausführung von Tasks unter unterschiedlichen CPU-Frequenzleveln ermöglichen.

Für das Vorgehen ist es außerdem erforderlich, dass die Tasks einer Anwendung auf diese prototypischen Tasks abgebildet werden können. Das bedeutet dass der Quelltext der prototypischen Tasks bekannt sein müsste oder dass die Arbeitsschritte der Tasks soweit dokumentiert sind, dass ein Vergleich mit konkreten Anwendungstasks möglich ist.

Ein geeigneter Benchmark ist epEBench aus nachfolgenden Gründen.

4.1.2 Benchmark epEBench

Die Software epEBench verwendet zur Ausführung von Tasks sog. Workloads, die einen Mix aus Modellen ausführen[7], die sich wie konkrete Anwendungen (vidplay, gzip) verhalten.

epEBench ermöglicht die Ausführung einzelner Tasks, die als Modell in einer Schnittstelle definiert sind und als Parameter zur Ausführung angegeben werden. Das Listing A.1 im Anhang auf Seite 81 entstammt aus der Schnittstelle eblloads.h und zeigt einen Auszug der vorhandenen Modelle.

4.2 Kriterien ermitteln

Für das Experiment wurde überprüft, ob die Modelle aus epEBench für einen Vergleich mit Anwendungstasks geeignet sind. Nachfolgend wurde exemplarisch das Modell dadd überprüft.

4.2.1 Überprüfung des Modells dadd

epEBench ermöglicht die Ausführung der einzelnen Modelle anhand des Parameters m. Mit dem Parameter t wird angegeben, über wie viele Sekunden ein Modell ausgeführt werden soll. Der Parameter n ermöglicht die parallele Ausführung des Modells in unterschiedlichen Threads. Zusätzlich kann mit dem Parameter a die Ausführung auf eine bestimmte Anzahl CPU-Kerne verteilt werden. Das Listing A.2 im Anhang auf Seite 81 zeigt die Ausführung des Modells dadd.

Die Tasks aus epEBench liegen als Quelltext in der Programmiersprache C vor. Im nachfolgenden Listing wird der Quelltext des Modells dadd vorgestellt:

```
1 void* run_dadd(int counts)
2 {
3     volatile double x = 3.21, y = 0.2, z = 1.34;
4
5     instent += counts*CNT_DIV;
6     while ((counts--) && !done0) {
7         x+=1.121;
8         y+=1.122;
9         z+=1.123;
10        x+=1.124;
11        y+=1.125;
12        z+=1.126;
13        x+=1.127;
14        y+=1.128;
15        z+=1.129;
16        x+=1.120;
17        y+=1.121;
18        z+=1.122;
19        x+=1.123;
20        y+=1.124;
21        z+=1.125;
22        x+=1.126;
23        y+=1.127;
24        z+=1.128;
25        x+=1.129;
26        y+=1.120;
27    }
28    return NULL;
29 }
```

Listing 4.1: eblloads.cpp[7]

Gemäß dieses Listings werden im Modell dadd, in mehreren Iterationen, die Werte von drei Variablen des Datentyps double erhöht. Zur Ausführung dieser Methode werden, für die oberste

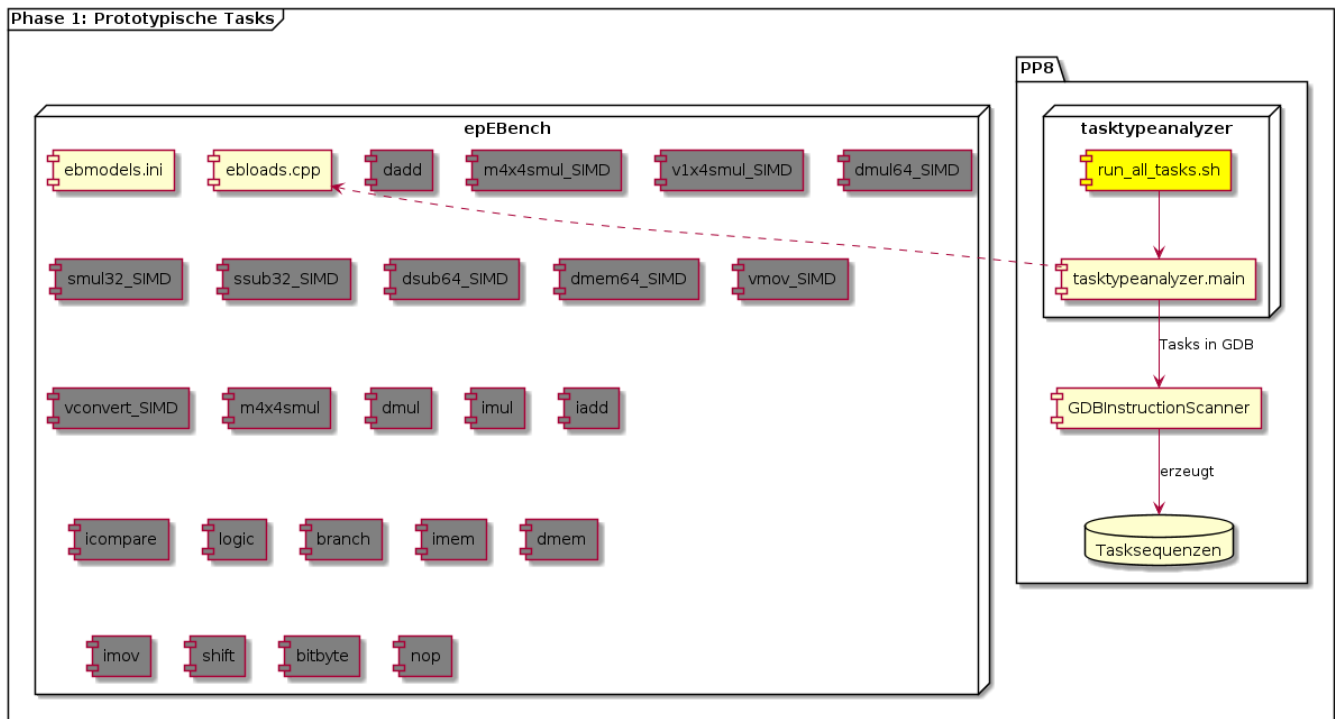


Abbildung 4.1: Komponentendiagramm für Artefakte der ersten Phase

Ebene der Rechnerarchitektur, CPU-Instruktionen erstellt und ausgeführt. Mithilfe des Instruktionsscanners von Holmbacka [5] wurde zunächst das Modell `dadd` ausgeführt, um die CPU-Instruktionen zu erhalten, die bei Ausführung dieses Tasks verwendet werden. Diese sind im Kapitel A.2.4 im Anhang auf Seite 82 hinterlegt.

Eine Anwendung könnte nun einen Task verwenden, in dem ebenfalls Additionen auf double Variablen ausgeführt werden. Dieser Task würde folglich vergleichbare Instruktionen verwenden. Wäre dieser Task auf das Modell `dadd` abbildbar, dann könnte zur Messung der Leistungsaufnahme anstelle des Anwendungstask das Modell `dadd` ausgeführt werden.

Damit alle Modelle aus `epEBench` als prototypische Tasks zur Verfügung stehen, wurden diese ausgeführt und die verwendeten CPU-Instruktionen ermittelt. Dies wurde in der Anwendung `Tasktypeanalyzer` implementiert.

4.3 Implementierung

Aufgrund der zuvor eruierten Anwendungen und Schnittstellen erfolgte die Implementierung der benötigten Artefakte in der Programmiersprache C/C++. Zur Einrichtung der Laufzeitumgebung wurden außerdem Shell-Skripte für das linuxbasierte Testsystem bereitgestellt. Die Abbildung 4.1

auf Seite 15 dient zur Übersicht der implementierten Artefakte dieser Phase.

4.3.1 Vorbereitung

Zur Durchführung des Experiments werden die CPU-Instruktionen der prototypischen Tasks aus dem Benchmark benötigt. Aufgrund des Programmiermodells der CPU, unterscheiden sich die verwendeten Instruktionen je nach System und müssen daher für jedes Testsystem individuell erstellt werden. Die CPU-Instruktionen werden in Form von Sequenzdateien erstellt. Für die Erstellung der Sequenzdateien wird der Instruktionsscanner verwendet. Damit eine Erstellung der Sequenzdateien möglich ist, müssen die Quelltexte aus dem Benchmark mit dem debug-Flag kompiliert werden und unter Linux, mithilfe des Debugging-Werkzeugs GDB, geladen werden. Damit die Sequenzdateien nur die Instruktionen enthalten, die einem bestimmten prototypischen Task entsprechen, wird der zu untersuchende Bereich des Quelltextes mithilfe von Breakpoints auf den eigentlichen Tasks eingeschränkt. Die Breakpoints können einerseits durch Methodennamen definiert werden, andererseits durch Zeilennummern.

Da der Benchmark 24 Task-Typen enthält und zu jedem einzelnen eine Sequenzdatei benötigt wird, ist es wünschenswert die Sequenzerstellung zu automatisieren. Aus diesem Grund wurde die Anwendung Tasktypeanalyzer umgesetzt.

4.3.2 Tasktypeanalyzer

Die Anwendung verwendet Teile des Quelltextes von epEBench, um die einzelnen Tasks zu kompilieren und auszuführen. Dabei ist der Aufruf eines bestimmten Tasks über einen Parameter steuerbar. Das Listing 4.2 zeigt einen Auszug der Mainmethode der Anwendung Tasktypeanalyzer. Bei Aufruf der Mainmethode wird geprüft, ob der Name eines prototypischen Tasks übergeben wurde. Für das Modell dadd wurde eine Konstante im Quelltext hinterlegt. Wenn dieser Taskname übergeben wurde, dann wird die Methode run_dadd aus epEBench aufgerufen.

```
1 #include "ebloads.cpp"
2
3 static const char *const dadd = "dadd";
4
5 int main(int argc, char *argv[]) {
6     if (argc==2) {
7         char* taskname = argv[1];
8
9         if (strcmp(taskname, dadd) == 0)
10        {
11            run_dadd(1);
12            printf("run_%s wurde ausgefuehrt", dadd);
13        }
14        ...
15    }
```

Listing 4.2: main.cpp aus Tasktypeanalyzer

Die eigentliche Erstellung der Sequenzdateien erfolgt außerhalb der Anwendung, durch Aufruf des Shellskriptes run_all_tasks.sh.

4.3.3 run_all_tasks.sh

Das Shellskript run_all_tasks.sh dient zur Generierung der Sequenzdateien und wird im nachfolgenden Listing 4.3 näher vorgestellt.

Das Skript enthält das Datenfeld tasks, welches die Namen der auszuführenden Task-Typen enthält. Das zweite Datenfeld stops enthält die Zeilennummer, in welcher der Task-Typ im Quelltext endet. In der fünften Zeile wird die Anwendung im Debug-Modus kompiliert, danach werden die Ordner für die Ausgabe vorbereitet. Ab der neunten Zeile erfolgt eine Iteration über alle Task-Typen aus dem Datenfeld tasks.

Da die Tasks in einem Debugging-Werkzeug ausgeführt werden, wird ein GDB-Skript erstellt, in dem u. A. die Breakpoints definiert und der Task ausgeführt werden soll. Für die Sequenzerstellung wird außerdem der Instruktionsscanner benötigt, weshalb die Quelldatei ci.py eingefügt wird.

Nach Erstellung des GDB-Skripts wird das Debugging-Werkzeug gdb ausgeführt. Dabei wird das generierte GDB-Skript als Parameter für eine Stapelverarbeitung übergeben. In GDB wird die Anwendung im Debug-Modus ausgeführt. Dabei wird die Anwendung vor Ausführung des prototypischen Tasks gestoppt und die Quelle für den Instruktionsscanner geladen. Der zweite Breakpoint wird an das Ende des prototypischen Tasks gesetzt, sodass mit dem ci-Befehl eine Sequenzdatei mit den Instruktionen geschrieben wird. Dies wird für jeden der hinterlegten prototypischen Tasks wiederholt.

Die Ergebnisse werden im seq-Ordner unter dem Namen des Task-Typen hinterlegt. Zusätzlich werden die Sequenzen an die Anwendung Tasktype-Estimator übergeben.

```
1 #!/bin/bash
2 tasks=('dadd' 'm4x4smul_SIMD' ...)
3 stops=( 1085 137 ...)
4
5 gcc -g main.cpp -o main
6 mkdir -p gdb
7 mkdir -p seq
8
9 for ((i = 0; i < ${#tasks[@]}; i++)); do
10
11     taskname=${tasks[i]}
12     taskstop=${stops[i]}
13     gdbname=$taskname".gdb"
14
15     #GDB-Datei vorbereiten
16     echo "b run_$taskname > $gdbname" >> $gdbname
17     echo "x" >> $gdbname
18     echo "source ci.py" >> $gdbname
19     echo "b $taskstop >> $gdbname" >> $gdbname
20     echo "ci" >> $gdbname
21     echo "exit" >> $gdbname
22
23     gdb --batch --command=$gdbname --args main $taskname > $taskname.gdb.log
24     mv instructionseq.txt seq/$taskname.seq
25     mv $taskname".gdb" gdb/$taskname".gdb"
26     mv $taskname.gdb.log gdb/$taskname.gdb.log
27
28 done
29 cp -f seq/*.seq ../3_tasktype_estimator/taskmapper/taskseq
```

Listing 4.3: run_all_tasks.sh aus Tasktypeanalyzer

4.4 Zusammenfassung der ersten Phase des Vorgehens

Das Benchmarkprogramm epEBench verfügt über 24 Modelle, die laut Holmbacka, aus den Instruktionen realer Anwendungen abgeleitet wurden. Diese Modelle verhalten sich hinsichtlich der Leistungsaufnahme wie reale Anwendungen und sind daher als prototypische Tasks geeignet. Die Modelle können unter unterschiedlichen CPU-Frequenzen ausgeführt werden, die Ausführungszeit und der Parallelitätsgrad sind konfigurierbar. Zu den einzelnen Modellen liegt ein kompilierbarer Quelltext vor, sodass ein Vergleich mit den Tasks einer Anwendung möglich ist. Aus diesen Gründen wurden die Modelle aus epEBench im weiteren Verlauf als prototypische Tasks bzw. Task-Typen verwendet.

Um die Kriterien der prototypischen Tasks auf einem Testsystem ermitteln zu können, wurde die Anwendung Tasktypeanalyzer umgesetzt. Nach Ausführung dieser Anwendung liegen zu den Task-Typen Sequenzdateien vor. Die Abbildung A.2 im Anhang auf Seite 78 zeigt einen Auszug der Sequenzdatei zum Task-Typen bitbyte.

Die Task-Typen werden im Kapitel 8.1 auf Seite 54 präsentiert. Die Logdatei des Tasktypeanalyzers ist im Kapitel B.1 im Anhang auf Seite 101 einzusehen.

Kapitel 5

Phase 2: Tasks einer Anwendung

5.1 Ermittlung einer Beispielanwendung

5.1.1 Kriterien zur Ermittlung geeigneter Anwendungen

Zunächst ist für das Experiment eine taskbasierte Beispielanwendung zu bestimmen. Diese Beispielanwendung wird einerseits benötigt, um die Leistungsaufnahme der CPU bei Ausführung der Tasks abzuschätzen. Andererseits soll auch die tatsächliche Leistungsaufnahme gemessen und mit der Schätzung verglichen werden. Die Anwendung sollte mehrere Tasks ausführen, die sich hinsichtlich der Instruktionen und der Leistungsaufnahme voneinander unterscheiden. Da auch der Parallelitätsgrad zu untersuchen ist, sollte die Anwendung eine Ausführung dieser Tasks auf mehreren CPU-Kernen unterstützen.

Um die Vergleichbarkeit der Anwendungstasks mit den prototypischen Tasks zu ermöglichen, wird der Quelltext dieser Anwendung benötigt, damit die Anwendung für den Instruktionsscanner kompiliert werden kann.

5.1.2 geeignete Anwendungen

Das Listing im Kapitel A.2.5 auf Seite 83 im Anhang zeigt ein mögliches C-Programm welches, als Anwendung für das Experiment in Frage kommen würde. Die Anwendung führt eine Matrixmultiplikation aus und führt diese in eigenen Threads aus. Die Untersuchung hat gezeigt, dass diese Anwendung nicht umfangreich genug ist, da zu wenige unterschiedliche Task-Typen vorhanden waren. Aus diesem Grund wurden andere Beispielanwendungen geprüft.

Im Jahr 2019 beschrieb Hautala die Anwendung stereo depth estimation, welche aus Stereokamerabildern eine Tiefenkarte ableitet[8]. Diese Anwendung kommt aufgrund der implementierten Tasks in Frage, jedoch liegt der Quelltext laut [9] in Python vor, während für die Auswertung der Instruktionen eine C/C++-Implementierung erforderlich ist.

Eine weitere Anwendung ist edgedetection, die im nachfolgenden Abschnitt vorgestellt wird.

5.1.3 edgedetection

Die Anwendung edgedetection führt eine grundlegende Kantenerkennung für RGB-Bilder durch. edgedetection ist als Bildverarbeitungspipeline implementiert und besteht aus folgenden Tasks [22]:

1. loadimage - Laden eines Bildes aus einer CSV-Datei,
2. greyscale - Konvertierung des Bildes in Graustufen,
3. checkcontrast - Überprüfung des Kontrastes,
4. sharpencontrast - Verbesserung des Kontrastes eines Bildes,
5. copyimage - Kopieren einer Bilddatei,
6. sobelv- Anwendung eines vertikalen Kantendetektions-Filters,
7. sobelh - Anwendung eines horizontalen Kantendetektions-Filters,
8. combineimgs - Kombination des vertikalen und horizontalen Kantendetektions-Filters,
9. writeimage - Schreiben des Ergebnis in eine CSV-Datei.

5.1.4 Zusammenfassung

Die Anwendung edgedetection wurde in der Programmiersprache C implementiert und der Quelltext wurde für das Experiment zur Verfügung gestellt. Die Tasks dieser Anwendung sind durch Methoden definiert, wodurch entsprechende Breakpoints für den Instruktionsscanner gesetzt werden konnten. Die Anwendung erlaubt außerdem den Aufruf einzelner Tasks, bei Übergabe entsprechender Methodenparameter. Die Anwendung wurde im Verlauf des Experiments außerdem so angepasst, dass die Ausführung der Tasks auf mehreren Kernen unterstützt wird. Aus diesem Gründen wurde edgedetection als Beispielanwendung ausgewählt.

5.2 Implementierung

Die Tasks der Anwendung edgedetection sollen auf prototypische Tasks abgebildet werden, damit eine Schätzung der Leistungsaufnahme durchgeführt werden kann. Damit diese Abbildung möglich ist, werden Sequenzdateien zu den einzelnen Tasks benötigt. Diese Sequenzdateien enthalten die Instruktionen, die auf Ebene des Programmiermodells der CPU, ausgeführt werden. Wie zuvor bei den prototypischen Tasks, müssen diese Instruktionen für das jeweilige Testsystem individuell erstellt werden. Da die Beispielanwendung aus neun Tasks besteht und zu jedem Task eine Sequenzdatei benötigt wird, ist es wünschenswert die Sequenzerstellung zu automatisieren.

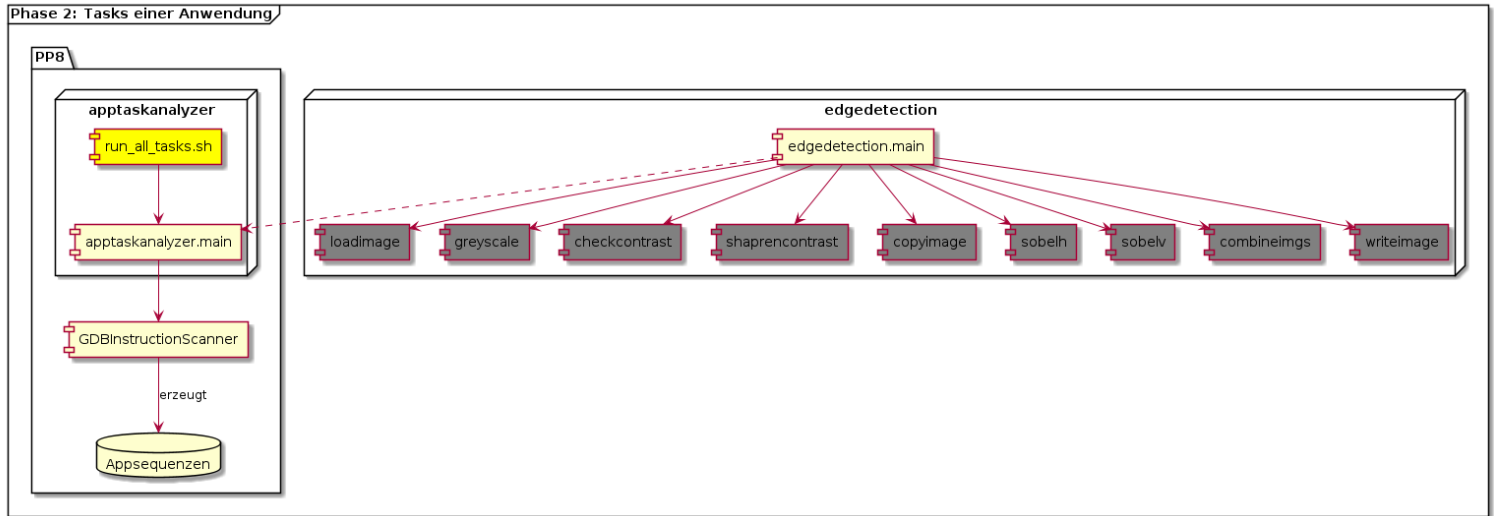


Abbildung 5.1: Komponentendiagramm für Artefakte der zweiten Phase

Desweiteren sollte es mit überschaubarem Aufwand auch möglich sein, die Sequenzerstellung für andere Anwendungen durchzuführen. Aus diesem Grund wurde die Anwendung AppTaskAnalyzer umgesetzt. Die Abbildung 5.1 dient zur Übersicht des implementierten Artefaktes.

5.2.1 Apptaskanalyzer

5.2.1.1 main-Methode

Die Implementierung basiert auf das Artefakt des Tasktypeanalyzer und wird ebenfalls über ein Shellskript gestartet. Der Apptaskanalyzer verwendete den Quellcode der Beispielanwendung edgedetection und führt die einzelnen Anwendungstasks aus. Dabei ist der Name des auszuführende Tasks als Programmparameter zu übergeben. Damit der Anwendungstask ausgeführt werden kann, werden die benötigten Eingabedaten vorbereitet. Das Listing 5.1 zeigt einen Auszug der Main-Methode. Demnach wurden für die Namen der Tasks Konstanten angelegt. Bei Start der main-Methode wird überprüft, ob der Parameter mit dem Wert einer der Konstanten übereinstimmt. Wenn dies der Fall ist, dann wird eine Bilddatei vorbereitet und der entsprechende Task aufgerufen. Da es sich bei edgedetection um eine Bildverarbeitungspipeline handelt, ist es sinnvoll, wenn vor Aufruf des zu untersuchenden Tasks, die vorgelagerten Tasks ausgeführt werden. Für die Erstellung der Instruktionen werden Breakpoints gesetzt, die den eigentlichen Task umfassen, daher können die vorgelagerten Tasks der Bildbearbeitungspipeline aufgerufen werden, ohne dass dessen Instruktionen in die Sequenzdatei geschrieben werden.

```

1 static const char *const task_loadimage = "loadimage";
2 static const char *const task_greyscale = "greyscale";
3 static const char *const task_checkcontrast = "checkcontrast";
4
5 int main(int argc, char *argv[]) {
6     ...
7     char* taskname = argv[1];
8     char filename[64] = "Motorcycle";
  
```

```

9     MyIMG *imgrgb, *imgh, *imgv;
10    imgrgb = createimagergb(XWIDTH, YWIDTH);
11    imgh = createimage(XWIDTH, YWIDTH);
12
13    if (strcmp(taskname, task_loadimage) == 0) {
14        loadimage(imgrgb, filename);
15    } else if (strcmp(taskname, task_greyscale) == 0) {
16        loadimage(imgrgb, filename);
17        greyscale(imgrgb, imgh);
18    } else if (strcmp(taskname, task_checkcontrast) == 0) {
19        loadimage(imgrgb, filename);
20        greyscale(imgrgb, imgh);
21        checkcontrast(imgh);
22    }
23    ...

```

Listing 5.1: Auszug aus main-Methode apptask_analyzer

5.2.1.2 run_all_tasks.sh

Das Shellskript führt nacheinander alle Tasks der Anwendung edgedetection aus und bereitet jeweils eine GDB-Kommandodatei vor. Dabei wird der GDB-Instruktionsscanner ausgeführt und für jeden Task eine Sequenzdatei generiert und im seq-Ordner abgelegt. Die Shellskript entspricht im wesentlichen dem Shellskript aus dem ersten Artefakt, wurde jedoch für die Anwendung edgedetection angepasst. Das Skript ist im Kapitel A.2.6 auf Seite 84 im Anhang hinterlegt.

5.2.1.3 Beschränkung der Sequenzdateien

Bei einigen Tasks hat sich gezeigt, dass die Erstellung der Sequenzdateien sehr viel Zeit in Anspruch nimmt und die Dateien einen Umfang im dreistelligen MegaByte-Bereich erreichen. Das ist deshalb problematisch, da die Sequenzen für eine Abbildung auf prototypische Tasks zu untersuchen sind und das Experiment dadurch erheblich aufwändiger wäre. Die Ursache wurde im Quelltext des Tasks greyscale geprüft und kann im Listing 5.2 nachvollzogen werden.

Die Methode iteriert durch die einzelnen Pixel eines Bildes. Für die Sequenzerstellung wurde ein Bild in der Auflösung 2.964x2.000 gewählt, sodass insgesamt Instruktionen über 5.928.000 Schleifenläufe geschrieben werden würden. Der Zweck dieser Sequenzdateien ist jedoch eine Abbildung des Anwendungstask auf einen oder auf mehrere äquivalente prototypische Tasks. Aus diesem Grund würde die komplette Sequenzdatei keine wesentlich neueren Erkenntnisse hervorbringen. Für die Sequenzerstellung wurden daher Konstanten definiert, die den Pixelbereich auf 20x20 Punkten bzw. 400 Schleifenläufe beschränken. Bei der tatsächlichen Ausführung des Tasks (in der vierten Phase) waren diese Konstanten hingegen nicht vorhanden d. h. bei Ausführung der Tasks wurden dann tatsächlich alle Pixel des Bildes verarbeitet.

Nachdem der Pixelbereich beschränkt wurde, war die Generierung von Sequenzdateien möglich, die für die Abbildungsverfahren geeignet waren.

```

1 static const int maxXWidth = 20;
2 static const int maxYWidth = 20;
3
4 int getwidth(MyIMG *source, int which)
5 {
6     if (which == XDIM) {
7         //return source->xwidth;
8         return maxXWidth;

```

```

9   }
10  if (which == YDIM) {
11      return maxYWidth;
12      //return source->ywidth;
13  }
14
15  return 0;
16 }
17
18
19 void greyscale(MyIMG* imgrgb, MyIMG* img)
20 {
21     int x, y, xwidth, ywidth;
22     int red, green, blue, grey;
23
24     printf("Converting to greyscale...\n");
25     xwidth = getwidth(imgrgb, XDIM);
26     ywidth = getwidth(imgrgb, YDIM);
27
28     for (y = 0; y < ywidth; y++) {
29         for (x = 0; x < xwidth; x++)
30         {
31             red = (int)getpixelrgb(imgrgb, x, y, 0);
32             green = (int)getpixelrgb(imgrgb, x, y, 1);
33             blue = (int)getpixelrgb(imgrgb, x, y, 2);
34             grey = (int)(0.2126 * red + 0.7152 * green + 0.0727 * blue);
35             setpixel(img, x, y, (unsigned char)grey);
36         }
37     }
38 }

```

Listing 5.2: greyscale aus edgedetection

5.3 Zusammenfassung der zweiten Phase des Vorgehens

Mit der Artefakt Apptaskanalyser wurden zu den Tasks der Beispielanwendung Sequenzdateien erstellt, die für eine Abbildung von Anwendungstasks auf prototypische Tasks geeignet waren. Bei der Erstellung der Sequenzdatei ist aufgefallen, dass Schleifenläufe im Millionenbereich umfangreiche Logdateien erzeugen und das Verfahren bei diesen Tasks besonders aufwändig werden könnte.

Die Abbildung 5.2 zeigt einen Auszug der Sequenzdatei zum Anwendungstask checkcontrast. Die Anwendungstasks werden im Kapitel 8.2 auf Seite 55 präsentiert. Das Listing B.2 im Anhang auf Seite 101 zeigt das Protokoll der Anwendung Apptaskanalyser und dient zur Übersicht der Sequenzdateien, die auf dem Testsystem generiert wurden.

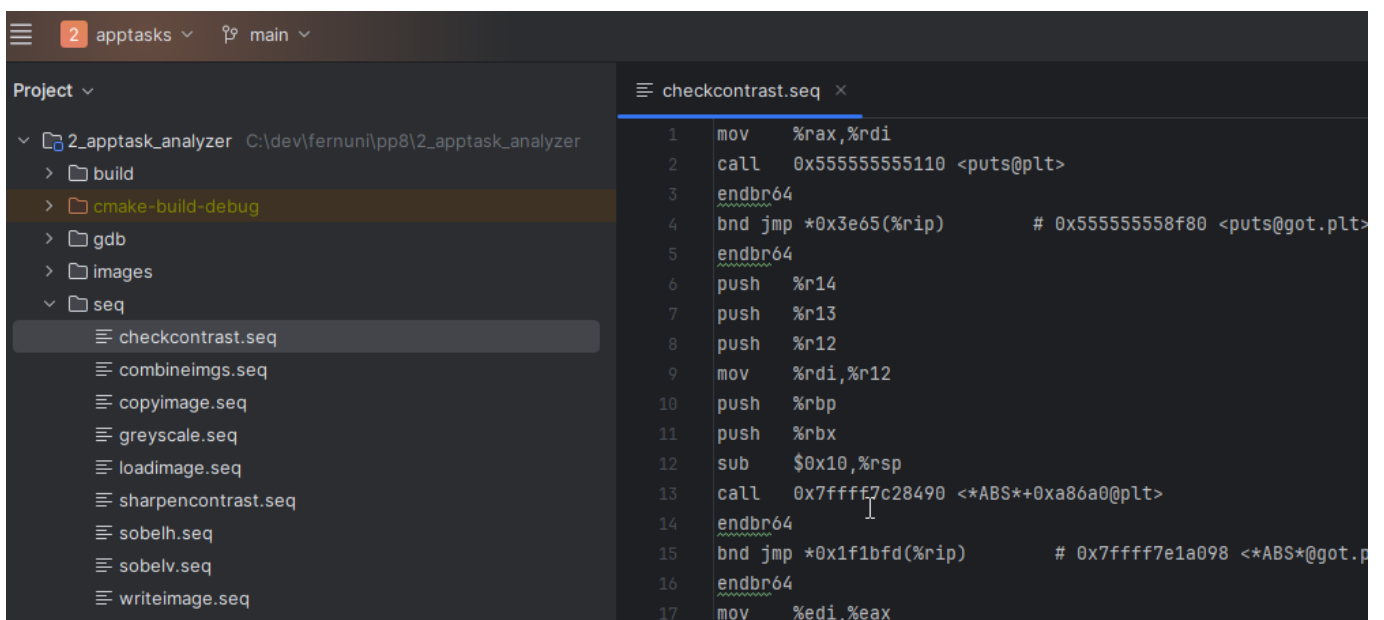


Abbildung 5.2: Sequenzdateien für Tasks von edgedetection

Kapitel 6

Phase 3: Abbildung der Anwendung auf Task-Typen

6.1 Abbildung von Tasks

In den vorangegangenen Phasen wurden Sequenzdateien zu den Task-Typen, sowie zu den Tasks einer Anwendung erstellt. Im nächsten Schritt müssen die Tasks der Anwendung auf äquivalente Task-Typen abgebildet werden. In diesem Kapitel werden mehrere Möglichkeiten geprüft, um eine Abbildung umzusetzen.

6.1.1 Vergleiche zwischen Anwendungstask und prototypischen Tasks

Zunächst einmal könnte der Quelltext des Anwendungstasks mit dem Quelltext der prototypischen Tasks verglichen werden. Es könnten bspw. Kontrollstrukturen wie Schleifen oder die Verwendung bestimmter Datentypen verglichen werden. Der Vergleich muss nicht zwangsläufig ein Textvergleich sein, sondern es wäre auch ein visueller Ansatz möglich, wenn die Quelltexte in Struktogrammen dargestellt werden, um dann Ähnlichkeiten in den Strukturen zu ermitteln.

Eine anderer Ansatz könnte sein, anhand von Logdateien auf Ähnlichkeiten zu schliessen. Dies setzt jedoch voraus, dass sowohl in den Anwendungen, als auch bei den prototypischen Tasks bei vergleichbaren Ereignissen auch vergleichbare Meldungen ausgegeben werden.

Es ist auch denkbar, die Ein- und Ausgabeparameter von Methoden zu vergleichen. Die meisten Anwendungstasks benötigen Eingabeparameter und liefern nach Ausführung ein Ergebnis zurück. Eine Zuordnung könnte dadurch erfolgen, dass ein prototypischer Tasks vergleichbare Eingabeparameter verwendet und nach Verarbeitung ein vergleichbares Ergebnis zurückliefert.

Die angesprochenen Abbildungsverfahren setzen eine hohe Vergleichbarkeit im Quelltext der Anwendungen voraus, die nicht zu erwarten ist. Aus diesem Grund wurde zuvor im Kapitel 2.1.7

auf Seite 7 ein Verfahren genannt, welches eine Abbildung der Tasks auf Grundlage des Programmiermodells der CPU durchführen würde. Dieses Verfahren wird im weiteren Verlauf näher erläutert.

6.2 Vergleich über CPU-Instruktionen

Das Listing A.2.4 im Anhang auf Seite 82 zeigt einen Auszug der Instruktionen zum prototypischen Task dadd. Dabei besteht jede Zeile aus einer Instruktion, die i. d. R. eine Assembleranweisung und je nach Anweisung keine oder mehrere Operanden enthält. Da die Operanden und insbesondere die Adressen für einen Vergleich nicht relevant sind, wurden diese entfernt, sodass nur die eigentlichen Anweisungen bewertet wurde.

6.2.1 CPU-Instruktionen

Im Kapitel 1.3 auf Seite 4 wurde auf die Herstellerseite zur eingesetzten CPU des Testsystems verwiesen. Laut Hersteller wird bei der CPU i5-8365U der Befehlssatz 64-Bit verwendet. Laut Keller legt der Befehlssatz die Grundoperationen eines Prozessors fest [2][s. S. 16]. Weiterhin wären folgende Befehlsarten zu unterscheiden: Datenbewegungsbefehle, Arithmetisch-logische Befehle, Schiebe- und Rotationsbefehle, Multimediabefehle, Gleitkommabefehle, Programmsteuerbefehle, Systemsteuerbefehle und Synchronisationsbefehle. Die verwendeten Befehle aus den Sequenzdateien können im Dokument ‘Intel® 64 and IA-32 Architectures Software Developer’s Manual’ [10] nachgelesen werden.

Beispiel: Die Sequenzdatei zum Task-Typen dadd enthält den Befehl aus dem nachfolgenden Listing:

```
l movsd 0xecc(%rip),%xmm0 \# 0x555555556008
```

Listing 6.1: Assembleranweisung in dadd

Wie man leicht erkennen kann, handelt es sich um einen Datenbewegungsbefehl. Laut Hersteller [10] wird mit dem Befehl movsd ein Doppelwort vom Typ String von einem Quellregister zu einem Zielregister übertragen. Für den Vergleich von Anwendungstasks mit Task-Typen wurden nur die verwendeten Befehle betrachtet, da die verwendeten Operanden keinen relevanten Einfluss auf die Task-Kategorie haben sollten.

Die Instruktionen aus den Anwendungstasks wurden folglich durchsucht und mit den Instruktionen der prototypischen Tasks verglichen. Für diesen Vergleich wurden mehrere Verfahren geprüft und nachfolgend beschrieben.

6.2.2 Komplettsuche

Jeder prototypische Task lässt sich in eine Sequenz aus mehreren Assembleranweisungen überführen. Es wäre nun möglich, in der Sequenz des Anwendungstasks nach der genauen Abfolge der Sequenzen aus den Task-Typen zu suchen. Würde ein bestimmter prototypischer Task aus den Befehlen push, mov, mov bestehen und diese Befehle wären im Anwendungstask vorhanden, dann wäre eine Zuordnung möglich.

Das Kapitel B.1 im Anhang auf Seite 101 zeigt den Umfang der Sequenzen zu den prototypischen Tasks. Demnach besteht ein Task-Typ im Testsystem annähernd aus 2000 Einträgen. Außerdem ist nicht gewährleistet, dass die Befehle in genau der gleichen Reihenfolge ausgeführt werden. Aus diesem Grund sind die Sequenzen aus den prototypischen Tasks in den Sequenzen der Anwendungstask nicht komplett zu finden.

6.2.3 Ähnlichkeitssuche

Anstelle einer Komplettsuche wurde eine Ähnlichkeitssuche verwendet, die einer Mustersuche in Zeichenketten entspricht. Mit einer Ähnlichkeitssuche ist eine Suche nach solchen Ketten gemeint, die dem Muster entsprechen bzw. ihm ähnlich sind[19][s.S. 406].

Das Ziel dieser Ähnlichkeitssuche ist, dass auch dann eine Zuordnung erfolgt, wenn sich die Instruktionen des Anwendungstasks teilweise von den Befehlen eines prototypischen Tasks unterscheiden. Die Schätzung der Leistungsaufnahme anhand von Task-Typen kann nur dann erfolgen, wenn der Anwendungstask auf mindestens einen Task-Typen abgebildet wurde. Aus diesem Grund muss das Suchverfahren für einen Anwendungstask den Task-Typen finden, welcher dem Anwendungstask am ähnlichsten ist. Der ähnlichste Task-Typ ist somit ein Task-Typ aus der Menge der vorhandenen Task-Typen, welcher die meisten Instruktionen des Anwendungstasks enthält.

6.3 Beziehungen zwischen Anwendungs- und prototypischen Task

Die Problemstellung lässt etwas Interpretationsspielraum bei der Definition von prototypischen Tasks zu. So wäre es einerseits möglich, dass ein Anwendungstask genau einem prototypischen Task entspricht. Andererseits wäre es auch denkbar, dass ein Anwendungstask auf mehrere prototypischen Tasks abgebildet werden könnte. Im Rahmen der Agilen Design Research-Methode wurden daher beide Möglichkeiten überprüft und nachfolgend beschrieben.

6.3.1 1:1 Zuordnung

Bei einer 1:1 Zuordnung wird ein Anwendungstask auf genau einen prototypischen Task abgebildet. Dabei sollte der Anwendungstask auf den prototypischen Task abgebildet werden, welcher

dem Anwendungstask hinsichtlich der verwendeten Befehle am ähnlichsten ist, damit eine möglichst präzise Leistungsaufnahme abgeschätzt werden kann. Da die Task-Typen aus konkreten Anwendungen abgeleitet wurden, müssten die prototypischen Tasks bereits soweit variieren, damit ein sinnvolles Abbildungsverfahren möglich ist. Die 1:1-Ähnlichkeitssuche wurde nachfolgend formalisiert:

Sei A die Menge der Befehle des Anwendungstasks und T_i die Menge der Befehle des i -ten Task-Typs. Wir suchen den Task-Typ T_i , der die meisten Befehle mit dem Anwendungstask A gemeinsam hat. Formal ausgedrückt:

$$T_{\text{best}} = \arg \max_{T_i} |A \cap T_i|$$

Hierbei bedeutet:

- $\arg \max_{T_i} |A \cap T_i|$: der Task-Typ T_i , für den die Anzahl der Elemente in der Schnittmenge $A \cap T_i$ maximiert wird,
- $|A \cap T_i|$: die Anzahl der gemeinsamen Befehle (also die Kardinalität der Schnittmenge) zwischen dem Anwendungstask A und dem i -ten Task-Typ T_i .

In Worten: Der Task-Typ, der dem Anwendungstask zugeordnet wird, ist derjenige, der die größte Anzahl gemeinsamer Befehle mit dem Anwendungstask hat.

6.3.2 1:N Zuordnung

Bei einer 1:N Zuordnung wird ein Anwendungstask auf mehrere prototypischen Tasks abgebildet. Dies ermöglicht ggf. eine präzisere Schätzung der Leistungsaufnahme, wenn der Anwendungstask verschiedene Merkmale eines prototypischen Tasks enthält. Da der Benchmark die Konfiguration eines Modells, also eines prozentualen Anteils der auszuführenden prototypischen Tasks ermöglicht, kann folglich auch eine Kombination von prototypischen Tasks, für die Messung der Leistungsaufnahme, ausgeführt werden. Damit jedoch eine Abbildung eines Anwendungstask auf mehrere prototypischen Tasks möglich ist, müssen die Instruktionen des Anwendungstasks segmentweise überprüft werden. Wenn bspw. ein Anwendungstask aus 10.000 Instruktionen besteht und die prototypischen Tasks 2.500 Instruktionen enthalten, dann würde der zu prüfende Bereich aus vier Segmenten bestehen. Jedes dieser Segmente würde dann mit den Instruktionen der Task-Typen verglichen werden und der ähnlichste würde dann zugeordnet werden. Im Ergebnis würde eine Abbildung des Anwendungstasks auf bis zu vier Task-Typen vorliegen.

Dieses Vorgehen ist, im Vergleich zur 1:1-Zuordnung, erheblich aufwändiger in der Ausführung, da der Anwendungstask sehr umfangreicher sein kann und jedes Segment mit allen Task-Typen zu vergleichen ist. Die Ausführungsdauer steigt also mit dem Umfang des Anwendungstasks sowie mit jedem weiteren, zu überprüfenden, prototypischen Task an. Dem gegenüber steht die Möglichkeit einer präziseren Schätzung der Leistungsaufnahme, sofern sich das Ergebnis der 1:1-Abbildung zu weit von dem Anwendungstask unterscheidet. Dieses Verfahren ist also dann von

Vorteil, wenn sich ein Anwendungstask nicht eindeutig einem prototypischen Task zuordnen lässt, etwa weil Instruktionen aus mehreren Task-Typen vorhanden sind. Die 1:N-Ähnlichkeitssuche wurde nachfolgend formalisiert:

Sei $A = \{A_1, A_2, \dots, A_k\}$ die Segmentierung des Anwendungstasks in k Segmente und $T_1, T_2, \dots, T_n$ die Menge der verfügbaren Task-Typen. Für jedes Segment A_j suchen wir den Task-Typ T_i , der die meisten Befehle mit dem Segment A_j gemeinsam hat. Formal können wir dies wie folgt ausdrücken:

$$T_{\text{best}}^{(j)} = \arg \max_{T_i} |A_j \cap T_i| \quad \text{für } j = 1, 2, \dots, k$$

Die Ergebnismenge der zugeordneten Task-Typen ist dann:

$$\{T_{\text{best}}^{(1)}, T_{\text{best}}^{(2)}, \dots, T_{\text{best}}^{(k)}\}$$

Hierbei bedeutet:

- $T_{\text{best}}^{(j)}$: der am besten passende Task-Typ für das j -te Segment A_j des Anwendungstasks,
- $\arg \max_{T_i} |A_j \cap T_i|$: der Task-Typ T_i , für den die Anzahl der Elemente in der Schnittmenge $A_j \cap T_i$ maximiert wird,
- $|A_j \cap T_i|$: die Anzahl der gemeinsamen Befehle zwischen dem j -ten Segment A_j und dem i -ten Task-Typ T_i .

6.4 Implementierung

Die Abbildung der Anwendungstasks auf Task-Typen könnte grundsätzlich manuell durch Vergleich der Textdateien durchgeführt und hergeleitet werden. In den vorangegangenen Phasen wurde festgestellt, dass die Sequenzdateien zu den einzelnen Tasks recht umfangreich sind. Weiterhin wurde festgestellt, dass eine Abbildung von Anwendungstasks auf prototypische Tasks auf Grundlage einer Ähnlichkeitssuche sinnvoll erscheint. Außerdem wäre, neben einer 1:1-Zuordnung, auch

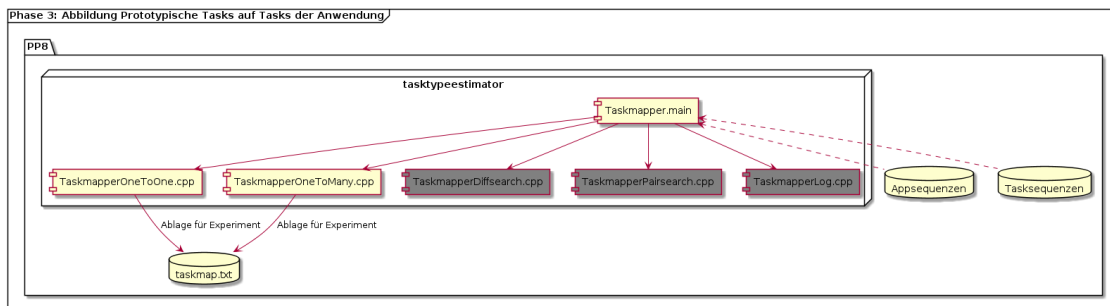


Abbildung 6.1: Komponentendiagramm für Artefakte der dritten Phase

eine 1:N-Zuordnung zur Bewertung beider Verfahren interessant.

Aus diesen Gründen ist, anstelle einer händischen Herleitung, eine Implementierung des Abbildungsverfahrens wünschenswert, zumal das Verfahren dann auch, mit überschaubarem Aufwand, auf anderen Testsystemen oder mit anderen Beispielanwendungen, wiederholt werden könnte. Das Diagramm 6.1 auf Seite 29 zeigt die Komponenten des Taskmappers. Die Sequenzdateien dienen als Eingabeparameter. Es wurde sowohl eine 1:1 -Zuordnung als auch eine 1:N-Zuordnung implementiert. Das Ergebnis wird in der Datei taskmap.txt festgehalten und dient als Eingabeparameter für die Schätzung der Leistungsaufnahme.

6.4.1 Tasktypeestimator

Der Tasktypeestimator besteht aus den Modulen Taskmapper und Estimator. Der Taskmapper ist die Implementierung der dritten Phase, während das Modul Estimator zur Implementierung der vierten Phase gehört.

6.4.1.1 Taskmapper

Die Dateistruktur des Taskmappers wird im Kapitel A.1.3 im Anhang auf Seite 79 vorgestellt. In den nachfolgenden Abschnitten werden einige Komponenten und Quelltexte des Taskmappers vorgestellt.

6.4.2 main.cpp

Nach Start der Mainmethode stehen mehrere Funktionen des Tasktypeestimators zur Auswahl. Die Abbildung A.1 im Anhang auf Seite 77 zeigt die Menüstruktur beim Aufruf der Anwendung. Für den Taskmapper kommen folgende Menüpunkte in Frage:

- Abbildung - 1 AnwTask auf 1 Prototyptask - Test
- Abbildung - Abbildung - 1 AnwTask auf 1 Prototyptask - alle
- Abbildung - Abbildung - 1 AnwTask auf N Prototyptasks - Test
- Abbildung - Abbildung - 1 AnwTask auf N Prototyptasks - alle
- Transfer to epebench

Die Abbildungsverfahren 1:1 und 1:N unterscheiden sich in der Ausführungszeit erheblich, weshalb die Verfahren separat auszuführen sind. Das Abbildungsverfahren wird nach Eingabe des entsprechenden Programmparameters gestartet. Die Ergebnissen der Abbildungen werden in die Datei taskmap.txt geschrieben.

Die einzelnen Task-Typen wurden aus dem Benchmark epEBench abgeleitet. Die einzelnen Task-Typen stehen daher für die Schätzung der Leistungsaufnahme im Benchmark als Modelle zur Verfügung. Bei einer 1:N-Abbildung wird jedoch ein Modell aus mehreren Task-Typen benötigt. Damit dieses Modell im Benchmark vorhanden ist, muss dieses Modell übertragen werden. Für diesen Zweck ist die Funktion ‘Transfer to epEBench’ vorgesehen.

Ein Auszug der Quelldatei main.cpp wurde im Kapitel A.2.7 im Anhang auf Seite 84 vorgestellt.

6.4.2.1 Testdateien für Abbildungsverfahren

Um die Abbildungsverfahren anhand einzelner Anwendungstasks zu überprüfen, besteht die Möglichkeit diese zu testen. Bei diesen Tests wird ein beliebiger Anwendungstask ausgewählt und nach dem zuvor ausgewählten Verfahren auf Task-Typen abgebildet.

Um sicherzustellen, dass die Abbildungsverfahren korrekt implementiert wurden, sind in der Menge der Anwendungstasks zusätzliche Sequenzdateien hinterlegt, welche die Sequenzen einiger Task-Typen beinhalten. Das 1:1-Verfahren lässt sich daher durch Abbildung anhand der Testdatei testWithNop.seq überprüfen. Das 1:N-Verfahren lässt sich anhand der Datei testWithNopBitByteShiftIMulLogic.seq überprüfen. In den Protokolldateien im Kapitel 8.4 auf Seite 58 lässt sich feststellen, dass bei Ausführung des Taskmappers diese Testdaten ausgeführt und zum gewünschten Ergebnis geführt haben. Dies sollte folglich auch für die eigentlichen Abbildungen gelten.

6.4.3 taskmapperOneToOne.cpp

Der Algorithmus 1 auf Seite 32 stellt die Implementierung des 1:1-Abbildungsverfahrens vereinfacht dar. Das Listing A.9 im Anhang auf Seite 85 zeigt die Implementierung aus der Quelldatei taskmapperOneToOne.cpp.

In der Methode compareAppTaskProtTasksOneToOne wird durch die Menge der Anwendungstasks iteriert und für jeden Task findet eine Analyse statt. Diese Analyse wird durch Aufruf der Methode analyseAppTask initiiert. In dieser Methode werden nacheinander die vorhandenen Task-Typen hinsichtlich ihrer Sequenzdateien überprüft. Dabei wird für jeden Befehl aus dem Task-Typen überprüft, ob dieser im Anwendungstask vorkommt. Wenn dieser Befehl gefunden wurde, dann wird dies in einem boolschen Datenfeld markiert. Diese Überprüfung erfolgt in der Methode compareProtTaskSequenEntryWithAppTaskEntry.

Für die 1:1-Abbildung ist zu beachten, wie umfangreich die Sequenzdatei des Anwendungstasks ist. Der Anwendungstask sollte zwar auf den Task-Typen abgebildet werden, dessen Instruktionen am häufigsten im Anwendungstask vorhanden sind. Wenn jedoch der Anwendungstask aus

Algorithm 1: 1:1 Abbildung - vereinfachter Algorithmus

Input: `std::vector<AnwTask> apptaskVektor`, `std::vector<PrototypTask> prottaskVektor`

Output: `taskmap.txt`

```
1 Function main():
2   | Call compareAppTaskProtTasksOneToOne();
3 Function compareAppTaskProtTasksOneToOne():
4   | Call initTaskVektors();
5   | for AnwTask t : apptaskVektor do
6   |   | Call analyseAppTask(t);
7   |   | t.calcBestTask();
8   |   | logBestTask(t, false, true);
9 Function initTaskVektors():
10  | initProttaskVektor();
11  | initAppTaskCollections();
12 Function analyseAppTask(t):
13  | for PrototypTask protTypTask : prottaskVektor do
14  |   | Call prepareAnwTaskAndProtTypTaskForCompareOneToOne(t,
15  |   |   | protTypTask);
16 Function prepareAnwTaskAndProtTypTaskForCompareOneToOne(appTask,
17  | protTypTask):
18  | Call compareAppTaskWithPrototypTasks(appTask, protTypTask);
19 Function compareAppTaskWithPrototypTasks():
20  | for std::string protTaskSequenceEntry : protTypTask.sequenzen do
21  |   | Call
22  |   |   | compareProtTaskSequenEntryWithAppTaskEntry((protTaskSequenceEntry,
23  |   |   |   | appTask, 0));
24 Function
25  | compareProtTaskSequenEntryWithAppTaskEntry(protTaskSequenceEntry,
26  | appTask, maxIndex):
27  | for int i=0; i < appTask.sequenzen.size(); i++ do
28  |   | if appTask.sequenzen[i].compare(protTaskSequenceEntry) == 0 then
29  |   |   | appTask.found[i]=true; // Sequenz gefunden, wird auf true gesetzt!
```

einem Vielfachen von Befehlen des Task-Typens besteht, dann könnten mit hoher Wahrscheinlichkeit, mehrere Task-Typen in Frage kommen, da für die meisten Task-Typen sämtliche Befehle in der Sequenz des Anwendungstasks gefunden werden würden. Es wird jedoch ein Verfahren benötigt, das eine möglichst hohe Ähnlichkeit zwischen Anwendungstasks und Task-Typen festgestellt.

Aus diesem Grund wird die Anzahl der Instruktionen des Anwendungstasks, durch die Anzahl der Instruktionen des Prototypen geteilt, um einen Faktor zu erhalten. Mit diesem Faktor werden die Instruktionen des Task-Typen erneut im Anwendungstask gesucht. Dadurch wird sichergestellt, dass nur der Task-Typ zugeordnet wird, welcher bei vergleichbarem Umfang der Sequenzen, die häufigsten Übereinstimmungen hat.

Beispiel: Der Task greyscale hat, laut Listing B.2 auf Seite 101, 18.296 Einträge. Der Tasktyp nop hat, laut Listing B.1 auf Seite 101, 1.999 Einträge. Würde man die Instruktionen des Task-Typen nop nur einmal in den 18.296 Einträgen des Anwendungstasks suchen, dann würden sehr wahrscheinlich alle Befehle gefunden werden. Dies würde auch für andere Task-Typen gelten, weshalb die Abbildung nun auf mehrere Task-Typen möglich wäre und somit zufällig erfolgen müsste. Um dies zu vermeiden, wird nun die Anzahl der Einträge des Anwendungstasks 18.296 durch die Anzahl aus dem Task-Typen 1.999 geteilt. Dies ergibt einen Faktor von 9,15, welcher abgerundet werden würde. Die Befehle aus dem Task-typ nop werden daher neun mal im Anwendungstask gesucht. Dabei wird jeder gefundene Befehl die Trefferanzahl um 1 erhöhen und bei weiteren Durchgängen nicht noch einmal mitgezählt werden. Da dieses Verfahren auch für alle anderen Task-Typen angewendet wird, sollte der Task-Typ mit den meisten Treffern auch die höchstens Übereinstimmung mit dem Anwendungstask haben.

6.4.4 taskmapperOneToMany.cpp

Der Algorithmus 2 auf Seite 35 stellt die Implementierung des 1:N-Abbildungsverfahrens vereinfacht dar. Das Listing A.10 im Anhang auf Seite 86 zeigt die Implementierung aus der Quelldatei taskmapperOneToMany.cpp.

Die 1:N-Abbildung hat den Zweck, in der Sequenzdatei des Anwendungstasks das Vorkommen unterschiedlicher prototypischer Tasks festzustellen. Der Anwendungstask wird also auf mehrere Task-Typen abgebildet. Der Algorithmus für die 1:N-Abbildung unterscheidet sich von der 1:1-Abbildung dadurch, dass die Sequenzdatei des Anwendungstasks in Segmente unterteilt wird. Für jedes dieser Segmente wird geprüft, welchem Task-Typen dieses Segment am ähnlichsten ist.

Der Algorithmus basiert auf die Implementierung des 1:1-Verfahrens. In der Methode analyseAppTaskMany wird zunächst berechnet, in wieviele Segmente die Sequenz des Anwendungstasks unterteilt wird. Die Anzahl der Segmente berechnet sich aus der Sequenzgröße des Anwendungstasks, durch den Mittelwert der Sequenzgrößen der Task-Typen. In der Methode prepareAnwTaskAndProtTypTaskForCompareOneToOne wird nun ein Index festgelegt, bis zu welcher Position der Anwendungstask durchsucht werden soll. Diese Position be-

rechnet sich aus dem letzten Treffer einer vorherigen Suche, zuzüglich der Sequenzgröße des Task-Typen. Der Suchbereich wird durch diesen Index nur beschränkt, es werden daher auch Positionen gefunden, die vor dem letzten Treffer einer vorherigen Suche liegen.

Um das Suchverfahren zu beschleunigen, wurden ein Suchindex mithilfe einer Map umgesetzt. Die Methode `compareProtTaskSequenEntryWithAppTaskEntryMany` zeigt, wie eine Position anhand des Indexes als gefunden markiert wurde. Da der Anwendungstask in mehrere Segmente unterteilt wird und für jedes dieser Segmente geprüft wird, welchem Task-Typen dieses am ähnlichsten ist, wurde die Verarbeitung parallelisiert. Durch die parallele Verarbeitung wird das Verfahren erheblich früher beendet, als dies bei einer sequentiellen Verarbeitung der Fall wäre. Dennoch kann die Ausführung des 1:N-Verfahrens, insbesondere bei den umfangreicheren Tasks, mehrere Stunden benötigen.

6.4.5 Transfer to epEBench

Während der Ausführung des Taskmappers werden die Ergebnisse der Abbildung in die Textdatei `taskmap.txt` geschrieben. Für die einzelnen Task-Typen sind die Modelle bereits im Benchmark `epEBench` vorhanden. Bei Ausführung des Taskmappers werden jedoch, für die 1:N-Abbildung, weitere Modelle geschrieben, die zunächst im Benchmark `epEBench` hinterlegt werden müssen, damit diese ebenfalls ausgeführt werden können. Damit diese Übertragung nicht händisch erfolgen muss, besteht die Möglichkeit eine Funktion aufzurufen, welche die Modelle aus der Datei `taskmap.txt` in die Datei `ebmodels.ini` übertragen.

Die Implementierung ist im Anhang A.16 auf Seite 91 aufgeführt.

6.5 Eruierung von alternativen Abbildungen

Während der Vorbereitung der Experimente hat sich gezeigt, dass einige Anwendungstasks mit vergleichsweise wenigen Instruktionen, nicht sehr präzise abzuschätzen sind. Im Rahmen des agilen Design Researchs wurden daher auch andere Abbildungsverfahren überprüft und implementiert. Diese werden hier kurz vorgestellt.

6.5.1 Diffsuche

Zu jedem Task-Typen liegt eine Sequenzdatei der verwendeten Assemblerbefehle vor. Dabei bestehen die Sequenzen aus unterschiedlichen Befehlen und es wäre möglich, dass jeder Task-Typ einzelne Befehle verwendet, die in keinem anderen Task-Typen vorkommen. Wäre dies der Fall, dann könnte der Anwendungstask auf das Vorkommen dieser Befehle untersucht werden und ggfls. auf den Task-Typen abgebildet werden.

Algorithm 2: 1:N Abbildung - vereinfachter Algorithmus

Input: `std::vector<AnwTask> apptaskVektor, std::vector<PrototypTask> prottaskVektor`

Output: `taskmap.txt`

```
1 Function main() :
2   | Call compareAppTaskProtTasksOneToMany();
3 Function compareAppTaskProtTasksOneToMany() :
4   | Call initTaskVektors();
5   | for AnwTask t : apptaskVektor do
6     |   t = Call analyseAppTaskMany(t);
7     |   Call logBestTasks(t);
8 Function initTaskVektors() :
9   |   initProttaskVektor();
10  |   initAppTaskCollections();
11 Function analyseAppTaskMany(t) :
12  |   int sizeAppDivProt = ceil(float(appTask.sequenzen.size() / calcSequenzSize()))+1; //
13  |   Der Faktor wird immer aufgerundet!
14  |   for int i = 0; i < sizeAppDivProt; i++ do
15    |   | Call analyse(t, protTypTask);
16 Function analyse(appTask) :
17  |   for int j = 0; j < prottaskVektor.size(); j++ do
18    |   | Call prepareAnwTaskAndProtTypTaskForCompare(appTask,
19    |   |   protTypTask);
20 Function prepareAnwTaskAndProtTypTaskForCompare(appTask,
21   |   protTypTask) :
22  |   int maxIndex = appTask.lastIndexOfTrue() + protTypTask.sequenzen.size();
23  |   appTask.initTaskHitList(protTypTask);
24  |   appTask = Call compareAppTaskWithPrototypTasksMany(appTask,
25  |   |   protTypTask, maxIndex);
26 Function compareAppTaskWithPrototypTasksMany(appTask, protTypTask,
27   |   maxIndex) :
28  |   for std::string protTaskSequenceEntry : protTypTask.sequenzen do
29    |   | appTask = Call
30    |   |   compareProtTaskSequenEntryWithAppTaskEntryMany((protTaskSequenceEntry,
31    |   |   appTask, 0));
32 Function compareProtTaskSequenEntryWithAppTaskEntryMany(protTaskSequenceEntry,
33   |   appTask, protypTask, maxIndex) :
34  |   for int index : indexes do
35    |   | ...
36    |   | appTask.resultOneToOne.pptFoundMapWithBool[protTypTask.taskname][index]=true;
```


Das Listing A.11 im Anhang auf Seite 86 zeigt die Implementierung dieses Suchverfahrens. Die Logausgaben sind im Anhang A.2.11 auf Seite 87 dokumentiert.

Gemäß dieses Listings wurde der Task-Typ `bitbyte` ausgewählt und dessen Instruktionen mit den Instruktionen der übrigen Task-Typen verglichen. Sobald ein Befehl aus `bitbyte` gefunden wurde, wurde die Position des Eintrags in einem boolschen Datenfeld mit dem Wert `true` belegt. Zunächst wurde `bitbyte` mit den einzelnen Task-Typen verglichen und es hat sich gezeigt, dass `bitbyte` im Einzelvergleich mindestens 13 Befehle enthält, die in anderen Task-Typen nicht enthalten sind.

Würden diese 13 Befehle in keinem der anderen Task-Typen vorkommen, dann würde es sich um ein Suchkriterium des Task-Typen `bitbyte` handeln. Daher wurden die Befehle des Task-Typen `bitbyte` nun nacheinander mit allen Task-Typen verglichen. Der Vergleich mit dem Task-Typen `icompare` hat ergeben, dass `bitbyte` 41 Befehle enthält, die nicht in `icompare` vorkommen. Der nachfolgende Vergleich mit dem Task-Typen `imul` hat jedoch gezeigt, dass diese 41 Befehle in `imul` enthalten sind. Folglich enthält der Task-Typ `bitbyte` keine Befehle, die nur in diesem Task-Typen vorhanden sind.

Aus diesem Grund ist diese Implementierung einer Diffsuche für ein Abbildungsverfahren nicht geeignet. Die einzelnen Instruktionen unterscheiden sich zu wenig voneinander, um eindeutig auf einen bestimmten Task-Typen schliessen zu können.

6.5.2 Paarsuche

Bei der Paarsuche wurden Kombinationen aus Vorgänger- und Nachfolgerinstruktion gebildet. Wenn ein Task-Typ eine Folge von zwei Befehlen enthält, die in keinem anderen Task-Typen vorkommen, könnten diese Folgen als Kriterium für eine Abbildung verwendet werden.

Das Listing A.13 im Anhang auf Seite 88 zeigt einen Auszug der Implementierung, das nachfolgende Listing A.14 zeigt außerdem die Bildschirmausgabe bei Ausführung. Zunächst erfolgt eine Iteration durch die einzelnen Befehle des Anwendungstasks `writeimage`, um Paare aus Vorgänger und Nachfolger zu bilden und in einer Map zu speichern. Aus den Instruktionen der Task-Typen wurden ebenfalls Paare gebildet und mit den Paaren der anderen Task-Typen verglichen, um eindeutige Paare zu ermitteln. Diese eindeutigen Paare wurden für jeden Task-Typen gesichert. Danach wurden nacheinander alle Paare aus dem Anwendungstask mit den eindeutigen Paaren des Task-Typen verglichen. Wenn im Anwendungstask ein Paar gefunden wurde, welches sich eindeutig einem Paar des Task-Typen zuordnen lässt, wurde dies als Treffer gezählt.

In dem Anwendungstask waren zu jedem der Task-Typen ungefähr 350 eindeutige Paare enthalten d. h. im Anwendungstask kommen zwar Folgen vor, die sich eindeutig einem Task-Typ zuordnen lassen, dies war jedoch für alle Task-Typen der Fall. Daher wurde im nächsten Schritt geprüft, aus welchem Task-Typen die meisten eindeutige Paare vorgekommen sind.

Beispiel: Der Tasktyp `m4x4smul_SIMD` enthält laut der Bildschirmausgabe A.14 180 eindeutige Paare sowie 916 nicht eindeutige Paare. Das bedeutet, dass dieser Task-Typ aus 180 Vorgänger-/Nachfolgerkombinationen besteht, die in keinem anderen Task-Typen gefunden wurden. Nun wurden aus den Instruktionen des Anwendungstasks `writeimage` ebenfalls Paare gebildet und nacheinander mit den eindeutigen Paaren der Task-Typen verglichen. Der Anwendungstask `writeimage` enthält 356 eindeutige Paare aus dem Task-Typen `icompare`. Im Anwendungstask `writeimage` wurden jedoch die meisten eindeutigen Paare aus dem Task-Typen `m4x4smul_SIMD` gefunden, nämlich 368. Daher wäre der Task-Typ `m4x4smul_SIMD` ein Kandidat für eine 1:1-Abbildung.

Im vorgestellten Beispiel variiert die Anzahl der eindeutigen Kombinationen pro Task-Typ nicht stark genug. Des weiteren erscheint die Anzahl der eindeutigen Kombinationen unerwartet hoch. Die berechnete Abbildung wurde exemplarisch für eine Abschätzung der Leistungsaufnahme getestet, die Auswertung hat aber in den betrachteten Fällen zu keinem präziseren Ergebnis geführt, weshalb dieses Abbildungsverfahren nicht weiter untersucht wurde. Die Messwerte aus diesem Versuch sind in den Logdateien nachzulesen, die in der Tabelle 6.1 aufgeführt sind.

Art	URL zur Protokolldatei
Paarsuche vom 13.04.2024	https://github.com/allankaufmann/pp8/blob/main/3_tasktype_estimator/results/20240413_18_00_result_paarsuche.txt
Paarsuche vom 14.04.2024	https://github.com/allankaufmann/pp8/blob/main/3_tasktype_estimator/results/20240414_12_30_result_paarsuche.txt

Tabelle 6.1: Logdateien für Paarsuche

6.5.3 Drillingsuche

Auf Grundlage der Paarsuche wurde auch eine Drillingsuche eruiert. Bei diesem Verfahren wurden aus den Befehlen des Task-Typen ‘`imul`’ Dreiergruppen gebildet und mit den Dreiergruppen der übrigen Task-Typen verglichen. Das Ziel war für jeden Task-Typen einige charakteristische Befehlsfolgen zu identifizieren, die eine Zuordnung zu einem Anwendungstask ermöglichen. Die Implementierung wird im Listing A.15 im Anhang auf Seite 90 vorgestellt. Im wesentlichen wurde die Paarsuche um einen Befehl erweitert.

Für den Task-Typen ‘`imul`’ wurden laut Logdatei 586 eindeutige Drillinge identifiziert. Es existieren daher 586 Befehlsfolgen, die in den anderen Task-Typen nicht gefunden wurden. Da dieser Task-Typ aus insgesamt 1.668 Drillingen besteht und somit etwa ein Drittel nur in diesem Task-Typen vorkommen, ist bei den anderen Task-Typen mit vergleichbaren Umfängen zu rechnen. Die Zuordnung wäre dadurch möglich, dass der Anwendungstask daraufhin geprüft wird, aus welchem Task-Typen die meisten Dreierkombinationen enthalten sind.

Aufgrund der festgestellten Menge der eindeutigen Dreierkombinationen wurde dieses Verfahren nicht weiterverfolgt und das Experiment auf Grundlage der Ähnlichkeitssuche fortgesetzt.

6.6 Zusammenfassung der dritten Phase des Vorgehens

Mit dem Taskmapper wurden die Tasks von edgedetection auf die prototypischen Tasks abgebildet. Dabei wurde sowohl ein 1:1-Abbildungsverfahren, als auch ein 1:N-Abbildungsverfahren umgesetzt. Es wurden weitere Abbildungsverfahren eruiert, mit denen sich anscheinend jedoch keine präzisere Schätzung der Leistungsaufnahme erreichen lässt.

Die Abbildung der Anwendungstasks auf die prototypischen Tasks wird im Kapitel 8.3 auf Seite 56 präsentiert.

Kapitel 7

Phase 4: Schätzung der Leistungsaufnahme

7.1 Leistungsaufnahme der CPU messen

7.1.1 Auswertung des Energiezählers

Wie im Kapitel 2.1.2 auf Seite 5 erläutert, wurden die Energiezähler mithilfe des Powercap-Frameworks, über die Intel-RAPL-Schnittstelle, ausgelesen. Ein Beispiel, wie der Energiezähler mithilfe der Anwendung `rapl-read` ausgelesen werden kann, ist im Kapitel A.2.3 auf Seite 82 im Anhang zu entnehmen. Damit die Leistungsaufnahme eines Tasks festgestellt werden kann, wäre es nun denkbar, diese Anwendung während der Ausführung eines Tasks zu beobachten und die Änderungen zu dokumentieren. Bei der Auswertung der Leistungsaufnahme ist jedoch zu beachten, dass der Energiezähler stets den Energieverbrauch der kompletten CPU anzeigt. Es steht kein Energiezähler für die einzelnen CPU Kerne zur Verfügung[25]. Der Energiezähler lässt sich auch nicht ohne weiteres auf einzelne Anwendungen oder Prozesse beschränken, sodass bei der Auswertung stets der komplette Energieverbrauch des Testsystems zu berücksichtigen ist.

Damit möglichst präzise Messungen möglich sind, sollte der Stand des Energiezählers unmittelbar vor Ausführung des Tasks ausgewertet werden. Sobald die Ausführung des Tasks beendet ist, sollte der Stand des Energiezählers erneut ausgelesen werden, damit die Differenz gebildet werden kann. Die Verwendung von Anwendungen wie `rapl-read` würde ggfs. zu Ungenauigkeiten führen, da die gewünschten Zeitpunkte zur Messung des Energiezählers schwer zu konfigurieren sind. Für die Auswertungen wurde daher eine Implementierung benötigt, die einerseits die gewünschten Tasks ausführt, andererseits unmittelbar vor und nach Ausführung, den Stand des Energiezählers ausliest und bereitstellt.

Bei den ersten Messungen des Energieverbrauchs wurden einige Tasks mehrmals hintereinander gemessen und es hat sich gezeigt, dass die Leistungsaufnahme stark variierte. Das lag daran, dass das Betriebssystem auch andere Dienste ausgeführt hat, die Energie verbraucht haben. Aus diesem Grund wurden die Messungen im abgesicherten Modus des Betriebssystems ausgeführt. Im abgesicherten Modus sind nur essentielle Dienste aktiv und die grafische Oberfläche ist nicht gestartet. Die Messungen im abgesicherten Modus haben schliesslich konstantere Messergebnisse

hervorgebracht, weshalb daraufhin alle weiteren Messungen im abgesicherten Modus des Betriebssystems ausgeführt wurden.

7.1.2 Parallelitätsgrad

Im Kapitel 2.1.4 auf Seite 6 wurde der Parallelitätsgrad als ein Faktor zur Bewertung der Leistungsaufnahme genannt. Weiterhin wurde erklärt, dass zur Bewertung der Leistungsaufnahme die Ausführung eines Tasks auf virtuellen Prozessoren nicht zu untersuchen ist, sondern die Tasks auf mehreren physikalischen Kernen ausgeführt werden sollen. Da das verwendete Testsystem über eine CPU mit vier Kernen verfügt, ergeben sich folglich vier Ausprägungen des Parallelitätsgrades. Der Parallelitätsgrad ist sowohl bei Ausführung der Task-Typen, als auch bei Ausführung der Anwendungstask zu berücksichtigen. Wenn bspw. ein Anwendungstask auf einem Kern ausgeführt wird, dann sollte der entsprechende Task-Typ ebenfalls auf einem Kern ausgeführt werden. Dies gilt entsprechend für die weiteren Parallelitätsgrade.

Für die Task-Typen wurde bereits im Kapitel 4.2.1 auf Seite 14 festgestellt, dass der Parallelitätsgrad mithilfe eines Parameters konfiguriert werden kann. Wenn ein Anwendungstask auf einen bestimmten Task-Typen abgebildet und auf einer bestimmten Anzahl Kernen ausgeführt wurde, dann sollte auch dieser Task-Typ auf entsprechend vielen Kernen ausgeführt werden.

Für die Anwendungstasks wurde der Quellcode der Beispielanwendung erweitert, damit die Tasks auf entsprechend vielen Kernen ausgeführt werden.

7.1.3 CPU-Frequenz

Das Testsystem verwendet eine CPU des Modells i5-8365U. Laut Herstellerspezifikation ist der Grundtakt bei 1.60 GHz eingestellt und kann auf bis zu 4.10 GHz konfiguriert werden. Für die Messung und Bewertung der Leistungsaufnahme ist die Frequenz ein zu untersuchender Faktor, weshalb Messungen unter unterschiedlichen Frequenzleveln durchgeführt wurden.

Für das Testsystem wurden folgende Frequenzlevel in MHz konfiguriert:

```
1 [CPUFrequency]
2 level=1500
3 level=2000
4 level=2200
5 level=2400
6 level=2500
7 level=3000
8 level=3500
9 level=4000
```

Listing 7.1: Frequenzlevel für i5-8365U

Die grundlegende CPU-Frequenz ist im BIOS des Testsystems eingestellt. Da bei den Messungen unterschiedliche Frequenzlevel zu untersuchen sind, ist es wünschenswert die Taktfrequenz der CPU während der Laufzeit ändern zu können. Dies ist auf Linuxsystemen mit der Anwendung

cpupower möglich. Die gewünschte Taktfrequenz wird daher mit der Anwendung cpupower eingestellt. Näheres zur Anwendung cpupower ist auf der Herstellerseite zu entnehmen: https://wiki.archlinux.org/title/CPU_frequency_scaling

7.1.3.1 Änderung der CPU-Frequenz zur Laufzeit

Um die CPU-Frequenz zur Laufzeit anzupassen, enthält der Tasktypeestimator eine Implementierung, die das gewünschte Frequenzlevel setzt. Es ist zu beachten, dass die Umstellung der CPU-Frequenz nur mit root-Berechtigungen funktioniert. Das nachfolgende Listing 7.2 zeigt die Implementierung. Im wesentlichen wird auf Systemebene die Anwendung cpupower aufgerufen. Damit die CPU Taktfrequenz geändert werden kann, muss diese Anwendung auf dem Betriebssystem vorhanden sein. Das Kapitel C im Anhang auf Seite 124 stellt eine Funktion vor, die alle relevanten Vorbedingungen prüfen, damit das Experiment ausgeführt werden kann.

```
1 void setupCpuFrequenzlevel(std::string frequenz) {  
2     currentCPUFreq = frequenz;  
3     std::string frequency = "sudo -S cpupower frequency-set -u " + currentCPUFreq + "mhz";  
4     system(frequency.c_str());  
5     system("cpupower frequency-info");  
6     printf("CPU Frequenz wurde auf %s MHz eingestellt!\n", frequenz.c_str());  
7 }
```

Listing 7.2: Setzen des CPU-Frequenzlevels

7.1.4 Leistungsaufnahme von Task-Typen messen

In den vorangegangenen Schritten wurde sichergestellt, dass sowohl der Parallelitätsgrad als auch das Frequenzlevel konfiguriert wird. Des weiteren wurden die Anwendungstasks auf Task-Typen abgebildet. Für die nächsten Schritte muss die Leistungsaufnahme dieser Task-Typen festgestellt werden, damit eine Schätzung der Leistungsaufnahme für die Anwendungstasks möglich ist. Dazu werden die Task-Typen im Benchmark epEBench fünf Sekunden lang ausgeführt.

Um den Aufwand zur Feststellung der Leistungsaufnahme zu reduzieren, wird die Leistungsaufnahme eines Task-Typen nur einmalig pro Kombination ermittelt. Die Messungen werden zwar fünfmal wiederholt, um sicherzustellen, dass es sich um konstante Messergebnisse handelt. Wenn jedoch ein Anwendungstask auf einen bestimmten Task-Typen abgebildet wurde, der bereits zuvor gemessen wurde, dann wird dieser Task-Typ nicht erneut gemessen. Der Aufwand zur Feststellung der Leistungsaufnahme würde also reduziert werden, wenn mehrere Anwendungstasks auf einen Task-Typen abgebildet worden sind.

Damit die Leistungsaufnahme eines Task-Typen festgestellt werden kann, wird ein Artefakt benötigt, welches zunächst den Stand des Energiezähler ausliest, unmittelbar danach den entsprechenden Task-Typen im epEBench startet und direkt im Anschluss den Energiezähler erneut ausliest.

7.1.5 Leistungsaufnahme der Anwendungs-Tasks messen

Damit eine möglichst präzise Messung der Anwendungs-Tasks möglich ist, wurde der Quelltext der Beispielanwendung `edgedetection` soweit angepasst, dass einzelne Tasks mithilfe eines Parameters ausgeführt werden. Unmittelbar vor und nach Ausführung der Tasks wird der Energiezähler gelesen und das Ergebnis in eine Textdatei geschrieben, die im weiteren Verlauf ausgewertet wird.

7.2 Schätzung der Leistungsaufnahme

Aus der Datei `taskmap.txt` geht hervor, welchen Task-Typen ein Anwendungstask am ähnlichsten ist. Damit zu einem Anwendungstask die Leistungsaufnahme der CPU geschätzt werden kann, werden die entsprechenden Task-Typen ausgeführt und dessen Leistungsaufnahme festgestellt. Bei dieser festgestellten Leistungsaufnahme handelt es sich folglich um die geschätzte Leistungsaufnahme bei Ausführung des Anwendungstasks.

Die Schätzung der Leistungsaufnahme wird für alle Frequenzlevel und Parallelitätsgraden wiederholt, damit für jede Kombination eine Schätzung vorhanden ist, die mit der tatsächlichen Leistungsaufnahme verglichen werden kann.

7.3 Vergleich Schätzung und Messung der Leistungsaufnahme

Zuvor wurde erklärt, dass für jede Kombination aus Frequenzleveln und Parallelitätsgraden die Leistungsaufnahme der CPU geschätzt wurde. Damit die Präzision dieser Schätzungen bewertet werden kann, wurden die Anwendungstask ebenfalls in den jeweiligen Frequenzleveln und Parallelitätsgraden ausgeführt und die tatsächliche Leistungsaufnahme gemessen. Die tatsächliche Leistungsaufnahme wurde daraufhin mit den Schätzungen verglichen um die absolute, sowie die prozentuale, Abweichung zu ermitteln.

7.4 Implementierung

Das Vorgehen zur Schätzung und Bewertung der Leistungsaufnahme wurde in der Anwendung `Tasktypeestimator` implementiert. Das Diagramm 7.1 stellt die wichtigsten Komponenten dieser Phase dar. Damit eine Schätzung der Leistungsaufnahme möglich ist, werden Schnittstellen zum Benchmark `epEBench`, sowie zur Beispielanwendung `edgedetection` benötigt. Weiterhin benötigt die Anwendung Zugriff auf die RAPL-Schnittstelle, sowie auf die Anwendung `cpupower`.

Phase 4: Schätzung der Leistungsaufnahme

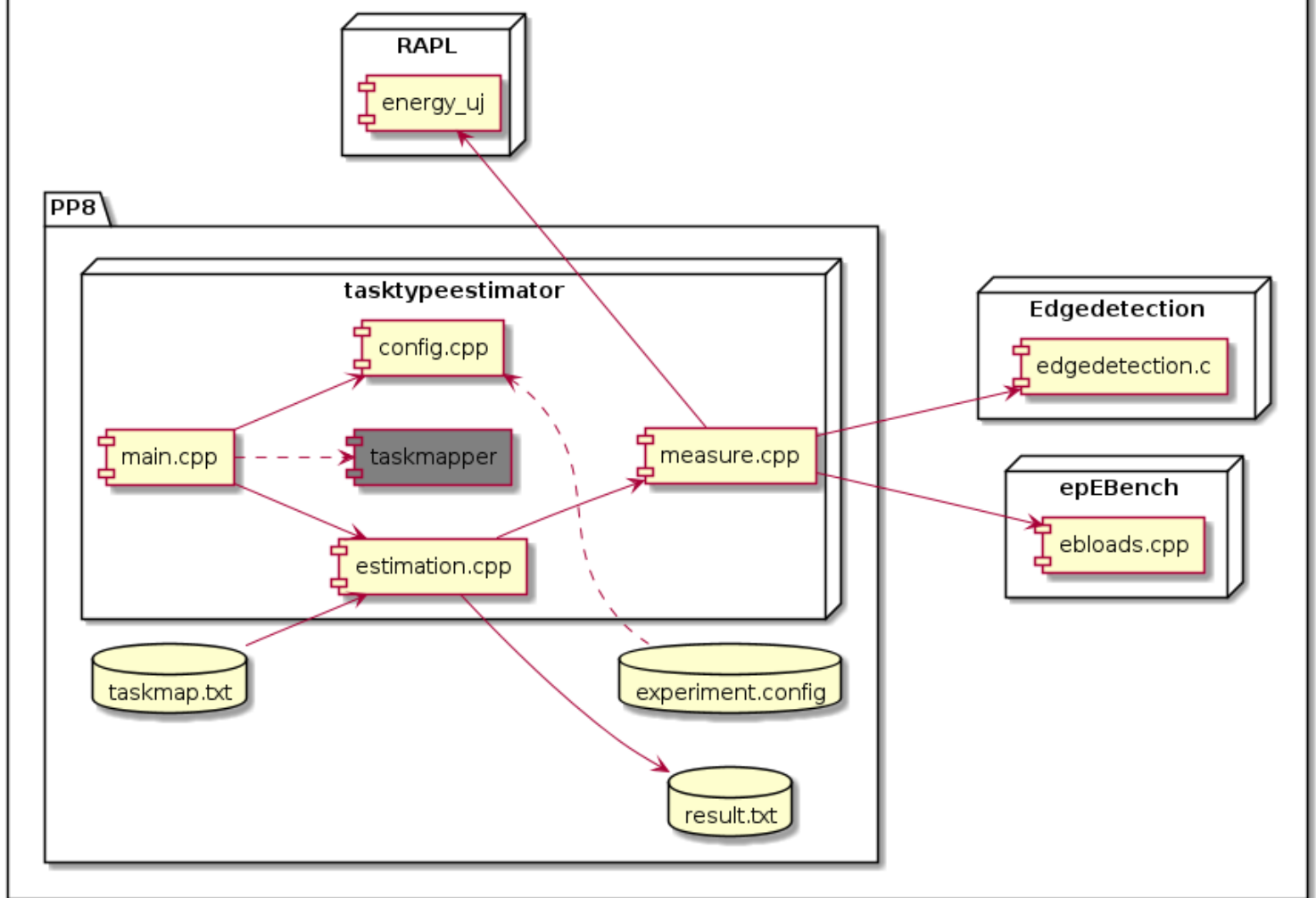


Abbildung 7.1: Komponentendiagramm für Artefakte der vierten Phase

7.4.1 Dateistruktur des Artefakts

Die Abbildung C.1 im Anhang auf Seite 125 zeigt die Dateistruktur des Artefakts. Die Quelldateien zur Ausführung des Estimators befinden sich im Unterordner `estimation`. Die Datei `measure.cpp` enthält Methoden zur Messung des Energieverbrauchs, während die Datei `estimator.cpp` die Methoden zur Durchführung der Schätzung enthält.

Die Anwendung wird anhand der Datei `experiment.config` konfiguriert. In dieser Datei werden u. A. die Frequenzlevel und Tasks definiert. Die Konfigurationsdatei wird in der Quelldatei `config.cpp` verarbeitet. Bei dieser Verarbeitung werden u. A. Skripte zur Ausführung der Task-Typen, sowie Skripte zur Ausführung der Anwendungstasks, generiert und in den ‘gen’-Ordner geschrieben.

Die Ergebnisse nach Ausführung des Experiments werden als Textdatei in den Ordner `results` geschrieben.

7.4.2 `config.cpp`

Das Listing A.17 im Anhang auf Seite 92 zeigt die Konfigurationsdatei `experiment.config` für den Tasktypeestimator. Die Konfigurationsdatei enthält die Namen der Task-Typen und der Anwendungstasks. Außerdem werden die CPU-Frequenzlevel sowie die Parallelitätsgrade definiert.

Der Tasktypeestimator ist weitestgehend von der Beispielanwendung, sowie von dem Benchmark, entkoppelt. Das Artefakt verwendet also nicht den Quelltext dieser Anwendungen, sondern die Interaktion und der Datenaustausch erfolgt über Shellskripte und Textdateien. Der Tasktypeestimator kann daher mit überschaubarem Anpassungsaufwand auch für andere Anwendungen oder Benchmarks verwendet werden. Die Shellskripte werden benötigt, um die auszuführenden Tasks zu parametrisieren und zu starten.

Zur Durchführung müsste nun für jeden Task ein entsprechendes Skript erstellt werden. Sollten zukünftig weitere Tasks definiert werden, müssten auch weitere Skripte hinterlegt werden. Für die Schätzung der Leistungsaufnahme muss außerdem beachtet werden, dass im 1:N-Abbildungsverfahren ein entsprechendes Modell auszuführen ist und folglich ein weiteres Skript für jeden Anwendungstask benötigt wird.

Um die Erstellung dieser Skripte zu vereinfachen, wurde diese Erstellung automatisiert. Das Listing A.18 im Anhang auf Seite 93 zeigt die Implementierung. Die Namen der einzelnen Tasks wurden zuvor aus der Konfigurationsdatei gelesen. Bei Ausführung der Skripte wird der Parallelitätsgrad benötigt, weshalb ein entsprechender Parameter vorbereitet wird. Des Weiteren wird eine Protokolldatei vorbereitet und die eigentliche Anwendung aufgerufen.

7.4.3 `measure.cpp`

7.4.3.1 Messung der Leistungsaufnahme von Task-Typen

Die Quelldatei `measure.cpp` enthält die Methoden, die zur Auswertung des Energiezählers dienlich sind. Das Listing A.19 im Anhang auf Seite 93 zeigt die Implementierung.

Bei der Messung des Energieverbrauchs von Task-Typen, werden diese im Benchmark ausgeführt und der Energieverbrauch über den Energiezähler ermittelt. Der Aufruf des Benchmarks erfolgt über die generierten Shellskripte, welche über die Methode `runAndMeasureScript` gestartet werden. Vor dem Aufruf des Tasks werden der Zeitpunkt gesichert und der Energiezähler über die Methode `readEnergy_UJ_secure` ausgelesen. Bei dem Auslesen des Energiezählers wird anhand einer Schleife sichergestellt, dass der Energiezähler zuvor einen aktuellen Zählerstand eingenommen hat[16][s.S. 2]. Der Aufruf des Tasks erfolgt in einem eigenen Thread und das Hauptprogramm wartet die Abarbeitung dieses Skriptes ab. Unmittelbar nach Ende der Ausführung werden sowohl Zeitpunkt als auch Zählerstand erneut ausgelesen.

Die Berechnung der Leistungsaufnahme wurde bereits im Kapitel 2.1.2 auf Seite 5 erklärt. Zunächst wird die Ausführungsdauer in Millisekunden ermittelt. Die Dauer der Ausführung ergibt sich durch Subtraktion des Zeitpunktes (in Millisekunden) bei Ende der Ausführung, mit dem Zeitpunkt des Ausführungsbeginns. Der Energieverbrauch ergibt sich durch Subtraktion des Energiezählers bei Ende der Ausführung, mit dem Stand des Energiezählers vor Ausführungsbeginn und liegt in Mikrojoule (μJ) vor.

Die Leistungsaufnahme der CPU wird dadurch berechnet, dass dieser Energieverbrauch durch die Dauer geteilt wird und liegt in der Einheit $\mu\text{J}/\text{ms}$ (Mikrojoule pro Millisekunde) bzw. Milliwatt (mW) vor.

Die ausgelesenen Daten werden in einer Instanz der Klasse `MeasureResult` gespeichert und im weiteren Verlauf in die Ergebnisdatei geschrieben. Die Berechnung der Leistungsaufnahme erfolgt durch Aufruf der Methode `power()`. Die Messwerte werden zusätzlich im Unterordner `logs/measure` protokolliert.

7.4.3.2 Parallelitätsgrad von Task-Typen bei Messung

Im Kapitel 4.2.1 auf Seite 14 wurden die Parameter zur Ausführung eines Modelles im Benchmark `epEBench` vorgestellt. Der Parameter `n` ermöglicht die Ausführung eines Modelles auf unterschiedlichen Threads. Da auf dem Testsystem außerdem das Multithreading deaktiviert wurde, werden diese Threads auf unterschiedlichen CPU-Kernen verteilt. Im vorherigen Kapitel A.18 auf Seite 93 wurden außerdem die Shellskripte zum Aufruf der Task-Typen generiert. Damit nicht für jeden Parallelitätsgrad ein Skript benötigt wird, wird dieser beim Aufruf des Shellskriptes als Parameter `$CORES` übergeben und wird beim Aufruf von `epEBench` verwendet.

Das nachfolgende Listing 7.3 zeigt wie der Wert der Variablen \$CORES übergeben wird. Zunächst wurde der Dateiname des Shellskriptes auf Grundlage des Tasknamens ermittelt. An diesen Dateinamen wird nun, von einem Leerzeichen getrennt, der Wert der Variable cores angehängt. Bei Aufruf dieses Skriptes wird also der gewünschte Parallelitätsgrad als Parameter angehängt. Dieser Wert wird schliesslich auch beim Aufruf von epEBench gesetzt.

```
1 std::string getFilenameWithParam(char* filename, std::string cores) {
2     std::string fileNameWithParam = filename;
3     fileNameWithParam+= " " + cores;
4     return fileNameWithParam;
5 }
6 void smtoff() {
7     system("echo off > /sys/devices/system/cpu/smt/control");
8 }
9
10 void smton() {
11     system("echo on > /sys/devices/system/cpu/smt/control");
12 }
```

Listing 7.3: Messung Anwendungstask mit Parallelitätsgrad

Im Kaptel 2.1.4 auf Seite 6 wurde erklärt, dass zur Messung der Leistungsaufnahme das simultane Multithreading nicht berücksichtigt werden soll. Aus diesem Grund wurde das simultane Multithreading im Testsystem deaktiviert. Um sicherzustellen, dass die Tasks auch tatsächlich auf mehreren Kernen ausgeführt werden, stellt der Tasktypeestimator im Betriebssystem das simultane Multithreading aus. Das vorangegangene Listing zeigt die Methode smtoff(). Für die Deaktivierung des simultanen Multithreadings sind root-Berechtigungen erforderlich.

7.4.3.3 Messung der Leistungsaufnahme von Anwendungstasks

Es wäre denkbar, dass die Feststellung der Leistungsaufnahme bei Ausführung der Anwendungstask auf gleiche Art und Weise erfolgen würde. Der Tasktype-Estimator würde demnach einen bestimmten Anwendungstask über ein Shellskript starten und währenddessen die Dauer und den Energiezähler auswerten. Dies würde jedoch dazu führen, dass die Messung nicht nur den eigentlichen Task umfasst, sondern auch vorgelagerte und nachgelagerte Anwendungstasks.

Beispiel: Die Leistungsaufnahme der CPU wurde für den Anwendungstask greyscale abgeschätzt. Diese Schätzung soll nun mit der tatsächlichen Leistungsaufnahme verglichen werden. Damit die Leistungsaufnahme von greyscale gemessen werden kann, muss zunächst die Anwendung edgedetection gestartet werden. Damit dieser Task nun eine Bilddatei konvertieren kann, muss dieses Bild zunächst geladen werden. Nach Ausführung dieses Tasks muss zwar die Bildverarbeitungspipeline nicht fortgesetzt werden, es findet aber noch eine Beendigung der Anwendung statt. Damit eine genauere Messung des Anwendungstasks möglich ist, müsste der Tasktypeestimator den Quelltext von edgedetection verwenden, um entsprechende Messpunkte platzieren zu können. Wenn dieses Verfahren jedoch für andere Beispielanwendungen wiederholt werden soll, müssten auch diese Quelltexte verwendet werden. Dennoch wäre es wünschenswert, wenn die Messpunkte direkt vor und nach Ausführung des Anwendungstasks greyscale gesetzt werden könnten. Dies wurde im Rahmen der Agilen Design Research Methode behandelt.

Um einerseits eine genauere Messung des Anwendungstask zu ermöglichen, andererseits den Tasktype-Estimator von der Beispielanwendung zu entkoppeln, wurden die Messungen in edgedetection implementiert. Der Tasktype-Estimator ruft zwar die Anwendung auf und übergibt entsprechende Parameter, die Messung findet aber nicht im Tasktype-Estimator statt. Damit der Tasktype-Estimator das Ergebnis dieser Messung erhält, wird während der Ausführung von edgedetection eine Textdatei geschrieben, die vom Estimator ausgelesen wird.

Die Listing A.20 im Anhang auf Seite 95 zeigt die Implementierung zur Messung der Anwendungstasks anhand der Methode `measureAppTask`. Die Methode bereitet zunächst die Parameter für ein Shellskript vor, damit ein bestimmter Task unter einem bestimmten Parallelitätsgrad ausgeführt werden wird. Der Aufruf findet in einem separaten Thread statt, während das Hauptprogramm auf die Beendigung dieses Threads wartet. Nachdem der Anwendungstask ausgeführt wurde, wird im Anwendungsordner die Textdatei mit dem Ergebnis ausgelesen. Die Daten werden in ein Objekt vom Typ `MeasureResult` geschrieben und an den estimator zurückgeliefert.

7.4.4 Vorbereitung des Experiments in edgedetection

Im vorangegangenen Kapitel wurde erklärt, dass die Messung der Leistungsaufnahme in edgedetection implementiert wurde. Weiterhin wurde im Kapitel 5.1.3 auf Seite 20 erklärt, dass eine Ausführung der Tasks auf mehreren CPU-Kernen erforderlich ist. Damit die Anwendungstasks auf mehreren Kernen ausgeführt werden, wurde die Anwendung edgedetection entsprechend angepasst.

7.4.4.1 Parametrisierung der Anwendungstasks

Bei der Messung der Leistungsaufnahme wird stets ein bestimmter Anwendungstask betrachtet. Dieser Anwendungstask soll ausgeführt und dessen Leistungsaufnahme gemessen werden. Damit dieser Anwendungstask bestimmt werden kann, wurde ein Parameter mit dem Namen des auszuführenden Anwendungstasks eingeführt. Ein weiterer Parameter definiert den Parallelitätsgrad. Das Listing A.21 im Anhang auf Seite 95 zeigt die geänderte Main-Methode aus edgedetection, sowie den Aufruf der Anwendungstasks.

Bisher war als Programmparameter eine Liste von Dateinamen mit Bildern erforderlich. Dieser Parameter ist nach wie vor zu übergeben, da für die Bildverarbeitungspipeline eine Bilddatei benötigt wird. Für das Experiment wird aus der entsprechenden Liste jedoch nur der erste Eintrag als Beispielbild verwendet. Dieses Beispielbild wird aus dem Dateisystem gelesen und in ein Objekt des Typs `MyIMG` abgebildet. Mit Aufruf der Methode `runEdgedetection` beginnt die eigentliche Ausführung des Tasks.

Die originäre Implementierung der Anwendungstasks wurden beibehalten. Damit eine Messung der Leistungsaufnahme, sowie eine parallele Ausführung ohne Änderungen der Anwendungs-

tasks möglich ist, wurden Funktionszeiger eingeführt. Diese Funktionszeiger verweisen auf die Implementierung der Tasks und übergeben die benötigten Parameter. Die Funktionszeiger werden wiederum von Methoden aufgerufen, die sowohl eine parallele Ausführung des Tasks, als auch eine Messung des Energieverbrauchs durchführen.

7.4.4.2 Messung der Leistungsaufnahme in edgedetection

Das Listing A.22 im Anhang auf Seite 97 zeigt den Aufruf eines Anwendungstasks mithilfe eines Funktionszeigers. Dieser Funktionszeiger verwendet als Parameter zwei Zeiger des Typs MyIMG und ist mit den Anwendungstasks greyscale, copyimage sowie combineimgs kompatibel, da diese Tasks die entsprechenden Parameter verwenden. Das Listing zeigt, dass vor Ausführung sowohl der Zeitpunkt, als auch der Energiezähler aufgenommen wird. Anschliessend findet der Aufruf des Tasks über den Funktionszeiger statt. Sobald der Task ausgeführt wurde, werden Zählerstand und Zeitstempel erneut ausgelesen und in eine Textdatei geschrieben.

7.4.4.3 Parallelisierung mit OMP

Damit der Energieverbrauch auch bei paralleler Ausführung des Anwendungstasks festgestellt werden kann, wurde die Verarbeitung mit OpenMP parallelisiert. OpenMP ist eine Spezifikation von Übersetzerdirektiven, Bibliotheksfunktionen und Umgebungsvariablen, die [...] einen einheitlichen Standard für die Programmierung von Parallelrechnern mit gemeinsamen Adressraum zur Verfügung stellt [21][s.S. 357].

Das Listing A.22 im Anhang auf Seite 97 zeigt wie OpenMP verwendet wurde. Der gewünschte Parallelitätsgrad wurde in einer for-Direktiven als Wert für den Parameter num_threads eingesetzt. Bei Ausführung werden nun entsprechend viele Threads auf unterschiedlichen CPU-Kernen erzeugt. Da die Anwendungstasks hinsichtlich ihrer Laufzeit sehr unterschiedlich sein können, wurde eine Laufzeit von fünf Sekunden festgelegt. Die Ausführung wird daher solange wiederholt, bis fünf Sekunden vergangen sind. Die Festlegung auf fünf Sekunden erleichtert außerdem die Vergleichbarkeit mit der Schätzung anhand der Task-Typen, da diese ebenfalls für einen Zeitraum von fünf Sekunden ausgeführt werden.

7.4.4.4 Parallelisierung mit PThread

Wie zuvor geschrieben, wurde zur parallelen Ausführung die Spezifikation OpenMP verwendet. Um einen noch präziseren Vergleich zu ermöglichen, wurde überprüft ob und wie eine parallele Ausführung im Benchmark epEBench implementiert wurde. Diese Überprüfung ist durch Sichtung des Quelltextes möglich. Aus diesem lässt sich ablesen, dass zur parallelen Ausführung pthreads verwendet wurde. Der Posix-Threads-Standard ist eine mögliche Realisierung des Threadmodells [21][s.S. 286]. Zur besseren Vergleichbarkeit wurde daher auch eine Implementierung nach dem

PThread-Standard umgesetzt und wird im Listing A.23 im Anhang auf Seite 98 gezeigt.

Mit der Änderung auf PThreads wurde das Ziel verfolgt, eine noch präzisere Bewertung der Leistungsaufnahme bei Ausführung von Tasks auf mehreren Kernen zu erreichen. Die Messungen der Anwendungstasks bei Ausführung mit PThreads haben jedoch keine erheblichen Verbesserungen gezeigt, weshalb weiterhin OMP verwendet wurde. Das Protokoll mit den Messwerten bei Ausführung mit PThreads befindet sich im Repository:

https://github.com/allankaufmann/pp8/blob/main/3_tasktype_estimator/results/202405111307_pthread.result

7.4.5 estimator.cpp

Die Quelldatei estimator.cpp enthält die Methoden zur Ausführung des eigentlichen Experiments. Die Abbildung A.1 auf Seite 77 zeigt das Startmenü des Tasktypeestimators. Für die Schätzung der Leistungsaufnahme stehen zwei Möglichkeiten zur Verfügung:

- Estimation - alle
- Estimation - TEST

Der Tasktypeestimator ist so konzipiert, dass die Leistungsaufnahme der CPU aller Anwendungstasks anhand von Task-Typen abgeschätzt wird. Demnach werden nacheinander, für jeden Anwendungstask, die entsprechenden Abbildungen aus der Taskmap verwendet, um Task-Typen auszuführen und die Leistungsaufnahme zu messen, um zu einer Schätzung der Leistungsaufnahme des Anwendungstasks zu gelangen. Diese Ausführung findet in allen spezifizierten Kombinationen aus CPU Taktfrequenz und Parallelitätsgraden statt. Um sicherzustellen, dass die Messungen nicht durch laufende Dienste des Betriebssystems verfälscht werden, werden die Messungen wiederholt, damit mögliche Schwankungen des Energieverbrauchs aufgedeckt werden können. Um die Schätzungen anschließend validieren zu können, werden außerdem die tatsächlichen Anwendungstasks ausgeführt und ebenfalls gemessen. Die Messergebnisse werden danach miteinander verglichen, um die Differenz zu überprüfen.

Entsprechend wird bei Auswahl des Menüpunkts ‘Estimation - alle’ die Leistungsaufnahme aller Anwendungstasks in allen Kombinationen abgeschätzt. Der zweite Menüpunkt ‘Estimation - TEST’ ermöglicht die Ausführung eines bestimmten Anwendungstasks.

Der Algorithmus 3 stellt die Implementierung vereinfacht dar. In der Methode `startEstimation` wird zunächst das simultane Multithreading deaktiviert. Daraufhin erfolgt eine Iteration durch die Liste der Anwendungstasks. Für die Messungen werden Wiederholungen über den Parameter `repeats` deklariert, wobei bei Ausführung des Experimentes fünf Wiederholungen konfiguriert waren.

Die Methode `repeatEstimationsForAppTask` iteriert nun durch die Taktfrequenzen und Parallelitätsgrade und startet pro Kombination die Schätzung der Leistungsaufnahme über

die Methode `startApptaskEstimation`. Für diese Schätzung sind mehrere Messungen der Leistungsaufnahme erforderlich. Zunächst werden die entsprechenden Task-Typen ausgeführt und deren Leistungsaufnahme gemessen. Danach wird der eigentliche Anwendungstask ausgeführt und dessen Leistungsaufnahme ebenfalls gemessen. Die Messergebnisse werden dann miteinander verglichen. Für die Ausführung und Messen der prototypischen Tasks wird mit Aufruf der Methode `estimateAppTask` ausgeführt.

Die Methode `estimateAppTask` zeigt außerdem, dass die Messung der Task-Typen nur einmalig pro Kombination aus Taktfrequenz und Parallelitätsgrad erfolgt. Die Messwerten werden in einer Map gespeichert. Wenn bereits ein Messwert hinterlegt wurde, dann wird dieser Messwert aus der Map entnommen.

In der vorangegangenen Phase wurde eine Abbildung der Anwendungstask auf Task-Typen umgesetzt. Dabei stehen zwei Abbildungsverfahren zur Verfügung, eine 1:1-Abbildung, sowie eine 1:N-Abbildung. Für die 1:1-Abbildung wird im Benchmark zunächst der Task-Typ ausgeführt, der diesem Anwendungstask am ähnlichsten ist. Als nächstes wird die 1:N-Abbildung des Anwendungstasks ausgeführt. Bei der jeweiligen Ausführung dieser Task-Typen wird die Leistungsaufnahme gemessen und in ein Ergebnisobjekt geschrieben. Dieses Ergebnisobjekt wird zur Bewertung ausgeliefert.

Die Messung des Anwendungstask wird in der Methode `measureAppTask` ausgeführt. Dabei werden die Skripte zum Aufruf der Beispielanwendung `edgedetection` ausgeführt und die Leistungsaufnahme aus dem Dateisystem gelesen.

Die Ergebnisse werden in eine Ergebnisdatei geschrieben und stehen zur Evaluierung des Vorgehens zur Verfügung.

Die genauere Implementierung ist im Listing A.24 im Anhang auf Seite 98 einzusehen.

Algorithm 3: Schätzung der Leistungsaufnahme anhand von Task-Typen

Input: experiment.config, taskmap.txt

Output: result.txt

```
1 Function main() :
2   if strcasecmp(parameter, U)==0 then
3     readConfigFile();
4     Call startEstimation(5);
5   else if strcasecmp(parameter, Z)==0 then
6     testEstimation();
7 Function startEstimation(repeats) :
8   smtoff();
9   for std::string apptaskname: apptypeVektor do
10    Call repeatEstimationsForAppTask(apptaskname, repeats)
11 Function repeatEstimationsForAppTask(apptaskname, repeats) :
12   for std::string cpuFreq : cpuFrequencyVektor do
13     setupCpuFrequenzlevel(cpuFreq);
14     for std::string cores : parallelismVektor do
15       setCurrentParallelism(cores);
16       for int i = 0; i < repeats; i++ do
17         Call startApptaskEstimation(apptaskname, i);
18 Function startApptaskEstimation(apptaskname, repeat) :
19   MeasureResult result = Call estimateAppTask(apptaskname, currentCPUFreq,
        currentParallelism, repeat);
20   MeasureResult appTaskresult = Call measureAppTask(apptaskname,
        currentCPUFreq, currentParallelism);
21   Result: result.txt
22 Function estimateAppTask(apptaskname, cpufreq, cores, repeat) :
23   ...
24   std::string oneToOneTaskname = readOneToOneMapping(apptaskname);
25   MeasureResult result;
26   if measureResultMap[oneToOneTaskname][cpufreq][cores].find(repeat) !=
        measureResultMap[oneToOneTaskname][cpufreq][cores].end() then
27     result = measureResultMap[oneToOneTaskname][cpufreq][cores][repeat];
28   else
29     result = runAndMeasureScript(filenameOneToOneWithParam.c_str());
30     measureResultMap[oneToOneTaskname][cpufreq][cores][repeat]=result;
31   MeasureResult resultOneToMany =
        runAndMeasureScript(filenameOneToManyWithParam.c_str());
32   return result;
33 Function measureAppTask(apptaskname, cpufreq, cores) :
34   MeasureResult result; std::thread t1(runCommand, fileWithParam.c_str());
35   t1.join();
36   return result;
```

7.5 Zusammenfassung der vierten Phase des Vorgehens

Mit dem Artefakt des Tasktypeestimators wurde ein Vorgehen zur Schätzung der Leistungsaufnahme der CPU anhand von Task-Typen implementiert. Das erstellte Artefakt wurde am 24.06.2024 auf dem Testsystem ausgeführt, um die Leistungsaufnahme der Anwendung edgedetection zu schätzen und mit der tatsächlichen Leistungsaufnahme zu vergleichen. Dabei wurden alle spezifizierten Kombinationen aus CPU-Taktfrequenz und Parallelitätsgraden ausgeführt und mit der tatsächlichen Leistungsaufnahme der Anwendungstasks verglichen. Die Ergebnisse sind in einer Ergebnisdatei gespeichert und wurden im Ergebnisteil, im Kapitel 8.4 auf Seite 58, aufbereitet und diskutiert.

Teil III

Evaluierung des Vorgehens

Kapitel 8

Ergebnisse und Analyse

8.1 Phase 1: Prototypische Tasks

Im Kapitel 4.4 auf Seite 18 wurde gezeigt, dass die prototypischen Tasks aus dem Benchmarkprogramm epEBench für die Schätzung der Leistungsaufnahme anhand von Task-Typen geeignet sind. Diese wurden daher als Grundlage für das Verfahren definiert. Um Fehler bei der Erstellung der Sequenzdateien zu vermeiden und damit das Verfahren auf anderen Testsystemen wiederholbar ist, wurde die Anwendung Tasktypeanalyzer umgesetzt. Mit dem Tasktypeanalyzer wurden die Sequenzdateien erstellt, die in das Experiment eingeflossen sind.

Die nachfolgende Tabelle 8.1 dient zur Übersicht der ermittelten Task-Typen. Die zweite Spalte zeigt den Umfang der Sequenzdateien des jeweiligen Task-Typen.

Tasktyp	Umfang der Sequenzdatei
nop	1999 Einträge
vconvert_SIMD	2090 Einträge
imov	1996 Einträge
imem	2080 Einträge
icompare	2095 Einträge
branch	1997 Einträge
bitbyte	2037 Einträge
v1x4smul_SIMD	2287 Einträge
dmem	2091 Einträge
dsub64_SIMD	2114 Einträge
dadd	2062 Einträge
m4x4smul_SIMD	2328 Einträge
dmul64_SIMD	2114 Einträge
dmul	2062 Einträge
m4x4smul	3199 Einträge
iadd	2039 Einträge
logic	2036 Einträge
imul	2071 Einträge
dmem64_SIMD	2081 Einträge
ssub32_SIMD	2114 Einträge
vmov_SIMD	2157 Einträge
smul32_SIMD	2114 Einträge
shift	1996 Einträge

Tabelle 8.1: Ergebnisse - Prototypische Tasks

8.2 Phase 2: Tasks einer Anwendung

Im Rahmen des agilen Design Researchs wurde als Beispielanwendung für das Verfahren die Anwendung edgedetection evaluiert und ausgewählt. Damit die Leistungsaufnahme der CPU anhand von Task-Typen geschätzt werden kann, wurden für die Tasks der Anwendung Sequenzdateien erstellt, um entsprechende Task-Typen zu identifizieren. Damit das Verfahren auch auf anderen Testsystemen angewandt werden kann, wurde die Anwendung Apptaskanalyzer umgesetzt.

Bei der Erstellung der Sequenzdateien für die Tasks von edgedetection hat sich gezeigt, dass die einige Tasks durch hohe Schleifenläufe sehr viele Instruktionen generieren und folglich eine Auswertung dieser Sequenzdateien erschweren würden. Um dies zu behandeln wurden die Schleifenläufe angemessen beschränkt, damit Sequenzdateien in auswertbarem Umfang gewonnen wurde. Diese Beschränkung wurde jedoch nur für die Sequenzerstellung vorgenommen, nicht bei der Messung des Energiebedarfs.

Mit dem Apptaskanalyzer wurden die entsprechenden Sequenzdateien erstellt, die in das Experiment eingeflossen sind.

Die nachfolgende Tabelle 8.2 dient zur Übersicht der ermittelten Anwendungstasks. Die zweite Spalte zeigt den Umfang der Sequenzdateien des jeweiligen Anwendungstasks.

Anwendungstask	Umfang der Sequenzdatei
sobelv	17.432 Einträge
checkcontrast	1.922 Einträge
copyimage	3.590 Einträge
combineimgs	8.295 Einträge
sharpencontrast	17.599 Einträge
loadimage	4.951 Einträge
sobelh	17.432 Einträge
writeimage	5.340 Einträge
greyscale	18.296 Einträge

Tabelle 8.2: Ergebnisse - Tasks einer Anwendung

8.3 Phase 3: Abbildung der Anwendung auf Task-Typen

Der Taskmapper ist ein Modul aus dem Artefakt des Tasktypeestimators und wurde umgesetzt, um die Tasks von edgedetection auf äquivalente Task-Typen abzubilden. Im Rahmen des agilen Design Researchs wurde neben einer 1:1-Abbildung auch eine 1:N-Abbildung umgesetzt. Die nachfolgende Tabelle 8.3 zeigt das Ergebnis dieser Abbildung. Demnach wurden alle neun Anwendungstasks aus edgedetection einerseits einem Task-Typen zugeordnet, andererseits wurde auch eine Zuordnung zu einem Modell aus mehreren Task-Typen ermittelt. Der Taskmapper kann, mit überschaubarem Aufwand, für andere Beispielanwendungen oder mit anderen Abbildungsvorschriften angepasst werden.

Anwendungstask	1:1 Abbildung	1:N Abbildung
sobelv	m4x4smul	branch=0.10 dmem=0.10 m4x4smul=0.50 vmov_SIMD=0.30
checkcontrast	m4x4smul_SIMD	dmem64_SIMD=0.50 shift=0.50
copyimage	m4x4smul	imem=0.33 m4x4smul=0.33 shift=0.33
combineimgs	imul	branch=0.40 m4x4smul=0.40 vmov_SIMD=0.20
sharpencontrast	vmov_SIMD	bitbyte=0.10 imul=0.10 m4x4smul=0.20 vmov_SIMD=0.60
loadimage	dmem64_SIMD	imem=0.25 imov=0.25 logic=0.25 shift=0.25
sobelh	m4x4smul	branch=0.10 dmem=0.10 m4x4smul=0.50 vmov_SIMD=0.30
writeimage	dmem	imem=0.25 imov=0.25 logic=0.25 shift=0.25
greyscale	m4x4smul	dadd=0.20 dmem=0.10 m4x4smul=0.60 vmov_SIMD=0.10

Tabelle 8.3: Abbildung der Anwendungstask auf Task-Typen

Die Tabelle wurde auf Grundlage der Ergebnisdatei taskmap.txt erstellt. Das Listing B.3 im Anhang auf Seite 102 zeigt den Inhalt der Datei taskmap.txt. Während der Ausführung des Taskmapers wurden außerdem Logdateien erstellt, die im Quellcoderepository eingesehen werden können:

Art	URL zur Protokolldatei
1:N	https://github.com/allankaufmann/pp8/blob/main/3_tasktype_estimator/logs/taskmapper/202404101919.log
1:1	https://github.com/allankaufmann/pp8/blob/main/3_tasktype_estimator/logs/taskmapper/202404110959.log

Tabelle 8.4: Logdateien für Taskmapper

8.4 Phase 4: Schätzung der Leistungsaufnahme

In den nachfolgenden Kapiteln werden die Ergebnisse der einzelnen Anwendungstasks tabellarisch präsentiert. Die Messungen wurden für jede Kombination fünfmal wiederholt, um sicherzustellen, dass es sich um konstante Messergebnisse handelt. Für die Ergebnispräsentation wurden die Messergebnisse daher aggregiert und aus den einzelnen Werten wurde der Median gebildet. Dabei wurden die Leistungsaufnahme der Task-Typen aber nur einmalig pro Kombination ermittelt. Wenn mehrere Anwendungstasks auf den selben Task-Typen abgebildet wurden, dann wurde dieser nicht erneut gemessen. Die Ergebnistabelle wird beispielhaft am ersten Anwendungstask erläutert.

Die einzelnen Messergebnisse wurden protokolliert und sind im Repository in einer Ergebnisdatei gesichert. Die Aufbereitung des Ergebnistabelle wurde ebenfalls im Repository gesichert. Zusätzlich wurden die Messwerte in einer Protokolldatei dokumentiert. Die Links für Protokolldateien sind in der Tabelle 8.5 hinterlegt. Die Ergebnisdatei ist außerdem im Kapitel B.4 im Anhang auf Seite 103 einzusehen.

Art	URL zur Protokolldatei
Aufbereitung der Ergebnisdatei	https://github.com/allankaufmann/pp8/blob/main/3_tasktype_estimator/results/Aufbereitung/Ergebnisaufbereitung.xlsx
Ergebnisdatei vom 24.06.2024	https://github.com/allankaufmann/pp8/blob/main/3_tasktype_estimator/results/202406242109.result
Protokoll der Messwerte vom 24.06.2024	https://github.com/allankaufmann/pp8/blob/main/3_tasktype_estimator/logs/measure/20240624210952.log

Tabelle 8.5: Logdateien für Tasktypeestimator

8.4.1 greyscale

Die Tabelle 8.6 zeigt die Ergebnisse zum Anwendungstask greyscale.

Die Tabelle ist nach Takt und Parallelitätsgrad gruppiert. Eine Zeile entspricht daher einer Kombination aus Taktfrequenz in Megahertz und der Anzahl der verwendeten CPU-Kerne. Der Anwendungstask wurde also bei dieser Kombination ausgeführt und sowohl nach dem 1:1-Verfahren, als auch nach dem 1:N-Verfahren fünf mal abgeschätzt. Die einzelnen Messungen wurden aggregiert und die Ergebnisse werden ab der vierten Spalte aufgeführt.

Die Spalte 'IST- P_{mW} ' zeigt die Leistungsaufnahme der CPU bei Ausführung des Anwendungstasks greyscale in Mikrojoule pro Mikrosekunde (mW).

Die fünfte und sechste Spalte gehört zur Schätzung der Leistungsaufnahme, bei einer 1:1-Abbildung. Der Anwendungstask greyscale wurde auf den Task-Typen m4x4smul abgebildet. Dieser Tasktyp wurde daher bei gleicher Kombination fünf Sekunden lang ausgeführt und die Leistungsaufnahme wurde protokolliert. Die Spalte 'Diff' zeigt die prozentuale Abweichung bei dieser 1:1-Abbildung im Vergleich zur tatsächlichen Leistungsaufnahme.

Die sechste und siebte Spalte gehört zur Schätzung der Leistungsaufnahme bei einer 1:N-Abbildung. Der Anwendungstask greyscale wurde auf ein Modell aus vier Task-Typen abgebildet. Dieses Modell wurde bei gleicher Kombination fünf Sekunden lang ausgeführt und die Leistungsaufnahme wurde protokolliert. Die Spalte 'Diff' zeigt die prozentuale Abweichung dieser 1:N-Abbildung im Vergleich zur tatsächlichen Leistungsaufnahme.

Task	Takt (Mhz)	Cores	IST- P_{mW}	1:1 Abbildung		1:N Abbildung	
				P_{mW}	Diff	P_{mW}	Diff
greyscale	1.500	1	2.330	2.572	9,7%	2.553	7,6 %
greyscale	1.500	2	3.679	4.055	10,2%	3.865	5,4 %
greyscale	1.500	3	4.902	5.455	11,5%	5.213	6,5 %
greyscale	1.500	4	6.168	6.848	11,2%	6.505	6,1 %
greyscale	2.000	1	3.243	3.480	7,1%	3.370	3,8 %
greyscale	2.000	2	5.087	5.611	10,1%	5.319	4,8 %
greyscale	2.000	3	6.944	7.799	12,5%	7.337	5,7 %
greyscale	2.000	4	8.865	10.023	12,9%	9.355	5,3 %
greyscale	2.200	1	3.768	4.017	7,3%	3.947	4,7 %
greyscale	2.200	2	5.931	6.686	12,7%	6.308	6,4 %
greyscale	2.200	3	8.273	9.356	13,1%	8.946	8,1 %
greyscale	2.200	4	10.507	11.902	13,3%	10.853	3,3 %
greyscale	2.400	1	4.343	4.772	10,7%	4.569	5,6 %
greyscale	2.400	2	7.057	7.834	11,0%	7.396	5,0 %
greyscale	2.400	3	9.876	11.235	13,3%	10.540	7,5 %
greyscale	2.400	4	12.510	14.063	12,4%	13.180	5,5 %
greyscale	2.500	1	4.648	5.099	9,7%	4.879	4,9 %
greyscale	2.500	2	7.630	8.603	12,4%	8.253	8,2 %
greyscale	2.500	3	10.551	12.093	14,6%	11.427	8,1 %
greyscale	2.500	4	13.587	15.365	12,8%	14.356	5,8 %
greyscale	3.000	1	6.717	7.416	10,7%	6.978	3,4 %
greyscale	3.000	2	11.171	12.568	12,5%	11.620	4,1 %
greyscale	3.000	3	15.869	17.819	12,3%	16.718	5,3 %
greyscale	3.000	4	20.138	23.501	16,6%	21.584	7,1 %
greyscale	3.500	1	9.497	10.607	11,4%	9.848	3,7 %
greyscale	3.500	2	15.783	18.148	12,5%	16.853	5,2 %
greyscale	3.500	3	22.735	26.240	15,4%	24.214	6,5 %
greyscale	3.500	4	25.673	24.912	-2,8%	24.917	-2,8 %
greyscale	4.000	1	13.619	14.996	10,1%	13.894	2,4 %
greyscale	4.000	2	22.086	25.625	15,4%	23.922	8,3 %
greyscale	4.000	3	25.889	24.904	-3,6%	24.843	-4,0 %
greyscale	4.000	4	25.757	24.978	-3,0%	24.855	-3,3 %

Tabelle 8.6: Ergebnisse - Task greyscale

8.4.2 sharpencontrast

Task	Takt (Mhz)	Cores	IST- P_{mw}	1:1 Abbildung		1:N Abbildung	
				P_{mw}	Diff	P_{mw}	Diff
sharpencontrast	1.500	1	2.441	1.946	-19,6%	2.319	-4,1 %
sharpencontrast	1.500	2	3.755	2.838	-24,6%	3.501	-6,6 %
sharpencontrast	1.500	3	5.061	3.688	-27,2%	4.670	-7,7 %
sharpencontrast	1.500	4	6.410	4.568	-28,7%	5.798	-9,4 %
sharpencontrast	2.000	1	3.294	2.630	-20,2%	3.074	-7,0 %
sharpencontrast	2.000	2	5.196	3.831	-26,2%	4.786	-7,9 %
sharpencontrast	2.000	3	7.145	5.097	-28,7%	6.541	-8,5 %
sharpencontrast	2.000	4	9.026	6.457	-28,5%	8.275	-8,2 %
sharpencontrast	2.200	1	3.863	2.945	-22,1%	3.574	-6,5 %
sharpencontrast	2.200	2	6.191	4.451	-28,0%	5.651	-8,7 %
sharpencontrast	2.200	3	8.561	6.039	-29,2%	7.798	-8,7 %
sharpencontrast	2.200	4	10.871	7.657	-29,6%	9.897	-8,8 %
sharpencontrast	2.400	1	4.471	3.471	-21,9%	4.190	-6,1 %
sharpencontrast	2.400	2	7.303	5.291	-27,4%	6.635	-8,8 %
sharpencontrast	2.400	3	10.255	7.280	-29,1%	9.255	-9,2 %
sharpencontrast	2.400	4	12.971	9.252	-28,5%	11.143	-14,3 %
sharpencontrast	2.500	1	4.764	3.746	-21,4%	4.492	-5,9 %
sharpencontrast	2.500	2	7.872	5.616	-28,5%	7.196	-8,6 %
sharpencontrast	2.500	3	11.018	7.820	-29,6%	10.034	-8,4 %
sharpencontrast	2.500	4	14.152	9.993	-29,5%	12.675	-10,5 %
sharpencontrast	3.000	1	6.928	5.287	-23,3%	6.401	-7,6 %
sharpencontrast	3.000	2	11.584	8.249	-29,3%	10.578	-9,0 %
sharpencontrast	3.000	3	16.485	11.533	-29,5%	14.667	-10,7 %
sharpencontrast	3.000	4	20.784	14.504	-29,9%	18.967	-8,5 %
sharpencontrast	3.500	1	9.880	7.453	-24,1%	9.035	-8,0 %
sharpencontrast	3.500	2	16.743	11.861	-29,4%	14.917	-11,3 %
sharpencontrast	3.500	3	23.651	16.780	-29,1%	21.550	-8,7 %
sharpencontrast	3.500	4	25.671	21.908	-14,7%	27.715	6,5 %
sharpencontrast	4.000	1	13.994	10.523	-25,8%	12.820	-9,2 %
sharpencontrast	4.000	2	23.469	16.651	-29,1%	21.242	-9,5 %
sharpencontrast	4.000	3	28.615	20.942	-26,8%	27.114	-4,4 %
sharpencontrast	4.000	4	25.696	24.721	-4,0%	25.066	-2,4 %

Tabelle 8.7: Ergebnisse - Task Sharpencontrast

8.4.3 combineimngs

Task	Takt (Mhz)	Cores	IST- P_{mW}	1:1 Abbildung		1:N Abbildung	
				P_{mW}	Diff	P_{mW}	Diff
combineimgs	1.500	1	2.527	2.126	-14,9%	2.510	0,5 %
combineimgs	1.500	2	3.774	2.930	-22,4%	3.875	2,9 %
combineimgs	1.500	3	5.117	3.891	-24,0%	5.193	1,5 %
combineimgs	1.500	4	6.414	4.847	-24,5%	6.469	0,5 %
combineimgs	2.000	1	3.289	2.647	-19,2%	3.359	2,1 %
combineimgs	2.000	2	5.342	4.061	-24,0%	5.310	-0,6 %
combineimgs	2.000	3	7.249	5.432	-25,1%	7.318	1,5 %
combineimgs	2.000	4	9.170	6.750	-26,2%	9.326	-0,1 %
combineimgs	2.200	1	3.827	3.161	-17,5%	3.907	1,7 %
combineimgs	2.200	2	6.218	4.751	-23,6%	6.290	1,5 %
combineimgs	2.200	3	8.661	6.429	-25,9%	8.723	0,4 %
combineimgs	2.200	4	11.066	8.238	-26,0%	11.209	1,3 %
combineimgs	2.400	1	4.565	3.577	-21,5%	4.612	0,9 %
combineimgs	2.400	2	7.375	5.527	-25,0%	7.418	0,5 %
combineimgs	2.400	3	10.386	7.640	-26,5%	10.484	1,1 %
combineimgs	2.400	4	13.139	9.583	-26,9%	13.051	-0,9 %
combineimgs	2.500	1	4.915	3.864	-21,4%	4.916	0,0 %
combineimgs	2.500	2	7.932	5.999	-24,4%	8.069	1,7 %
combineimgs	2.500	3	11.155	8.220	-26,1%	11.352	1,5 %
combineimgs	2.500	4	14.340	10.640	-25,8%	14.192	-0,9 %
combineimgs	3.000	1	6.949	5.572	-20,0%	7.058	1,7 %
combineimgs	3.000	2	11.774	8.756	-25,4%	11.802	0,3 %
combineimgs	3.000	3	16.673	12.294	-26,5%	16.453	-1,3 %
combineimgs	3.000	4	20.970	15.654	-26,0%	21.629	2,2 %
combineimgs	3.500	1	9.925	7.826	-21,5%	9.969	0,0 %
combineimgs	3.500	2	17.012	12.627	-26,8%	16.778	-1,9 %
combineimgs	3.500	3	24.032	17.821	-26,0%	24.506	3,2 %
combineimgs	3.500	4	25.798	23.108	-10,4%	27.038	3,8 %
combineimgs	4.000	1	14.258	11.004	-22,2%	14.236	-0,1 %
combineimgs	4.000	2	23.868	17.778	-25,5%	24.309	1,7 %
combineimgs	4.000	3	26.158	22.337	-14,6%	27.536	6,0 %
combineimgs	4.000	4	25.744	24.810	-3,8%	25.060	-3,0 %

Tabelle 8.8: Ergebnisse - Task combineimgs

8.4.4 copyimage

Task	Takt (Mhz)	Cores	IST- P_{mW}	1:1 Abbildung		1:N Abbildung	
				P_{mW}	Diff	P_{mW}	Diff
copyimage	1.500	1	2.400	2.572	7,1%	2.529	5,4 %
copyimage	1.500	2	3.663	4.055	10,9%	3.797	3,7 %
copyimage	1.500	3	4.926	5.455	11,2%	5.116	4,2 %
copyimage	1.500	4	5.984	6.848	14,6%	6.412	7,3 %
copyimage	2.000	1	3.147	3.480	10,5%	3.337	5,9 %
copyimage	2.000	2	4.922	5.611	13,8%	5.263	6,9 %
copyimage	2.000	3	6.839	7.799	14,0%	7.230	5,8 %
copyimage	2.000	4	8.503	10.023	17,9%	9.181	8,4 %
copyimage	2.200	1	3.703	4.017	8,5%	3.891	5,1 %
copyimage	2.200	2	5.849	6.686	13,9%	6.230	6,7 %
copyimage	2.200	3	8.058	9.356	16,2%	8.654	7,3 %
copyimage	2.200	4	10.275	11.902	17,1%	11.277	9,1 %
copyimage	2.400	1	4.291	4.772	11,7%	4.582	6,8 %
copyimage	2.400	2	6.943	7.834	13,4%	7.307	5,4 %
copyimage	2.400	3	9.650	11.235	16,8%	10.313	7,0 %
copyimage	2.400	4	11.871	14.063	19,0%	12.927	8,9 %
copyimage	2.500	1	4.594	5.099	11,0%	4.887	6,4 %
copyimage	2.500	2	7.547	8.603	14,0%	7.968	5,5 %
copyimage	2.500	3	10.429	12.093	16,0%	11.202	6,8 %
copyimage	2.500	4	13.005	15.365	17,9%	14.111	8,4 %
copyimage	3.000	1	6.597	7.416	12,9%	6.979	5,8 %
copyimage	3.000	2	10.938	12.568	14,6%	11.616	5,6 %
copyimage	3.000	3	15.170	17.819	17,1%	16.373	8,0 %
copyimage	3.000	4	19.548	23.501	20,4%	21.320	9,0 %
copyimage	3.500	1	9.437	10.607	12,4%	9.998	5,9 %
copyimage	3.500	2	15.766	18.148	15,1%	16.664	6,3 %
copyimage	3.500	3	22.467	26.240	16,7%	24.413	8,2 %
copyimage	3.500	4	25.727	24.912	-2,9%	25.880	0,6 %
copyimage	4.000	1	13.402	14.996	12,9%	14.150	6,9 %
copyimage	4.000	2	22.353	25.625	14,2%	23.791	6,3 %
copyimage	4.000	3	26.056	24.904	-3,6%	25.850	0,5 %
copyimage	4.000	4	25.702	24.978	-2,9%	25.614	-0,3 %

Tabelle 8.9: Ergebnisse - Task copyimage

8.4.5 sobelh

Task	Takt (Mhz)	Cores	IST- P_{mw}	1:1 Abbildung		1:N Abbildung	
				P_{mw}	Diff	P_{mw}	Diff
sobelh	1.500	1	2.434	2.572	6,2%	2.642	6,3 %
sobelh	1.500	2	3.151	4.055	28,7%	3.941	25,1 %
sobelh	1.500	3	4.132	5.455	32,0%	5.263	27,9 %
sobelh	1.500	4	5.278	6.848	30,7%	6.695	26,9 %
sobelh	2.000	1	3.251	3.480	7,0%	3.413	4,8 %
sobelh	2.000	2	4.425	5.611	26,8%	5.347	20,9 %
sobelh	2.000	3	5.942	7.799	31,3%	7.524	24,2 %
sobelh	2.000	4	7.439	10.023	33,6%	9.600	29,2 %
sobelh	2.200	1	3.762	4.017	6,8%	3.986	5,9 %
sobelh	2.200	2	5.177	6.686	28,6%	6.352	22,1 %
sobelh	2.200	3	7.037	9.356	33,4%	8.948	28,0 %
sobelh	2.200	4	8.927	11.902	35,3%	11.526	28,2 %
sobelh	2.400	1	4.382	4.772	7,2%	4.664	5,6 %
sobelh	2.400	2	6.137	7.834	27,6%	7.544	22,9 %
sobelh	2.400	3	8.408	11.235	34,3%	10.533	25,3 %
sobelh	2.400	4	11.060	14.063	27,9%	13.441	22,3 %
sobelh	2.500	1	4.764	5.099	8,5%	4.986	5,3 %
sobelh	2.500	2	6.619	8.603	28,8%	8.133	21,9 %
sobelh	2.500	3	9.351	12.093	29,3%	11.582	21,2 %
sobelh	2.500	4	11.424	15.365	34,6%	14.572	27,2 %
sobelh	3.000	1	6.783	7.416	10,1%	7.019	4,1 %
sobelh	3.000	2	9.587	12.568	31,1%	12.048	24,9 %
sobelh	3.000	3	13.320	17.819	33,4%	16.725	25,2 %
sobelh	3.000	4	17.460	23.501	34,6%	21.479	22,5 %
sobelh	3.500	1	9.572	10.607	11,1%	9.924	3,6 %
sobelh	3.500	2	14.888	18.148	21,7%	17.055	14,6 %
sobelh	3.500	3	19.619	26.240	32,3%	24.350	23,0 %
sobelh	3.500	4	24.955	24.912	-,2%	27.833	11,7 %
sobelh	4.000	1	13.402	14.996	12,1%	14.070	4,9 %
sobelh	4.000	2	19.437	25.625	31,7%	24.044	23,4 %
sobelh	4.000	3	23.910	24.904	4,8%	28.953	20,7 %
sobelh	4.000	4	25.546	24.978	-2,1%	25.056	-1,9 %

Tabelle 8.10: Ergebnisse - Task sobelh

8.4.6 sobelv

Task	Takt (Mhz)	Cores	IST- P_{mW}	1:1 Abbildung		1:N Abbildung	
				P_{mW}	Diff	P_{mW}	Diff
sobelv	1.500	1	2.430	2.572	5,7%	2.500	5,2 %
sobelv	1.500	2	3.124	4.055	30,4%	3.896	24,6 %
sobelv	1.500	3	4.152	5.455	31,7%	5.273	27,0 %
sobelv	1.500	4	5.175	6.848	32,6%	6.688	30,8 %
sobelv	2.000	1	3.228	3.480	8,0%	3.414	5,7 %
sobelv	2.000	2	4.339	5.611	29,3%	5.386	24,2 %
sobelv	2.000	3	5.839	7.799	33,3%	7.639	26,7 %
sobelv	2.000	4	7.270	10.023	36,8%	9.649	31,7 %
sobelv	2.200	1	3.775	4.017	7,5%	3.982	4,8 %
sobelv	2.200	2	5.160	6.686	27,7%	6.381	23,9 %
sobelv	2.200	3	6.973	9.356	33,3%	8.882	26,2 %
sobelv	2.200	4	8.768	11.902	35,3%	11.513	31,3 %
sobelv	2.400	1	4.342	4.772	9,5%	4.634	6,5 %
sobelv	2.400	2	6.002	7.834	29,2%	7.422	23,7 %
sobelv	2.400	3	8.685	11.235	31,0%	10.711	23,3 %
sobelv	2.400	4	10.389	14.063	35,9%	13.517	30,3 %
sobelv	2.500	1	4.642	5.099	10,0%	4.965	6,6 %
sobelv	2.500	2	6.489	8.603	32,6%	8.056	24,1 %
sobelv	2.500	3	8.868	12.093	36,2%	11.429	27,4 %
sobelv	2.500	4	11.246	15.365	36,5%	14.595	29,4 %
sobelv	3.000	1	6.670	7.416	11,6%	7.055	5,6 %
sobelv	3.000	2	9.543	12.568	32,2%	11.947	23,7 %
sobelv	3.000	3	14.066	17.819	27,2%	16.705	19,1 %
sobelv	3.000	4	16.433	23.501	42,4%	21.600	32,1 %
sobelv	3.500	1	9.565	10.607	10,6%	9.836	3,5 %
sobelv	3.500	2	13.720	18.148	31,4%	16.929	22,5 %
sobelv	3.500	3	18.780	26.240	40,1%	24.523	30,6 %
sobelv	3.500	4	24.547	24.912	1,5%	30.511	24,2 %
sobelv	4.000	1	13.266	14.996	13,9%	13.785	4,1 %
sobelv	4.000	2	19.079	25.625	34,1%	24.003	25,4 %
sobelv	4.000	3	23.494	24.904	6,8%	29.334	26,0 %
sobelv	4.000	4	25.520	24.978	-2,1%	25.046	-1,7 %

Tabelle 8.11: Ergebnisse - Task sobelv

8.4.7 writeimage

Task	Takt (Mhz)	Cores	IST- P_{mW}	1:1 Abbildung		1:N Abbildung	
				P_{mW}	Diff	P_{mW}	Diff
writeimage	1.500	1	4.407	2.552	-41,6%	1.965	-55,4 %
writeimage	1.500	2	4.899	3.959	-20,0%	2.643	-46,2 %
writeimage	1.500	3	5.252	5.158	-1,9%	3.429	-34,5 %
writeimage	1.500	4	5.385	6.120	13,0%	4.219	-22,4 %
writeimage	2.000	1	5.174	3.418	-34,3%	2.488	-51,9 %
writeimage	2.000	2	6.148	5.187	-15,8%	3.569	-42,0 %
writeimage	2.000	3	6.677	6.958	3,9%	4.714	-29,8 %
writeimage	2.000	4	6.818	8.440	23,8%	5.869	-14,8 %
writeimage	2.200	1	5.589	3.551	-36,4%	2.823	-49,6 %
writeimage	2.200	2	7.059	5.854	-18,5%	4.193	-40,9 %
writeimage	2.200	3	7.657	7.996	4,4%	5.605	-26,4 %
writeimage	2.200	4	7.652	9.831	28,5%	6.980	-8,4 %
writeimage	2.400	1	6.169	3.632	-41,1%	3.312	-46,4 %
writeimage	2.400	2	8.004	6.493	-18,2%	4.963	-37,9 %
writeimage	2.400	3	8.561	9.430	9,9%	6.598	-22,9 %
writeimage	2.400	4	8.882	11.972	35,8%	8.448	-5,0 %
writeimage	2.500	1	6.496	3.697	-42,1%	3.602	-44,5 %
writeimage	2.500	2	8.467	6.305	-25,4%	5.253	-38,1 %
writeimage	2.500	3	9.110	9.932	9,0%	7.220	-20,6 %
writeimage	2.500	4	8.860	12.667	42,8%	8.948	-2,3 %
writeimage	3.000	1	7.884	3.647	-53,6%	4.929	-37,4 %
writeimage	3.000	2	11.110	6.652	-40,1%	7.602	-31,6 %
writeimage	3.000	3	12.363	11.957	-1,2%	10.529	-14,3 %
writeimage	3.000	4	12.681	17.592	37,8%	13.134	3,0 %
writeimage	3.500	1	8.953	3.698	-58,1%	6.808	-23,9 %
writeimage	3.500	2	14.476	6.715	-53,3%	10.767	-25,6 %
writeimage	3.500	3	17.046	12.514	-28,0%	14.761	-14,5 %
writeimage	3.500	4	11.560	16.010	23,8%	17.901	53,4 %
writeimage	4.000	1	10.576	3.716	-65,0%	9.400	-12,4 %
writeimage	4.000	2	17.062	6.506	-59,8%	14.785	-13,3 %
writeimage	4.000	3	20.090	12.764	-35,2%	18.139	-10,0 %
writeimage	4.000	4	16.242	16.745	8,1%	20.702	44,3 %

Tabelle 8.12: Ergebnisse - Task writeimage

8.4.8 checkcontrast

Task	Takt (Mhz)	Cores	IST- P_{mw}	1:1 Abbildung		1:N Abbildung	
				P_{mw}	Diff	P_{mw}	Diff
checkcontrast	1.500	1	2.686	12.925	395,4%	10.888	307,0 %
checkcontrast	1.500	2	2.701	21.550	701,0%	17.700	553,0 %
checkcontrast	1.500	3	2.741	25.813	851,4%	20.858	663,3 %
checkcontrast	1.500	4	2.940	33.231	1.044,5%	28.069	858,4 %
checkcontrast	2.000	1	3.663	3.402	-6,5%	2.729	-25,2 %
checkcontrast	2.000	2	3.559	4.767	33,5%	4.002	12,6 %
checkcontrast	2.000	3	3.651	6.440	77,2%	5.364	47,0 %
checkcontrast	2.000	4	3.654	8.099	121,9%	6.590	79,5 %
checkcontrast	2.200	1	3.958	3.918	-6,5%	3.084	-21,6 %
checkcontrast	2.200	2	4.138	5.862	41,7%	4.690	13,1 %
checkcontrast	2.200	3	4.215	7.876	87,4%	6.326	50,3 %
checkcontrast	2.200	4	4.222	9.632	128,2%	7.994	89,8 %
checkcontrast	2.400	1	4.861	4.270	-8,4%	3.562	-26,2 %
checkcontrast	2.400	2	4.865	6.579	35,1%	5.444	12,1 %
checkcontrast	2.400	3	4.859	9.003	85,4%	7.454	52,5 %
checkcontrast	2.400	4	4.857	11.248	131,6%	9.445	93,7 %
checkcontrast	2.500	1	4.939	4.764	-2,4%	3.841	-21,9 %
checkcontrast	2.500	2	5.179	7.366	42,3%	5.918	14,0 %
checkcontrast	2.500	3	5.203	10.030	92,6%	8.140	57,0 %
checkcontrast	2.500	4	5.290	12.390	134,2%	10.332	97,1 %
checkcontrast	3.000	1	7.395	6.777	-7,7%	5.442	-26,4 %
checkcontrast	3.000	2	7.428	10.394	39,9%	8.677	17,1 %
checkcontrast	3.000	3	7.424	14.309	92,6%	11.992	61,9 %
checkcontrast	3.000	4	7.510	18.511	148,3%	14.935	98,9 %
checkcontrast	3.500	1	10.361	8.846	-14,6%	7.550	-27,3 %
checkcontrast	3.500	2	10.411	14.404	38,7%	12.199	17,5 %
checkcontrast	3.500	3	10.627	20.978	101,8%	17.060	60,8 %
checkcontrast	3.500	4	10.890	27.404	151,6%	22.273	104,9 %
checkcontrast	4.000	1	15.077	12.447	-17,3%	10.690	-28,6 %
checkcontrast	4.000	2	15.166	20.245	35,8%	17.106	13,3 %
checkcontrast	4.000	3	15.149	26.616	75,7%	21.606	42,0 %
checkcontrast	4.000	4	15.450	32.848	113,0%	27.301	77,1 %

Tabelle 8.13: Ergebnisse - Task checkcontrast

8.4.9 loadimage

Task	Takt (Mhz)	Cores	IST- P_{mW}	1:1 Abbildung		1:N Abbildung	
				P_{mW}	Diff	P_{mW}	Diff
loadimage	1.500	1	4.261	2.153	-49,4%	1.953	-54,2 %
loadimage	1.500	2	5.856	3.097	-47,2%	2.650	-54,3 %
loadimage	1.500	3	6.930	4.101	-40,8%	3.396	-51,2 %
loadimage	1.500	4	6.798	5.132	-23,8%	4.186	-38,6 %
loadimage	2.000	1	5.574	2.846	-48,8%	2.461	-55,9 %
loadimage	2.000	2	7.676	4.285	-44,4%	3.568	-53,8 %
loadimage	2.000	3	9.643	5.733	-39,7%	4.727	-51,1 %
loadimage	2.000	4	10.027	7.197	-28,0%	5.887	-41,2 %
loadimage	2.200	1	6.402	3.336	-48,1%	2.808	-55,8 %
loadimage	2.200	2	8.942	5.074	-43,4%	4.163	-53,6 %
loadimage	2.200	3	11.526	6.831	-40,8%	5.508	-52,2 %
loadimage	2.200	4	11.498	8.676	-24,8%	6.675	-42,1 %
loadimage	2.400	1	7.276	3.800	-47,8%	3.334	-54,2 %
loadimage	2.400	2	10.384	5.944	-42,7%	4.937	-52,4 %
loadimage	2.400	3	13.266	8.147	-38,4%	6.620	-50,3 %
loadimage	2.400	4	12.879	10.146	-20,9%	8.267	-35,5 %
loadimage	2.500	1	7.736	4.048	-47,6%	3.587	-53,6 %
loadimage	2.500	2	11.209	6.334	-42,6%	5.287	-52,8 %
loadimage	2.500	3	14.349	8.790	-37,6%	7.143	-50,2 %
loadimage	2.500	4	13.543	11.094	-17,6%	9.046	-33,0 %
loadimage	3.000	1	9.954	5.850	-40,8%	4.968	-50,1 %
loadimage	3.000	2	15.813	9.391	-40,6%	7.666	-51,5 %
loadimage	3.000	3	19.926	12.930	-35,1%	10.647	-46,9 %
loadimage	3.000	4	16.984	16.316	-4,2%	13.034	-23,3 %
loadimage	3.500	1	10.138	8.153	-19,4%	6.808	-32,4 %
loadimage	3.500	2	21.850	13.419	-38,9%	10.949	-49,9 %
loadimage	3.500	3	29.495	19.156	-35,5%	15.053	-49,0 %
loadimage	3.500	4	19.067	24.461	27,7%	19.600	0,1 %
loadimage	4.000	1	10.231	11.453	12,2%	9.596	-5,6 %
loadimage	4.000	2	25.172	18.898	-24,7%	15.283	-38,8 %
loadimage	4.000	3	33.967	23.605	-29,5%	19.224	-44,2 %
loadimage	4.000	4	20.946	29.530	41,6%	24.535	14,4 %

Tabelle 8.14: Ergebnisse - Task loadimage

Kapitel 9

Diskussion und Interpretation

9.1 Diskussion

Die Tabellen zeigen dass die Leistungsaufnahme der CPU bei Ausführung eines Anwendungstasks mit höherer Taktfrequenz und/oder höherem Parallelitätsgrad steigt. Dies zeigt auch die Ausführung der entsprechenden Task-Typen, die Leistungsaufnahme der CPU steigt ebenfalls, wenn ein höherer Takt einstellt und/oder der Task auf mehreren Kernen ausgeführt wird. Die Ergebnistabellen zeigen zu beiden Abbildungsverfahren sowohl die gemessenen Werte, als auch die prozentualen Abweichungen im Vergleich mit der Leistungsaufnahme der Anwendungstasks.

Zur besseren Übersicht wurden prozentuale Abweichungen, die sich im Bereich von -20% bis +20% bewegen farblich hervorgehoben. Bei den meisten Messungen hat sich das 1:N-Verfahren als ein präziseres Verfahren gezeigt. Es ist zwar möglich, die Anwendungstasks auf einen prototypischen Task abzubilden, die Abweichungen sind jedoch in den meisten Fällen höher.

9.1.1 Abweichungen im niedrigen Prozentbereich

Bei den Tasks greyscale, sharpencontrast, combineimgs und copyimage bewegen sich die Abweichungen im 1:N-Verfahren im einstelligen oder niedrigen zweistelligen Prozentbereich, in jeder der verwendeten Kombinationen. Die Leistungsaufnahme der CPU ist bei diesen Tasks somit sehr verlässlich, anhand von Task-Typen, abschätzbar.

9.1.2 Abweichungen bei Ausführung auf mehreren Kernen

Bei den Tasks sobelh und sobelv wurden bei Ausführung auf einem CPU-Kern Abweichungen im einstelligen Prozentbereich festgestellt. Bei der Ausführung auf mehreren Kernen befinden sich die Abweichung jedoch im 30% Bereich. Die Leistungsaufnahme ist bei Ausführung der Anwendungstask auf mehreren Kernen tatsächlich geringer, als es abgeschätzt wurde. Dies könnte daran

liegen, dass sich die parallele Verarbeitung in der Anwendung effizienter ausführen lässt, als dies im Benchmark-Programm der Fall ist. Aus diesem Grund wurde im Kapitel 7.4.4.4 auf Seite 48 gezeigt, dass die parallele Verarbeitung von OMP auf PThread umgestellt wurde, um bei der Schätzung dieser Tasks einen präziseren Energiebedarf zu erreichen. Bei den Tasks `sobelh` und `sobelv` lässt sich demnach die Leistungsaufnahme der CPU verlässlich abschätzen, wenn die Tasks auf einem Kern ausgeführt werden, bei Ausführung auf mehreren Kernen ist mit Abweichungen im 30% Bereich zu rechnen.

9.1.3 Abweichungen bei Tasks mit geringerer Sequenzgröße

Bei den Tasks `writeimage`, `checkcontrast` und `loadimage` sind deutlich größere Abweichungen festzustellen. Dies liegt sehr wahrscheinlich daran, dass die Sequenzen dieser Tasks nicht sehr umfangreich sind und sich daher nicht so einfach auf die Task-Typen abbilden lassen. Laut der Ergebnistabelle 8.2 auf Seite 56 umfassen diese Tasks weniger Einträge, als die meisten anderen Tasks. So besteht der Anwendungstask `checkcontrast` aus 1.922 Instruktionen und entspricht hinsichtlich des Umfangs ungefähr einem prototypischen Task. Im Vergleich dazu, besteht der Task `copyimage` zwar ebenfalls nur aus 3.590 Instruktionen, dieser lässt sich jedoch sowohl im 1:1-Verfahren, als auch im 1:N-Verfahren präziser abbilden. Die drei genannten Tasks sind somit hinsichtlich ihrer Instruktionen nicht umfangreich genug, um anhand der vorhandenen Task-Typen eine verlässliche Schätzung durchzuführen. Wenn eine verlässliche Schätzung dieser drei Task-Typen erforderlich ist, dann müsste der Benchmark um entsprechende prototypische Tasks erweitert werden, die den Aufgaben von `writeimage`, `checkcontrast` und `loadimage` ähnlicher sind.

9.1.4 Reduzierung des Aufwands

Laut der Taskmap 8.3 auf Seite 57 wurden die Anwendungstasks `sobelv`, `copyimage`, `sobelh` und `greyscale` im 1:1-Abbildungsverfahren auf den Task-Typen `m4x4smul` abgebildet. Die Leistungsaufnahme der CPU wurde für diese vier Anwendungstasks daher nur einmalig, anhand des Task-Typen `m4x4smul` abgeschätzt und musste somit nicht viermal festgestellt werden. Die übrigen Anwendungstasks wurden auf unterschiedliche Task-Typen abgebildet, weshalb hier keine Reduzierungen der Messungen möglich waren. Im 1:N-Verfahren findet die Abbildung auf ein Modell aus mehreren Task-Typen statt. Bei diesem Verfahren sind äquivalente Modelle zwischen den Anwendungstasks nicht zu erwarten. Dennoch hat das Abbildungsergebnis gezeigt, dass die Anwendungstasks `loadimage` und `writeimage` aus den gleichen Kombinationen aus Task-Typen besteht und somit auch hier der Aufwand zur Feststellung der Leistungsaufnahme reduziert werden konnte.

Eine weitere Reduzierung des Aufwandes zur Erhebung der Leistungsaufnahme anhand von Task-Typen ergibt sich also dann, wenn eine Anwendung aus mehreren Tasks besteht, die sich auf dieselben Task-Typen abbilden lassen.

Kapitel 10

Fazit

10.1 Beantwortung der Forschungsfragen

10.1.1 Wie werden prototypische Tasks definiert?

Die erste Forschungsfrage wurde zuvor im Kapitel 8.1 beantwortet. Die prototypischen Tasks wurden aus dem Benchmarkprogramm epEBench abgeleitet.

10.1.2 Wie wird die Leistungsaufnahme dieser prototypischen Tasks ermittelt?

Die zweite Forschungsfrage wurde im Kapitel 7.1.4 auf Seite 41 beantwortet. Die Task-Typen werden ausgeführt, dabei werden Dauer und Energiezähler aufgenommen. Die Leistungsaufnahme berechnet sich auf Grundlage der Formel, die im Kapitel 2.1.2 auf Seite 5 hergeleitet wurde.

10.1.3 Wie werden Tasks einer konkreten Anwendung auf prototypischen Tasks abgebildet?

Die dritte Forschungsfrage wurde zuvor im Kapitel 8.3 beantwortet. Mit dem Taskmapper wurden zwei Abbildungsverfahren auf Grundlage einer Ähnlichkeitssuche implementiert, dabei hat sich die 1:N-Zuordnung in den meisten Schätzungen als präziser herausgestellt.

10.1.4 Wie wird die Leistungsaufnahme der CPU anhand von Task-Typen abgeschätzt?

Die vierte Forschungsfrage wurde im Kapitel 7.2 auf Seite 42 beantwortet. Die Task-Typen werden einmalig ausgeführt und die Leistungsaufnahme bestimmt. Mit diesem Wert erfolgt die Schätzung

der Leistungsaufnahme eines Anwendungstasks.

10.1.5 Wie weit weichen die Schätzungen von der tatsächlichen Leistungsaufnahme konkreter Anwendungstasks ab?

Die fünfte Forschungsfrage wird mit der vorangegangene Diskussion im Kapitel 9.1 beantwortet.

10.2 Zusammenfassung

Mit dem Tasktypeanalyzer wurde ein Artefakt implementiert, mit dem Kriterien für prototypische Tasks aus einem Benchmark-Programm gewonnen wurden.

Mit dem Apptaskanalyzer wurde ein Artefakt implementiert, mit dem Kriterien für die Tasks einer Beispielanwendung ermittelt wurden.

Der Taskmapper ist ein Modul aus dem Artefakt Tasktypeestimator, welches die Anwendungstasks auf Task-Typen abbildet hat.

Der Tasktypeestimator ist ein Artefakt, welches die Schätzung der Leistungsaufnahme der CPU anhand von Task-Typen durchführt hat.

Das Experiment hat gezeigt, dass die Schätzung der Leistungsaufnahme der CPU anhand von Task-Typen möglich ist und daraus verlässliche Informationen zu gewinnen sind. Die Leistungsaufnahme der CPU wird einmalig durch prototypische Tasks erhoben, was zu einer Reduzierung des Aufwandes zur Feststellung der Leistungsaufnahme führt, wenn mehrere Tasks einer Anwendung diese prototypischen Tasks nutzen.

Auf Grundlage eines Abbildungsverfahrens ist eine Schätzung der Leistungsaufnahme für die Tasks einer Anwendung möglich. Die Ergebnisse dieser Schätzung können in einem Modell für einen statischen Scheduler verwendet werden.

Dieses Verfahren ermöglicht folglich, dass Informationen zur Leistungsaufnahme der CPU nicht mehr nur durch Benchmarking der Anwendung erhoben werden müssten, sondern anhand von Task-Typen abzuschätzen sind. Dieses Verfahren sollte daher den Aufwand zur Erhebung der Leistungsaufnahme deutlich reduzieren können.

Literaturverzeichnis

- [1] Alan R. Hevner, Salvatore T. March, Jinsoo Park and Sudha Ram. *Design Science in Information Systems Research*. 2004.
- [2] Bähring, Gotthardt, Keller, Schiffmann, Ungerer. “Einführung in die technische und theoretische Informatik”. Diss.
- [3] E. Zimmermann. *Das Experiment in den Sozialwissenschaften*. ISBN: 978-3-519-00037-2.
- [4] Fowler, M., Highsmith, J.: *The agile manifesto. Software Development*. 2001. URL: <https://agilemanifesto.org/iso/de/manifesto.html>.
- [5] Simon Holmbacka. *GDBInstructionScanner*. 2017. URL: <https://registry.abo.fi/sholmbac/GDBInstructionScanner>.
- [6] Simon Holmbacka und Jörg Keller. “Workload Type-Aware Scheduling on big.LITTLE Platforms”. In: *Algorithms and Architectures* (2017).
- [7] Simon Holmbacka und Robert Müller. “epEBench: True Energy Benchmark”. In: *2017 25th Euromicro International Conference on Parallel, Distributed and Network-Based Processing* (2017).
- [8] Ilkka Hautala. *FROM DATAFLOW MODELS TO ENERGY EFFICIENT APPLICATION SPECIFIC PROCESSORS*. 2019.
- [9] Ilkka Hautala, Jani Boutellier, Senior Member, IEEE, and Olli Silvén, Member, IEEE. *TTADF: Power Efficient Dataflow-Based Multicore Co-Design Flow*. Hrsg. von IEEE TRANSACTIONS ON COMPUTERS VOL. XX, NO. XX AUGUST 2019. 2019.
- [10] Intel Corporation. *Intel 64 and IA-32 Architectures Software Developer’s Manual*. Hrsg. von Intel Corporation. URL: <https://software.intel.com/en-us/download/intel-64-and-ia-32-architectures-sdm-combined-volumes-1-2a-2b-2c-2d-3a-3b-3c-3d-and-4>.
- [11] Kashif Nizam Khan. *RAPL in Action : Experiences in Using RAPL for Power Measurements*. Hrsg. von University of Helsinki. URL: <https://helda.helsinki.fi/server/api/core/bitstreams/bdc6c9a5-74d4-494b-ae83-860625a665ce/content>.
- [12] Kieran Conboy, Rob Gleasure, Eoin Cullina. *Agile Design Science Research*. Hrsg. von -. URL: https://link.springer.com/chapter/10.1007/978-3-319-18714-3_11.

- [13] Klaus Gebhardt. *Leistungsaufnahme*. Hrsg. von Umweltdatenbank. URL: <https://www.umweltdatenbank.de/cms/lexikon/38-lexikon-1/2263-leistungsaufnahme.html>.
- [14] Sebastian Litzinger. *Raising Energy Efficiency and Fault Tolerance with Parallel Streaming Application Scheduling on Multicore Systems*. Hrsg. von FernUniversität in Hagen.
- [15] Sebastian Litzinger, Jörg Keller und Christoph Kessler. “Scheduling Moldable Parallel Streaming Tasks on Heterogeneous Platforms with Frequency Scaling”. In: (2019).
- [16] Marcus Hähnel. *Measuring Energy Consumption for Short Code Paths Using RAPL*. Hrsg. von Technische Universität Dresden. 2012.
- [17] Robert Mueller. 2017. URL: <https://github.com/RobertMueller/epEBench>.
- [18] Nicola Döring. *Forschungsmethoden und Evaluation: in den Sozial- und Humanwissenschaften*. 2023. ISBN: 9783662647615.
- [19] Gustav Pomberger und Heinz Dobler. *Algorithmen und Datenstrukturen*. 2008. ISBN: 978-3-8273-7268-0.
- [20] Thomas Rauber und Gudula Rünger. *Multicore: Parallele Programmierung*. 2008. ISBN: 978-3-540-73113-9.
- [21] Thomas Rauber und Gudula Rünger. *Parallele Programmierung*. ISBN: 978-3-642-13603-0.
- [22] Jörg Keller Sebastian Litzinger. *Code generation for energy-efficient execution of dynamic streaming task graphs on parallel and heterogeneous platforms*. Hrsg. von John Wiley & Sons Ltd. 2020.
- [23] Tom Stempel. *Measuring the Energy Consumption of Software written in C on x86-64 Processors*. Hrsg. von Leipzig University. URL: <https://ul.qucosa.de/api/qucosa%3A77194/attachment/ATT-0/>.
- [24] The Linux Foundation. *The Linux Kernel documentation: Powercap*. Hrsg. von The Linux Foundation. URL: <https://www.kernel.org/doc/html/next/power/powercap/powercap.html>.
- [25] Vince Weaver. *Reading RAPL energy measurements from Linux*. Hrsg. von VMW Research Group. URL: <https://web.eece.maine.edu/~vweaver/projects/rapl/>.

Anhang

Anhang A

Ergänzungen

A.1 Abbildungen

A.1.1 Menü des Tasktypeestimators

```

allan@Allan-W11:~/pp8/3_tasktype_estimator$ ./main
Bitte einen der folgenden Parameter eingeben:
----- Init -----
I (=Init - Vorbedingungen für Experiment prüfen!)
C (=Config - Konfigurationsdatei auslesen und Skripte für prototyp tasks erstellen)
----- Taskmapper-----
A (Abbildung - 1 AnwTask auf 1 Prototyp task - Test)
B (Abbildung - 1 AnwTask auf 1 Prototyp task - alle)
D (Abbildung - 1 AnwTask auf N Prototyp tasks - Test)
E (Abbildung - 1 AnwTask auf N Prototyp tasks - alle)
T (=Transfer to epebench)
----- Tasktype Estimator -----
R (Run - Leistungsaufnahme aller prototyp tasks messen)
S (Anw. Tasks messen)
U (Estimation - alle)
Z (Estimation - TEST)
----- Sonstiges -----
V (Alternative Abbildung - Diffsuche)
Y (Alternative Abbildung - Paarsuche)
Q (Alternative Abbildung - Drillingsuche)
W (CPU Frequency)
X (=Exit)

```

Abbildung A.1: Tasktypeestimator Menü

A.1.2 Auszug einer Sequenzdatei zum prototypischen Task bitbyte

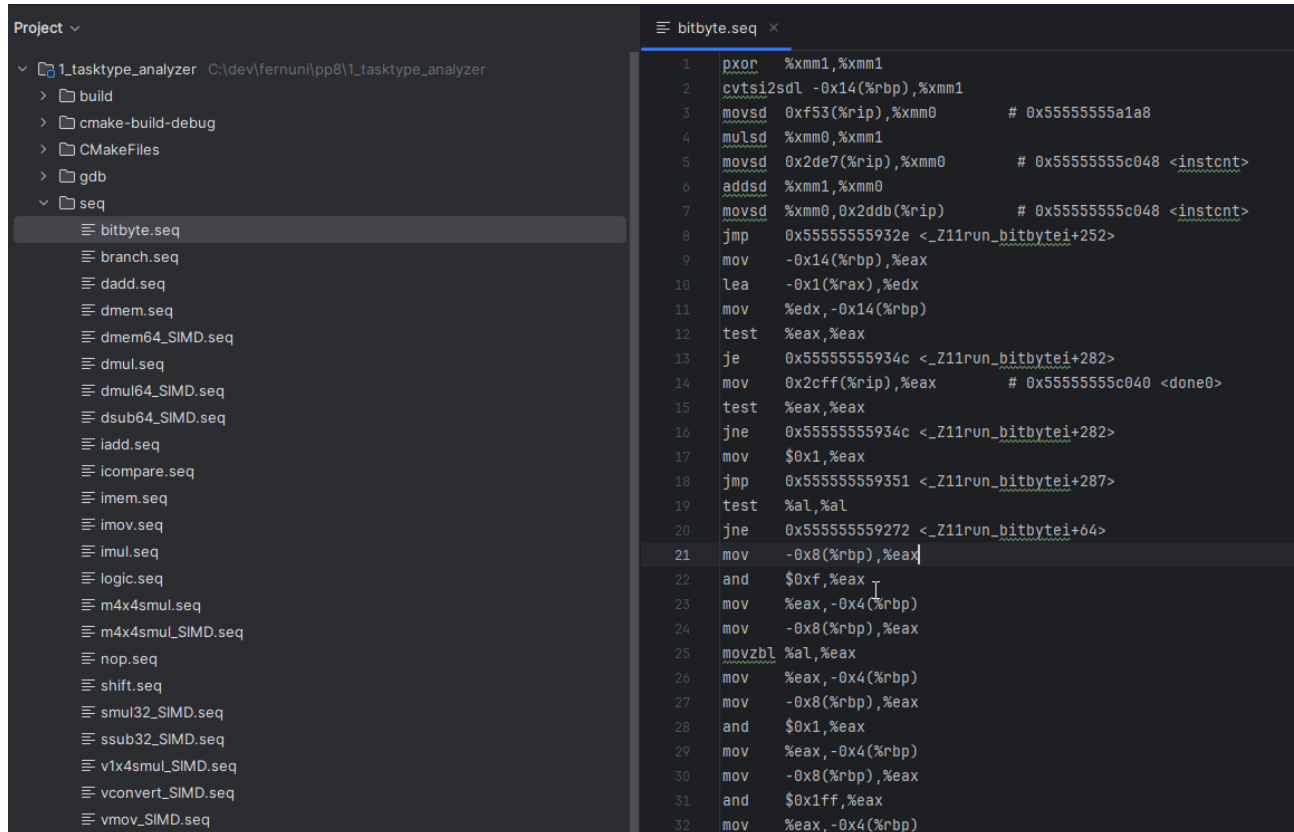
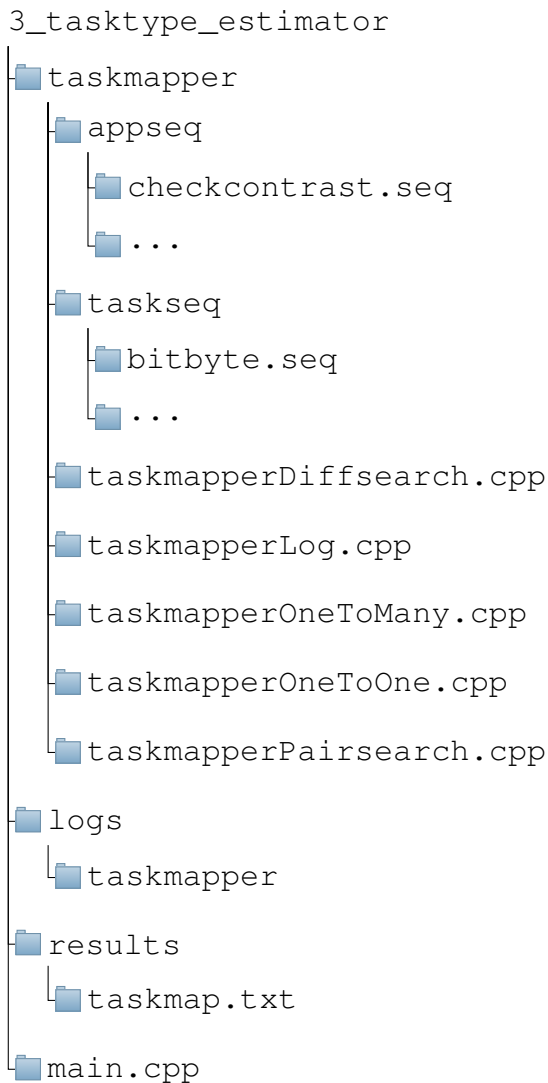


Abbildung A.2: Sequenzdateien der prototypischen Tasks

A.1.3 Dateistruktur des Taskmappers

In der Abbildung A.1.3 wird die Dateistruktur des Taskmappers dargestellt. Die Quellen befinden sich im entsprechenden Unterordner taskmapper. Die Ordner appseq und taskseq enthalten die zuvor erstellten Sequenzdateien der Tasks. Während die Anwendungstasks auf Task-Typen abgebildet werden, wird eine Protokolldatei geschrieben, die im Ordner logs/taskmapper gespeichert wird. Das Ergebnis einer Abbildung wird in der Datei taskmap.txt im Ordner results gespeichert.



A.1.4 Dateistruktur des Tasktypeestimators

```
3_tasktype_estimator
├── estimation
│   ├── estimator.cpp
│   └── measure.cpp
├── gen
│   ├── apptasks
│   │   ├── checkcontrast.sh
│   │   └── ...
│   ├── tasktypes
│   │   ├── onetomany
│   │   │   ├── checkcontrast.sh
│   │   │   └── ...
│   │   └── ...
├── logs
│   └── measure
├── results
│   └── result.txt
├── config.cpp
├── main.cpp
└── experiment.config
```

Abbildung A.3: Dateistruktur für Tasktypeestimator

A.2 Listings

A.2.1 Prototypische Tasks aus epEBench

```
1 // SIMD
2 void *run_m4x4smul_SIMD(int);
3 void* run_v1x4smul_SIMD(int);
4 void* run_dm4x4smul_SIMD(int);
5 void* run_smul32_SIMD(int);
6 void* run_ssub32_SIMD(int);
7 void* run_dsub64_SIMD(int);
8 void* run_dmem64_SIMD(int);
9 void* run_smem32_SIMD(int);
10 void* run_vmov_SIMD(int);
11 void* run_vconvert_SIMD(int);
12
13 // ALU / FP
14 void *run_m4x4smul(int);
15 void *run_dm4x4smul(int);
16 void *run_dadd(int);
17 void *run_iadd(int);
18 void *run_imul(int);
19 void *run_icompare(int);
20 void *run_logic(int);
21 void *run_branch(int);
22 void *run_imem(int);
23 void *run_dmem(int);
24 void *run_imov(int);
25 void *run_shift(int);
26 void *run_bitbyte(int);
27 void *run_nop(int);
```

Listing A.1: Prototypische Tasks aus epEBench ebloads.h[17]

A.2.2 Task-Typ dadd im Benchmark epEBench

```
1 ./epbench -m dadd -t 1 -a 1 -n 1
2
3 *****
4 * epEBench - Energy Benchmark *
5 * Version 1.1.2 *
6 * build Dec 2 2023 *
7 * Robert Mueller *
8 * 2016 *
9 *****
10 > !Too less allocated CPU cores to reach 30% CPU load!
11
12 Initial settings:
13   CPU load: 12%
14   cores# : 8
15   threads# : 1
16   model : dadd
17   runtime : 1.0s
18
19 Returned pthread affinity and core load:
20   used cores: 1
21   CPU 0: 100.0%
22
23
24 [1Acpu usage [12.72% / 12.72%] max [ 0.00%] min [ 0.00%]
25 Run: 1010.96 ms
26 Counts: 1326.447140 Mio
27
28 Benchmarking finished.
```

Listing A.2: Ausführung des Tasks dadd

Das Ergebnis dieser Ausführung ist aus der Logdatei zu entnehmen:

```
1 # Testrun settings:
2 # Model : dadd
3 # Cpu# : 8
4 # Thread# : 1
5 time usage avg_usage
6 0 0.1195 0.1272
```

Listing A.3: Ergebnis Leistungsaufnahme von dadd

A.2.3 Energiezähler über RAPL-Schnittstelle

Das nachfolgende Listing zeigt, wie die Leistungsaufnahme der CPU, die mithilfe der Anwendung `rapl-read` ausgegeben wird. Weaver hatte mehrere Möglichkeiten genannt, um den Energieverbrauch über die RAPL-Schnittstelle auszuwerten[25]. Eine dieser Möglichkeiten ist die gezeigte Anwendung, die unter folgender URL bereitgestellt wurde: <https://github.com/deater/uarch-configure>

```
1 root@allan-Latitude-5500:/home/allan/git/uarch-configure/rapl-read# ./rapl-plot
2
3 RAPL read -- use -s for sysfs, -p for perf_event, -m for msr
4
5 Found Kaby Lake Processor type
6   0 (0), 1 (0), 2 (0), 3 (0), 4 (0), 5 (0), 6 (0), 7 (0)
7
8   Detected 8 cores in 1 packages
9
10
11 Using sysfs powercap interface to gather results
12
13 Time (s)   Package0 (W) Cores(W)   GPU(W)   DRAM(W)   Psys(W) |
14 0.500160   18.228521 17.099118 0.025626 0.507044 0.000000 |
15 1.000436   18.318819 16.902001 0.042215 0.526685 0.000000
```

Listing A.4: Leistungsaufnahme der CPU über RAPL[25]

A.2.4 Assemblercode von `dadd`

```
1 call 0x55555555129 <run\_dadd>
2 endbr64
3 push %rbp
4 mov %rsp,%rbp
5 mov %edi,-0x24(%rbp)
6 movsd 0xec(%rip),%xmm0 # 0x555555556008
7 movsd %xmm0,-0x18(%rbp)
8 movsd 0xec7(%rip),%xmm0 # 0x555555556010
9 movsd %xmm0,-0x10(%rbp)
10 movsd 0xec2(%rip),%xmm0 # 0x555555556018
11 movsd %xmm0,-0x8(%rbp)
12 jmp 0x55555555318 <run\_dadd+495>
13 mov -0x24(%rbp),%eax
14 lea -0x1(%rax),%edx
15 mov %edx,-0x24(%rbp)
16 test %eax,%eax
17 jg 0x55555555160 <run\_dadd+55>
18 movsd -0x18(%rbp),%xmm1
19 movsd 0xeb3(%rip),%xmm0 # 0x555555556020
20 addsd %xmm1,%xmm0
21 movsd %xmm0,-0x18(%rbp)
22 movsd -0x10(%rbp),%xmm1
23 movsd 0xea5(%rip),%xmm0 # 0x555555556028
24 addsd %xmm1,%xmm0
25 movsd %xmm0,-0x10(%rbp)
26 movsd -0x8(%rbp),%xmm1
27 movsd 0xe97(%rip),%xmm0 # 0x555555556030
28 addsd %xmm1,%xmm0
29 movsd %xmm0,-0x8(%rbp)
30 movsd -0x18(%rbp),%xmm1
31 movsd 0xe89(%rip),%xmm0 # 0x555555556038
32 addsd %xmm1,%xmm0
33 movsd %xmm0,-0x18(%rbp)
34 movsd -0x10(%rbp),%xmm1
35 movsd 0xe7b(%rip),%xmm0 # 0x555555556040
36 addsd %xmm1,%xmm0
37 movsd %xmm0,-0x10(%rbp)
38 movsd -0x8(%rbp),%xmm1
39 movsd 0xe6d(%rip),%xmm0 # 0x555555556048
40 addsd %xmm1,%xmm0
41 movsd %xmm0,-0x8(%rbp)
42 movsd -0x18(%rbp),%xmm1
43 movsd 0xe5f(%rip),%xmm0 # 0x555555556050
44 addsd %xmm1,%xmm0
45 movsd %xmm0,-0x18(%rbp)
46 movsd -0x10(%rbp),%xmm1
47 movsd 0xe51(%rip),%xmm0 # 0x555555556058
48 addsd %xmm1,%xmm0
49 movsd %xmm0,-0x10(%rbp)
50 movsd -0x8(%rbp),%xmm1
51 movsd 0xe43(%rip),%xmm0 # 0x555555556060
```

```

52 addsd %xmm1,%xmm0
53 movsd %xmm0,-0x8(%rbp)
54 movsd -0x18(%rbp),%xmm1
55 movsd 0xe35(%rip),%xmm0 # 0x555555556068
56 addsd %xmm1,%xmm0
57 movsd %xmm0,-0x18(%rbp)
58 movsd -0x10(%rbp),%xmm1
59 movsd 0xdd7(%rip),%xmm0 # 0x555555556020
60 addsd %xmm1,%xmm0
61 movsd %xmm0,-0x10(%rbp)
62 movsd -0x8(%rbp),%xmm1
63 movsd 0xdc9(%rip),%xmm0 # 0x555555556028
64 addsd %xmm1,%xmm0
65 movsd %xmm0,-0x8(%rbp)
66 movsd -0x18(%rbp),%xmm1
67 movsd 0xdbb(%rip),%xmm0 # 0x555555556030
68 addsd %xmm1,%xmm0
69 movsd %xmm0,-0x18(%rbp)
70 movsd -0x10(%rbp),%xmm1
71 movsd 0xdad(%rip),%xmm0 # 0x555555556038
72 addsd %xmm1,%xmm0
73 movsd %xmm0,-0x10(%rbp)
74 movsd -0x8(%rbp),%xmm1
75 movsd 0xd9f(%rip),%xmm0 # 0x555555556040
76 addsd %xmm1,%xmm0
77 movsd %xmm0,-0x8(%rbp)
78 movsd -0x18(%rbp),%xmm1
79 movsd 0xd91(%rip),%xmm0 # 0x555555556048
80 addsd %xmm1,%xmm0
81 movsd %xmm0,-0x18(%rbp)
82 movsd -0x10(%rbp),%xmm1
83 movsd 0xd83(%rip),%xmm0 # 0x555555556050
84 addsd %xmm1,%xmm0
85 movsd %xmm0,-0x10(%rbp)
86 movsd -0x8(%rbp),%xmm1
87 movsd 0xd75(%rip),%xmm0 # 0x555555556058
88 addsd %xmm1,%xmm0
89 movsd %xmm0,-0x8(%rbp)
90 movsd -0x18(%rbp),%xmm1
91 movsd 0xd67(%rip),%xmm0 # 0x555555556060
92 addsd %xmm1,%xmm0
93 movsd %xmm0,-0x18(%rbp)
94 movsd -0x10(%rbp),%xmm1
95 movsd 0xd59(%rip),%xmm0 # 0x555555556068
96 addsd %xmm1,%xmm0
97 movsd %xmm0,-0x10(%rbp)
98 mov -0x24(%rbp),%eax
99 lea -0x1(%rax),%edx
100 mov %edx,-0x24(%rbp)
101 test %eax,%eax
102 jg 0x55555555160 <run_dadd+55>
103 mov $0x0,%eax
104 pop %rbp

```

Listing A.5: Assemblercode von dadd[5]

A.2.5 Beispiel einer taskbasierte Anwendung in C

Das nachfolgende Listing zeigt ein mögliches C-Programm welches, hinsichtlich verwendeter Tasks, untersucht werden könnte.

```

1 #include <pthread.h>
2 #include <stdlib.h>
3
4 typedef struct {
5     int size, row, column;
6     double (*MA)[8], (*MB)[8], (*MC)[8];
7 } matrix_type_t;
8
9 void *thread_mult (void *w) {
10     matrix_type_t * work = (matrix_type_t *) w;
11     int i, row = work->row, column = work->column;
12     work->MC[row][column] = 0;
13     for (i = 0; i < work->size; i++)
14         work->MC[row][column] += work->MA[row][i] * work->MB[i][column];
15     return NULL;
16 }
17
18 int main() {
19     int row, column, size = 8, i;
20     double MA[8][8], MB[8][8], MC[8][8];
21     matrix_type_t *work;

```



```

22 pthread_t thread[8 * 8];
23 for (row = 0; row < size; row++)
24     for (column = 0; column < size; column++) {
25         work = (matrix_type_t *) malloc(sizeof(matrix_type_t));
26         work->size = size;
27         work->row = row;
28         work->column = column;
29         work->MA = MA;
30         work->MB = MB;
31         work->MC = MC;
32         pthread_create(&(thread[column + row * 8]), NULL,
33             thread_mult, (void *) work);
34     }
35     for (i = 0; i < size * size; i++)
36         pthread_join(thread[i], NULL);
37 }

```

Listing A.6: Thread-Programm zur Realisierung einer Matrixmultiplikation[21][s.S. 291

A.2.6 run_all_tasks.sh aus apptaskanalyzer

```

1 #!/bin/bash
2 tasks=(loadimage greyscale checkcontrast sharpencontrast copyimage sobelx sobely combineimgs writeimage)
3 stops=(206          167          57          86          85          146          116          189          226)
4 mkdir -p gdb
5 mkdir -p seq
6
7
8 for ((i = 0; i < ${#tasks[@]}; i++)); do
9
10     taskname=${tasks[i]}
11     taskstop=${stops[i]}
12     gdbname=$taskname".gdb"
13
14     #GDB-Datei vorbereiten
15     echo "b $taskname > $gdbname"
16     echo "r" >> $gdbname
17     echo "source ci.py" >> $gdbname
18     echo "b $taskstop >> $gdbname"
19     echo "ci" >> $gdbname
20     echo "exit" >> $gdbname
21
22     gdb --batch --command=$gdbname --args apptasks $taskname > $taskname.gdb.log
23     mv instructionseq.txt seq/$taskname.seq
24     mv $taskname".gdb" gdb/$taskname".gdb"
25     mv $taskname.gdb.log gdb/$taskname.gdb.log
26 done
27 cp -f seq/*.seq ../3_tasktype_estimator/taskmapper/appseq

```

Listing A.7: run_all_tasks.sh aus apptasks

A.2.7 main.cpp des Taskmappers

Das nachfolgende Listing zeigt einen Auszug aus der Quelldatei main.cpp. Zunächst wird der eingegebene Parameter überprüft. Wenn eines der Abbildungsverfahren getestet werden soll, dann wird zunächst eine Liste der vorhandenen Tasks benötigt. Für diesen Zweck wird die Methode initTaskVektors() aufgerufen, die den Ordner mit den Sequenzdateien durchsucht und einen entsprechenden Vektor vorbereitet, aus dem ein Anwendungstask ausgewählt werden kann. Nach Auswahl eines Tasks wird die entsprechende Testmethode ausgeführt. Der ausgewählte Anwendungstask wird also im entsprechenden Verfahren auf Task-Typen abgebildet.

```

1 if (strcasecmp(parameter, "A")==0) {
2     initTaskVektors();
3     printf("Im Ordner taskmapper/appseq sind %lu Skripte hinterlegt, bitte durch Eingabe auswählen!\n", apptaskVektor.size());
4
5     for (int i = 0; i < apptaskVektor.size(); i++) {
6         printf("[%d]: %s\n", i, apptaskVektor[i].taskname.c_str());
7     }
8
9     int index = 0;
10    if (scanf("%d", &index) == 1) {

```

```

11         compareAppTaskProtTasksOneToOneTest(apptaskVektor[index].taskname);
12     }
13 } else if (strcasecmp(parameter, "B")==0) {
14     compareAppTaskProtTasksOneToOne();
15 } else if (strcasecmp(parameter, "D")==0) {
16     initTaskVektors();
17     printf("Im Ordner taskmapper/appseq sind %lu Skripte hinterlegt, bitte durch Eingabe auswählen!\n", apptaskVektor.size());
18
19     for (int i = 0; i < apptaskVektor.size(); i++) {
20         printf("[%d]: %s\n", i, apptaskVektor[i].taskname.c_str());
21     }
22
23     int index = 0;
24     if (scanf("%d", &index) == 1) {
25         compareAppTaskProtTasksOneToManyTest(apptaskVektor[index].taskname);
26     }
27 } else if (strcasecmp(parameter, "E")==0) {
28     compareAppTaskProtTasksOneToMany();
29 } else if (strcasecmp(parameter, "T")==0) {
30     transferTaskMapToEpEBench();
31 }

```

Listing A.8: Auszug aus main.cpp

A.2.8 Implementierung des 1:1-Abbildungsverfahrens

```

1
2 AnwTask compareAppTaskWithPrototypTasks(AnwTask appTask, PrototypTask protTypTask ) {
3
4     float sizeAppDivProt = (float) appTask.sequenzen.size() / protTypTask.sequenzen.size();
5
6     if (sizeAppDivProt > 0.5) {
7         for (int i = 0; i < sizeAppDivProt; i++) {
8             for (std::string protTaskSequenceEntry : protTypTask.sequenzen) {
9                 appTask = compareProtTaskSequenEntryWithAppTaskEntry(protTaskSequenceEntry, appTask, 0);
10            }
11        }
12    }
13
14    int anzahl_hits = 0;
15    for (bool found : appTask.found) {
16        if (found) {
17            anzahl_hits++;
18        }
19    }
20
21    logTaskMapperOne2One(appTask.taskname, protTypTask.taskname, anzahl_hits, appTask.sequenzen.size(), protTypTask.sequenzen.size(), sizeAppDivProt);
22
23    if (appTask.resultOneToOne.protTaskAnzTrefferMap[appTask.taskname][protTypTask.taskname] == 0 || appTask.resultOneToOne.protTaskAnzTrefferMap[appTask.taskname][protTypTask.taskname] < anzahl_hits) {
24        appTask.resultOneToOne.protTaskAnzTrefferMap[appTask.taskname][protTypTask.taskname]=anzahl_hits;
25    }
26
27    appTask.resetFound();
28    return appTask;
29 }
30
31 AnwTask compareProtTaskSequenEntryWithAppTaskEntry(std::string protTaskSequenceEntry, AnwTask appTask, int maxIndex) {
32     for (int i=0; i < appTask.sequenzen.size(); i++) {
33         if (appTask.found[i] == true) {
34             continue; // Sequenz bereits gefunden, nächster Treffer!
35         }
36
37         if (maxIndex!=0 && i > maxIndex) {
38             return appTask;
39         }
40
41         if (appTask.sequenzen[i].compare(protTaskSequenceEntry) == 0) {
42             appTask.found[i]=true; // Sequenz gefunden, wird auf true gesetzt!
43             return appTask;
44         }
45     }
46     return appTask;
47 }

```

Listing A.9: Auszug aus taskmapperOneToOne.cpp

A.2.9 Implementierung des 1:N-Abbildungsverfahrens

```
1 AnwTask analyseAppTaskMany(AnwTask appTask) {
2     int sizeAppDivProt = ceil(float(appTask.sequenzen.size() / calcSequenzSize()))+1;
3
4     for (int i = 0; i < sizeAppDivProt; i++) {
5         resultOneToMany=appTask.resultOneToOne;
6
7         if (appTask.allFound()){
8             return appTask;
9         }
10
11         appTask= analyse(appTask);
12     }
13     ...
14 }
15
16 AnwTask analyse(AnwTask appTask) {
17     std::list<std::thread> threadlist;
18     std::thread myThreads[prottaskvektor.size()];
19     for (int j = 0; j < prottaskvektor.size(); j++) {
20         myThreads[j] = std::thread(prepareAnwTaskAndProtTypTaskForCompare, appTask, prottaskvektor[j]);
21     }
22
23     for (int j = 0; j < prottaskvektor.size(); j++) {
24         myThreads[j].join();
25     }
26
27     appTask.resultOneToOne=resultOneToMany;
28     appTask = calcBestTaskMany(appTask);
29     logBestTask(appTask, true, false);
30     appTask.resultOneToOne.list.push_back(appTask.bestName);
31     appTask.resultOneToOne.abgebildeteTaskMap[appTask.bestName]+=1;
32
33     appTask.mergeFound(appTask.resultOneToOne.pptFoundMapWithBool[appTask.bestName]);
34     appTask.resultOneToOne.resetPptFoundMap(appTask.resultOneToOne.pptFoundMapWithBool[appTask.bestName]); // wieder zurücksetzen!
35     appTask.resultOneToOne.protTaskAnzTrefferMap.clear();
36     appTask.bestName="";
37     appTask.besthit=0;
38     return appTask;
39 }
40
41 AnwTask compareProtTaskSequenEntryWithAppTaskEntryMany(std::string protTaskSequenceEntry, AnwTask appTask, PrototypTask
42     protTypTask, int maxIndex) {
43     std::list<int> indexes = appTask.indexOfMap[protTaskSequenceEntry];
44
45     if (indexes.empty()) {
46         return appTask;
47     }
48
49     for (int index : indexes) {
50         if (appTask.resultOneToOne.pptFoundMapWithBool[protTypTask.taskname][index]) {
51             continue; // Sequenz bereits gefunden, nächster Treffer!
52         }
53         if (appTask.found[index]) {
54             continue; // Sequenz bereits gefunden, nächster Treffer!
55         }
56         if (maxIndex!=0 && index > maxIndex) {
57             return appTask;
58         }
59
60         appTask.resultOneToOne.pptFoundMapWithBool[protTypTask.taskname][index]=true;
61
62         return appTask;
63     }
64
65     return appTask;
66 }
```

Listing A.10: Auszug aus taskmapperOneToMany.cpp

A.2.10 Implementierung eines alternativen Abbildungsverfahrens: Diff-Suche

```
1
2 void testDiffSearch() {
3     initTaskVektors();
4     if (checkSequenzfiles()==false) {
5         return;
6     }
7     currentPrototypTask = prottaskvektor[3];
8     currentPrototypTask.initHit();
9
10    for (PrototypTask ptt : prottaskvektor) {
```

```

11     if (currentPrototypTask.taskname == ptt.taskname) {
12         continue;
13     }
14     ptt.initHit();
15     comparePrototypTaskWithOtherPrototypTask(ptt);
16
17     std::cout << " Aus " << currentPrototypTask.taskname << " sind in " << ptt.taskname << " nicht enthalten: " <<
currentPrototypTask.countNotFound() << " von " << currentPrototypTask.sequenzen.size() << "\n";
18     currentPrototypTask.initHit();
19 }
20
21 std::cout << "\nEs wird nun geprüft, ob der Tasktyp '" << currentPrototypTask.taskname << "' Sequenzen enthält, die in keinem
anderen Tasktyp vorkommen!\n";
22 for (PrototypTask ptt : prottaskVektor) {
23     if (currentPrototypTask.taskname == ptt.taskname) {
24         continue;
25     }
26     if (isSimilarTask(ptt.taskname)) {
27         continue;
28     }
29
30     std::cout << "Vergleiche " << currentPrototypTask.taskname << " mit " << ptt.taskname << "\n";
31     ptt.initHit();
32     comparePrototypTaskWithOtherPrototypTask(ptt);
33     std::cout << " Aus dem Task-Typen '" << currentPrototypTask.taskname << "' wurden bisher " << currentPrototypTask.
countNotFound() << " nicht gefunden!\n";
34 }
35
36 std::cout << "Insgesamt nicht enthalten sind:\n";
37 for (int i = 0; i < currentPrototypTask.sequenzen.size(); i++) {
38     if (!currentPrototypTask.hit[i]) {
39         std::cout << currentPrototypTask.sequenzen[i] << "\n";
40     }
41 }
42
43 std::cout << "Das sind " << currentPrototypTask.countNotFound() << " von " << currentPrototypTask.sequenzen.size() << "\n";
44 }
45
46 PrototypTask compareProtTaskSequenEntries(int indexToCompare, PrototypTask protTypTaskToCompare) {
47     std::string protTaskSequenceEntry = currentPrototypTask.sequenzen[indexToCompare];
48
49     for (int i=0; i < protTypTaskToCompare.sequenzen.size(); i++) {
50         if (protTypTaskToCompare.hit[i] == true) {
51             continue; // Sequenz bereits gefunden, nächster Treffer!
52         }
53
54         if (protTypTaskToCompare.sequenzen[i].compare(protTaskSequenceEntry) == 0) {
55             currentPrototypTask.hit[indexToCompare]=true;
56             protTypTaskToCompare.hit[i]=true; // Sequenz gefunden, wird auf true gesetzt!
57             return protTypTaskToCompare;
58         }
59     }
60     return protTypTaskToCompare;
61 }

```

Listing A.11: Implementierung der Diff-Suche

A.2.11 Logdatei für alternative Abbildung: Diffsuche

Für die Eruierung der Diffsuche wurde der Tasktyp bitbyte verwendet. Zunächst wurden die Befehle des Task-Typen mit den Befehlen der anderen Task-Typen verglichen. Der Tasktyp bitbyte besteht aus 2.037 Befehlen. Bei dem Vergleich mit dem Task-Typen icompare wurden 41 Befehle festgestellt, die zwar in bitbyte enthalten sind, aber nicht in icompare gefunden wurde. Diese Suche wird mit den anderen Task-Typen wiederholt und die Anzahl der nicht gefundenen Befehle variiert im zweistelligen Bereich.

Der Zweck dieses Suchverfahren war für jeden Task-Typen einen Befehl zu finden, der charakteristisch für diesen Task-Typen ist und in keinem anderen Task-Typen vorkommt. Aus diesem Grund wurden im nächsten Schritt noch einmal die Befehle des Task-Typen bitbyte mit denen von icompare verglichen und erneut die 41 Befehle festgesellt, die nur in bitbyte vorkommen. Diese Befehle wurden nun im nächsten Tasktyp imul gesucht. Die Suche hat ergeben, dass diese Befehle

in imul vorgekommen sind, sodass festzustellen ist, dass der Tasktyp bitbyte keine Befehle enthält, die nur in diesem Task-Typen vorkommen.

Das Listing A.12 zeigt die Logausgabe dieses Versuches.

```
1 Der Tasktyp bitbyte wird mit den anderen Task-Typen verglichen!
2 Aus bitbyte sind in icompare nicht enthalten: 41 von 2037
3 Aus bitbyte sind in imul nicht enthalten: 20 von 2037
4 Aus bitbyte sind in logic nicht enthalten: 13 von 2037
5 Aus bitbyte sind in dsub64_SIMD nicht enthalten: 64 von 2037
6 Aus bitbyte sind in branch nicht enthalten: 61 von 2037
7 Aus bitbyte sind in dmem64_SIMD nicht enthalten: 23 von 2037
8 Aus bitbyte sind in iadd nicht enthalten: 20 von 2037
9 Aus bitbyte sind in shift nicht enthalten: 61 von 2037
10 Aus bitbyte sind in ssub32_SIMD nicht enthalten: 64 von 2037
11 Aus bitbyte sind in smul32_SIMD nicht enthalten: 64 von 2037
12 Aus bitbyte sind in dmul nicht enthalten: 60 von 2037
13 Aus bitbyte sind in vlx4smul_SIMD nicht enthalten: 23 von 2037
14 Aus bitbyte sind in m4x4smul nicht enthalten: 21 von 2037
15 Aus bitbyte sind in vconvert_SIMD nicht enthalten: 23 von 2037
16 Aus bitbyte sind in nop nicht enthalten: 62 von 2037
17 Aus bitbyte sind in m4x4smul_SIMD nicht enthalten: 23 von 2037
18 Aus bitbyte sind in vmov_SIMD nicht enthalten: 23 von 2037
19 Aus bitbyte sind in dadd nicht enthalten: 60 von 2037
20 Aus bitbyte sind in imem nicht enthalten: 39 von 2037
21 Aus bitbyte sind in dmul64_SIMD nicht enthalten: 64 von 2037
22 Aus bitbyte sind in imov nicht enthalten: 61 von 2037
23 Aus bitbyte sind in dmem nicht enthalten: 35 von 2037
24
25 Es wird nun geprüft, ob der Tasktyp 'bitbyte' Sequenzen enthält, die in keinem anderen Tasktyp vorkommen!
26 Vergleiche bitbyte mit icompare
27 Aus dem Tasktypen 'bitbyte' wurden bisher 41 nicht gefunden!
28 Vergleiche bitbyte mit imul
29 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
30 Vergleiche bitbyte mit logic
31 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
32 Vergleiche bitbyte mit dsub64_SIMD
33 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
34 Vergleiche bitbyte mit branch
35 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
36 Vergleiche bitbyte mit dmem64_SIMD
37 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
38 Vergleiche bitbyte mit iadd
39 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
40 Vergleiche bitbyte mit shift
41 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
42 Vergleiche bitbyte mit ssub32_SIMD
43 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
44 Vergleiche bitbyte mit smul32_SIMD
45 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
46 Vergleiche bitbyte mit dmul
47 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
48 Vergleiche bitbyte mit vlx4smul_SIMD
49 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
50 Vergleiche bitbyte mit m4x4smul
51 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
52 Vergleiche bitbyte mit vconvert_SIMD
53 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
54 Vergleiche bitbyte mit nop
55 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
56 Vergleiche bitbyte mit m4x4smul_SIMD
57 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
58 Vergleiche bitbyte mit vmov_SIMD
59 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
60 Vergleiche bitbyte mit dadd
61 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
62 Vergleiche bitbyte mit imem
63 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
64 Vergleiche bitbyte mit dmul64_SIMD
65 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
66 Vergleiche bitbyte mit imov
67 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
68 Vergleiche bitbyte mit dmem
69 Aus dem Tasktypen 'bitbyte' wurden bisher 0 nicht gefunden!
70 Insgesamt nicht enthalten sind:
71 Das sind 0 von 2037
```

Listing A.12: Logausgabe Diffsuche

A.2.12 Implementierung eines alternativen Abbildungsverfahrens: Paar-Suche

```
1 AnwTask analyseAppTaskPair (AnwTask anwTak) {
2
```

```

3     std::string prevEntry = anwTak.sequenzen[0];
4     std::string currentEntry;
5
6     std::map<std::string, std::map<std::string, bool>> pairEntryMap;
7
8     for (int i = 1; i < anwTak.sequenzen.size(); i++) {
9         currentEntry=anwTak.sequenzen[i];
10        pairEntryMap[prevEntry][currentEntry]=true;
11        prevEntry=currentEntry;
12    }
13
14    for (PrototypTask t : newprottaskVektor) {
15        int counter = 0;
16        std::map<std::string, std::map<std::string, bool>>::iterator it;
17        for(it=t.uniqueEntryPairMap.begin(); it != t.uniqueEntryPairMap.end(); it++) {
18            std::string prev = it->first;
19            std::map<std::string, bool> innerMap = it->second;
20            std::map<std::string, bool>::iterator innerIt;
21            for(innerIt=innerMap.begin(); innerIt!=innerMap.end(); innerIt++) {
22                std::string current = innerIt->first;
23                if (pairEntryMap[prev][current]) {
24                    t.uniqueEntryPairMap[prev][current]=true;
25                    counter++;
26                }
27            }
28        }
29        logMessageOnTaskmapperFileAndCout("Im Anwendungstask " + anwTak.taskname + " sind aus " + t.taskname + " " + std::
to_string(counter) + " eindeutige Kombinationen enthalten\n", true);
30
31        if (anwTak.besthit<counter) {
32            anwTak.bestName=t.taskname;
33            anwTak.besthit=counter;
34        }
35    }
36
37    logMessageOnTaskmapperFileAndCout("Für den Anwendungstask " + anwTak.taskname + " ist der Tasktyp " + anwTak.bestName + " bei
Paarsuche am Ähnlichsten: " + std::to_string(anwTak.besthit) + " Treffer!\n\n", true);
38
39    return anwTak;
40 }

```

Listing A.13: Implementierung der Paar-Suche

```

1
2 Im Tasktyp icompare sind insgesamt 182 Kombinationen eindeutig, nicht eindeutig sind: 843
3 Im Tasktyp imul sind insgesamt 184 Kombinationen eindeutig, nicht eindeutig sind: 868
4 Im Tasktyp logic sind insgesamt 158 Kombinationen eindeutig, nicht eindeutig sind: 996
5 Im Tasktyp bitbyte sind insgesamt 181 Kombinationen eindeutig, nicht eindeutig sind: 932
6 Im Tasktyp dsub64_SIMD sind insgesamt 163 Kombinationen eindeutig, nicht eindeutig sind: 994
7 Im Tasktyp branch sind insgesamt 157 Kombinationen eindeutig, nicht eindeutig sind: 995
8 Im Tasktyp dmem64_SIMD sind insgesamt 185 Kombinationen eindeutig, nicht eindeutig sind: 846
9 Im Tasktyp iadd sind insgesamt 184 Kombinationen eindeutig, nicht eindeutig sind: 868
10 Im Tasktyp shift sind insgesamt 159 Kombinationen eindeutig, nicht eindeutig sind: 995
11 Im Tasktyp ssub32_SIMD sind insgesamt 160 Kombinationen eindeutig, nicht eindeutig sind: 994
12 Im Tasktyp smul32_SIMD sind insgesamt 160 Kombinationen eindeutig, nicht eindeutig sind: 994
13 Im Tasktyp dmul sind insgesamt 163 Kombinationen eindeutig, nicht eindeutig sind: 990
14 Im Tasktyp vlx4smul_SIMD sind insgesamt 192 Kombinationen eindeutig, nicht eindeutig sind: 974
15 Im Tasktyp m4x4smul sind insgesamt 196 Kombinationen eindeutig, nicht eindeutig sind: 875
16 Im Tasktyp vconvert_SIMD sind insgesamt 185 Kombinationen eindeutig, nicht eindeutig sind: 974
17 Im Tasktyp nop sind insgesamt 161 Kombinationen eindeutig, nicht eindeutig sind: 994
18 Im Tasktyp m4x4smul_SIMD sind insgesamt 180 Kombinationen eindeutig, nicht eindeutig sind: 916
19 Im Tasktyp vmov_SIMD sind insgesamt 175 Kombinationen eindeutig, nicht eindeutig sind: 886
20 Im Tasktyp dadd sind insgesamt 163 Kombinationen eindeutig, nicht eindeutig sind: 990
21 Im Tasktyp imem sind insgesamt 165 Kombinationen eindeutig, nicht eindeutig sind: 1051
22 Im Tasktyp dmul64_SIMD sind insgesamt 163 Kombinationen eindeutig, nicht eindeutig sind: 994
23 Im Tasktyp imov sind insgesamt 156 Kombinationen eindeutig, nicht eindeutig sind: 995
24 Im Tasktyp dmem sind insgesamt 160 Kombinationen eindeutig, nicht eindeutig sind: 1093
25
26 Im Anwendungstask writeimage sind aus icompare 356 eindeutige Kombinationen enthalten
27 Im Anwendungstask writeimage sind aus imul 356 eindeutige Kombinationen enthalten
28 Im Anwendungstask writeimage sind aus logic 358 eindeutige Kombinationen enthalten
29 Im Anwendungstask writeimage sind aus bitbyte 356 eindeutige Kombinationen enthalten
30 Im Anwendungstask writeimage sind aus dsub64_SIMD 356 eindeutige Kombinationen enthalten
31 Im Anwendungstask writeimage sind aus branch 356 eindeutige Kombinationen enthalten
32 Im Anwendungstask writeimage sind aus dmem64_SIMD 357 eindeutige Kombinationen enthalten
33 Im Anwendungstask writeimage sind aus iadd 356 eindeutige Kombinationen enthalten
34 Im Anwendungstask writeimage sind aus shift 358 eindeutige Kombinationen enthalten
35 Im Anwendungstask writeimage sind aus ssub32_SIMD 356 eindeutige Kombinationen enthalten
36 Im Anwendungstask writeimage sind aus smul32_SIMD 356 eindeutige Kombinationen enthalten
37 Im Anwendungstask writeimage sind aus dmul 356 eindeutige Kombinationen enthalten
38 Im Anwendungstask writeimage sind aus vlx4smul_SIMD 358 eindeutige Kombinationen enthalten
39 Im Anwendungstask writeimage sind aus m4x4smul 359 eindeutige Kombinationen enthalten
40 Im Anwendungstask writeimage sind aus vconvert_SIMD 358 eindeutige Kombinationen enthalten
41 Im Anwendungstask writeimage sind aus nop 358 eindeutige Kombinationen enthalten
42 Im Anwendungstask writeimage sind aus m4x4smul_SIMD 358 eindeutige Kombinationen enthalten
43 Im Anwendungstask writeimage sind aus vmov_SIMD 356 eindeutige Kombinationen enthalten
44 Im Anwendungstask writeimage sind aus dadd 356 eindeutige Kombinationen enthalten
45 Im Anwendungstask writeimage sind aus imem 356 eindeutige Kombinationen enthalten

```

```

46 Im Anwendungstask writeimage sind aus dmul64_SIMD 356 eindeutige Kombinationen enthalten
47 Im Anwendungstask writeimage sind aus imov 356 eindeutige Kombinationen enthalten
48 Im Anwendungstask writeimage sind aus dmem 357 eindeutige Kombinationen enthalten
49 Für den Anwendungstask writeimage ist der Tasktyp m4x4smul bei Paarsuche am ähnlichsten: 359.000000 Treffer!

```

Listing A.14: Bildschirmausgabe bei Paar-Suche

A.2.13 Implementierung eines alternativen Abbildungsverfahrens: Drilling-Suche

```

1  initTaskVektors();
2  if (checkSequenzfiles()==false) {
3      return;
4  }
5
6  currentPrototypTask = prottaskVektor[1];
7  currentPrototypTask.initHit();
8
9  std::string prevPrevEntry = currentPrototypTask.sequenzen[0];
10 std::string prevEntry = currentPrototypTask.sequenzen[1];
11 std::string currentEntry;
12
13 std::map<std::string, std::map<std::string, std::map<std::string, bool>>> myMap;
14
15 for (int i = 2; i < currentPrototypTask.sequenzen.size(); i++) {
16     currentEntry=currentPrototypTask.sequenzen[i];
17     myMap[prevPrevEntry][prevEntry][currentEntry]=true;
18     prevPrevEntry=prevEntry;
19     prevEntry=currentEntry;
20 }
21
22
23 for (PrototypTask prototypTaskToCompare : prottaskVektor) {
24     if (currentPrototypTask.taskname == prototypTaskToCompare.taskname) {
25         continue;
26     }
27
28     std::cout << "Triplesuche " << currentPrototypTask.taskname << " mit " << prototypTaskToCompare.taskname << "\n";
29
30     prevPrevEntry = prototypTaskToCompare.sequenzen[0];
31     prevEntry = prototypTaskToCompare.sequenzen[1];
32     for (int i = 2; i < prototypTaskToCompare.sequenzen.size(); i++) {
33         currentEntry=prototypTaskToCompare.sequenzen[i];
34         if (myMap[prevPrevEntry][prevEntry][currentEntry]) {
35             myMap[prevPrevEntry][prevEntry][currentEntry]=false;
36             prevPrevEntry=prevEntry;
37             prevEntry=currentEntry;
38         }
39     }
40 }
41
42 std::map<std::string, std::map<std::string, std::map<std::string, bool>>> ::iterator it;
43 std::cout << "Nicht gefunden wurden folgende Kombinationen:\n";
44 int counter = 0;
45 int counterNot = 0;
46 for (it = myMap.begin(); it != myMap.end(); it++) {
47     std::string prevprev = it->first;
48     std::map<std::string, std::map<std::string, bool>> innerMap = it->second;
49     std::map<std::string, std::map<std::string, bool>>::iterator innerIt;
50     for (innerIt = innerMap.begin(); innerIt != innerMap.end(); innerIt++) {
51         std::string prev = innerIt->first;
52         std::map<std::string, bool> innerinnerMap = innerIt->second;
53         std::map<std::string, bool>::iterator innerInnerIt;
54         for (innerInnerIt = innerinnerMap.begin(); innerInnerIt != innerinnerMap.end(); innerInnerIt++) {
55             std::string current = innerInnerIt->first;
56             bool b = innerInnerIt->second;
57             if (b) {
58                 counter++;
59                 std::cout << "[" << prevprev << "/" << prev << "/" << current << "]\n";
60             } else {
61                 counterNot++;
62             }
63         }
64     }
65 }
66
67 std::cout << "Demnach sind im Tasktyp '" << currentPrototypTask.taskname << "' insgesamt " << counter << " Kombinationen
    eindeutig, nicht eindeutig sind: " << counterNot << "\n";
68 }
69 Triplesuche imul mit icompare
70 Triplesuche imul mit logic
71 Triplesuche imul mit bitbyte

```

```

72 Triplesuche imul mit dsub64_SIMD
73 Triplesuche imul mit branch
74 Triplesuche imul mit dmem64_SIMD
75 Triplesuche imul mit iadd
76 Triplesuche imul mit shift
77 Triplesuche imul mit ssub32_SIMD
78 Triplesuche imul mit smul32_SIMD
79 Triplesuche imul mit dmul
80 Triplesuche imul mit v1x4smul_SIMD
81 Triplesuche imul mit m4x4smul
82 Triplesuche imul mit vconvert_SIMD
83 Triplesuche imul mit nop
84 Triplesuche imul mit m4x4smul_SIMD
85 Triplesuche imul mit vmov_SIMD
86 Triplesuche imul mit dadd
87 Triplesuche imul mit imem
88 Triplesuche imul mit dmul64_SIMD
89 Triplesuche imul mit imov
90 Triplesuche imul mit dmem
91 Demnach sind im Tasktyp 'imul' insgesamt 586 Kombinationen eindeutig, nicht eindeutig sind: 1082

```

Listing A.15: Implementierung der Drilling-Suche

A.2.14 Transfer Taskmap nach epEBench

Das nachfolgende Listing A.16 zeigt die Implementierung zur Übertragung der Taskmap in den Benchmark epEBench.

```

1 void transferTaskMapToEpEBench() {
2     std::ifstream infile(filename_taskmap_result_from_folder);
3     std::string currentSection;
4     std::string currentModel;
5
6     std::string line;
7     std::map<std::string, std::list<std::string>> mapOfLine;
8
9     if (!infile.is_open()) {
10         std::cout << "Taskmapdatei " << filename_taskmap_result_from_folder << " nicht gefunden!\n";
11         return;
12     } else {
13         std::cout << "Taskmapdatei " << filename_taskmap_result_from_folder << " gefunden!\n";
14     }
15
16     while (std::getline(infile, line)) {
17         size_t delimiterPos = line.find('=');
18         std::string keyInline = (delimiterPos!=0) ? line.substr(0, delimiterPos): NULL;
19         std::string valueInline = (delimiterPos!=0) ? line.substr(delimiterPos + 1):NULL;
20         if (line[0] == '[' && line[line.length() - 1] == ']') {
21             currentSection = line.substr(1, line.length() - 2);
22             continue;
23         }
24         if (currentSection == result_section_one2many) {
25             if (keyInline == result_section_model) {
26                 currentModel = valueInline;
27                 mapOfLine[currentModel].push_back(line);
28                 continue;
29             }
30             mapOfLine[currentModel].push_back(line);
31         } else {
32             continue;
33         }
34     }
35     infile.close();
36
37     std::list<std::string> linesEbmodels;
38
39     currentModel = "";
40     std::ifstream ebmodelsfile(filename_epebench_ebmodels);
41
42     if (!ebmodelsfile.is_open()) {
43         std::cout << "epEBench-ebmodels-Datei " << filename_epebench_ebmodels << " nicht gefunden!\n";
44         return;
45     } else {
46         std::cout << "epEBench-ebmodels-Datei " << filename_epebench_ebmodels << " gefunden!\n";
47     }
48
49     std::set<std::string> modellsInepeBench;
50
51     bool skipModel = false;
52     while (std::getline(ebmodelsfile, line)) {
53         size_t delimiterPos = line.find('=');
54         std::string keyInline = (delimiterPos!=0) ? line.substr(0, delimiterPos): NULL;

```



```

55     std::string valueInLine = (delimiterPos!=0) ? line.substr(delimiterPos + 1):NULL;
56
57     if (keyInline == result_section_model) {
58         currentModel = valueInLine;
59         modelsInepeBench.insert(currentModel);
60         if (mapOfLine[currentModel].size()==0){
61             linesEbmodels.push_back(line);
62             continue;
63         } else if (!skipModel){
64             skipModel=true;
65             for (std::string s : mapOfLine[currentModel]) {
66                 linesEbmodels.push_back(s);
67             }
68         }
69     } else if (keyInline == result_section_end_model || keyInline == result_section_end_model_space) {
70         if (skipModel) {
71             skipModel=false;
72             continue;
73         }
74         linesEbmodels.push_back(line);
75     } else if (!skipModel) {
76         linesEbmodels.push_back(line);
77     }
78 }
79 ebmodelsfile.close();
80
81 // Fehlende Modelle hinzufügen!
82 std::map<std::string, std::list<std::string>>::iterator it;
83 for (it = mapOfLine.begin(); it != mapOfLine.end(); ++it) {
84     std::string key = it->first;
85     std::list<std::string> values = it->second;
86     if (modelsInepeBench.find(key) == modelsInepeBench.end()) {
87         std::cout << key << " wird übertragen!\n";
88         linesEbmodels.insert(linesEbmodels.end(), values.begin(), values.end());
89     }
90 }
91
92
93 std::ofstream outfile(filename_epebench_ebmodels);
94 for (std::string s : linesEbmodels) {
95     outfile << s << "\n";
96 }
97 outfile.close();
98 std::cout << "Taskmap wurde nach epEBench übertragen!\n";
99 }

```

Listing A.16: Transfer nach epEBench

A.2.15 experiment.config

```

1 [TaskTypes]
2 tasktype=m4x4smul_SIMD
3 tasktype=vlx4smul_SIMD
4 tasktype=dmul64_SIMD
5 tasktype=smul32_SIMD
6 tasktype=dsub64_SIMD
7 tasktype=dmem64_SIMD
8 tasktype=vmov_SIMD
9 tasktype=vconvert_SIMD
10 tasktype=m4x4smul
11 tasktype=dmul
12 tasktype=dadd
13 tasktype=iadd
14 tasktype=imul
15 tasktype=icompare
16 tasktype=logic
17 tasktype=branch
18 tasktype=imem
19 tasktype=dmem
20 tasktype=imov
21 tasktype=shift
22 tasktype=bitbyte
23 tasktype=nop
24 tasktype=ssub32_SIMD
25
26 [ApplicationTasks]
27 apptask=checkcontrast
28 apptask=loadimage
29 apptask=greyscale
30 apptask=sharpencontrast
31 apptask=copyimage
32 apptask=sobelh
33 apptask=sobelv

```

```

34 apptask=combineimgs
35 apptask=writeimage
36
37 [CPUFrequency]
38 level=1500
39 level=2000
40 level=2200
41 level=2400
42 level=2500
43 level=3000
44
45 [Parallelism]
46 level=1
47 level=2
48 level=3
49 level=4

```

Listing A.17: Konfigurationsdatei experiment.config

A.2.16 Generierung von Skripten im Tasktypeestimator

```

1 void generateTaskTypeScripts(std::string folder, std::string task) {
2     // $CORES ist der Parameter für Anzahl Kerne!
3     std::string filename = folder + "/" + task + ".sh";
4     std::ofstream scriptfile (filename);
5     if (scriptfile.is_open()) {
6         scriptfile << "#!/bin/bash\n";
7         scriptfile << "CORES=$1\n";
8         scriptfile << "if [ -z \"$CORES\" ]\n";
9         scriptfile << "then\n";
10        scriptfile << "CORES=1\n";
11        scriptfile << "fi\n";
12        scriptfile << "echo Anzahl ist $CORES\n";
13        scriptfile << "cd ..\n";
14        scriptfile << "cd epEBench/bin/Release\n";
15        scriptfile << "./epebench -m " + task + " -t 5 -n $CORES -u 100 \n";
16        scriptfile << "chmod a+w epebench_loadlog.txt\n";
17        scriptfile << "mv epebench_loadlog.txt epebench_" + task + ".log\n";
18        scriptfile << "cd ../../..\n";
19    }
20    scriptfile.close();
21    chmod(filename.c_str(), 0777);
22 }
23
24 void generateAppTaskScripts(std::string task) {
25     // $CORES ist der Parameter für Anzahl Kerne!
26     std::string filename = foldername_generated_scripts_apptasks + "/" + task + ".sh";
27     std::ofstream scriptfile (filename);
28     if (scriptfile.is_open()) {
29         scriptfile << "#!/bin/bash\n";
30         scriptfile << "CORES=$1\n";
31         scriptfile << "if [ -z \"$CORES\" ]\n";
32         scriptfile << "then\n";
33         scriptfile << "CORES=1\n";
34         scriptfile << "fi\n";
35         scriptfile << "echo Anzahl ist $CORES\n";
36         scriptfile << "cd ..\n";
37         scriptfile << "cd edgedetection\n";
38         scriptfile << "echo \"$PWD\"\n";
39         scriptfile << "./edgedetection imgfilenames $CORES " + task + "\n";
40     }
41     scriptfile.close();
42     chmod(filename.c_str(), 0777);
43 }

```

Listing A.18: Generierung von Skripten in config.cpp

A.2.17 Messung der Leistungsaufnahme von Task-Typen

```

1 #ifndef INC_3_TASKTYPE_ESTIMATOR_MEASURERESULT_H
2 #define INC_3_TASKTYPE_ESTIMATOR_MEASURERESULT_H
3 #include <string>
4 /**
5  * Class to hold the result of a measurement
6  */
7 class MeasureResult {
8 public:
9
10    // Name of the task

```

```

11     std::string taskname;
12
13     // Duration of the task
14     long long duration;
15
16     // Energy used by the task in microjoules
17     long long energy_mj;
18
19     // CPU frequency during the task
20     std::string cpuFreq;
21
22     // Level of parallelism during the task
23     std::string parallelism;
24
25     // Duration of the task in a one-to-many scenario
26     long long duration_one_to_many;
27
28     // Energy used by the task in a one-to-many scenario in microjoules
29     long long energy_my_one_to_many;
30
31     /**
32      * Function to calculate the power used by the task
33      * @return float - power used by the task, -1 if power is 0
34      */
35     float power() {
36         if (energy_mj == 0 || duration==0) {
37             return 0;
38         }
39         float power = (float) energy_mj/duration;
40         if (power==0) {
41             return -1;
42         }
43         return power;
44     }
45
46     /**
47      * Function to calculate the power used by the task in a one-to-many scenario
48      * @return float - power used by the task in a one-to-many scenario, -1 if power is 0
49      */
50     float powerOneToMany() {
51         if (energy_mj == 0 || duration==0) {
52             return 0;
53         }
54         float power = (float) energy_my_one_to_many/duration_one_to_many;
55         if (power==0) {
56             return -1;
57         }
58         return power;
59     }
60 };
61 #endif //INC_3_TASKTYPE_ESTIMATOR_MEASURERESULT_H
62
63
64 long unsigned readEnergy_UJ() {
65     FILE *filePointer;
66     filePointer = popen("cat /sys/class/powercap/intel-rapl/intel-rapl\\:0/energy_uj", "r");
67
68     long unsigned energy_ui=0;
69
70     fscanf(filePointer, "%lu", &energy_ui);
71     int status = pclose(filePointer);
72     if (WIFEXITED(status)) {
73         int exit_status = WEXITSTATUS(status);
74         if (exit_status == 1) {
75             std::cout << RED << "Fehler beim Lesen von intel-rapl/energy_uj! Root-User? "<< RESET;
76         }
77     }
78     return energy_ui;
79 }
80
81 long unsigned readEnergy_UJ_secure() {
82     long unsigned energy_ui = readEnergy_UJ();
83     long unsigned new_energy_ui = readEnergy_UJ();
84     while (new_energy_ui == energy_ui) {
85         new_energy_ui = readEnergy_UJ();
86     }
87     return new_energy_ui;
88 }
89
90
91 MeasureResult runAndMeasureScript(const char* script) {
92     uint64_t timestamp_begin = timeSinceEpochMillisec();
93     long long counter_begin = readEnergy_UJ_secure();
94     std::thread t1(runCommand, script);
95     t1.join();
96     long long counter_end = readEnergy_UJ_secure();
97     uint64_t timestamp_end = timeSinceEpochMillisec();

```

```

98     uint64_t duration = timestamp_end - timestamp_begin;
99     std::cout << "Dauer: " << duration << " MS" << std::endl;
100     long long counter_diff = counter_end - counter_begin;
101     long long energy_mj = counter_diff;
102     std::cout << "Leistungsaufnahme in Mikojoule: " << (energy_mj) << std::endl;
103
104     MeasureResult result;
105     result.duration=duration;
106     result.energy_mj=energy_mj;
107
108     logMeasure(script, duration, energy_mj);
109     logMeasureFlush();
110     return result;
111 }

```

Listing A.19: Messung

A.2.18 Messung der Leistungsaufnahme von Anwendungstasks

```

1  MeasureResult measureAppTask(std::string apptaskname, std::string cpufreq, std::string cores) {
2      char* filename = searchTasktypeFile(apptaskname, foldername_generated_scripts_apptasks);
3
4
5      std::string fileWithParam = getFilenameWithParam(filename, cores);
6
7      std::thread t1(runCommand, fileWithParam.c_str());
8      t1.join();
9      char* measureFilename = searchTasktypeFile(apptaskname, folder_measure);
10
11     MeasureResult result;
12     result.parallelism=-1;
13     result.duration=-1;
14     result.energy_mj=-1;
15     result.taskname=apptaskname;
16     result.cpuFreq=cpufreq;
17
18     if (measureFilename!=NULL) {
19         std::ifstream measureAppTaskFile(measureFilename);
20         std::string line;
21
22         while (std::getline(measureAppTaskFile, line)) {
23             if (line.empty()) {
24                 continue;
25             }
26
27             size_t delimiterPos = line.find('=');
28             if (delimiterPos == std::string::npos) {
29                 continue;
30             }
31
32             std::string keyOfCurrentLine = (delimiterPos != 0) ? line.substr(0, delimiterPos) : NULL;
33             std::string valueOfCurrentLine = (delimiterPos != 0) ? line.substr(delimiterPos + 1) : NULL;
34
35             if (keyOfCurrentLine=="parallelism") {
36                 result.parallelism=valueOfCurrentLine;
37             } else if (keyOfCurrentLine=="duration") {
38                 result.duration=stoi(valueOfCurrentLine);
39             } else if (keyOfCurrentLine=="counter_diff") {
40                 result.energy_mj=stoll(valueOfCurrentLine);
41             }
42         }
43         measureAppTaskFile.close();
44         logMeasure(apptaskname.c_str(), result.duration, result.energy_mj);
45         logMeasureNewLine();
46     }
47
48     return result;
49 }

```

Listing A.20: Messung der Anwendungstasks

A.2.19 Aufruf von Anwendungstasks in edgedetection

```

1
2  int main(int argc, char *argv[])
3  {
4      char *listfilename;
5      char filename[64];
6      FILE *fp;

```

```

7   MyIMG *imgrgb, *imggh;
8
9   listfilename = argv[1];
10  fp = fopen(listfilename, "r");
11  imgrgb = createimagergb(XWIDTH, YWIDTH);
12  imggh = createimage(XWIDTH, YWIDTH);
13  int parallelism = (argc>2) ? atoi(argv[2]) : 1;
14  char* taskname = (argc>3) ? argv[3] : NULL;
15
16  printf("Parallelitätsgrad: %d\n", parallelism);
17  printf("Task: %s\n", taskname);
18  char filenames[23][64];
19  int imageCounter = 0;
20  while(fgets(filename, sizeof(filename), fp) != NULL) {
21      filename[strlen(filename)] = 0;
22      strcpy(filenames[imageCounter], filename );
23      imageCounter++;
24  }
25
26  runEdgedetection(taskname, parallelism, filenames[0], imgrgb, imggh);
27  return 0;
28 }
29
30 void runEdgedetection(char* taskname, int parallelism, char* filename, MyIMG* imgrgb, MyIMG* imggh) {
31     MyIMG *imgv;
32
33     FunctionPtrWithImgrgbAndImg gs = greyscale;
34     FunctionPtrWithImg cc = checkcontrast;
35     FunctionPtrWithImg sc = sharpencontrast;
36     FunctionPtrWithImgrgbAndImg ci = copyimage;
37     FunctionPtrWithImg sh = sobel;
38     FunctionPtrWithImg sv = sobel;
39     FunctionPtrWithImgrgbAndImg combine = combineimgs;
40     FunctionPtrWithImgAndChar wi = writeimage;
41     FunctionPtrWithImgAndChar li = loadimage;
42
43     ThreadParams params;
44     params.taskname=taskname;
45     params.paramImgh=imggh;
46     params.paramFilename=filename;
47     params.paramImgrgb=imgrgb;
48     params.imgv=imgv;
49     params.timestamp_begin=0;
50
51     if (withPThread) {
52         measureAppTaskWithPThread(taskname, parallelism, params);
53         smton();
54         return;
55     }
56
57     if (taskname==NULL) {
58         callFunctionWithImgAndChar(task_loadimage, li, parallelism, imgrgb,filename);
59         callFunctionWithImgrgbAndImg(task_greyscale, gs, parallelism, imgrgb,imggh);
60         callFunctionWithImg(task_checkcontrast, cc, parallelism, imggh);
61         callFunctionWithImg(task_sharpencontrast, sc, parallelism, imggh);
62         callFunctionWithImgrgbAndImg(task_copyimage, ci, parallelism, &imgv,imggh);
63         callFunctionWithImg(task_sobel, sh, parallelism, imggh);
64         callFunctionWithImg(task_sobel, sv, parallelism, imggh);
65         callFunctionWithImgrgbAndImg(task_combineimgs, combine,parallelism, imggh,imgv);
66         callFunctionWithImgAndChar(task_writeimage, wi, parallelism, imggh,filename);
67     } else if (strcmp(taskname, task_greyscale) == 0) {
68         loadimage(imgrgb,filename);
69         callFunctionWithImgrgbAndImg(taskname, gs, parallelism, imgrgb,imggh);
70     } else if (strcmp(taskname, task_checkcontrast) == 0) {
71         loadimage(imgrgb,filename);
72         callFunctionWithImg(taskname, cc, parallelism, imggh);
73     } else if (strcmp(taskname, task_sharpencontrast) == 0) {
74         loadimage(imgrgb,filename);
75         callFunctionWithImg(taskname, sc, parallelism, imggh);
76     } else if (strcmp(taskname, task_copyimage) == 0) {
77         loadimage(imgrgb,filename);
78         callFunctionWithImgrgbAndImg(taskname, ci, parallelism, &imgv,imggh);
79     } else if (strcmp(taskname, task_sobel) == 0) {
80         loadimage(imgrgb,filename);
81         callFunctionWithImg(taskname, sh, parallelism, imggh);
82     } else if (strcmp(taskname, task_sobel) == 0) {
83         loadimage(imgrgb,filename);
84         callFunctionWithImg(taskname, sv, parallelism, imggh);
85     } else if (strcmp(taskname, task_combineimgs) == 0) {
86         loadimage(imgrgb,filename);
87         copyimage(&imgv,imggh);
88         callFunctionWithImgrgbAndImg(taskname, combine, parallelism, imggh,imgv);
89     } else if (strcmp(taskname, task_writeimage) == 0) {
90         loadimage(imgrgb,filename);
91         callFunctionWithImgAndChar(taskname, wi, parallelism, imggh,filename);
92     } else if (strcmp(taskname, task_loadimage) == 0) {

```

```

94     callFunctionWithImgAndChar(task_loadimage, li, parallelism, imgrgb,filename);
95 }
96     smton();
97 }

```

Listing A.21: Aufruf von Anwendungstasks in edgedetection

A.2.20 Messung des Energieverbrauchs in edgedetection mit Funktionszeiger

```

1
2 void smtoff() {
3     system("echo off > /sys/devices/system/cpu/smt/control");
4 }
5
6 void prepareParallelism(int parallelism) {
7     printf("dyn: %d\n", omp_get_dynamic());
8     printf("num_treads: %d\n", omp_get_num_threads());
9     smtoff();
10 }
11
12 bool checkDuration(unsigned long long timestamp_begin) {
13     unsigned long long timestamp_end = millisecondsSinceEpoch();
14     unsigned long long duration = timestamp_end - timestamp_begin;
15     return duration > 5000;
16 }
17
18 long unsigned readEnergy_UJ() {
19     FILE *filePointer;
20     filePointer = popen("cat /sys/class/powercap/intel-rapl/intel-rapl\\:0/energy_uj", "r");
21
22     long unsigned energy_ui=0;
23
24     fscanf(filePointer, "%lu", &energy_ui);
25     pclose(filePointer);
26     return energy_ui;
27 }
28
29 void callFunctionWithImgrgbAndImg(char* taskname, FunctionPtrWithImgrgbAndImg functionPtrWithImgrgbAndImg, int parallelism, MyIMG*
    imgrgb,MyIMG* img) {
30     prepareParallelism(parallelism);
31     unsigned long long timestamp_begin = millisecondsSinceEpoch();
32     long long counter_begin = readEnergy_UJ();
33     bool stop = false;
34
35     #pragma omp parallel num_threads(parallelism) shared(timestamp_begin, stop)
36     {
37         #pragma omp for
38         for (int j = 0; j < parallelism; j++) {
39             int counter = 1;
40             while(!stop) {
41                 printf("OMP-Lauf=%d, CPU %d / Durchgang#%d, Thread %d\n", j, sched_getcpu(), counter++, omp_get_thread_num
42                     ());
43                 functionPtrWithImgrgbAndImg(imgrgb, img);
44                 stop=checkDuration(timestamp_begin);
45             }
46         }
47     };
48     long long counter_end = readEnergy_UJ();
49     unsigned long long timestamp_end = millisecondsSinceEpoch();
50
51     long long counter_diff = counter_end - counter_begin;
52     unsigned long long duration = timestamp_end - timestamp_begin;
53     printf("dauer %llu, zähler %llu, power %llu\n", duration, counter_diff, counter_diff/duration);
54
55     mkdir("measure", 0777);
56     char* filename = (char*) malloc(sizeof(char) * 30);
57     sprintf(filename, "%s%s.log", "measure/", taskname);
58     FILE* log = fopen(filename, "w");
59     fprintf(log, "parallelism=%d\n", parallelism);
60     fprintf(log, "duration=%lld\n", duration);
61     fprintf(log, "counter_diff=%lld\n", counter_diff);
62     fclose(log);
63     free(filename);
64 }

```

Listing A.22: Messung der Leistungsaufnahme mit Funktionszeiger

A.2.21 Parallelisierung von edgedetection mit PThread

Anstelle von OMP kann die parallele Verarbeitung auch mit PThread erfolgen. Hierfür ist die boolsche Variable withPThread auf true zu setzen. Wenn gesetzt, dann würde bei Ausführung von edgedetection eine parallele Verarbeitung mit PThreads ausgeführt werden.

```
1 bool withPThread = false; // try pthread for more cores
2
3 void runEdgedetection(char* taskname, int parallelism, char* filename, MyIMG* imgrgb, MyIMG* imgh) {
4     ...
5     if (withPThread) {
6         measureAppTaskWithPThread(taskname, parallelism, params);
7         smton();
8         return;
9     }
10    ...
11 }
12
13 void measureAppTaskWithPThread(char* taskname, int parallelism, ThreadParams params) {
14     params = prepareTask(params);
15     prepareParallelism(parallelism);
16     unsigned long long timestamp_begin = millisecondsSinceEpoch();
17     long long counter_begin = readEnergy_UJ();
18     params.timestamp_begin=timestamp_begin;
19
20     pthread_t threads[parallelism];
21     for (int j = 0; j < parallelism; j++) {
22         pthread_create(&threads[j], NULL, thread_function, (void *)&params);
23     }
24     for (int j = 0; j < parallelism; j++) {
25         pthread_join(threads[j], NULL);
26     }
27
28     long long counter_end = readEnergy_UJ();
29     unsigned long long timestamp_end = millisecondsSinceEpoch();
30
31     long long counter_diff = counter_end - counter_begin;
32     unsigned long long duration = timestamp_end - timestamp_begin;
33     printf("dauer %llu, zähler %llu, power %llu\n", duration, counter_diff, counter_diff/duration);
34
35     mkdir("measure", 0777);
36     char* filename = (char*) malloc(sizeof(char) * 30);
37     sprintf(filename, "%s%s.log", "measure/", taskname);
38     FILE* log = fopen(filename, "w");
39     fprintf(log, "parallelism=%d\n", parallelism);
40     fprintf(log, "duration=%lld\n", duration);
41     fprintf(log, "counter_diff=%lld\n", counter_diff);
42     fclose(log);
43     free(filename);
44 }
```

Listing A.23: Parallelisierung mit PThread

A.2.22 Schätzung der Leistungsaufnahme im Tasktypeestimator

```
1
2
3 MeasureResult measureAppTask(std::string apptaskname, std::string cpufreq, std::string cores) {
4     char* filename = searchTasktypeFile(apptaskname, foldername_generated_scripts_apptasks);
5
6     std::string fileWithParam = getFilenameWithParam(filename, cores);
7
8     std::thread t1(runCommand, fileWithParam.c_str());
9     t1.join();
10    char* measureFilename = searchTasktypeFile(apptaskname, folder_measure);
11
12    MeasureResult result;
13    result.parallelism=-1;
14    result.duration=-1;
15    result.energy_mj=-1;
16    result.taskname=apptaskname;
17    result.cpuFreq=cpufreq;
18
19    if (measureFilename!=NULL) {
20        std::ifstream measureAppTaskFile(measureFilename);
21        std::string line;
22
23        while (std::getline(measureAppTaskFile, line)) {
24            if (line.empty()) {
25                continue;
26            }
27        }
28    }
```

```

27     size_t delimiterPos = line.find('=');
28     if (delimiterPos == std::string::npos) {
29         continue;
30     }
31
32     std::string keyOfCurrentLine = (delimiterPos != 0) ? line.substr(0, delimiterPos) : NULL;
33     std::string valueOfCurrentLine = (delimiterPos != 0) ? line.substr(delimiterPos + 1) : NULL;
34
35     if (keyOfCurrentLine=="parallelism") {
36         result.parallelism=valueOfCurrentLine;
37     } else if (keyOfCurrentLine=="duration") {
38         result.duration=stoi(valueOfCurrentLine);
39     } else if (keyOfCurrentLine=="counter_diff") {
40         result.energy_mj=stoll(valueOfCurrentLine);
41     }
42 }
43
44 measureAppTaskFile.close();
45 logMeasure(apptaskname.c_str(), result.duration, result.energy_mj);
46 logMeasureNewLine();
47 }
48
49 return result;
50 }
51
52 MeasureResult estimateAppTask(std::string apptaskname, std::string cpufreq, std::string cores, int repeat) {
53     std::string oneToOneTaskname = readOneToOneMapping(apptaskname);
54     char* filenameOneToOne = searchTasktypeFile(oneToOneTaskname, foldername_generated_scripts_tasktypes_from_folder);
55
56     std::string filenameOneToOneWithParam = getFilenameWithParam(filenameOneToOne, cores);
57
58     MeasureResult result;
59     if (measureResultMaps[oneToOneTaskname][cpufreq][cores].find(repeat) != measureResultMaps[oneToOneTaskname][cpufreq][cores].
60         end()) {
61         result = measureResultMaps[oneToOneTaskname][cpufreq][cores][repeat];
62     } else {
63         result = runAndMeasureScript(filenameOneToOneWithParam.c_str());
64         result.taskname=oneToOneTaskname;
65         result.cpuFreq=cpufreq;
66         result.parallelism=cores;
67         measureResultMaps[oneToOneTaskname][cpufreq][cores][repeat]=result;
68     }
69
70     char* filenameOneToMany = searchTasktypeFile(apptaskname, foldername_generated_scripts_tasktypes_onetomany);
71     std::string filenameOneToManyWithParam = getFilenameWithParam(filenameOneToMany, cores);
72
73     MeasureResult resultOneToMany = runAndMeasureScript(filenameOneToManyWithParam.c_str());
74     result.duration_one_to_many = resultOneToMany.duration;
75     result.energy_my_one_to_many = resultOneToMany.energy_mj;
76     logMeasureFlush();
77
78     return result;
79 }
80
81 void startApptaskEstimation(std::string apptaskname, int repeat) {
82     MeasureResult result = estimateAppTask(apptaskname, currentCPUFreq, currentParallelism, repeat);
83     MeasureResult result = estimateAppTask(apptaskname, currentCPUFreq, currentParallelism);
84     MeasureResult appTaskresult = measureAppTask(apptaskname, currentCPUFreq, currentParallelism);
85     logMeasureNewLine();
86     logMeasureFlush();
87
88     resultFile << currentCPUFreq << " ";
89     resultFile << currentParallelism << " ";
90     resultFile << apptaskname << " ";
91     resultFile << appTaskresult.duration << " MS; ";
92     resultFile << appTaskresult.power() << " MJ/MS:\t";
93     resultFile << result.power() << " MJ/MS; ";
94     if (appTaskresult.power() != 0) {
95         resultFile << (result.power() - appTaskresult.power()) * 100 / appTaskresult.power() << "%;\t";
96         resultFile << result.powerOneToMany() << " MJ/MS; ";
97         resultFile << (result.powerOneToMany() - appTaskresult.power()) * 100 / appTaskresult.power() << "%; ";
98         resultFile << "\n";
99     } else {
100         resultFile << 0 << "%;\t";
101         resultFile << result.powerOneToMany() << " MJ/MS; ";
102         resultFile << 0 << "%;\t";
103         resultFile << "\n";
104     }
105     resultFile.flush();
106 }
107
108 void repeatEstimationsForAppTask(std::string apptaskname, int repeats) {
109     for (std::string cpuFreq : cpuFrequencyVektor) {
110         setupCpuFrequenzlevel(cpuFreq);
111
112         for (std::string cores : parallelismVektor) {

```



```

113         setCurrentParallelism(cores);
114
115         for (int i = 0; i < repeats; i++) {
116             startApptaskEstimation(apptaskname, i);
117         }
118         logEmptyline();
119     }
120 }
121 }
122
123 void startEstimation(int repeats) {
124     smtOff();
125     openResultFile();
126     logHeadline();
127     openMeasurLogFile();
128
129
130     for (std::string apptaskname: apptypeVektor) {
131         repeatEstimationsForAppTask(apptaskname, repeats);
132     }
133
134     closeMeasureLogFile();
135     closeResultFile();
136     smtOn();
137 }

```

Listing A.24: Schätzung der Leistungsaufnahme

Anhang B

Ergänzungen zu den Ergebnissen

B.1 Prototypische Tasks

Das Listing B.1 zeigt eine Übersicht zu den Sequenzdateien, die auf dem Testsystem generiert wurden. Die Übersicht zeigt außerdem, aus wievielen Befehlen ein Task-Typ besteht.

```
1 Der Task nop enthält 1999 Einträge
2 Der Task vconvert_SIMD enthält 2090 Einträge
3 Der Task imov enthält 1996 Einträge
4 Der Task imem enthält 2080 Einträge
5 Der Task icompare enthält 2095 Einträge
6 Der Task branch enthält 1997 Einträge
7 Der Task bitbyte enthält 2037 Einträge
8 Der Task v1x4smul_SIMD enthält 2287 Einträge
9 Der Task dmem enthält 2091 Einträge
10 Der Task dsub64_SIMD enthält 2114 Einträge
11 Der Task dadd enthält 2062 Einträge
12 Der Task m4x4smul_SIMD enthält 2328 Einträge
13 Der Task dmul64_SIMD enthält 2114 Einträge
14 Der Task dmul enthält 2062 Einträge
15 Der Task m4x4smul enthält 3199 Einträge
16 Der Task iadd enthält 2039 Einträge
17 Der Task logic enthält 2036 Einträge
18 Der Task imul enthält 2071 Einträge
19 Der Task dmem64_SIMD enthält 2081 Einträge
20 Der Task ssub32_SIMD enthält 2114 Einträge
21 Der Task vmov_SIMD enthält 2157 Einträge
22 Der Task smul32_SIMD enthält 2114 Einträge
23 Der Task shift enthält 1996 Einträge
```

Listing B.1: Umfang der Sequenzen von Task-Typen

B.2 Tasks einer Anwendung

```
1 Der Task sobelv enthält 17432 Einträge
2 Der Task checkcontrast enthält 1922 Einträge
3 Der Task copyimage enthält 3590 Einträge
4 Der Task combineimgs enthält 8295 Einträge
5 Der Task sharpencontrast enthält 17599 Einträge
6 Der Task loadimage enthält 4951 Einträge
7 Der Task sobelh enthält 17432 Einträge
8 Der Task writeimage enthält 5340 Einträge
9 Der Task greyscale enthält 18296 Einträge
```

Listing B.2: Umfang der Sequenzen von Anwendungstasks

B.3 Abbildung der Anwendung auf Task-Typen

B.3.1 taskmap.txt

```
1 [OneToOne]
2 sobelv=m4x4smul
3 checkcontrast=m4x4smul_SIMD
4 copyimage=m4x4smul
5 testWithNopBitByteShiftIMulLogic=nop
6 testWithNopAndBitByteAndShift=nop
7 testWithNopAndBitByte=nop
8 combineimgs=imul
9 sharpencontrast=vmov_SIMD
10 loadimage=dmem64_SIMD
11 sobelh=m4x4smul
12 writeimage=dmem
13 greyscale=m4x4smul
14 testWithNop=nop
15
16 [OneToMany]
17 model=sobelv
18   branch=0.100000
19   dmem=0.100000
20   m4x4smul=0.500000
21   vmov_SIMD=0.300000
22 end_model
23 model=checkcontrast
24   dmem64_SIMD=0.500000
25   shift=0.500000
26 end_model
27 model=copyimage
28   imem=0.330000
29   m4x4smul=0.330000
30   shift=0.330000
31 end_model
32 model=testWithNopAndBitByteAndShift
33   bitbyte=0.330000
34   nop=0.330000
35   shift=0.330000
36 end_model
37 model=testWithNopAndBitByte
38   bitbyte=0.500000
39   nop=0.500000
40 end_model
41 model=combineimgs
42   branch=0.400000
43   m4x4smul=0.400000
44   vmov_SIMD=0.200000
45 end_model
46 model=sharpencontrast
47   bitbyte=0.100000
48   imul=0.100000
49   m4x4smul=0.200000
50   vmov_SIMD=0.600000
51 end_model
52 model=loadimage
53   imem=0.250000
54   imov=0.250000
55   logic=0.250000
56   shift=0.250000
57 end_model
58 model=sobelh
59   branch=0.100000
60   dmem=0.100000
61   m4x4smul=0.500000
62   vmov_SIMD=0.300000
63 end_model
64 model=writeimage
65   imem=0.250000
66   imov=0.250000
67   logic=0.250000
68   shift=0.250000
69 end_model
70 model=greyscale
71   dadd=0.200000
72   dmem=0.100000
73   m4x4smul=0.600000
74   vmov_SIMD=0.100000
75 end_model
76 model=testWithNop
77   nop=1.000000
78 end_model
79 model=testWithNopBitByteShiftIMulLogic
80   bitbyte=0.200000
81   imul=0.200000
```

```

82 logic=0.200000
83 nop=0.200000
84 shift=0.200000
85 end_model

```

Listing B.3: Inhalt der Ergebnisdatei taskmap.txt

B.4 Leistungsaufnahme der CPU anhand von Task-Typen

B.4.1 Ergebnisdatei zur Schätzung der Leistungsaufnahme vom 24.06.2024

```

1 CPUFrequency;Parallelism;apptask;apptaskduration (MS); apptaskpower (MJ/MS); estimationOneToOne (MJ/MS); diffOneToOne (%);
  estimationOneToMany (MJ/MS); diffOneToMany (%)
2 1500;1;checkcontrast;5034;2711.72;2382.29;-12.1482; 2061.73;-23.9695;
3 1500;1;checkcontrast;5133;2562.19;12882.7;402.801; 10861.8;323.927;
4 1500;1;checkcontrast;5033;2537.77;12924.7;409.294; 11076.1;336.451;
5 1500;1;checkcontrast;5042;2685.53;13303.2;395.365; 10929.3;306.97;
6 1500;1;checkcontrast;5140;2697.19;13126.1;386.658; 10888.4;303.694;
7
8 1500;2;checkcontrast;5110;2726.13;20633.3;656.871; 17587.3;545.138;
9 1500;2;checkcontrast;5111;2699.94;21830.4;708.552; 17630.1;552.982;
10 1500;2;checkcontrast;5004;2690.47;21550.1;700.98; 17699.7;557.867;
11 1500;2;checkcontrast;5049;2701.33;21871.8;709.67; 17733.3;556.466;
12 1500;2;checkcontrast;5098;2717.55;20977.2;671.916; 17741.6;552.85;
13
14 1500;3;checkcontrast;5131;2731.37;25206.1;822.836; 20847.6;663.265;
15 1500;3;checkcontrast;5069;2741.45;26308.1;859.643; 20857.8;660.831;
16 1500;3;checkcontrast;5135;2771.26;26407.3;852.896; 20774.6;649.644;
17 1500;3;checkcontrast;5094;2761.25;25773.6;833.401; 21189.8;667.397;
18 1500;3;checkcontrast;5174;2713.14;25812.8;851.397; 21233.3;682.607;
19
20 1500;4;checkcontrast;5034;2878.03;33231.4;1054.66; 27794.4;865.744;
21 1500;4;checkcontrast;5176;2939.93;33647.1;1044.49; 28068.5;854.734;
22 1500;4;checkcontrast;5046;2931.78;32958.1;1024.17; 28099.1;858.43;
23 1500;4;checkcontrast;5155;2943.46;33164.4;1026.71; 28265.9;860.294;
24 1500;4;checkcontrast;5165;2983.99;34293.9;1049.26; 28012.3;838.752;
25
26 2000;1;checkcontrast;5142;3663.01;3424.71;-6.50566; 2818.51;-23.0549;
27 2000;1;checkcontrast;5144;3690;3489.88;-5.42321; 2720.7;-26.2681;
28 2000;1;checkcontrast;5069;3409.48;3397.64;-0.347284; 2728.67;-19.9682;
29 2000;1;checkcontrast;5069;3678.62;3402.14;-7.51585; 2712.1;-26.2741;
30 2000;1;checkcontrast;5143;3654.86;3233.3;-11.5341; 2735.03;-25.1674;
31
32 2000;2;checkcontrast;5117;3407.87;4772.93;40.0561; 4043.98;18.666;
33 2000;2;checkcontrast;5139;3667.59;4767.36;29.9861; 4000.97;9.09005;
34 2000;2;checkcontrast;5106;3397.44;4766.78;40.3053; 4001.93;17.7927;
35 2000;2;checkcontrast;5047;3558.97;4749.92;33.4634; 4008.72;12.6371;
36 2000;2;checkcontrast;5125;3634.63;4768.19;31.1881; 3998.55;10.0126;
37
38 2000;3;checkcontrast;5071;3650.71;6491.5;77.8145; 5340.05;46.274;
39 2000;3;checkcontrast;5059;3588.02;6026.18;67.9526; 5351.21;49.141;
40 2000;3;checkcontrast;5151;3661.66;6749.23;84.3214; 5364.26;46.498;
41 2000;3;checkcontrast;5044;3624.72;6421.56;77.1601; 5397.59;48.9104;
42 2000;3;checkcontrast;5117;3652.01;6439.61;76.3306; 5368.22;46.9936;
43
44 2000;4;checkcontrast;5148;3650.2;8098.59;121.867; 5402.55;48.0071;
45 2000;4;checkcontrast;5161;3654.13;8174.88;123.716; 6706.9;83.5431;
46 2000;4;checkcontrast;5028;3670.51;8093.41;120.498; 6590.23;79.5452;
47 2000;4;checkcontrast;5075;3650.98;8127.22;122.604; 6239.5;70.8996;
48 2000;4;checkcontrast;5149;3688.91;8006.81;117.051; 6713.68;81.9962;
49
50 2200;1;checkcontrast;5023;3958.06;3670.17;-7.27366; 3122.47;-21.1112;
51 2200;1;checkcontrast;5093;3938.25;3665.7;-6.92069; 3100.3;-21.2772;
52 2200;1;checkcontrast;5026;4198.48;3926.72;-6.47281; 3080.21;-26.6352;
53 2200;1;checkcontrast;5027;3933.79;3926.49;-0.185598; 3084.34;-21.5937;
54 2200;1;checkcontrast;5096;4148.79;3917.84;-5.56666; 3059.93;-26.2452;
55
56 2200;2;checkcontrast;5093;4148.64;5862.31;41.3067; 4677.43;12.7461;
57 2200;2;checkcontrast;5070;3947.57;5844.46;48.0522; 4693.6;18.8984;
58 2200;2;checkcontrast;5008;4144.96;5872.78;41.685; 4689.75;13.1436;
59 2200;2;checkcontrast;5073;4137.88;5848.86;41.3492; 4642.98;12.2069;
60 2200;2;checkcontrast;5088;3948.91;5870.96;48.673; 4696.84;18.9402;
61
62 2200;3;checkcontrast;5114;4208.04;7919.46;88.1984; 6326.41;50.3412;
63 2200;3;checkcontrast;5045;4203.47;7876.48;87.3807; 6322.4;50.4093;
64 2200;3;checkcontrast;5008;4215.56;7954.84;88.7019; 6336.15;50.3038;
65 2200;3;checkcontrast;5047;4225.17;7367.27;74.3665; 6258.33;48.1204;
66 2200;3;checkcontrast;5104;4214.74;7445.71;76.6589; 6326.89;50.1133;
67
68 2200;4;checkcontrast;5123;4221.05;9631.79;128.185; 7951.86;88.3856;

```

69 2200;4;checkcontrast;5046;4202.83;9556.34;127.379; 7974.88;89.7505;
70 2200;4;checkcontrast;5143;4230.25;9729;129.986; 8036.52;89.9775;
71 2200;4;checkcontrast;5157;4252.46;9534.54;124.212; 7993.96;87.9844;
72 2200;4;checkcontrast;5098;4222.41;9718.41;130.163; 8052.95;90.7195;
73
74 2400;1;checkcontrast;5114;4524.38;4256.43;-5.92235; 3561.42;-21.2839;
75 2400;1;checkcontrast;5053;4875.36;4256.39;-12.6959; 3562.34;-26.9318;
76 2400;1;checkcontrast;5119;4865.75;4269.96;-12.2445; 3590.21;-26.2146;
77 2400;1;checkcontrast;5054;4861.28;4522.84;-6.96187; 3585.9;-26.2355;
78 2400;1;checkcontrast;5120;4850.67;4445.11;-8.36106; 3542.7;-26.9647;
79
80 2400;2;checkcontrast;5073;4840.33;6962.57;43.8449; 5444.23;12.4764;
81 2400;2;checkcontrast;5119;4842.07;6954;43.6162; 5429.44;12.1306;
82 2400;2;checkcontrast;5039;4864.58;6570.02;35.0583; 5444.45;11.9202;
83 2400;2;checkcontrast;5039;4868.71;6579.28;35.1338; 5456.85;12.0799;
84 2400;2;checkcontrast;5117;4867.12;6555.7;34.6937; 5453.26;12.043;
85
86 2400;3;checkcontrast;5024;4853.9;9257.22;90.7172; 7400.17;52.4583;
87 2400;3;checkcontrast;5118;4883.66;9370.25;91.8696; 7420.17;51.9388;
88 2400;3;checkcontrast;5044;4858.82;8989.46;85.0133; 7507.52;54.5134;
89 2400;3;checkcontrast;5039;4887.81;8836.26;80.7813; 7454.34;52.5087;
90 2400;3;checkcontrast;5031;4854.97;9003.37;85.4465; 7545.57;55.4195;
91
92 2400;4;checkcontrast;5114;4857.32;9712.68;99.9596; 9387.7;93.2692;
93 2400;4;checkcontrast;5094;4869.99;11247.9;130.963; 9472.73;94.5123;
94 2400;4;checkcontrast;5125;4876.11;11292.2;131.582; 9444.9;93.6976;
95 2400;4;checkcontrast;5094;4842.17;11274.2;132.834; 9510.73;96.4147;
96 2400;4;checkcontrast;5106;4851.64;11243.7;131.752; 9250.67;90.6711;
97
98 2500;1;checkcontrast;5101;5207.63;4577.64;-12.0976; 3896.24;-25.1822;
99 2500;1;checkcontrast;5102;4938.85;4889.35;-1.00228; 3886.38;-21.31;
100 2500;1;checkcontrast;5040;4920.54;4801.18;-2.42571; 3840.56;-21.9483;
101 2500;1;checkcontrast;5105;5187.75;4758.87;-8.26708; 3829.33;-26.1851;
102 2500;1;checkcontrast;5043;4848.55;4763.87;-1.74658; 3814.29;-21.3314;
103
104 2500;2;checkcontrast;5102;5177.62;7366.22;42.2704; 5927.66;14.4863;
105 2500;2;checkcontrast;5049;5178.29;7378.51;42.4893; 5904.51;14.0243;
106 2500;2;checkcontrast;5118;5189.22;7112.3;37.0593; 5917.86;14.0414;
107 2500;2;checkcontrast;5037;5178.92;7126.2;37.6001; 5929.32;14.4894;
108 2500;2;checkcontrast;5048;5202.08;7522.52;44.6058; 5903.64;13.486;
109
110 2500;3;checkcontrast;5052;5188.35;7790.01;50.1442; 8071.58;55.5713;
111 2500;3;checkcontrast;5087;5185.12;10030.4;93.4467; 8140.1;56.9895;
112 2500;3;checkcontrast;5101;5209.42;10032.7;92.588; 8139.47;56.2453;
113 2500;3;checkcontrast;5036;5202.76;9905.62;90.3917; 8197.94;57.569;
114 2500;3;checkcontrast;5080;5212.06;10199.3;95.6861; 8206.8;57.4578;
115
116 2500;4;checkcontrast;5035;5210.05;12266.4;135.436; 10297.3;97.643;
117 2500;4;checkcontrast;5017;5301.61;12363.3;133.199; 10306.3;94.3988;
118 2500;4;checkcontrast;5090;5363.07;12460.2;132.332; 10398.7;93.895;
119 2500;4;checkcontrast;5032;5290.04;12389.6;134.206; 10441.2;97.3739;
120 2500;4;checkcontrast;5109;5242.82;12390.5;136.333; 10332;97.0697;
121
122 3000;1;checkcontrast;5036;7400.97;6382.01;-13.7679; 5447.58;-26.3937;
123 3000;1;checkcontrast;5083;6947.57;6823.67;-1.78329; 5431.57;-21.8206;
124 3000;1;checkcontrast;5071;7407.73;6700.17;-9.55162; 5442.04;-26.5357;
125 3000;1;checkcontrast;5083;7394.91;6827.75;-7.6697; 5426.12;-26.6236;
126 3000;1;checkcontrast;5033;7014.33;6777.46;-3.37697; 5447.57;-22.3366;
127
128 3000;2;checkcontrast;5037;7411.89;10727.7;44.7364; 8662.07;16.8672;
129 3000;2;checkcontrast;5045;7440.79;10170;36.6797; 8715.7;17.1341;
130 3000;2;checkcontrast;5090;7038.7;10931.8;55.3099; 8670.53;23.1837;
131 3000;2;checkcontrast;5086;7427.78;10394.3;39.9378; 8704.29;17.1856;
132 3000;2;checkcontrast;5040;7429.47;10360.2;39.4477; 8677.13;16.7934;
133
134 3000;3;checkcontrast;5062;6992.93;14684.3;109.988; 11992.4;71.4925;
135 3000;3;checkcontrast;5013;7394.76;14399;94.7185; 12042.1;62.8458;
136 3000;3;checkcontrast;5017;7446.77;14157.1;90.1111; 11890.6;59.674;
137 3000;3;checkcontrast;5053;7424.32;14182.3;91.0249; 12018.2;61.8766;
138 3000;3;checkcontrast;5072;7427.56;14309.1;92.6488; 11933.1;60.6595;
139
140 3000;4;checkcontrast;5080;7430.69;18511.3;149.119; 15038.2;102.38;
141 3000;4;checkcontrast;5048;7440.23;16285.5;118.884; 14759.6;98.3763;
142 3000;4;checkcontrast;5034;7510.06;18684.3;148.791; 14934.7;98.8624;
143 3000;4;checkcontrast;5080;7569.42;17661.1;133.322; 14823.5;95.834;
144 3000;4;checkcontrast;5049;7513.76;18657.3;148.308; 15114.4;101.156;
145
146 3500;1;checkcontrast;5025;9910.64;8950.01;-9.69293; 7722.44;-22.0793;
147 3500;1;checkcontrast;5021;10425.1;8909.35;-14.5397; 7569.78;-27.3892;
148 3500;1;checkcontrast;5064;10361.4;8845.99;-14.6253; 7550.21;-27.1311;
149 3500;1;checkcontrast;5020;10346.6;8819.36;-14.7609; 7525.69;-27.2642;
150 3500;1;checkcontrast;5064;10381.3;8821.98;-15.02; 7527.37;-27.4907;
151
152 3500;2;checkcontrast;5081;10318.7;14327.8;38.8519; 12122;17.4751;
153 3500;2;checkcontrast;5035;10374.5;14457.2;39.3525; 12198.8;17.5844;
154 3500;2;checkcontrast;5086;10465.4;14403.5;37.6296; 12195.6;16.533;
155 3500;2;checkcontrast;5047;10440.2;14485.7;38.7491; 12235.2;17.1927;

156 3500;2;checkcontrast;5088;10411.4;14054.3;34.9893; 12275.3;17.902;
157
158 3500;3;checkcontrast;5073;10394.2;20978.4;101.827; 16960.3;63.1707;
159 3500;3;checkcontrast;5070;10499;21525;105.019; 17016.2;62.0737;
160 3500;3;checkcontrast;5071;10627.3;21650.3;103.724; 17060.3;60.5332;
161 3500;3;checkcontrast;5061;10698;20847.1;94.8701; 17200;60.7785;
162 3500;3;checkcontrast;5029;10744.5;20879.2;94.3256; 17267;60.7061;
163
164 3500;4;checkcontrast;5052;10791.2;27146.9;151.565; 22210.2;105.818;
165 3500;4;checkcontrast;5030;10915;27313.8;150.241; 22186.4;103.265;
166 3500;4;checkcontrast;5061;10844.1;27404.2;152.71; 22314.5;105.775;
167 3500;4;checkcontrast;5029;10890.3;27688.8;154.251; 22318.5;104.938;
168 3500;4;checkcontrast;5036;10909.9;27432;151.441; 22273;104.154;
169
170 4000;1;checkcontrast;5039;15163.7;12343;-18.6016; 10848.2;-28.4592;
171 4000;1;checkcontrast;5053;15163.1;12576.8;-17.0566; 10785.9;-28.8676;
172 4000;1;checkcontrast;5020;15076.9;12447;-17.4431; 10689.8;-29.0984;
173 4000;1;checkcontrast;5031;14966.9;12381.5;-17.2742; 10690;-28.5761;
174 4000;1;checkcontrast;5047;14768.7;13017;-11.861; 10660.5;-27.8171;
175
176 4000;2;checkcontrast;5053;14720;20670.1;40.4222; 16820.8;14.272;
177 4000;2;checkcontrast;5011;14206.2;20186.9;42.0996; 17076.3;20.2033;
178 4000;2;checkcontrast;5004;15172.3;20245.4;33.4366; 17105.6;12.7424;
179 4000;2;checkcontrast;5055;15225;20227.5;32.857; 17174.5;12.8043;
180 4000;2;checkcontrast;5053;15165.7;20601.3;35.8412; 17175.6;13.2525;
181
182 4000;3;checkcontrast;5044;15082.7;25588;69.6515; 21422.8;42.0355;
183 4000;3;checkcontrast;5017;15149.3;26616.2;75.6922; 21612.6;42.6639;
184 4000;3;checkcontrast;5026;15216.2;27037.4;77.6884; 21605.6;41.9907;
185 4000;3;checkcontrast;5028;15404.1;26556.6;72.3997; 21657.7;40.5971;
186 4000;3;checkcontrast;5022;15117.3;26675.9;76.4595; 21599.9;42.8824;
187
188 4000;4;checkcontrast;5037;15476;34059.5;120.08; 28075.6;81.4145;
189 4000;4;checkcontrast;5084;15449.8;32847.6;112.609; 28282.9;83.0637;
190 4000;4;checkcontrast;5038;15412.9;32834.7;113.034; 27301.3;77.1333;
191 4000;4;checkcontrast;5080;15449.8;33703.2;118.146; 24646.9;59.5285;
192 4000;4;checkcontrast;5045;15375.2;32690.7;112.62; 25625.6;66.6685;
193
194 1500;1;loadimage;5003;4426.78;2736.91;-38.1736; 2076.26;-53.0978;
195 1500;1;loadimage;5005;4259.41;2153.44;-49.4428; 1961.15;-53.9574;
196 1500;1;loadimage;5004;4260.78;2172.01;-49.0232; 1942.98;-54.3984;
197 1500;1;loadimage;5005;4247.66;2137.23;-49.6845; 1944.26;-54.2276;
198 1500;1;loadimage;5005;4261.34;2150.94;-49.5244; 1953.04;-54.1685;
199
200 1500;2;loadimage;5005;5940.42;3110.51;-47.6382; 2650.29;-55.3855;
201 1500;2;loadimage;5004;5856.29;3094.14;-47.1654; 2673.82;-54.3427;
202 1500;2;loadimage;5004;5835.84;3096.58;-46.9387; 2667.84;-54.2852;
203 1500;2;loadimage;5006;5753.32;3085.22;-46.3749; 2638.04;-54.1476;
204 1500;2;loadimage;5005;5867.51;3100.9;-47.1513; 2634.83;-55.0945;
205
206 1500;3;loadimage;5003;6930.48;4100.73;-40.8305; 3146.1;-54.6049;
207 1500;3;loadimage;5004;6988.86;4077.41;-41.6585; 3396.01;-51.4082;
208 1500;3;loadimage;5031;6947.31;4095.82;-41.0446; 3388.84;-51.2208;
209 1500;3;loadimage;5004;6489.69;4230.88;-34.8061; 3404.4;-47.5415;
210 1500;3;loadimage;5003;6528.52;4187.08;-35.8649; 3435.43;-47.3782;
211
212 1500;4;loadimage;5041;6797.66;5182.73;-23.7572; 3501.99;-48.4824;
213 1500;4;loadimage;5019;6907.1;5134.87;-25.6581; 3940.74;-42.9465;
214 1500;4;loadimage;5033;6816.09;5126.48;-24.7886; 4186.47;-38.5797;
215 1500;4;loadimage;5032;6629.62;5057.05;-23.7204; 4220.65;-36.3365;
216 1500;4;loadimage;5016;6680.24;5132.22;-23.1731; 4223.29;-36.7793;
217
218 2000;1;loadimage;5003;5572.91;3040.36;-45.4439; 2470.73;-55.6653;
219 2000;1;loadimage;5004;5574.42;2846.15;-48.9427; 2471.96;-55.6553;
220 2000;1;loadimage;5003;5581.68;2859.86;-48.7635; 2442.42;-56.2422;
221 2000;1;loadimage;5004;5577.62;2834.81;-49.1753; 2461.13;-55.8749;
222 2000;1;loadimage;5005;5561.4;2844.92;-48.8452; 2434.54;-56.2244;
223
224 2000;2;loadimage;5004;7657.09;4254.89;-44.432; 3535.61;-53.8257;
225 2000;2;loadimage;5005;7933.2;4289.43;-45.9306; 3540.55;-55.3704;
226 2000;2;loadimage;5004;7676.38;4266.82;-44.4162; 3568.22;-53.5169;
227 2000;2;loadimage;5004;7979.01;4285.19;-46.2942; 3571.8;-55.2351;
228 2000;2;loadimage;5003;7626.83;4321.9;-43.3329; 3575.11;-53.1245;
229
230 2000;3;loadimage;5004;9293.03;5733.03;-38.3083; 4726.79;-49.1361;
231 2000;3;loadimage;5008;9409.13;5906.72;-37.2235; 4735.34;-49.6729;
232 2000;3;loadimage;5004;9944;4936.33;-50.3587; 4656.52;-53.1726;
233 2000;3;loadimage;5016;9642.79;5817.49;-39.6701; 4714.86;-51.1048;
234 2000;3;loadimage;5003;9929.45;5732.79;-42.2648; 4753.44;-52.1279;
235
236 2000;4;loadimage;5009;9915.43;7128.82;-28.1037; 5847.42;-41.027;
237 2000;4;loadimage;5013;9902.1;7196.89;-27.3196; 5760.86;-41.8218;
238 2000;4;loadimage;5017;10038.1;7189.37;-28.379; 5903.81;-41.1859;
239 2000;4;loadimage;5030;10026.8;7214.78;-28.0454; 5886.53;-41.2923;
240 2000;4;loadimage;5031;10035.1;7300.01;-27.2555; 5900.22;-41.2044;
241
242 2200;1;loadimage;5004;6439.06;3340.86;-48.1157; 2886.3;-55.1751;

243 2200;1;loadimage;5003;6442.81;3345.52;-48.0735; 2798.03;-56.5713;
244 2200;1;loadimage;5004;6358.74;3275.04;-48.4955; 2807.85;-55.8426;
245 2200;1;loadimage;5003;6390.4;3244.32;-49.2313; 2826.86;-55.7639;
246 2200;1;loadimage;5002;6401.51;3335.94;-47.8882; 2801.9;-56.2306;
247
248 2200;2;loadimage;5004;8911.44;5002.79;-43.861; 4139.28;-53.5509;
249 2200;2;loadimage;5003;8862.95;5074.13;-42.749; 4162.04;-53.04;
250 2200;2;loadimage;5004;8941.64;5064.04;-43.3657; 4191.37;-53.1253;
251 2200;2;loadimage;5003;9149.13;5087.52;-44.3933; 4170.42;-54.4173;
252 2200;2;loadimage;5003;9006.76;5100.82;-43.3667; 4163.08;-53.7783;
253
254 2200;3;loadimage;5003;11179;6834.75;-38.8607; 5508.18;-50.7274;
255 2200;3;loadimage;5004;11525.6;6823.35;-40.7982; 5511.42;-52.181;
256 2200;3;loadimage;5002;11462.6;6821.49;-40.489; 5601.76;-51.13;
257 2200;3;loadimage;5026;11621.2;6836.05;-41.1761; 5115.91;-55.9778;
258 2200;3;loadimage;5003;11694.5;6831.41;-41.5843; 5331.74;-54.4081;
259
260 2200;4;loadimage;5008;11356.7;8675.84;-23.6057; 6456.52;-43.1477;
261 2200;4;loadimage;5037;11498.4;8591.86;-25.278; 6659.6;-42.0825;
262 2200;4;loadimage;5029;11745.4;8632.95;-26.4995; 6674.59;-43.1728;
263 2200;4;loadimage;5027;11586.7;8712.62;-24.8051; 7037.23;-39.2647;
264 2200;4;loadimage;5031;11489.9;8873.77;-22.7691; 6788.64;-40.9166;
265
266 2400;1;loadimage;5003;7332.31;4049.35;-44.7739; 3351.59;-54.2901;
267 2400;1;loadimage;5004;7300.28;3860.76;-47.1149; 3343.59;-54.1992;
268 2400;1;loadimage;5003;7275.66;3800.05;-47.7704; 3333.64;-54.181;
269 2400;1;loadimage;5004;7242.75;3713.49;-48.7282; 3319.58;-54.1669;
270 2400;1;loadimage;5002;7242.56;3725.41;-48.5622; 3298.19;-54.461;
271
272 2400;2;loadimage;5003;10384.4;5955.33;-42.6514; 4942.21;-52.4076;
273 2400;2;loadimage;5004;10435.3;5943.55;-43.0437; 4943.68;-52.6254;
274 2400;2;loadimage;5002;10355.7;5940.68;-42.6338; 4928.73;-52.4057;
275 2400;2;loadimage;5002;10260.8;5941.9;-42.0915; 4937.15;-51.8836;
276 2400;2;loadimage;5004;10441.9;5945.17;-43.0642; 4926.61;-52.8187;
277
278 2400;3;loadimage;5004;13288.4;8181.31;-38.4326; 6610.92;-50.2504;
279 2400;3;loadimage;5010;13266;8072.72;-39.1475; 6651.76;-49.8587;
280 2400;3;loadimage;5006;13208.7;8237.55;-37.6352; 6373.97;-51.744;
281 2400;3;loadimage;5002;13129.1;8146.52;-37.9508; 6690.9;-49.0378;
282 2400;3;loadimage;5004;13317.9;8078.87;-39.3384; 6620.07;-50.2921;
283
284 2400;4;loadimage;5030;12761;10135.9;-20.5715; 8266.91;-35.2174;
285 2400;4;loadimage;5030;13103.6;10146;-22.5713; 8293.06;-36.7117;
286 2400;4;loadimage;5021;12921.5;10220.7;-20.9016; 8337.16;-35.4783;
287 2400;4;loadimage;5006;12611;10155.3;-19.4731; 8240.44;-34.6569;
288 2400;4;loadimage;5017;12879.2;9491.73;-26.3021; 8257.46;-35.8855;
289
290 2500;1;loadimage;5004;7753.01;4164.92;-46.2799; 3590.9;-53.6837;
291 2500;1;loadimage;5003;7738.89;4031.8;-47.9021; 3587.23;-53.6468;
292 2500;1;loadimage;5004;7703.88;4035.7;-47.6147; 3591.07;-53.3862;
293 2500;1;loadimage;5002;7735.77;4048.4;-47.6664; 3578.06;-53.7466;
294 2500;1;loadimage;5002;7710.38;4051.66;-47.452; 3577.25;-53.6047;
295
296 2500;2;loadimage;5002;11121.3;6315.85;-43.2094; 5253.02;-52.7662;
297 2500;2;loadimage;5004;11039;6332.55;-42.6349; 5301.39;-51.9759;
298 2500;2;loadimage;5004;11274.4;6334.4;-43.8159; 5286.5;-53.1104;
299 2500;2;loadimage;5003;11257.7;6495.05;-42.3056; 5284.06;-53.0626;
300 2500;2;loadimage;5003;11208.5;6497.17;-42.0337; 5337.05;-52.3841;
301
302 2500;3;loadimage;5003;14469.2;8772.21;-39.3734; 7138.02;-50.6677;
303 2500;3;loadimage;5003;14385;9001.63;-37.4235; 7167.27;-50.1754;
304 2500;3;loadimage;5004;14075.4;8790.01;-37.5505; 7086.04;-49.6565;
305 2500;3;loadimage;5003;14068;8788.75;-37.5266; 7165.09;-49.0681;
306 2500;3;loadimage;5002;14349;8944.75;-37.6627; 7142.71;-50.2214;
307
308 2500;4;loadimage;5007;14099.2;11034.4;-21.7378; 8426.17;-40.2367;
309 2500;4;loadimage;5020;13483.4;11116.7;-17.5527; 9046.28;-32.9082;
310 2500;4;loadimage;5030;13412.1;11094;-17.2837; 9072.96;-32.3525;
311 2500;4;loadimage;5008;13930.4;10956.2;-21.3503; 8795.57;-36.8608;
312 2500;4;loadimage;5025;13543;11464.5;-15.3473; 9067.59;-33.0461;
313
314 3000;1;loadimage;5003;9979.55;5985.25;-40.0249; 4995.83;-49.9393;
315 3000;1;loadimage;5002;9856.41;5849.06;-40.6573; 4979.44;-49.4802;
316 3000;1;loadimage;5003;9936.97;5881.59;-40.811; 4950.16;-50.1844;
317 3000;1;loadimage;5003;9964.49;5850.31;-41.2884; 4967.69;-50.1461;
318 3000;1;loadimage;5004;9954.22;5848.48;-41.2462; 4965.53;-50.1163;
319
320 3000;2;loadimage;5002;15730.3;9238.49;-41.2694; 7666.49;-51.2628;
321 3000;2;loadimage;5002;15813.4;9454.75;-40.2104; 7665.36;-51.5261;
322 3000;2;loadimage;5003;15842;9413.23;-40.5804; 7665.48;-51.6128;
323 3000;2;loadimage;5002;15978.2;9391.46;-41.2234; 7670.19;-51.9959;
324 3000;2;loadimage;5004;15612.9;9391.08;-39.8505; 7672.57;-50.8575;
325
326 3000;3;loadimage;5002;20159.5;13074.4;-35.1452; 10473.3;-48.0478;
327 3000;3;loadimage;5012;19863.3;12906;-35.0256; 10647.4;-46.3966;
328 3000;3;loadimage;5002;20165.2;12929.5;-35.882; 10708.6;-46.8958;
329 3000;3;loadimage;5012;19905.8;12915.2;-35.1184; 10688.6;-46.3043;

```

330 3000;3;loadimage;5003;19926.2;13034.9;-34.5839; 10579.8;-46.9049;
331
332 3000;4;loadimage;5020;16975.2;16704.4;-1.59533; 13049.8;-23.1248;
333 3000;4;loadimage;5030;16469.5;16316.2;-0.931034; 12804.1;-22.2562;
334 3000;4;loadimage;5025;17158.3;16433.2;-4.22623; 12918.5;-24.7098;
335 3000;4;loadimage;5006;17507.1;16311.2;-6.83098; 13249.8;-24.3178;
336 3000;4;loadimage;5027;16983.9;14777.4;-12.9915; 13033.7;-23.2582;
337
338 3500;1;loadimage;5003;10090;8460.07;-16.1538; 7153.75;-29.1005;
339 3500;1;loadimage;5004;10278.7;8326.58;-18.9915; 6949.49;-32.3891;
340 3500;1;loadimage;5004;10137.8;8152.68;-19.5814; 6807.34;-32.8519;
341 3500;1;loadimage;5003;10028.8;8083.6;-19.3962; 6807.99;-32.1157;
342 3500;1;loadimage;5003;10177.5;8048.13;-20.922; 6779.53;-33.3868;
343
344 3500;2;loadimage;5002;21630.4;13169.5;-39.1158; 10841;-49.8807;
345 3500;2;loadimage;5002;21850.4;13310.4;-39.0841; 10946.9;-49.9007;
346 3500;2;loadimage;5002;22035.2;13461.4;-38.9095; 10948.9;-50.3115;
347 3500;2;loadimage;5003;21913.6;13418.9;-38.7646; 11010.3;-49.756;
348 3500;2;loadimage;5003;21745.8;13458.6;-38.1093; 11057.4;-49.1515;
349
350 3500;3;loadimage;5004;29494.6;18676.4;-36.6785; 15053.3;-48.9627;
351 3500;3;loadimage;5003;29696.1;19156.2;-35.4927; 15217;-48.7575;
352 3500;3;loadimage;5002;29355.6;19312.4;-34.2122; 14027.9;-52.2138;
353 3500;3;loadimage;5002;29441.4;19424.1;-34.0247; 15207.4;-48.3469;
354 3500;3;loadimage;5003;30021.7;18749.3;-37.5475; 13558.7;-54.8369;
355
356 3500;4;loadimage;5016;19066.8;24357.1;27.7463; 19641.9;3.01649;
357 3500;4;loadimage;5027;18583.8;24910.7;34.0455; 18605.4;0.116586;
358 3500;4;loadimage;5024;20708.6;24460.9;18.1193; 19338.7;-6.61516;
359 3500;4;loadimage;5023;20941.8;24270.3;15.8941; 19600.2;-6.40647;
360 3500;4;loadimage;5027;19049.2;24605.2;29.1665; 19620.2;2.99752;
361
362 4000;1;loadimage;5003;10367.5;11796;13.7791; 9920.36;-4.31254;
363 4000;1;loadimage;5002;10284.3;11423;11.0731; 9853.02;-4.19308;
364 4000;1;loadimage;5003;10230.6;11483;12.2423; 9596.22;-6.20056;
365 4000;1;loadimage;5004;10181.1;11384.2;11.8173; 9534.79;-6.34778;
366 4000;1;loadimage;5003;10060.1;11453;13.8454; 9493.6;-5.63151;
367
368 4000;2;loadimage;5002;23908.8;17999.5;-24.7159; 14844;-37.9138;
369 4000;2;loadimage;5002;25469.6;18692.8;-26.6072; 15018.1;-41.0353;
370 4000;2;loadimage;5003;23592.3;18897.8;-19.8985; 15282.9;-35.2206;
371 4000;2;loadimage;5002;25276;18911.5;-25.18; 15319.8;-39.3898;
372 4000;2;loadimage;5002;25171.5;18996.4;-24.5321; 15405.5;-38.7979;
373
374 4000;3;loadimage;5004;34340;23604.9;-31.2612; 19028.5;-44.5881;
375 4000;3;loadimage;5004;33966.5;23963.1;-29.4509; 18945.3;-44.2236;
376 4000;3;loadimage;5004;36007.4;23347.8;-35.1583; 19224.1;-46.6106;
377 4000;3;loadimage;5011;31067.6;23911.2;-23.0348; 19285.2;-37.9249;
378 4000;3;loadimage;5002;31873.9;23150.2;-27.3696; 19344.7;-39.3088;
379
380 4000;4;loadimage;5023;21626;24757.7;14.4807; 21028.9;-2.76147;
381 4000;4;loadimage;5024;20945.7;30649.2;46.3271; 24535.4;17.1382;
382 4000;4;loadimage;5030;20848.4;29530.3;41.6433; 23851.9;14.4066;
383 4000;4;loadimage;5021;20816.7;29619.6;42.2878; 24757.2;18.9294;
384 4000;4;loadimage;5027;21654.4;29383.4;35.6923; 24738.6;14.2424;
385
386 1500;1;greyscale;5143;2553.06;2793.24;9.40762; 2734.86;7.12084;
387 1500;1;greyscale;5141;2491.68;2572.91;3.26003; 2593.38;4.08172;
388 1500;1;greyscale;5142;2325.21;2569.83;10.5203; 2552.58;9.77848;
389 1500;1;greyscale;5143;2329.57;2571.53;10.3864; 2512.92;7.87042;
390 1500;1;greyscale;5142;2324.8;2550.01;9.68702; 2501.54;7.60223;
391
392 1500;2;greyscale;5012;3683.26;3935.26;6.84162; 3847.43;4.45701;
393 1500;2;greyscale;5055;3668.21;4076.28;11.1245; 3865.09;5.36733;
394 1500;2;greyscale;5022;3678.86;4046.51;9.99359; 3858.25;4.87634;
395 1500;2;greyscale;5012;3677.94;4054.92;10.2497; 3876.59;5.40113;
396 1500;2;greyscale;5010;3685.86;4076.92;10.6099; 3883.72;5.3682;
397
398 1500;3;greyscale;5150;4884.03;5453.1;11.6516; 5177.26;6.00377;
399 1500;3;greyscale;5150;4901.79;5454.87;11.2834; 5219.64;6.48435;
400 1500;3;greyscale;5153;4874.11;5443.3;11.6778; 5195.45;6.59295;
401 1500;3;greyscale;5148;4919.74;5483.25;11.454; 5213.1;5.96291;
402 1500;3;greyscale;5149;4912.83;5457.77;11.0922; 5253.51;6.93451;
403
404 1500;4;greyscale;5035;6179.78;6803.74;10.0966; 5219.28;-15.5427;
405 1500;4;greyscale;5129;6118.97;6814.6;11.3685; 6493.55;6.12165;
406 1500;4;greyscale;5094;6180.42;6848.38;10.8077; 6778.82;9.68207;
407 1500;4;greyscale;5152;6167.63;6859.59;11.2193; 6552.98;6.24803;
408 1500;4;greyscale;5059;6164.81;6898.5;11.9012; 6504.77;5.51455;
409
410 2000;1;greyscale;5103;3228.68;3520.09;9.02543; 3399.58;5.29317;
411 2000;1;greyscale;5020;3207.26;3484.22;8.63546; 3369.55;5.06036;
412 2000;1;greyscale;5106;3269.99;3459.87;5.80679; 3377.06;3.27431;
413 2000;1;greyscale;5106;3250.98;3480.48;7.05952; 3366.7;3.55965;
414 2000;1;greyscale;5103;3243;3462.25;6.76081; 3367.68;3.84466;
415
416 2000;2;greyscale;5039;5074.25;5583.9;10.0438; 5319.12;4.82566;

```


417 2000;2;greyscale;5015;5120.61;5602.91;9.41876; 5312.12;3.74005;
418 2000;2;greyscale;5019;5086.82;5611.49;10.3143; 5328.86;4.75809;
419 2000;2;greyscale;5033;5097.11;5610.98;10.0817; 5341.58;4.79626;
420 2000;2;greyscale;5071;4804.37;5612.52;16.8211; 5303.55;10.3901;
421
422 2000;3;greyscale;5019;6944.16;7780.21;12.0396; 7269.85;4.69013;
423 2000;3;greyscale;5010;6949.16;8042.06;15.7271; 7539.66;8.4975;
424 2000;3;greyscale;5021;6932.21;7799.22;12.5071; 7273.94;4.92963;
425 2000;3;greyscale;5027;6947.63;7787.92;12.0946; 7614.94;9.60483;
426 2000;3;greyscale;5020;6943.22;7813.52;12.5345; 7336.69;5.66694;
427
428 2000;4;greyscale;5115;8800.11;9899.97;12.4983; 9529.87;8.29259;
429 2000;4;greyscale;5101;8840.59;9844.7;11.3579; 9286.06;5.03897;
430 2000;4;greyscale;5096;8892.15;10225.6;14.9963; 9555.19;7.45651;
431 2000;4;greyscale;5098;8865.27;10252.6;15.649; 7750.42;-12.5755;
432 2000;4;greyscale;5104;8881.35;10023.1;12.8554; 9354.63;5.32898;
433
434 2200;1;greyscale;5060;3773.25;4231.81;12.1528; 4019.65;6.5302;
435 2200;1;greyscale;5058;3772.42;4014.16;6.40811; 3950.42;4.71854;
436 2200;1;greyscale;5058;3768.1;4011.24;6.45267; 3925.41;4.17497;
437 2200;1;greyscale;5057;3743.34;4017.05;7.31202; 3946.85;5.4366;
438 2200;1;greyscale;5056;3758.65;4040.42;7.49663; 3907.34;3.956;
439
440 2200;2;greyscale;5040;5937.05;6533.71;10.0498; 6299.56;6.10599;
441 2200;2;greyscale;5079;5940.48;6708.79;12.9336; 6302.2;6.08915;
442 2200;2;greyscale;5039;5930.75;6652.77;12.1741; 6307.63;6.35461;
443 2200;2;greyscale;5007;5917.43;6690.95;13.0719; 6313.49;6.69299;
444 2200;2;greyscale;5077;5930.57;6686.1;12.7397; 6317.8;6.52939;
445
446 2200;3;greyscale;5081;8207.42;9414.9;14.712; 8953;9.08414;
447 2200;3;greyscale;5084;8272.9;9296.47;12.3726; 8946.1;8.13744;
448 2200;3;greyscale;5093;8282.92;9558.72;15.4029; 9018.09;8.87577;
449 2200;3;greyscale;5096;8272.66;9355.72;13.0921; 8802.07;6.39954;
450 2200;3;greyscale;5079;8326.79;9355.64;12.3559; 8941.52;7.38262;
451
452 2200;4;greyscale;5094;10478.1;11850;13.0933; 11036.2;5.32682;
453 2200;4;greyscale;5096;10509.3;12144.8;15.5621; 9372.69;-10.8157;
454 2200;4;greyscale;5101;10533.3;12210.2;15.9197; 11426.2;8.47739;
455 2200;4;greyscale;5105;10506.5;11901.5;13.2771; 10852.7;3.29429;
456 2200;4;greyscale;5095;10469.1;9603.97;-8.26324; 10367.9;-0.96639;
457
458 2400;1;greyscale;5021;4347.03;4829.28;11.094; 4600.37;5.82802;
459 2400;1;greyscale;5021;4346.19;4772.45;9.80777; 4589.87;5.6067;
460 2400;1;greyscale;5021;4343.48;4808.09;10.6967; 4565.01;5.10025;
461 2400;1;greyscale;5073;4135.3;4649.16;12.426; 4569.38;10.4969;
462 2400;1;greyscale;5021;4320.72;4772.45;10.4549; 4530.27;4.84994;
463
464 2400;2;greyscale;5066;7056.69;7833.59;11.0094; 7575.95;7.35851;
465 2400;2;greyscale;5096;7043.51;7643.2;8.51411; 7396.44;5.01077;
466 2400;2;greyscale;5058;7087.64;7756.65;9.43915; 7379.96;4.12437;
467 2400;2;greyscale;5070;7074.96;7885.11;11.451; 7580.73;7.14886;
468 2400;2;greyscale;5086;7047.09;7900.28;12.107; 7396.23;4.95433;
469
470 2400;3;greyscale;5059;9945.34;11189.6;12.511; 10539.9;5.97813;
471 2400;3;greyscale;5088;9850.66;11378.3;15.5076; 10601.1;7.61853;
472 2400;3;greyscale;5056;9875.91;11133.7;12.7364; 10536.4;6.68779;
473 2400;3;greyscale;5059;9915.02;11235.2;13.3147; 10659.5;7.50906;
474 2400;3;greyscale;5054;9732.47;11365.3;16.7766; 10468;7.55768;
475
476 2400;4;greyscale;5088;12438.3;13914.3;11.8662; 13185.4;6.00628;
477 2400;4;greyscale;5085;12498.9;14041.9;12.3458; 13180.2;5.45106;
478 2400;4;greyscale;5067;12534.1;14139.8;12.8101; 13113.1;4.61897;
479 2400;4;greyscale;5085;12509.5;14062.8;12.4169; 11573.3;-7.48445;
480 2400;4;greyscale;5068;12521.4;14218.6;13.5541; 13268.4;5.96614;
481
482 2500;1;greyscale;5002;4727.04;5326.93;12.6906; 4932.35;4.34327;
483 2500;1;greyscale;5003;4646.98;5098.86;9.72416; 4902.3;5.49439;
484 2500;1;greyscale;5092;4639.25;5114.28;10.2395; 4871.49;5.00595;
485 2500;1;greyscale;5002;4652.78;5080.72;9.19751; 4879.15;4.86523;
486 2500;1;greyscale;5092;4647.57;5078.28;9.26761; 4790.02;3.06517;
487
488 2500;2;greyscale;5079;7604.86;8462.37;11.2758; 8228.83;8.20485;
489 2500;2;greyscale;5036;7655.71;8603.15;12.3755; 8247.52;7.73033;
490 2500;2;greyscale;5039;7630.19;8765.31;14.8767; 8252.88;8.16094;
491 2500;2;greyscale;5050;7660.35;8626.42;12.6113; 8303.9;8.40098;
492 2500;2;greyscale;5036;7629.87;8517.82;11.6378; 8271.43;8.40856;
493
494 2500;3;greyscale;5041;10551;12092.8;14.6124; 11242.7;6.55544;
495 2500;3;greyscale;5069;10548.1;12235.3;15.9946; 11445.8;8.51006;
496 2500;3;greyscale;5084;10566.2;12066.8;14.202; 11428.4;8.15978;
497 2500;3;greyscale;5041;10575.3;12043.9;13.8874; 11427;8.05342;
498 2500;3;greyscale;5040;10499.8;12190.3;16.1001; 11293.1;7.55582;
499
500 2500;4;greyscale;5055;13576.2;15107.2;11.2766; 13944.3;2.71134;
501 2500;4;greyscale;5047;13563.9;15377.2;13.3681; 14356.2;5.84135;
502 2500;4;greyscale;5052;13590.2;15333.8;12.8302; 14493.6;6.64796;
503 2500;4;greyscale;5045;13624.4;15366.4;12.7854; 11775.6;-13.5698;

504 2500;4;greyscale;5057;13587.4;15365.3;13.0846; 14430.2;6.20247;
505
506 3000;1;greyscale;5007;6851.57;7581.91;10.6595; 7076.36;3.28078;
507 3000;1;greyscale;5009;6814.93;7416.07;8.82106; 6977.8;2.39003;
508 3000;1;greyscale;5072;6668.74;7406.73;11.0665; 6978.46;4.64443;
509 3000;1;greyscale;5072;6716.67;7419.46;10.4635; 6946.33;3.41933;
510 3000;1;greyscale;5014;6612.67;7409.71;12.0532; 6955.21;5.18013;
511
512 3000;2;greyscale;5067;11170.9;12665.8;13.3825; 11884.8;6.39117;
513 3000;2;greyscale;5024;11131.3;12520.1;12.4759; 11589.3;4.11402;
514 3000;2;greyscale;5007;11185.3;12559.6;12.2862; 11620.3;3.88861;
515 3000;2;greyscale;5006;11193.6;12568.4;12.282; 11590.7;3.54765;
516 3000;2;greyscale;5079;11146.5;12656.6;13.5483; 11981.5;7.49108;
517
518 3000;3;greyscale;5054;15742.5;17381.8;10.4128; 16106.9;2.31483;
519 3000;3;greyscale;5052;15868.8;17819.1;12.2902; 16718.2;5.35218;
520 3000;3;greyscale;5055;15769.4;17964.8;13.9221; 16570.6;5.0812;
521 3000;3;greyscale;5054;15892.8;17767.1;11.7934; 16738.2;5.31949;
522 3000;3;greyscale;5059;15970.3;18041.1;12.9662; 16849;5.50158;
523
524 3000;4;greyscale;5076;20006.2;21763.4;8.78327; 20539.1;2.66399;
525 3000;4;greyscale;5065;20151.9;23501;16.6196; 21583.5;7.10441;
526 3000;4;greyscale;5065;20147.9;23558.3;16.9264; 21792.9;8.16424;
527 3000;4;greyscale;5065;20138;23542.8;16.9072; 21731.5;7.91286;
528 3000;4;greyscale;5065;20121.7;23448.8;16.5349; 19050.5;-5.32379;
529
530 3500;1;greyscale;5019;9845.95;10961.8;11.3334; 10232.8;3.9286;
531 3500;1;greyscale;5030;9731.95;10845.5;11.4421; 10003.7;2.79266;
532 3500;1;greyscale;5064;9497.35;10607.3;11.6871; 9848.26;3.69481;
533 3500;1;greyscale;5046;9389.31;10473.3;11.5448; 9824.6;4.63592;
534 3500;1;greyscale;5015;9490.9;10397.5;9.55196; 9727.6;2.49398;
535
536 3500;2;greyscale;5070;15783.2;17634.2;11.7277; 16606.4;5.21578;
537 3500;2;greyscale;5032;15734.4;18025.1;14.5587; 16699.1;6.13103;
538 3500;2;greyscale;5037;15602.6;18168.2;16.4434; 16852.7;8.01208;
539 3500;2;greyscale;5026;16205.1;18223.5;12.4553; 16945.4;4.56879;
540 3500;2;greyscale;5047;16286.4;18147.8;11.429; 16858;3.50986;
541
542 3500;3;greyscale;5064;22450;25947;15.5769; 24214.4;7.85925;
543 3500;3;greyscale;5012;22692;26385.5;16.2764; 24482.6;7.89083;
544 3500;3;greyscale;5024;22776.7;26240.2;15.2064; 24097.7;5.79997;
545 3500;3;greyscale;5007;22850.2;26353.8;15.3333; 24158.6;5.72621;
546 3500;3;greyscale;5041;22734.6;26237.8;15.4091; 24219.3;6.53082;
547
548 3500;4;greyscale;5050;25672.6;32917.8;28.2213; 24769.3;-3.51883;
549 3500;4;greyscale;5058;25637.8;24910.9;-2.83529; 24952.1;-2.67462;
550 3500;4;greyscale;5062;25583.2;24949.5;-2.47705; 24869.1;-2.79162;
551 3500;4;greyscale;5053;25721.6;24901.8;-3.18726; 25002.1;-2.7972;
552 3500;4;greyscale;5067;25781.9;24912.1;-3.37353; 24917.4;-3.35312;
553
554 4000;1;greyscale;5026;13787.3;14874.4;7.88516; 14424.5;4.62187;
555 4000;1;greyscale;5022;13871.9;15268.9;10.0702; 14105.8;1.68597;
556 4000;1;greyscale;5025;13561.8;15108.8;11.4069; 13893.8;2.44772;
557 4000;1;greyscale;5025;13461.5;14996;11.3991; 13850.4;2.88881;
558 4000;1;greyscale;5008;13619.4;14889.4;9.32452; 13755.4;0.998321;
559
560 4000;2;greyscale;5031;22421.1;25102.9;11.9613; 23067.8;2.88461;
561 4000;2;greyscale;5019;22576.8;25801.4;14.2824; 23756.7;5.22606;
562 4000;2;greyscale;5005;21649;25950;19.8666; 23962.6;10.687;
563 4000;2;greyscale;5031;22085.5;25625.3;16.0274; 23922.4;8.31721;
564 4000;2;greyscale;5005;21911.8;25281.9;15.3802; 23941;9.26054;
565
566 4000;3;greyscale;5041;26983;30796.4;14.1323; 25674.6;-4.84892;
567 4000;3;greyscale;5014;25702.1;26481.9;3.0342; 24730.8;-3.77912;
568 4000;3;greyscale;5006;25805.2;24865.9;-3.64002; 24863.2;-3.65023;
569 4000;3;greyscale;5066;25888.8;24893;-3.84647; 24843.1;-4.03921;
570 4000;3;greyscale;5067;25963.4;24903.8;-4.08088; 24801.2;-4.4763;
571
572 4000;4;greyscale;5056;25653.8;24925;-2.84073; 24796.4;-3.34207;
573 4000;4;greyscale;5049;25769;24991.3;-3.01807; 24845.8;-3.58259;
574 4000;4;greyscale;5053;25756.6;24896.8;-3.33804; 24950.4;-3.12994;
575 4000;4;greyscale;5013;25700.6;25019.3;-2.65106; 24906.3;-3.09077;
576 4000;4;greyscale;5016;25786.4;24978;-3.13486; 24854.5;-3.61407;
577
578 1500;1;sharpencontrast;5041;2623.62;2270.03;-13.477; 2540.47;-3.16945;
579 1500;1;sharpencontrast;5040;2495.6;2124.48;-14.8711; 2420.59;-3.00585;
580 1500;1;sharpencontrast;5040;2418.6;1945.53;-19.5596; 2318.95;-4.11984;
581 1500;1;sharpencontrast;5041;2424.24;1935.24;-20.1715; 2290.9;-5.50046;
582 1500;1;sharpencontrast;5042;2440.99;1932.59;-20.8274; 2298.12;-5.85288;
583
584 1500;2;sharpencontrast;5154;3765.06;2838.31;-24.6144; 3497.87;-7.09658;
585 1500;2;sharpencontrast;5153;3755.09;2897.29;-22.8438; 3509.05;-6.55231;
586 1500;2;sharpencontrast;5154;3750.02;2815.02;-24.9331; 3500.75;-6.64716;
587 1500;2;sharpencontrast;5147;3753.35;2898.73;-22.7696; 3506.16;-6.58583;
588 1500;2;sharpencontrast;5148;3757.03;2781.5;-25.9655; 3490.05;-7.10624;
589
590 1500;3;sharpencontrast;5077;5099.19;3670.14;-28.0252; 4659;-8.63251;

591 1500;3;sharpencontrast;5067;5067.76;3689.43;-27.1979; 4663.89;-7.96932;
592 1500;3;sharpencontrast;5074;5050.96;3693.07;-26.8838; 4669.99;-7.54252;
593 1500;3;sharpencontrast;5066;5061.19;3687.84;-27.1349; 4673.23;-7.66545;
594 1500;3;sharpencontrast;5075;5052.24;3670.49;-27.3492; 4670.77;-7.55051;
595
596 1500;4;sharpencontrast;5142;6396.31;4631.73;-27.5874; 5797.09;-9.36813;
597 1500;4;sharpencontrast;5099;6413.93;4502.95;-29.7942; 5797.91;-9.60433;
598 1500;4;sharpencontrast;5103;6442.99;4564.62;-29.1537; 5389.75;-16.347;
599 1500;4;sharpencontrast;5100;6403.28;4568.32;-28.6565; 5817.43;-9.14916;
600 1500;4;sharpencontrast;5151;6410;4642.93;-27.5675; 5834.65;-8.9758;
601
602 2000;1;sharpencontrast;5040;3326.5;2636.48;-20.7433; 3086.46;-7.21604;
603 2000;1;sharpencontrast;5041;3288.98;2624.61;-20.1997; 3103.06;-5.65286;
604 2000;1;sharpencontrast;5041;3294.04;2630.38;-20.1471; 3061.12;-7.07101;
605 2000;1;sharpencontrast;5042;3306.51;2642.63;-20.078; 3073.56;-7.04511;
606 2000;1;sharpencontrast;5041;3282.83;2613.71;-20.3823; 3053.35;-6.99015;
607
608 2000;2;sharpencontrast;5112;5196.07;3780.39;-27.2451; 4785.9;-7.89379;
609 2000;2;sharpencontrast;5125;5155.83;3830.64;-25.7027; 4791.88;-7.05896;
610 2000;2;sharpencontrast;5127;5181.78;3845.64;-25.7854; 4791.52;-7.53147;
611 2000;2;sharpencontrast;5114;5210.11;3828.62;-26.5157; 4779.99;-8.2556;
612 2000;2;sharpencontrast;5113;5213.23;3847.71;-26.1935; 4773.73;-8.43059;
613
614 2000;3;sharpencontrast;5082;7145.18;5097.48;-28.6584; 6525.15;-8.67754;
615 2000;3;sharpencontrast;5080;7136.61;5084.72;-28.7516; 5873.08;-17.705;
616 2000;3;sharpencontrast;5087;7151.08;5113.69;-28.4906; 6541.01;-8.53115;
617 2000;3;sharpencontrast;5086;7134.61;5109.29;-28.3872; 6550.11;-8.19254;
618 2000;3;sharpencontrast;5082;7152.49;5076.54;-29.0243; 6555.94;-8.34054;
619
620 2000;4;sharpencontrast;5103;8968.04;6337.8;-29.3291; 8278.23;-7.69194;
621 2000;4;sharpencontrast;5102;9016.13;6457.01;-28.3838; 8275.18;-8.21806;
622 2000;4;sharpencontrast;5015;9097.99;6508.83;-28.4586; 8269.23;-9.10928;
623 2000;4;sharpencontrast;5106;9033.85;6477.55;-28.2969; 8304.49;-8.07366;
624 2000;4;sharpencontrast;5108;9026.19;5987.1;-33.6697; 6743.87;-25.2856;
625
626 2200;1;sharpencontrast;5043;3863.43;3036.08;-21.415; 3611.08;-6.53187;
627 2200;1;sharpencontrast;5041;3866.49;2944.89;-23.8355; 3581.76;-7.36396;
628 2200;1;sharpencontrast;5043;3895.27;2969.6;-23.7641; 3574.16;-8.2436;
629 2200;1;sharpencontrast;5043;3779.96;2944.53;-22.1016; 3543.6;-6.25295;
630 2200;1;sharpencontrast;5111;3708.61;2937.76;-20.7856; 3558.21;-4.0555;
631
632 2200;2;sharpencontrast;5008;6191.34;4432.28;-28.4116; 5609.9;-9.39109;
633 2200;2;sharpencontrast;5114;6198.69;4459.66;-28.0549; 5661.58;-8.66486;
634 2200;2;sharpencontrast;5114;6173.98;4450.89;-27.9089; 5651.34;-8.46508;
635 2200;2;sharpencontrast;5013;6173.46;4446.92;-27.9671; 5648.83;-8.49806;
636 2200;2;sharpencontrast;5115;6215.82;4481.55;-27.9009; 5658.22;-8.97064;
637
638 2200;3;sharpencontrast;5081;8529.07;6039.27;-29.1919; 7783.66;-8.73963;
639 2200;3;sharpencontrast;5086;8561.06;6010.73;-29.7899; 7818.2;-8.67728;
640 2200;3;sharpencontrast;5090;8594.87;6169.24;-28.2218; 7836.24;-8.82655;
641 2200;3;sharpencontrast;5088;8644.86;5670.8;-34.4027; 7773.57;-10.0787;
642 2200;3;sharpencontrast;5072;8341.98;6146.82;-26.3146; 7797.53;-6.52654;
643
644 2200;4;sharpencontrast;5030;10812.1;7547.98;-30.1897; 7850.19;-27.3947;
645 2200;4;sharpencontrast;5026;10870.7;7635.32;-29.7626; 9859.08;-9.30623;
646 2200;4;sharpencontrast;5029;10883;7658.32;-29.6306; 9943.16;-8.63608;
647 2200;4;sharpencontrast;5102;10857.8;7657.19;-29.4776; 9896.93;-8.84967;
648 2200;4;sharpencontrast;5104;10871.2;7704.78;-29.1265; 9996.1;-8.04941;
649
650 2400;1;sharpencontrast;5043;4471.46;3519.03;-21.3001; 4234.53;-5.29864;
651 2400;1;sharpencontrast;5046;4473.9;3502.77;-21.7066; 4238.11;-5.27053;
652 2400;1;sharpencontrast;5046;4442.81;3471.47;-21.8632; 4173.04;-6.07206;
653 2400;1;sharpencontrast;5046;4480.74;3467.61;-22.6108; 4190.17;-6.48489;
654 2400;1;sharpencontrast;5046;4458;3471.14;-22.1367; 4173.22;-6.38798;
655
656 2400;2;sharpencontrast;5106;7236.23;5252.9;-27.4084; 6621.12;-8.50046;
657 2400;2;sharpencontrast;5093;7302.93;5312.49;-27.2554; 6635.36;-9.1411;
658 2400;2;sharpencontrast;5095;7313.91;5277.32;-27.8454; 6630.16;-9.34859;
659 2400;2;sharpencontrast;5010;7319.96;5291.65;-27.7092; 6673.11;-8.83683;
660 2400;2;sharpencontrast;5101;7284.76;5291;-27.369; 6669.43;-8.44685;
661
662 2400;3;sharpencontrast;5095;10205.1;7181.27;-29.6306; 9099.26;-10.8362;
663 2400;3;sharpencontrast;5097;10182.3;7221.64;-29.0767; 9254.66;-9.11064;
664 2400;3;sharpencontrast;5093;10263.4;7289.03;-28.9804; 9189.44;-10.4639;
665 2400;3;sharpencontrast;5099;10255;7295.3;-28.8607; 9312.11;-9.19408;
666 2400;3;sharpencontrast;5093;10276.3;7280.41;-29.1532; 9339.9;-9.11209;
667
668 2400;4;sharpencontrast;5092;13060.6;9187.17;-29.6575; 11777.1;-9.8276;
669 2400;4;sharpencontrast;5091;13008.4;9319.61;-28.3571; 11142.8;-14.3414;
670 2400;4;sharpencontrast;5084;12970.7;9302.99;-28.2771; 11112.2;-14.3287;
671 2400;4;sharpencontrast;5098;12931.1;9252.03;-28.4514; 9297.22;-28.1019;
672 2400;4;sharpencontrast;5102;12930.8;9109.12;-29.5548; 11748.7;-9.1419;
673
674 2500;1;sharpencontrast;5048;4796.92;3783.99;-21.1163; 4512.58;-5.92749;
675 2500;1;sharpencontrast;5091;4760.02;3758.37;-21.0431; 4457.75;-6.35019;
676 2500;1;sharpencontrast;5047;4783.91;3745.1;-21.7148; 4500.51;-5.92398;
677 2500;1;sharpencontrast;5045;4764;3745.5;-21.3791; 4492.12;-5.70681;

678 2500;1;sharpencontrast;5039;4758.2;3725.2;-21.7098; 4462.99;-6.20431;
679
680 2500;2;sharpencontrast;5018;7844.29;5606.78;-28.5241; 7161.09;-8.70951;
681 2500;2;sharpencontrast;5101;7897.58;5615.99;-28.8898; 7193.1;-8.92013;
682 2500;2;sharpencontrast;5014;7882.35;5614.42;-28.7723; 7211.49;-8.51088;
683 2500;2;sharpencontrast;5099;7871.77;5634.07;-28.4268; 7196.32;-8.58064;
684 2500;2;sharpencontrast;5006;7854.67;5770.62;-26.5326; 7213.06;-8.16851;
685
686 2500;3;sharpencontrast;5100;10957.8;7709.05;-29.6477; 10034;-8.42994;
687 2500;3;sharpencontrast;5098;11018.4;7912.47;-28.1883; 10106.3;-8.27793;
688 2500;3;sharpencontrast;5067;11162.4;7844.96;-29.7195; 7299.99;-34.6017;
689 2500;3;sharpencontrast;5099;11097.2;7798.12;-29.7288; 9985.88;-10.0143;
690 2500;3;sharpencontrast;5099;11012.6;7819.89;-28.9916; 10128.1;-8.03166;
691
692 2500;4;sharpencontrast;5045;14100.3;9939.29;-29.5102; 11340.7;-19.5716;
693 2500;4;sharpencontrast;5101;14144.4;9921.95;-29.8527; 12706.9;-10.1635;
694 2500;4;sharpencontrast;5041;14152.3;9993.06;-29.3892; 12670.9;-10.4679;
695 2500;4;sharpencontrast;5043;14212.2;10024.1;-29.4684; 12675;-10.8158;
696 2500;4;sharpencontrast;5039;14229;10085.2;-29.1223; 12771.4;-10.2436;
697
698 3000;1;sharpencontrast;5038;7004.95;5407.76;-22.8008; 6450.9;-7.90932;
699 3000;1;sharpencontrast;5039;6963.65;5373.67;-22.8325; 6434.68;-7.59612;
700 3000;1;sharpencontrast;5052;6893.71;5287.46;-23.3002; 6401.13;-7.14529;
701 3000;1;sharpencontrast;5038;6778.45;5179.04;-23.5955; 6371.07;-6.00981;
702 3000;1;sharpencontrast;5051;6928.47;5153.51;-25.6183; 6382.88;-7.87456;
703
704 3000;2;sharpencontrast;5015;11583.2;8161.97;-29.5363; 10458.4;-9.71084;
705 3000;2;sharpencontrast;5017;11583.8;8185.2;-29.3389; 10584.6;-8.62567;
706 3000;2;sharpencontrast;5011;11662.7;8283.48;-28.9747; 10608.5;-9.03919;
707 3000;2;sharpencontrast;5012;11668.5;8248.6;-29.3086; 10577.8;-9.3472;
708 3000;2;sharpencontrast;5009;11568.6;8405.39;-27.3433; 10547.5;-8.82687;
709
710 3000;3;sharpencontrast;5028;16289;11480.2;-29.5215; 14448.8;-11.2974;
711 3000;3;sharpencontrast;5027;16355;11518.3;-29.5732; 14667.2;-10.3196;
712 3000;3;sharpencontrast;5029;16484.6;11532.9;-30.0382; 14716.2;-10.7274;
713 3000;3;sharpencontrast;5028;16541.3;11828.8;-28.4893; 14883.5;-10.0223;
714 3000;3;sharpencontrast;5030;16497.2;11704;-29.0545; 12825.4;-22.2573;
715
716 3000;4;sharpencontrast;5066;20720.6;14424.1;-30.3876; 18966.6;-8.46495;
717 3000;4;sharpencontrast;5063;20691.4;14504.4;-29.9015; 18921.6;-8.55312;
718 3000;4;sharpencontrast;5064;20784.8;14711.2;-29.2211; 18094.7;-12.9424;
719 3000;4;sharpencontrast;5063;20846.4;14804.5;-28.983; 19201.1;-7.8925;
720 3000;4;sharpencontrast;5062;20783.5;14475.9;-30.3492; 19099.4;-8.1032;
721
722 3500;1;sharpencontrast;5039;10086.1;7636.83;-24.2833; 9350.87;-7.28914;
723 3500;1;sharpencontrast;5053;9910.79;7616.83;-23.146; 9115.28;-8.02672;
724 3500;1;sharpencontrast;5040;9815.95;7453.24;-24.0701; 9034.81;-7.95792;
725 3500;1;sharpencontrast;5058;9697.22;7384.12;-23.8532; 8967.7;-7.52297;
726 3500;1;sharpencontrast;5041;9880.25;7322.38;-25.8887; 8944.06;-9.47536;
727
728 3500;2;sharpencontrast;5010;16533;11675.3;-29.382; 14718.5;-10.9746;
729 3500;2;sharpencontrast;5053;16742.6;11860.9;-29.1572; 14808.4;-11.5526;
730 3500;2;sharpencontrast;5013;16823.6;11844.9;-29.5934; 14916.8;-11.3344;
731 3500;2;sharpencontrast;5022;16689.9;11944.4;-28.4333; 14918.6;-10.6131;
732 3500;2;sharpencontrast;5006;16929.8;11920.6;-29.5878; 14999.8;-11.3996;
733
734 3500;3;sharpencontrast;5052;23400.7;16471.3;-29.6121; 21238.1;-9.24147;
735 3500;3;sharpencontrast;5044;23518.8;16600.6;-29.4156; 21515.1;-8.51944;
736 3500;3;sharpencontrast;5053;23650.8;16780.3;-29.0498; 21607.1;-8.64105;
737 3500;3;sharpencontrast;5054;23755.7;16835.2;-29.132; 21681.2;-8.73286;
738 3500;3;sharpencontrast;5051;23687.4;16813.7;-29.0184; 21549.5;-9.02569;
739
740 3500;4;sharpencontrast;5071;30813.9;21504.7;-30.2111; 27886.8;-9.49924;
741 3500;4;sharpencontrast;5075;30774.5;21929.8;-28.7404; 28814.2;-6.37018;
742 3500;4;sharpencontrast;5060;25588.4;22251.6;-13.0403; 27257.4;6.52268;
743 3500;4;sharpencontrast;5079;25670.9;21907.6;-14.6596; 27651.1;7.71378;
744 3500;4;sharpencontrast;5076;25559.6;21858.5;-14.4805; 27714.7;8.43137;
745
746 4000;1;sharpencontrast;5053;14234.6;10900.3;-23.4243; 13259.8;-6.84839;
747 4000;1;sharpencontrast;5064;13824.2;10605.9;-23.28; 12935.5;-6.42835;
748 4000;1;sharpencontrast;5047;14180.9;10522.9;-25.7955; 12820;-9.59718;
749 4000;1;sharpencontrast;5041;13993.9;10366.7;-25.9199; 12712.7;-9.1553;
750 4000;1;sharpencontrast;5048;13965.4;10279.7;-26.3916; 12652.1;-9.40374;
751
752 4000;2;sharpencontrast;5006;23414.6;16203.6;-30.7973; 20882.3;-10.8152;
753 4000;2;sharpencontrast;5047;23939.1;16515.5;-31.0104; 21143.6;-11.6773;
754 4000;2;sharpencontrast;5058;23167.8;16669.3;-28.0498; 21275;-8.17027;
755 4000;2;sharpencontrast;5037;23469;16650.7;-29.0526; 21357.6;-8.99683;
756 4000;2;sharpencontrast;5039;23473.6;16863;-28.1622; 21242.4;-9.50532;
757
758 4000;3;sharpencontrast;5065;29581.5;20458;-30.8421; 26635.1;-9.96048;
759 4000;3;sharpencontrast;5028;29181.9;20913.8;-28.3328; 26961.8;-7.60763;
760 4000;3;sharpencontrast;5016;28614.8;20942.3;-26.8132; 27358.7;-4.38989;
761 4000;3;sharpencontrast;5018;27593.2;21035.2;-23.7667; 27113.5;-1.73833;
762 4000;3;sharpencontrast;5026;26413.6;21141.4;-19.9604; 27272.8;3.25276;
763
764 4000;4;sharpencontrast;5071;25757.3;24721;-4.02317; 25064.2;-2.69071;

765 4000;4;sharpencontrast;5070;25779;22423.3;-13.0172; 27104.4;5.14154;
766 4000;4;sharpencontrast;5065;25598.8;23814.1;-6.97155; 25949.3;1.36947;
767 4000;4;sharpencontrast;5065;25676.4;24749.3;-3.61085; 25066.3;-2.3761;
768 4000;4;sharpencontrast;5074;25695.6;24837;-3.34143; 25041.8;-2.54441;
769
770 1500;1;copyimage;5013;2583.21;2793.24;8.13091; 2739.07;6.03383;
771 1500;1;copyimage;5011;2392.17;2572.91;7.55551; 2645.95;10.6087;
772 1500;1;copyimage;5012;2399.8;2569.83;7.08494; 2529.29;5.3957;
773 1500;1;copyimage;5005;2412.79;2571.53;6.57904; 2465.69;2.19248;
774 1500;1;copyimage;5017;2390.63;2550.01;6.66657; 2474.71;3.51715;
775
776 1500;2;copyimage;5020;3665.47;3935.26;7.36037; 3788.78;3.36427;
777 1500;2;copyimage;5021;3663.47;4076.28;11.2682; 3819.69;4.26411;
778 1500;2;copyimage;5015;3660.63;4046.51;10.5414; 3797.44;3.73735;
779 1500;2;copyimage;5018;3634.25;4054.92;11.5753; 3809.06;4.81013;
780 1500;2;copyimage;5014;3675.17;4076.92;10.9314; 3764.82;2.43938;
781
782 1500;3;copyimage;5017;4962.27;5453.1;9.89138; 5102.05;2.81692;
783 1500;3;copyimage;5016;4851.53;5454.87;12.4362; 5115.77;5.4466;
784 1500;3;copyimage;5016;4925.97;5443.3;10.502; 5135.06;4.24456;
785 1500;3;copyimage;5012;4929.05;5483.25;11.2436; 5104.19;3.55324;
786 1500;3;copyimage;5013;4892.67;5457.77;11.5498; 5129.31;4.83651;
787
788 1500;4;copyimage;5020;5894.81;6803.74;15.419; 6380.84;8.24497;
789 1500;4;copyimage;5016;5914.27;6814.6;15.223; 6399.03;8.19637;
790 1500;4;copyimage;5029;6013.62;6848.38;13.8813; 6415.75;6.687;
791 1500;4;copyimage;5020;5984.15;6859.59;14.6292; 6422.27;7.32121;
792 1500;4;copyimage;5021;6062.9;6898.5;13.7822; 6411.68;5.75274;
793
794 2000;1;copyimage;5015;3165.89;3520.09;11.1881; 3337.25;5.41276;
795 2000;1;copyimage;5011;3152.7;3484.22;10.5153; 3333.79;5.74407;
796 2000;1;copyimage;5005;3133.27;3459.87;10.4235; 3346.35;6.80062;
797 2000;1;copyimage;5015;3130.77;3480.48;11.1701; 3314.81;5.87826;
798 2000;1;copyimage;5004;3146.89;3462.25;10.0215; 3336.66;6.03048;
799
800 2000;2;copyimage;5015;4931.94;5583.9;13.219; 5235.82;6.1614;
801 2000;2;copyimage;5008;4921.78;5602.91;13.8392; 5262.92;6.93139;
802 2000;2;copyimage;5009;4930.31;5611.49;13.8163; 5266.32;6.81518;
803 2000;2;copyimage;5005;4920.92;5610.98;14.023; 5285.52;7.4091;
804 2000;2;copyimage;5014;4906.73;5612.52;14.3841; 5258.99;7.179;
805
806 2000;3;copyimage;5013;6707.48;7780.21;15.9931; 7185.44;7.12576;
807 2000;3;copyimage;5016;6821.46;8042.06;17.8935; 7230.24;5.99254;
808 2000;3;copyimage;5015;6839.05;7799.22;14.0396; 7065.82;3.31582;
809 2000;3;copyimage;5014;6891.23;7787.92;13.012; 7277.11;5.59953;
810 2000;3;copyimage;5016;6887.72;7813.52;13.4413; 7283.91;5.75214;
811
812 2000;4;copyimage;5011;8528.31;9899.97;16.0836; 7829.42;-8.19493;
813 2000;4;copyimage;5019;8471.64;9844.7;16.2076; 9180.83;8.37135;
814 2000;4;copyimage;5015;8494.88;10225.6;20.3742; 8077.19;-4.91694;
815 2000;4;copyimage;5014;8507.42;10252.6;20.5136; 9242.03;8.63489;
816 2000;4;copyimage;5016;8502.63;10023.1;17.8821; 9223.22;8.47498;
817
818 2200;1;copyimage;5007;3741.33;4231.81;13.1096; 3990.21;6.65205;
819 2200;1;copyimage;5010;3701.55;4014.16;8.44538; 3891.22;5.12416;
820 2200;1;copyimage;5004;3702.81;4011.24;8.32947; 3906.8;5.50887;
821 2200;1;copyimage;5009;3700.8;4017.05;8.54554; 3862.9;4.38024;
822 2200;1;copyimage;5004;3712.1;4040.42;8.84475; 3852.27;3.77619;
823
824 2200;2;copyimage;5009;5805.45;6533.71;12.5443; 6214.47;7.04536;
825 2200;2;copyimage;5006;5811.48;6708.79;15.4404; 6230.01;7.20177;
826 2200;2;copyimage;5012;6008.62;6652.77;10.7203; 6207.95;3.31734;
827 2200;2;copyimage;5006;5849.39;6690.95;14.3872; 6240.71;6.69005;
828 2200;2;copyimage;5012;5870.36;6686.1;13.896; 6242.81;6.34472;
829
830 2200;3;copyimage;5011;8103.33;9414.9;16.1855; 8823.18;8.88341;
831 2200;3;copyimage;5014;8058.06;9296.47;15.3686; 8620.3;6.97743;
832 2200;3;copyimage;5014;8050.77;9558.72;18.7306; 8584.29;6.62706;
833 2200;3;copyimage;5012;8068.55;9355.72;15.9529; 8654.31;7.2598;
834 2200;3;copyimage;5005;8013.78;9355.64;16.7444; 8699.12;8.552;
835
836 2200;4;copyimage;5013;10120.4;11850;17.0907; 11032.9;9.01661;
837 2200;4;copyimage;5025;10232.8;12144.8;18.6857; 11290.2;10.3342;
838 2200;4;copyimage;5014;10333.8;12210.2;18.1569; 11277.4;9.13092;
839 2200;4;copyimage;5015;10274.7;11901.5;15.8327; 11370.4;10.6636;
840 2200;4;copyimage;5010;10315.5;9603.97;-6.89723; 11190.2;8.47973;
841
842 2400;1;copyimage;5013;4312.41;4829.28;11.9857; 4607.01;6.83152;
843 2400;1;copyimage;5003;4271.66;4772.45;11.7237; 4587.77;7.40026;
844 2400;1;copyimage;5003;4287.54;4808.09;12.1409; 4581.68;6.86029;
845 2400;1;copyimage;5009;4302.94;4649.16;8.04614; 4533.78;5.3647;
846 2400;1;copyimage;5012;4291.06;4772.45;11.2184; 4520.78;5.35344;
847
848 2400;2;copyimage;5014;6905.24;7833.59;13.444; 7281.38;5.44717;
849 2400;2;copyimage;5005;6935.5;7643.2;10.204; 7324.2;5.60445;
850 2400;2;copyimage;5013;6956.97;7756.65;11.4947; 7298.88;4.91471;
851 2400;2;copyimage;5011;6942.95;7885.11;13.5701; 7347.57;5.82786;

852 2400;2;copyimage;5011;6960.73;7900.28;13.4979; 7306.5;4.96738;
853
854 2400;3;copyimage;5011;9542.73;11189.6;17.2579; 10219.3;7.09016;
855 2400;3;copyimage;5011;9652.52;11378.3;17.8787; 9779.25;1.31294;
856 2400;3;copyimage;5006;9638.91;11133.7;15.5084; 10313.4;6.99758;
857 2400;3;copyimage;5012;9650.46;11235.2;16.4212; 10323.9;6.97866;
858 2400;3;copyimage;5011;9728.86;11365.3;16.82; 10357;6.45635;
859
860 2400;4;copyimage;5008;11871;13914.3;17.213; 12926.6;8.89239;
861 2400;4;copyimage;5024;11900.8;14041.9;17.9916; 12925.9;8.61382;
862 2400;4;copyimage;5021;11833.4;14139.8;19.4907; 12824.5;8.3758;
863 2400;4;copyimage;5005;11813.8;14062.8;19.0371; 13255.8;12.2062;
864 2400;4;copyimage;5012;11898.9;14218.6;19.4943; 13142.4;10.4501;
865
866 2500;1;copyimage;5006;4609.15;5326.93;15.5728; 4950.21;7.39959;
867 2500;1;copyimage;5003;4594.34;5098.86;10.9814; 4895.81;6.5617;
868 2500;1;copyimage;5004;4591.63;5114.28;11.3828; 4872.36;6.11392;
869 2500;1;copyimage;5003;4599.49;5080.72;10.4627; 4886.57;6.24154;
870 2500;1;copyimage;5006;4580.33;5078.28;10.8715; 4872;6.36785;
871
872 2500;2;copyimage;5010;7511.73;8462.37;12.6555; 7889.27;5.02605;
873 2500;2;copyimage;5009;7547.26;8603.15;13.9903; 7959.57;5.46301;
874 2500;2;copyimage;5007;7593.26;8765.31;15.4354; 7967.79;4.9324;
875 2500;2;copyimage;5006;7535.32;8626.42;14.4798; 7981.03;5.91493;
876 2500;2;copyimage;5007;7555.85;8517.82;12.7316; 8009.13;5.99909;
877
878 2500;3;copyimage;5005;10428.5;12092.8;15.9594; 11140.9;6.83195;
879 2500;3;copyimage;5009;10500.8;12235.3;16.5179; 11201.5;6.67358;
880 2500;3;copyimage;5013;10445.2;12066.8;15.5251; 11461.1;9.72621;
881 2500;3;copyimage;5011;10414.1;12043.9;15.6505; 11238.9;7.92008;
882 2500;3;copyimage;5010;10363.2;12190.3;17.6307; 9995.79;-3.54494;
883
884 2500;4;copyimage;5024;12809.2;15107.2;17.9402; 13885.6;8.40365;
885 2500;4;copyimage;5006;13005.4;15377.2;18.2365; 13962.8;7.36119;
886 2500;4;copyimage;5006;13084.9;15333.8;17.1867; 14274.6;9.09151;
887 2500;4;copyimage;5016;13065.1;15366.4;17.614; 14119.3;8.06918;
888 2500;4;copyimage;5028;12999;15365.3;18.2038; 14110.6;8.55173;
889
890 3000;1;copyimage;5002;6677.08;7581.91;13.5513; 7033.87;5.34349;
891 3000;1;copyimage;5004;6597.06;7416.07;12.4148; 6979.98;5.80436;
892 3000;1;copyimage;5007;6563.16;7406.73;12.8532; 6979.4;6.34209;
893 3000;1;copyimage;5010;6604.08;7419.46;12.3465; 6902.52;4.51896;
894 3000;1;copyimage;5003;6535.33;7409.71;13.3792; 6978.63;6.78305;
895
896 3000;2;copyimage;5002;10937.7;12665.8;15.7996; 11421.8;4.42625;
897 3000;2;copyimage;5005;10927.8;12520.1;14.5704; 11586.5;6.02732;
898 3000;2;copyimage;5010;10933.8;12559.6;14.8691; 11615.8;6.23788;
899 3000;2;copyimage;5010;11049.2;12568.4;13.7491; 11665.5;5.5779;
900 3000;2;copyimage;5010;11041.8;12656.6;14.6252; 11656.9;5.57094;
901
902 3000;3;copyimage;5011;15043.9;17381.8;15.5405; 16376.4;8.8574;
903 3000;3;copyimage;5010;15251.1;17819.1;16.8381; 16199.2;6.21668;
904 3000;3;copyimage;5006;15164.2;17964.8;18.4686; 16372.5;7.96846;
905 3000;3;copyimage;5008;15169.9;17767.1;17.1203; 13984.3;-7.81587;
906 3000;3;copyimage;5011;15292.4;18041.1;17.9745; 16585.2;8.45381;
907
908 3000;4;copyimage;5016;19162.8;21763.4;13.5709; 20135.6;5.07649;
909 3000;4;copyimage;5011;19422.6;23501;20.9983; 21320.1;9.76935;
910 3000;4;copyimage;5007;19556.7;23558.3;20.4613; 21568.6;10.2873;
911 3000;4;copyimage;5006;19547.8;23542.8;20.4372; 19098.8;-2.29697;
912 3000;4;copyimage;5013;19721.9;23448.8;18.8973; 21493.7;8.98388;
913
914 3500;1;copyimage;5004;9464.58;10961.8;15.8194; 10332.2;9.1673;
915 3500;1;copyimage;5007;9520.47;10845.5;13.9175; 10093;6.0134;
916 3500;1;copyimage;5004;9436.8;10607.3;12.4038; 9997.69;5.9436;
917 3500;1;copyimage;5006;9434.7;10473.3;11.0083; 9921.03;5.1547;
918 3500;1;copyimage;5006;9353.53;10397.5;11.1608; 9843.55;5.23879;
919
920 3500;2;copyimage;5005;15475.3;17634.2;13.9505; 16663.5;7.67842;
921 3500;2;copyimage;5009;15534.5;18025.1;16.0324; 16435;5.79627;
922 3500;2;copyimage;5008;15801.9;18168.2;14.9746; 16859;6.68942;
923 3500;2;copyimage;5006;15765.9;18223.5;15.5878; 16755.5;6.27667;
924 3500;2;copyimage;5006;15768;18147.8;15.0923; 16442.3;4.27627;
925
926 3500;3;copyimage;5007;22271.2;25947;16.5047; 24095.8;8.19246;
927 3500;3;copyimage;5003;22467.1;26385.5;17.4406; 24294.6;8.13433;
928 3500;3;copyimage;5009;22448.4;26240.2;16.8911; 24412.5;8.74955;
929 3500;3;copyimage;5004;22573.4;26353.8;16.7476; 24477.4;8.43512;
930 3500;3;copyimage;5008;22795.1;26237.8;15.1029; 24510.9;7.5274;
931
932 3500;4;copyimage;5011;28775.5;32917.8;14.3953; 31835.4;10.6339;
933 3500;4;copyimage;5005;25653.5;24910.9;-2.8949; 30717.9;19.7417;
934 3500;4;copyimage;5016;25699.2;24949.5;-2.91706; 25769.2;0.272496;
935 3500;4;copyimage;5011;25727.1;24901.8;-3.20778; 25880.3;0.595471;
936 3500;4;copyimage;5005;25815.1;24912.1;-3.49784; 25854.5;0.152716;
937
938 4000;1;copyimage;5003;13670;14874.4;8.81016; 14640.2;7.09662;

939 4000;1;copyimage;5002;13437.2;15268.9;13.6316; 14361.9;6.88218;
940 4000;1;copyimage;5009;13402;15108.8;12.7358; 14150.1;5.58207;
941 4000;1;copyimage;5002;13031.4;14996;15.0756; 14089.3;8.11776;
942 4000;1;copyimage;5004;13185.6;14889.4;12.9219; 13921.5;5.58144;
943
944 4000;2;copyimage;5008;21983.3;25102.9;14.1909; 23153.6;5.32368;
945 4000;2;copyimage;5007;22377.7;25801.4;15.2995; 23791;6.31579;
946 4000;2;copyimage;5002;22279.9;25950;16.4724; 24047.6;7.93416;
947 4000;2;copyimage;5005;22522.3;25625.3;13.7772; 23694.6;5.20478;
948 4000;2;copyimage;5009;22352.6;25281.9;13.105; 24109.8;7.86129;
949
950 4000;3;copyimage;5007;28248.3;30796.4;9.02033; 30420.5;7.68964;
951 4000;3;copyimage;5008;27748;26481.9;-4.56258; 29120.6;4.947;
952 4000;3;copyimage;5007;25727.2;24865.9;-3.34787; 25849.5;0.475459;
953 4000;3;copyimage;5007;26055.6;24893;-4.46215; 25409.5;-2.48;
954 4000;3;copyimage;5007;25836.1;24903.8;-3.60831; 25403.5;-1.67417;
955
956 4000;4;copyimage;5007;25692.5;24925;-2.98724; 25488.9;-0.792334;
957 4000;4;copyimage;5005;25702.3;24991.3;-2.76636; 25613.8;-0.344175;
958 4000;4;copyimage;5006;25588.2;24896.8;-2.70206; 25712.9;0.487231;
959 4000;4;copyimage;5012;25854.6;25019.3;-3.23077; 25535.8;-1.23308;
960 4000;4;copyimage;5012;25727.8;24978;-2.91423; 25677.6;-0.195208;
961
962 1500;1;sobelh;5104;2630.59;2793.24;6.18312; 2796.2;6.29547;
963 1500;1;sobelh;5116;2520.82;2572.91;2.06634; 2731.05;8.33964;
964 1500;1;sobelh;5125;2433.9;2569.83;5.5848; 2641.82;8.54298;
965 1500;1;sobelh;5139;2385.86;2571.53;7.7819; 2523.65;5.77516;
966 1500;1;sobelh;5130;2391.12;2550.01;6.64481; 2492.25;4.22947;
967
968 1500;2;sobelh;5077;3146.47;3935.26;25.069; 3977.41;26.4085;
969 1500;2;sobelh;5078;3150.65;4076.28;29.3792; 3940.59;25.0726;
970 1500;2;sobelh;5212;3539.54;4046.51;14.3231; 3936.81;11.2237;
971 1500;2;sobelh;5445;3133.79;4054.92;29.3935; 3922.93;25.1817;
972 1500;2;sobelh;5424;3168.19;4076.92;28.6828; 3961.96;25.0542;
973
974 1500;3;sobelh;5227;4131.98;5453.1;31.9732; 5298.37;28.2284;
975 1500;3;sobelh;5234;4125.39;5454.87;32.2268; 5276.07;27.8925;
976 1500;3;sobelh;5231;4102.61;5443.3;32.6788; 5259.02;28.1871;
977 1500;3;sobelh;5174;4253.3;5483.25;28.9175; 5263.22;23.7442;
978 1500;3;sobelh;5059;4196.5;5457.77;30.0551; 5170.78;23.2164;
979
980 1500;4;sobelh;5169;5565.93;6803.74;22.2389; 6783.68;21.8786;
981 1500;4;sobelh;5175;5553.58;6814.6;22.7064; 6578.31;18.4517;
982 1500;4;sobelh;5365;5095.5;6848.38;34.4007; 6589.07;29.3116;
983 1500;4;sobelh;5371;5229.46;6859.59;31.1719; 6694.99;28.0243;
984 1500;4;sobelh;5349;5277.79;6898.5;30.7082; 6696.54;26.8817;
985
986 2000;1;sobelh;5103;3256;3520.09;8.11071; 3413.06;4.82354;
987 2000;1;sobelh;5117;3243.87;3484.22;7.40918; 3391.88;4.56279;
988 2000;1;sobelh;5104;3251.39;3459.87;6.41185; 3425.68;5.36038;
989 2000;1;sobelh;5087;3251.54;3480.48;7.04095; 3415.57;5.04463;
990 2000;1;sobelh;5109;3243.34;3462.25;6.74971; 3379.79;4.20718;
991
992 2000;2;sobelh;5109;4398.14;5583.9;26.9604; 5362.92;21.9361;
993 2000;2;sobelh;5010;4426.29;5602.91;26.5825; 5304.8;19.8474;
994 2000;2;sobelh;5016;4423.59;5611.49;26.8539; 5346.83;20.8709;
995 2000;2;sobelh;5018;4425.13;5610.98;26.798; 5379.25;21.5612;
996 2000;2;sobelh;5072;5048.22;5612.52;11.1783; 5261.91;4.23311;
997
998 2000;3;sobelh;5232;5941.91;7780.21;30.938; 7580.35;27.5743;
999 2000;3;sobelh;5196;6014.31;8042.06;33.7155; 7468.25;24.1747;
1000 2000;3;sobelh;5210;5938.72;7799.22;31.3284; 7631.97;28.512;
1001 2000;3;sobelh;5288;5819.37;7787.92;33.8276; 7212.11;23.933;
1002 2000;3;sobelh;5181;6180.04;7813.52;26.4315; 7523.93;21.7457;
1003
1004 2000;4;sobelh;5010;7411.76;9899.97;33.5712; 9599.99;29.5239;
1005 2000;4;sobelh;5355;7438.69;9844.7;32.3445; 8824.45;18.6291;
1006 2000;4;sobelh;5283;7484.72;10225.6;36.6202; 9668.85;29.1813;
1007 2000;4;sobelh;5051;7319.55;10252.6;40.0714; 9649.64;31.8338;
1008 2000;4;sobelh;5192;7916.49;10023.1;26.61; 9386.37;18.5673;
1009
1010 2200;1;sobelh;5066;3801.02;4231.81;11.3334; 4041.61;6.32945;
1011 2200;1;sobelh;5032;3762.06;4014.16;6.7011; 3985.81;5.94753;
1012 2200;1;sobelh;5031;3770.89;4011.24;6.37393; 4004.13;6.1853;
1013 2200;1;sobelh;5033;3759.96;4017.05;6.83752; 3980.34;5.8611;
1014 2200;1;sobelh;5065;3738.6;4040.42;8.07326; 3941.67;5.43185;
1015
1016 2200;2;sobelh;5214;5138.4;6533.71;27.1545; 6337.03;23.3269;
1017 2200;2;sobelh;5194;5177.39;6708.79;29.5788; 6183.94;19.4412;
1018 2200;2;sobelh;5024;5971.08;6652.77;11.4165; 6401.61;7.2103;
1019 2200;2;sobelh;5224;5202.77;6690.95;28.6037; 6351.6;22.0811;
1020 2200;2;sobelh;5248;5107.65;6686.1;30.9038; 6586.57;28.9551;
1021
1022 2200;3;sobelh;5084;7036.62;9414.9;33.7985; 9004.21;27.962;
1023 2200;3;sobelh;5109;6970.34;9296.47;33.3717; 8948.34;28.3774;
1024 2200;3;sobelh;5061;7487.03;9558.72;27.6704; 8933.95;19.3257;
1025 2200;3;sobelh;5087;6981.36;9355.72;34.01; 9038.78;29.4702;

1026 2200;3;sobelh;5201;7346.88;9355.64;27.3416; 8886.34;20.9539;
1027
1028 2200;4;sobelh;5217;8755.19;11850;35.3483; 11525.9;31.6466;
1029 2200;4;sobelh;5221;8819.58;12144.8;37.703; 11201.9;27.0117;
1030 2200;4;sobelh;5076;8926.61;12210.2;36.7837; 11232.5;25.8319;
1031 2200;4;sobelh;5201;8926.79;11901.5;33.3235; 11560.8;29.5066;
1032 2200;4;sobelh;5178;9085.45;9603.97;5.70721; 11643.4;28.1548;
1033
1034 2400;1;sobelh;5010;4514.86;4829.28;6.9642; 4721.13;4.5687;
1035 2400;1;sobelh;5020;4452.05;4772.45;7.19666; 4641.48;4.25493;
1036 2400;1;sobelh;5138;4382.37;4808.09;9.7144; 4627.29;5.58877;
1037 2400;1;sobelh;5026;4348.3;4649.16;6.91894; 4685.18;7.74741;
1038 2400;1;sobelh;5020;4364.28;4772.45;9.35239; 4664.38;6.87613;
1039
1040 2400;2;sobelh;5006;6136.92;7833.59;27.6469; 7543.58;22.9214;
1041 2400;2;sobelh;5095;6103.02;7643.2;25.2363; 7480.47;22.5699;
1042 2400;2;sobelh;5039;6122.24;7756.65;26.6963; 7448.57;21.6642;
1043 2400;2;sobelh;5004;6159.07;7885.11;28.0244; 7639.83;24.0419;
1044 2400;2;sobelh;5022;6164.58;7900.28;28.156; 7700.09;24.9085;
1045
1046 2400;3;sobelh;5240;8330.51;11189.6;34.3207; 10414.2;25.0122;
1047 2400;3;sobelh;5235;8304.67;11378.3;37.0105; 8195.92;-1.30942;
1048 2400;3;sobelh;5203;8446.71;11133.7;31.8116; 10710;26.7952;
1049 2400;3;sobelh;5247;8455.95;11235.2;32.8672; 10803.7;27.7643;
1050 2400;3;sobelh;5267;8407.95;11365.3;35.1727; 10532.7;25.2712;
1051
1052 2400;4;sobelh;5259;11492.8;13914.3;21.0697; 13441.1;16.9527;
1053 2400;4;sobelh;5232;11226.4;14041.9;25.0798; 13517.1;20.4045;
1054 2400;4;sobelh;5267;11059.5;14139.8;27.8515; 13624.3;23.1909;
1055 2400;4;sobelh;5073;10529;14062.8;33.5624; 12873.7;22.2682;
1056 2400;4;sobelh;5233;10722.1;14218.6;32.6096; 13430.9;25.2636;
1057
1058 2500;1;sobelh;5010;4806.94;5326.93;10.8175; 5075.18;5.58019;
1059 2500;1;sobelh;5167;4765.95;5098.86;6.98515; 5020.89;5.34916;
1060 2500;1;sobelh;5145;4711.95;5114.28;8.53856; 4948.4;5.01799;
1061 2500;1;sobelh;5169;4763.97;5080.72;6.64893; 4960.08;4.11667;
1062 2500;1;sobelh;5139;4646.57;5078.28;9.29102; 4986.43;7.31424;
1063
1064 2500;2;sobelh;5061;6568.26;8462.37;28.8373; 8133.33;23.8276;
1065 2500;2;sobelh;5164;7104.03;8603.15;21.1024; 8082.53;13.7739;
1066 2500;2;sobelh;5058;6618.53;8765.31;32.4358; 8069.65;21.925;
1067 2500;2;sobelh;5136;6602.2;8626.42;30.6598; 8214.65;24.4229;
1068 2500;2;sobelh;5143;7363.03;8517.82;15.6836; 8329.73;13.129;
1069
1070 2500;3;sobelh;5121;9351.05;12092.8;29.3201; 11331.9;21.1833;
1071 2500;3;sobelh;5127;8896.36;12235.3;37.531; 11581.6;30.1839;
1072 2500;3;sobelh;5264;9972.35;12066.8;21.003; 9954.16;-0.18233;
1073 2500;3;sobelh;5163;9828.77;12043.9;22.5376; 11611.9;18.1421;
1074 2500;3;sobelh;5118;8970.73;12190.3;35.8893; 11647;29.8331;
1075
1076 2500;4;sobelh;5132;11942;15107.2;26.5048; 14491.9;21.3529;
1077 2500;4;sobelh;5162;11423.9;15377.2;34.6047; 14463.4;26.6058;
1078 2500;4;sobelh;5167;11374.7;15333.8;34.8062; 14589.7;28.2641;
1079 2500;4;sobelh;5163;11474.8;15366.4;33.9138; 14590.3;27.1509;
1080 2500;4;sobelh;5175;11411.5;15365.3;34.6481; 14572.4;27.6993;
1081
1082 3000;1;sobelh;5066;6786.48;7581.91;11.7209; 7159.73;5.49992;
1083 3000;1;sobelh;5047;6789.94;7416.07;9.22148; 6981.27;2.81783;
1084 3000;1;sobelh;5064;6782.95;7406.73;9.19637; 7033.57;3.6949;
1085 3000;1;sobelh;5063;6644.17;7419.46;11.6687; 7018.98;5.64111;
1086 3000;1;sobelh;5073;6729.92;7409.71;10.1009; 7008.24;4.1355;
1087
1088 3000;2;sobelh;5132;9544.66;12665.8;32.7006; 11667.3;22.2395;
1089 3000;2;sobelh;5213;9547.9;12520.1;31.1289; 11646.1;21.976;
1090 3000;2;sobelh;5156;9663.09;12559.6;29.9746; 12073.1;24.9406;
1091 3000;2;sobelh;5150;9587.24;12568.4;31.0948; 12048;25.667;
1092 3000;2;sobelh;5117;9618.7;12656.6;31.5837; 12055.5;25.3339;
1093
1094 3000;3;sobelh;5120;13173.6;17381.8;31.944; 16498.6;25.2403;
1095 3000;3;sobelh;5038;14134.4;17819.1;26.0695; 16633.7;17.683;
1096 3000;3;sobelh;5084;13386.7;17964.8;34.1992; 16725.4;24.941;
1097 3000;3;sobelh;5155;13319.5;17767.1;33.3918; 16776;25.951;
1098 3000;3;sobelh;5190;13285.6;18041.1;35.7949; 16769;26.2196;
1099
1100 3000;4;sobelh;5138;17509.7;21763.4;24.293; 20385.9;16.4264;
1101 3000;4;sobelh;5157;17460.4;23501;34.5963; 21386.4;22.4851;
1102 3000;4;sobelh;5059;16618.9;23558.3;41.7558; 21478.7;29.2423;
1103 3000;4;sobelh;5038;16720.2;23542.8;40.8045; 21783.7;30.2838;
1104 3000;4;sobelh;5133;17702.3;23448.8;32.4624; 21635;22.216;
1105
1106 3500;1;sobelh;5046;9786.05;10961.8;12.0148; 10303.7;5.28948;
1107 3500;1;sobelh;5034;9760.41;10845.5;11.1171; 10107.5;3.55647;
1108 3500;1;sobelh;5054;9572.24;10607.3;10.8133; 9850.83;2.91041;
1109 3500;1;sobelh;5028;9420.83;10473.3;11.1717; 9923.88;5.33975;
1110 3500;1;sobelh;5049;9479.15;10397.5;9.68776; 9705.79;2.39094;
1111
1112 3500;2;sobelh;5171;13743.5;17634.2;28.3088; 16558.7;20.4834;

1113 3500;2;sobelh;5097;14812.4;18025.1;21.6891; 17022.6;14.9207;
1114 3500;2;sobelh;5038;15576.6;18168.2;16.6377; 17086.3;9.6925;
1115 3500;2;sobelh;5112;14888.1;18223.5;22.4026; 17054.9;14.5534;
1116 3500;2;sobelh;5032;16000.9;18147.8;13.4175; 17125.5;7.02874;
1117
1118 3500;3;sobelh;5136;19618.7;25947.32.2563; 24137.7;23.034;
1119 3500;3;sobelh;5073;18868.4;26385.5;39.8393; 22626.3;19.9164;
1120 3500;3;sobelh;5115;19860.1;26240.2;32.1251; 24478.5;23.2545;
1121 3500;3;sobelh;5035;19171.7;26353.8;37.462; 24350.2;27.0108;
1122 3500;3;sobelh;5113;20434.3;26237.8;28.4005; 24662.9;20.6933;
1123
1124 3500;4;sobelh;5167;24581.3;32917.8;33.914; 31694.2;28.9364;
1125 3500;4;sobelh;5034;26460.3;24910.9;-5.85574; 32162.4;21.5497;
1126 3500;4;sobelh;5160;24914.3;24949.5;0.141556; 27833.2;11.7159;
1127 3500;4;sobelh;5146;25589.4;24901.8;-2.68691; 25581;-0.0326216;
1128 3500;4;sobelh;5156;24955.1;24912.1;-0.172388; 25028;0.292204;
1129
1130 4000;1;sobelh;5093;13670;14874.4;8.81049; 14500.8;6.07748;
1131 4000;1;sobelh;5110;13623.2;15268.9;12.08; 14193.6;4.18736;
1132 4000;1;sobelh;5088;13351.7;15108.8;13.16; 14069.7;5.37755;
1133 4000;1;sobelh;5058;13301.3;14996;12.7409; 13955.9;4.92159;
1134 4000;1;sobelh;5022;13401.8;14889.4;11.0997; 13947.2;4.06961;
1135
1136 4000;2;sobelh;5105;19067.7;25102.9;31.6514; 23525.7;23.3794;
1137 4000;2;sobelh;5071;19371.6;25801.4;33.1916; 23785.2;22.784;
1138 4000;2;sobelh;5027;19751.7;25950;31.3807; 24044.2;21.7323;
1139 4000;2;sobelh;5154;19455.2;25625.3;31.7144; 24198.2;24.3794;
1140 4000;2;sobelh;5124;19436.9;25281.9;30.0715; 24155;24.2739;
1141
1142 4000;3;sobelh;5091;25299.6;30796.4;21.7265; 30547.9;20.7445;
1143 4000;3;sobelh;5027;23848.1;26481.9;11.0441; 30197.3;26.6233;
1144 4000;3;sobelh;5021;23910.2;24865.9;3.99677; 28953.2;21.0912;
1145 4000;3;sobelh;5063;23759.2;24893;4.77219; 28329.2;19.2347;
1146 4000;3;sobelh;5039;25125.4;24903.8;-0.881773; 26590.1;5.82951;
1147
1148 4000;4;sobelh;5089;25416;24925;-1.93182; 25457.1;0.161677;
1149 4000;4;sobelh;5106;25502.1;24991.3;-2.00296; 25081.8;-1.64814;
1150 4000;4;sobelh;5206;25663.8;24896.8;-2.98855; 24942.8;-2.80923;
1151 4000;4;sobelh;5116;25545.5;25019.3;-2.05997; 25055.6;-1.91792;
1152 4000;4;sobelh;5105;25715.2;24978;-2.8668; 25008;-2.75026;
1153
1154 1500;1;sobelv;5114;2647.87;2793.24;5.4901; 2786.68;5.24219;
1155 1500;1;sobelv;5099;2459.3;2572.91;4.61974; 2626.95;6.81699;
1156 1500;1;sobelv;5118;2430.34;2569.83;5.73911; 2489.5;2.43389;
1157 1500;1;sobelv;5108;2274.61;2571.53;13.0539; 2492.52;9.58029;
1158 1500;1;sobelv;5076;2408.78;2550.01;5.86306; 2499.64;3.77202;
1159
1160 1500;2;sobelv;5435;3129.01;3935.26;25.7668; 3895.93;24.5098;
1161 1500;2;sobelv;5445;3124.2;4076.28;30.4746; 3854.64;23.3802;
1162 1500;2;sobelv;5064;3108.35;4046.51;30.1819; 3953.9;27.2026;
1163 1500;2;sobelv;5041;3104.82;4054.92;30.601; 3869.89;24.6414;
1164 1500;2;sobelv;5044;3126.19;4076.92;30.4117; 3955.11;26.5153;
1165
1166 1500;3;sobelv;5241;4156.27;5453.1;31.2018; 5273.03;26.8692;
1167 1500;3;sobelv;5249;4143.01;5454.87;31.6645; 5259.89;26.9583;
1168 1500;3;sobelv;5217;4132.67;5443.3;31.7139; 5272.19;27.5734;
1169 1500;3;sobelv;5185;4152.25;5483.25;32.0549; 5284.46;27.2674;
1170 1500;3;sobelv;5197;4331.14;5457.77;26.0124; 5279.56;21.8978;
1171
1172 1500;4;sobelv;5315;5185.29;6803.74;31.2123; 6843.32;31.9757;
1173 1500;4;sobelv;5357;5110.69;6814.6;33.3401; 6599.95;29.14;
1174 1500;4;sobelv;5276;5252.19;6848.38;30.391; 5503.05;4.77629;
1175 1500;4;sobelv;5295;5174.82;6859.59;32.557; 6879.34;32.9386;
1176 1500;4;sobelv;5356;5113.68;6898.5;34.903; 6687.78;30.7822;
1177
1178 2000;1;sobelv;5099;3243.28;3520.09;8.53469; 3416.25;5.33312;
1179 2000;1;sobelv;5088;3213.31;3484.22;8.43075; 3414.34;6.25606;
1180 2000;1;sobelv;5085;3203.22;3459.87;8.01235; 3386.43;5.71975;
1181 2000;1;sobelv;5074;3236.49;3480.48;7.53882; 3381.57;4.48267;
1182 2000;1;sobelv;5093;3228.37;3462.25;7.24457; 3422.77;6.02167;
1183
1184 2000;2;sobelv;5083;4298.67;5583.9;29.8981; 5385.81;25.29;
1185 2000;2;sobelv;5090;4314.98;5602.91;29.8478; 5359.16;24.199;
1186 2000;2;sobelv;5074;4338.93;5611.49;29.3289; 5396.16;24.3661;
1187 2000;2;sobelv;5046;4359.01;5610.98;28.7216; 5387.97;23.6055;
1188 2000;2;sobelv;5333;4340.46;5612.52;29.3072; 5384.93;24.0638;
1189
1190 2000;3;sobelv;5203;5838.62;7780.21;33.2543; 7338.09;25.6819;
1191 2000;3;sobelv;5268;5702.98;8042.06;41.015; 7638.68;33.9418;
1192 2000;3;sobelv;5139;6077.38;7799.22;28.3319; 7700.91;26.7142;
1193 2000;3;sobelv;5209;5791;7787.92;34.4831; 7647.19;32.0529;
1194 2000;3;sobelv;5131;6071.87;7813.52;28.6839; 6508.14;7.18517;
1195
1196 2000;4;sobelv;5360;7270.48;9899.97;36.1668; 9620.6;32.3242;
1197 2000;4;sobelv;5046;7160.24;9844.7;37.4911; 9712.89;35.6503;
1198 2000;4;sobelv;5042;7229.42;10225.6;41.4449; 9338.84;29.1783;
1199 2000;4;sobelv;5221;7776.71;10252.6;31.8372; 9683.5;24.5192;

1200 2000;4;sobelv;5024;7327.97;10023.1;36.7784; 9648.85;31.6716;
1201
1202 2200;1;sobelv;5062;3789.89;4231.81;11.6606; 4045.93;6.75592;
1203 2200;1;sobelv;5044;3799.42;4014.16;5.65193; 3982.21;4.81115;
1204 2200;1;sobelv;5086;3774.77;4011.24;6.26437; 3955.52;4.78816;
1205 2200;1;sobelv;5023;3738.04;4017.05;7.46403; 3987.64;6.67729;
1206 2200;1;sobelv;5045;3708.51;4040.42;8.95007; 3885.24;4.76547;
1207
1208 2200;2;sobelv;5234;5115.14;6533.71;27.7328; 6336.03;23.8683;
1209 2200;2;sobelv;5196;5159.67;6708.79;30.0238; 6631.8;28.5315;
1210 2200;2;sobelv;5210;5400.52;6652.77;23.1875; 6392.45;18.3674;
1211 2200;2;sobelv;5170;5450.17;6690.95;22.766; 6380.6;17.0717;
1212 2200;2;sobelv;5195;5105.94;6686.1;30.9474; 6347.87;24.3231;
1213
1214 2200;3;sobelv;5242;7235.41;9414.9;30.1226; 8881.39;22.749;
1215 2200;3;sobelv;5058;6972.89;9296.47;33.323; 8798.02;26.1746;
1216 2200;3;sobelv;5140;6785.01;9558.72;40.8799; 8882.15;30.9084;
1217 2200;3;sobelv;5184;7292.65;9355.72;28.2898; 8919.61;22.3097;
1218 2200;3;sobelv;5150;6802.08;9355.64;37.5408; 9116.48;34.0248;
1219
1220 2200;4;sobelv;5199;8768.31;11850.35;1459; 11513.3;31.3063;
1221 2200;4;sobelv;5238;8657.9;12144.8;40.2744; 11441.7;32.1534;
1222 2200;4;sobelv;5215;8724.99;12210.2;39.9447; 11522.9;32.0684;
1223 2200;4;sobelv;5212;8794.83;11901.5;35.324; 11548.8;31.3132;
1224 2200;4;sobelv;5093;8830.55;9603.97;8.75842; 11190.7;26.7266;
1225
1226 2400;1;sobelv;5011;4460.9;4829.28;8.25801; 4750.62;6.49466;
1227 2400;1;sobelv;5167;4358.76;4772.45;9.49096; 4544.82;4.26846;
1228 2400;1;sobelv;5037;4339.36;4808.09;10.8017; 4634.37;6.79834;
1229 2400;1;sobelv;5010;4334.9;4649.16;7.24951; 4657.54;7.44286;
1230 2400;1;sobelv;5153;4341.99;4772.45;9.91382; 4486.62;3.33092;
1231
1232 2400;2;sobelv;5007;5990.49;7833.59;30.7671; 7520.61;25.5425;
1233 2400;2;sobelv;5012;5949.24;7643.2;28.4734; 7370.21;23.8848;
1234 2400;2;sobelv;5107;6002.33;7756.65;29.2273; 7422.36;23.6579;
1235 2400;2;sobelv;5071;6073.39;7885.11;29.8306; 7409.91;22.0063;
1236 2400;2;sobelv;5010;6383.98;7900.28;23.7517; 7489.83;17.3223;
1237
1238 2400;3;sobelv;5044;8751.77;11189.6;27.8552; 10671.7;21.9381;
1239 2400;3;sobelv;5063;8684.94;11378.3;31.0114; 10710.6;23.3238;
1240 2400;3;sobelv;5264;8283.47;11133.7;34.4092; 10760.7;29.9055;
1241 2400;3;sobelv;5283;8268.91;11235.2;35.8726; 10796.9;30.5723;
1242 2400;3;sobelv;5138;8917.56;11365.3;27.448; 10698;19.9657;
1243
1244 2400;4;sobelv;5075;10406.6;13914.3;33.7066; 13563.6;30.3362;
1245 2400;4;sobelv;5079;10351.8;14041.9;35.6475; 13503.6;30.4476;
1246 2400;4;sobelv;5259;10389.3;14139.8;36.0992; 13516.7;30.102;
1247 2400;4;sobelv;5092;10347.2;14062.8;35.91; 13515.3;30.6186;
1248 2400;4;sobelv;5092;10437.4;14218.6;36.2267; 13539.1;29.717;
1249
1250 2500;1;sobelv;5147;4800.1;5326.93;10.9754; 5063.7;5.49149;
1251 2500;1;sobelv;5142;4641.6;5098.86;9.85125; 4928.87;6.18894;
1252 2500;1;sobelv;5170;4643.11;5114.28;10.1478; 4994.92;7.57701;
1253 2500;1;sobelv;5126;4625.79;5080.72;9.83464; 4930.24;6.58166;
1254 2500;1;sobelv;5136;4618.05;5078.28;9.9659; 4965.19;7.51695;
1255
1256 2500;2;sobelv;5146;6488.79;8462.37;30.4152; 8042.8;23.949;
1257 2500;2;sobelv;5101;6485.01;8603.15;32.662; 8312.49;28.1799;
1258 2500;2;sobelv;5135;6489.49;8765.31;35.0693; 8055.87;24.1372;
1259 2500;2;sobelv;5141;6503.76;8626.42;32.6374; 8056.48;23.8741;
1260 2500;2;sobelv;5138;6496.98;8517.82;31.1042; 8079.48;24.3574;
1261
1262 2500;3;sobelv;5118;8759.61;12092.8;38.0517; 11464.7;30.8817;
1263 2500;3;sobelv;5037;8932.16;12235.3;36.9799; 11301;26.5203;
1264 2500;3;sobelv;5045;8858.54;12066.8;36.2169; 11428.8;29.0148;
1265 2500;3;sobelv;5019;8868.11;12043.9;35.8118; 11294;27.355;
1266 2500;3;sobelv;5122;9670.25;12190.3;26.0594; 11598.1;19.9355;
1267
1268 2500;4;sobelv;5137;11331.1;15107.2;33.3249; 14541.6;28.3334;
1269 2500;4;sobelv;5173;11206.9;15377.2;37.2111; 14133.1;26.1099;
1270 2500;4;sobelv;5167;11229.9;15333.8;36.5447; 14631.4;30.2899;
1271 2500;4;sobelv;5166;11246.3;15366.4;36.6346; 14635.2;30.1332;
1272 2500;4;sobelv;5157;11281;15365.3;36.2047; 14594.9;29.3758;
1273
1274 3000;1;sobelv;5080;6705.8;7581.91;13.065; 7145.03;6.54997;
1275 3000;1;sobelv;5074;6682.43;7416.07;10.9787; 7055.28;5.57957;
1276 3000;1;sobelv;5096;6670.41;7406.73;11.0386; 7010;5.09107;
1277 3000;1;sobelv;5098;6637.9;7419.46;11.7743; 7274.12;9.58473;
1278 3000;1;sobelv;5081;6638.41;7409.71;11.6187; 6875.4;3.56996;
1279
1280 3000;2;sobelv;5113;9543.25;12665.8;32.7203; 11805.1;23.701;
1281 3000;2;sobelv;5077;10181.9;12520.1;22.9639; 11947.1;17.3366;
1282 3000;2;sobelv;5126;9502.3;12559.6;32.1739; 12025.7;26.5557;
1283 3000;2;sobelv;5038;10057.7;12568.4;24.9629; 11794.7;17.2703;
1284 3000;2;sobelv;5218;9407.86;12656.6;34.5326; 12075.6;28.3564;
1285
1286 3000;3;sobelv;5119;13010.8;17381.8;33.5947; 16419.2;26.1964;

1287 3000;3;sobelv;5176;12986.6;17819.1;37.2116; 16680.3;28.4428;
1288 3000;3;sobelv;5113;14124.4;17964.8;27.1901; 16705.1;18.2713;
1289 3000;3;sobelv;5113;14066.1;17767.1;26.3113; 16752.3;19.0973;
1290 3000;3;sobelv;5101;14289;18041.1;26.2589; 16757.1;17.2731;
1291
1292 3000;4;sobelv;5034;16206.7;21763.4;34.2865; 21599.7;33.2769;
1293 3000;4;sobelv;5163;17176.1;23501.36;8239; 16965.7;-1.22519;
1294 3000;4;sobelv;5046;16246.8;23558.3;45.0021; 21577.9;32.813;
1295 3000;4;sobelv;5045;16433.2;23542.8;43.2638; 21709.8;32.1096;
1296 3000;4;sobelv;5039;16468.9;23448.8;42.3829; 21634.8;31.3679;
1297
1298 3500;1;sobelv;5012;9574.83;10961.8;14.4858; 10259.5;7.15078;
1299 3500;1;sobelv;5003;9564.62;10845.5;13.3917; 9900.52;3.51187;
1300 3500;1;sobelv;5116;9588.79;10607.3;10.622; 9763.28;1.81972;
1301 3500;1;sobelv;5017;9502.83;10473.3;10.2123; 9836.16;3.50761;
1302 3500;1;sobelv;5119;9500.19;10397.5;9.44485; 9590.38;0.949323;
1303
1304 3500;2;sobelv;5189;13666.4;17634.2;29.0326; 16620.2;21.6133;
1305 3500;2;sobelv;5157;13719.6;18025.1;31.3823; 16929.4;23.3961;
1306 3500;2;sobelv;5012;13713.8;18168.2;32.4805; 16803.9;22.5326;
1307 3500;2;sobelv;5021;14553.7;18223.5;25.2154; 17014.3;16.907;
1308 3500;2;sobelv;5007;13801.4;18147.8;31.4921; 17069.5;23.6793;
1309
1310 3500;3;sobelv;5063;18514.8;25947;40.1423; 24068.6;29.9971;
1311 3500;3;sobelv;5066;19933.6;26385.5;32.3669; 24195.8;21.3822;
1312 3500;3;sobelv;5083;18699.2;26240.2;40.3283; 24551.3;31.2964;
1313 3500;3;sobelv;5032;18780.3;26353.8;40.3273; 24522.7;30.577;
1314 3500;3;sobelv;5046;18847.3;26237.8;39.2123; 24613.4;30.5939;
1315
1316 3500;4;sobelv;5150;24298;32917.8;35.475; 31804.1;30.8917;
1317 3500;4;sobelv;5154;24547.1;24910.9;1.48205; 32200.1;31.1769;
1318 3500;4;sobelv;5160;24575.5;24949.5;1.52213; 30510.9;24.1518;
1319 3500;4;sobelv;5110;25712.2;24901.8;-3.15163; 25834.2;0.474506;
1320 3500;4;sobelv;5173;24383.3;24912.1;2.16886; 24979.1;2.44347;
1321
1322 4000;1;sobelv;5109;13858.7;14874.4;7.32865; 14424.2;4.07984;
1323 4000;1;sobelv;5012;13394.4;15268.9;13.9941; 14135.7;5.53451;
1324 4000;1;sobelv;5081;13265.6;15108.8;13.8945; 13784.5;3.91168;
1325 4000;1;sobelv;5066;13131.7;14996.14;1.969; 13700.4;4.33077;
1326 4000;1;sobelv;5065;13101.6;14889.4;13.646; 13622.8;3.97872;
1327
1328 4000;2;sobelv;5101;18839.5;25102.9;33.2465; 23275.1;23.5442;
1329 4000;2;sobelv;5110;19078.9;25801.4;35.235; 23823.6;24.869;
1330 4000;2;sobelv;5016;18960.6;25950;36.8626; 24002.7;26.5924;
1331 4000;2;sobelv;5109;19105.9;25625.3;34.1221; 24105.3;26.1667;
1332 4000;2;sobelv;5087;19253.8;25281.9;31.3088; 24140.2;25.3791;
1333
1334 4000;3;sobelv;5035;23454.1;30796.4;31.305; 30546.4;30.2391;
1335 4000;3;sobelv;5162;23493.9;26481.9;12.7182; 29742.8;26.5978;
1336 4000;3;sobelv;5073;23273.6;24865.9;6.84133; 29334.1;26.0401;
1337 4000;3;sobelv;5065;25084.5;24893;-0.76325; 28708.9;14.4489;
1338 4000;3;sobelv;5029;23611.8;24903.8;5.47205; 26417.4;11.8821;
1339
1340 4000;4;sobelv;5085;25520.1;24925;-2.33189; 26852.8;5.22233;
1341 4000;4;sobelv;5085;25520.5;24991.3;-2.07387; 24999.8;-2.04055;
1342 4000;4;sobelv;5095;25432.9;24896.8;-2.10802; 25045.8;-1.52232;
1343 4000;4;sobelv;5087;25510.4;25019.3;-1.9254; 25078.1;-1.69496;
1344 4000;4;sobelv;5074;25594.2;24978;-2.40727; 25034.4;-2.18705;
1345
1346 1500;1;combineimgs;5061;2563.64;2386.31;-6.91723; 2717.94;6.01865;
1347 1500;1;combineimgs;5061;2527.23;2156.04;-14.6874; 2596.72;2.74956;
1348 1500;1;combineimgs;5061;2497.16;2126.22;-14.8545; 2510.17;0.52063;
1349 1500;1;combineimgs;5062;2538.66;2123.62;-16.3487; 2485.36;-2.09938;
1350 1500;1;combineimgs;5061;2503.03;2059.49;-17.7199; 2463.28;-1.58796;
1351
1352 1500;2;combineimgs;5015;3774.19;2928.02;-22.4199; 3866.17;2.43699;
1353 1500;2;combineimgs;5096;3765.14;2937.39;-21.9846; 3882.32;3.11222;
1354 1500;2;combineimgs;5098;3782.35;2926.29;-22.6329; 3890.8;2.86734;
1355 1500;2;combineimgs;5098;3743.28;2929.92;-21.7286; 3867.32;3.31368;
1356 1500;2;combineimgs;5020;3814.87;2951.02;-22.6442; 3874.51;1.56343;
1357
1358 1500;3;combineimgs;5083;5112.96;3886.44;-23.9886; 5203.56;1.77185;
1359 1500;3;combineimgs;5082;5129.32;3901.14;-23.9443; 5176.07;0.911465;
1360 1500;3;combineimgs;5078;5122.82;3879.29;-24.2743; 5135.58;0.249125;
1361 1500;3;combineimgs;5094;5117.21;3891.11;-23.9604; 5193.26;1.48618;
1362 1500;3;combineimgs;5082;5103.84;3891.21;-23.7593; 5194.56;1.77742;
1363
1364 1500;4;combineimgs;5070;6426.9;4855.28;-24.4538; 6458.18;0.486686;
1365 1500;4;combineimgs;5100;6440.07;3897.59;-39.479; 6468.83;0.446606;
1366 1500;4;combineimgs;5100;6413.58;4871.82;-24.039; 5238.03;-18.3292;
1367 1500;4;combineimgs;5098;6388.01;4847.22;-24.12; 6507.65;1.87284;
1368 1500;4;combineimgs;5086;6401.38;4828.49;-24.5711; 6534.52;2.07991;
1369
1370 2000;1;combineimgs;5039;3293.53;2664.77;-19.0908; 3376.92;2.53207;
1371 2000;1;combineimgs;5040;3311.08;2674.29;-19.2321; 3363.36;1.57916;
1372 2000;1;combineimgs;5036;3269.98;2646.71;-19.0602; 3343.36;2.24425;
1373 2000;1;combineimgs;5036;3279.58;2644.83;-19.3545; 3341.48;1.88757;

1374 2000;1;combineimgs;5036;3288.51;2647.31;-19.498; 3359.11;2.14684;
1375
1376 2000;2;combineimgs;5013;5341.86;4059.4;-24.0078; 5299.13;-0.799899;
1377 2000;2;combineimgs;5016;5334.77;4083.91;-23.4473; 5301.67;-0.620533;
1378 2000;2;combineimgs;5049;5349.18;4044.47;-24.3908; 5316.34;-0.613878;
1379 2000;2;combineimgs;5072;5314.16;4060.86;-23.5841; 5311.44;-0.0512156;
1380 2000;2;combineimgs;5011;5345.98;4062.31;-24.0118; 5309.78;-0.677112;
1381
1382 2000;3;combineimgs;5067;7363.84;4992;-32.2093; 7296.75;-0.911084;
1383 2000;3;combineimgs;5058;7198.5;5499.56;-23.6013; 7317.8;1.65735;
1384 2000;3;combineimgs;5064;7200.32;5446.43;-24.3585; 7310.29;1.52721;
1385 2000;3;combineimgs;5062;7248.73;5432.49;-25.056; 7388.74;1.93142;
1386 2000;3;combineimgs;5056;7327.86;5385.77;-26.5027; 7339.86;0.163812;
1387
1388 2000;4;combineimgs;5071;9145.33;6750.26;-26.189; 8934.52;-2.30514;
1389 2000;4;combineimgs;5066;9169.76;5494.35;-40.0818; 9019.05;-1.64355;
1390 2000;4;combineimgs;5064;9154.45;6727.74;-26.5086; 9330.71;1.92538;
1391 2000;4;combineimgs;5067;9282.44;6889.96;-25.7742; 9325.84;0.467533;
1392 2000;4;combineimgs;5064;9334.57;6929.18;-25.7686; 9326.8;-0.0832758;
1393
1394 2200;1;combineimgs;5046;3883.97;3170.78;-18.3623; 3948.47;1.66054;
1395 2200;1;combineimgs;5047;3827.33;3162.43;-17.3725; 3946.54;3.11461;
1396 2200;1;combineimgs;5046;3794.21;3153.89;-16.8763; 3906.9;2.96995;
1397 2200;1;combineimgs;5049;3826.13;3158.22;-17.4567; 3888.45;1.62894;
1398 2200;1;combineimgs;5047;3836.78;3160.68;-17.6214; 3902.67;1.71751;
1399
1400 2200;2;combineimgs;5010;6237.67;4734.9;-24.0917; 6286.52;0.783194;
1401 2200;2;combineimgs;5011;6279.28;4814.93;-23.3204; 6287.55;0.131625;
1402 2200;2;combineimgs;5014;6198.17;4851.15;-21.7325; 6290.18;1.48448;
1403 2200;2;combineimgs;5067;6196.48;4704.64;-24.0755; 6307.1;1.7852;
1404 2200;2;combineimgs;5009;6218.23;4750.7;-23.6005; 6342.4;1.99685;
1405
1406 2200;3;combineimgs;5033;8654.35;6358.65;-26.5266; 8692.18;0.437167;
1407 2200;3;combineimgs;5011;8569.49;6494.29;-24.2162; 7740.37;-9.67529;
1408 2200;3;combineimgs;5013;8660.98;6420.75;-25.8658; 8724.44;0.732677;
1409 2200;3;combineimgs;5013;8717.74;6459.34;-25.9058; 8722.65;0.0563349;
1410 2200;3;combineimgs;5013;8715.96;6429.12;-26.2374; 8751.92;0.412497;
1411
1412 2200;4;combineimgs;5064;10944.9;8073.78;-26.2328; 11191.7;2.25472;
1413 2200;4;combineimgs;5063;10982.8;7821.43;-28.785; 8837.43;-19.5342;
1414 2200;4;combineimgs;5067;11066.3;8246.21;-25.4839; 11208.9;1.28856;
1415 2200;4;combineimgs;5065;11154.3;8300.04;-25.5886; 11548.8;3.53696;
1416 2200;4;combineimgs;5066;11129.2;8238.22;-25.9764; 11278.7;1.34327;
1417
1418 2400;1;combineimgs;5055;4599.36;3708.69;-19.365; 4641.33;0.912555;
1419 2400;1;combineimgs;5054;4596.27;3634.69;-20.9209; 4624.14;0.606247;
1420 2400;1;combineimgs;5054;4554.78;3577.12;-21.4643; 4611.73;1.25031;
1421 2400;1;combineimgs;5054;4564.56;3517.27;-22.9439; 4552.99;-0.253482;
1422 2400;1;combineimgs;5055;4494.76;3521.49;-21.6533; 4552.83;1.29193;
1423
1424 2400;2;combineimgs;5040;7345.53;5514.65;-24.925; 7384.53;0.531035;
1425 2400;2;combineimgs;5029;7369.17;5500.08;-25.3636; 7425.36;0.762593;
1426 2400;2;combineimgs;5008;7379.69;5535.16;-24.9947; 7417.75;0.515767;
1427 2400;2;combineimgs;5059;7374.5;5559.73;-24.6087; 7416.06;0.563624;
1428 2400;2;combineimgs;5028;7392.75;5526.72;-25.2413; 7417.66;0.336894;
1429
1430 2400;3;combineimgs;5035;10245.9;7528.19;-26.5248; 10406.2;1.56479;
1431 2400;3;combineimgs;5034;10386.3;7532.9;-27.4726; 10446.1;0.575823;
1432 2400;3;combineimgs;5056;10374.2;7755.7;-25.2406; 10483.7;1.05559;
1433 2400;3;combineimgs;5031;10463;7639.73;-26.9833; 10520.7;0.551255;
1434 2400;3;combineimgs;5028;10389.7;7713.83;-25.7552; 10506.5;1.12355;
1435
1436 2400;4;combineimgs;5045;13119.8;9581.61;-26.9681; 13137.7;0.136684;
1437 2400;4;combineimgs;5045;13104.2;9583.17;-26.8697; 13092.3;-0.0913573;
1438 2400;4;combineimgs;5049;13138.9;9766.02;-25.6707; 13014.4;-0.947133;
1439 2400;4;combineimgs;5044;13193.9;9359.79;-29.0595; 13050.7;-1.0854;
1440 2400;4;combineimgs;5043;13174.4;9698.42;-26.3846; 12826.5;-2.64123;
1441
1442 2500;1;combineimgs;5028;4951.01;4031.1;-18.5802; 4959.59;0.17328;
1443 2500;1;combineimgs;5030;4897.26;3936.39;-19.6206; 4938.68;0.845637;
1444 2500;1;combineimgs;5023;4930.63;3839.97;-22.1201; 4909.79;-0.422562;
1445 2500;1;combineimgs;5029;4915.36;3863.54;-21.3987; 4915.64;0.00557285;
1446 2500;1;combineimgs;5029;4895.68;3843.87;-21.4844; 4880.3;-0.314112;
1447
1448 2500;2;combineimgs;5057;7934.59;5999.31;-24.3904; 8031.43;1.2205;
1449 2500;2;combineimgs;5056;7931.82;5949.85;-24.9876; 8069.33;1.73367;
1450 2500;2;combineimgs;5060;7913.72;5981.36;-24.4179; 8120.42;2.61195;
1451 2500;2;combineimgs;5059;7915.38;6049.92;-23.5675; 8080.92;2.09144;
1452 2500;2;combineimgs;5055;7986.23;6036.01;-24.4198; 8063.13;0.962949;
1453
1454 2500;3;combineimgs;5012;11117.3;8199.13;-26.249; 11260.3;1.28645;
1455 2500;3;combineimgs;5015;11155.4;8352;-25.1305; 11351.6;1.7585;
1456 2500;3;combineimgs;5037;11222.5;8297.13;-26.0672; 11378.3;1.38762;
1457 2500;3;combineimgs;5017;11183.9;8220.1;-26.5005; 11352.4;1.5068;
1458 2500;3;combineimgs;5013;11079.8;8186.4;-26.1144; 11370.7;2.62512;
1459
1460 2500;4;combineimgs;5024;14310.2;10614.6;-25.8251; 11329.6;-20.8289;

1461 2500;4;combineimgs;5026;14384.2;10565.7;-26.5463; 14123.5;-1.81236;
1462 2500;4;combineimgs;5030;14399.1;10639.8;-26.1081; 14429.1;0.208421;
1463 2500;4;combineimgs;5056;14324.7;10685.8;-25.4033; 14191.9;-0.927319;
1464 2500;4;combineimgs;5037;14339.5;10684.8;-25.4872; 14523.7;1.28438;
1465
1466 3000;1;combineimgs;5023;7133.85;5705.42;-20.0233; 7048.84;-1.19171;
1467 3000;1;combineimgs;5015;6862.56;5610.29;-18.2478; 7068.92;3.00705;
1468 3000;1;combineimgs;5023;6948.68;5572.23;-19.8088; 7068.57;1.72532;
1469 3000;1;combineimgs;5015;6960.16;5531.49;-20.5263; 7057.66;1.40091;
1470 3000;1;combineimgs;5011;6867.33;5399.7;-21.3712; 7011.65;2.10154;
1471
1472 3000;2;combineimgs;5050;11704.6;8598.83;-26.5348; 11688.9;-0.134262;
1473 3000;2;combineimgs;5048;11773.6;8621.99;-26.7686; 11668.3;-0.894337;
1474 3000;2;combineimgs;5035;11732.7;8756.44;-25.3673; 11801.7;0.588241;
1475 3000;2;combineimgs;5038;11783.8;8818.28;-25.1664; 11814.6;0.261274;
1476 3000;2;combineimgs;5042;11790;8886.79;-24.6243; 11847.4;0.487213;
1477
1478 3000;3;combineimgs;5025;16559.8;12032.6;-27.3381; 16343.9;-1.30355;
1479 3000;3;combineimgs;5032;16808.5;12275.5;-26.9681; 16452.6;-2.11725;
1480 3000;3;combineimgs;5028;16637.3;12294.2;-26.1046; 16810.4;1.04037;
1481 3000;3;combineimgs;5027;16748.5;12301.9;-26.5489; 16526.6;-1.32478;
1482 3000;3;combineimgs;5049;16673.2;12343.2;-25.97; 16350.9;-1.93342;
1483
1484 3000;4;combineimgs;5044;20896.5;14594.4;-30.1587; 21193.6;1.42183;
1485 3000;4;combineimgs;5047;20970.2;15398.8;-26.5681; 21698.1;3.4713;
1486 3000;4;combineimgs;5042;21156;15653.9;-26.0071; 21628.8;2.2351;
1487 3000;4;combineimgs;5044;20946.6;15713.3;-24.9838; 21792.5;4.03866;
1488 3000;4;combineimgs;5042;21168.4;15703.8;-25.8151; 21556.8;1.83463;
1489
1490 3500;1;combineimgs;5023;10291.8;8136.83;-20.9386; 10366.7;0.728329;
1491 3500;1;combineimgs;5008;10207;8003.15;-21.5913; 10081.9;-1.22543;
1492 3500;1;combineimgs;5021;9920.89;7825.54;-21.1206; 9969.17;0.486593;
1493 3500;1;combineimgs;5019;9924.77;7787.04;-21.5393; 9921.89;-0.0290663;
1494 3500;1;combineimgs;5019;9924.37;7716.27;-22.2493; 9907.8;-0.166927;
1495
1496 3500;2;combineimgs;5033;16912.5;12268.3;-27.4602; 16587.1;-1.92369;
1497 3500;2;combineimgs;5035;17280.3;12503.1;-27.6453; 16697.3;-3.37387;
1498 3500;2;combineimgs;5034;16941.5;12753.4;-24.721; 16778;-0.964673;
1499 3500;2;combineimgs;5005;17012;12656.9;-25.6; 16882.2;-0.762985;
1500 3500;2;combineimgs;5034;17243.1;12626.9;-26.7712; 16876.7;-2.12525;
1501
1502 3500;3;combineimgs;5042;23737;17384.5;-26.7618; 24506.1;3.24032;
1503 3500;3;combineimgs;5043;23891;17820.6;-25.4086; 24790.7;3.76586;
1504 3500;3;combineimgs;5043;24091.7;17825.6;-26.0093; 24458.1;1.5212;
1505 3500;3;combineimgs;5038;24032.3;16579.6;-31.0112; 24429.5;1.65272;
1506 3500;3;combineimgs;5036;24060.7;17826.3;-25.9111; 24940;3.65469;
1507
1508 3500;4;combineimgs;5040;31298.2;23108.1;-26.168; 32333.9;3.30928;
1509 3500;4;combineimgs;5046;25690.1;23526;-8.42407; 27038.2;5.24732;
1510 3500;4;combineimgs;5045;25798.1;23104.6;-10.4407; 26786.5;3.83131;
1511 3500;4;combineimgs;5045;25770.6;20474.3;-20.552; 28906.6;12.1687;
1512 3500;4;combineimgs;5041;25819.6;23263.4;-9.90039; 26579.5;2.94292;
1513
1514 4000;1;combineimgs;5034;14270.9;11442.6;-19.8182; 14755.3;3.39461;
1515 4000;1;combineimgs;5025;14375.7;11184.1;-22.2017; 14343.9;-0.221368;
1516 4000;1;combineimgs;5025;14247.5;11003.7;-22.7672; 14236.2;-0.0794754;
1517 4000;1;combineimgs;5009;13958.5;10872.2;-22.1107; 14077.4;0.851745;
1518 4000;1;combineimgs;5011;14258.4;10824.6;-24.0828; 13975.1;-1.98702;
1519
1520 4000;2;combineimgs;5038;23965.8;17072.8;-28.7619; 23854;-0.466875;
1521 4000;2;combineimgs;5038;23966.8;17557.6;-26.7422; 24333.1;1.52842;
1522 4000;2;combineimgs;5011;23868.3;17824.1;-25.3232; 24442.6;2.40606;
1523 4000;2;combineimgs;5018;23868.4;17778;-25.5167; 24308.8;1.84519;
1524 4000;2;combineimgs;5010;23789.8;17830.6;-25.0491; 24198.1;1.71632;
1525
1526 4000;3;combineimgs;5021;29177;22018.8;-24.5339; 29767.5;2.02371;
1527 4000;3;combineimgs;5024;27302.3;22129;-18.9482; 29363.1;7.54802;
1528 4000;3;combineimgs;5019;25455.5;22365.3;-12.1395; 27535.8;8.17211;
1529 4000;3;combineimgs;5024;25714.5;22442.1;-12.726; 27258.9;6.00587;
1530 4000;3;combineimgs;5007;26158.1;22337;-14.6077; 26644.7;1.86023;
1531
1532 4000;4;combineimgs;5039;25855.4;24821.7;-3.99825; 24973;-3.413;
1533 4000;4;combineimgs;5046;25743.7;24809.5;-3.62873; 25059.9;-2.65622;
1534 4000;4;combineimgs;5041;25704.5;24821.5;-3.43487; 24919.3;-3.05445;
1535 4000;4;combineimgs;5046;25716;24728.8;-3.83908; 25146.1;-2.21632;
1536 4000;4;combineimgs;5035;25874.4;24787.3;-4.2013; 25104.6;-2.97483;
1537
1538 1500;1;writeimage;5009;4571.73;2874.74;-37.1193; 2139.67;-53.1978;
1539 1500;1;writeimage;5007;4412.22;2705.14;-38.6898; 1968.97;-55.3745;
1540 1500;1;writeimage;5005;4400.82;2516.72;-42.8125; 1964.85;-55.3528;
1541 1500;1;writeimage;5004;4406.53;2552.09;-42.0839; 1959.82;-55.5246;
1542 1500;1;writeimage;5008;4365.96;2550.43;-41.5838; 1948.05;-55.381;
1543
1544 1500;2;writeimage;5010;4880.89;3833.2;-21.4652; 2628.03;-46.1567;
1545 1500;2;writeimage;5004;4970.1;3977.22;-19.977; 2643.33;-46.8153;
1546 1500;2;writeimage;5007;4896.73;3958.69;-19.1566; 2660.15;-45.675;
1547 1500;2;writeimage;5009;4952.36;3944.99;-20.3413; 2626.37;-46.9674;

1548 1500;2;writeimage;5007;4899.39;4003.95;-18.2765; 2651.03;-45.8907;
1549
1550 1500;3;writeimage;5007;5185.79;4960.71;-4.3404; 3435.66;-33.7486;
1551 1500;3;writeimage;5015;5259.21;5158.04;-1.92372; 3415.73;-35.0524;
1552 1500;3;writeimage;5014;5201.99;5180.96;-0.404227; 3429.28;-34.0775;
1553 1500;3;writeimage;5009;5288.47;4984.39;-5.74982; 3463.98;-34.4992;
1554 1500;3;writeimage;5020;5252.02;5208.05;-0.837205; 3427.62;-34.7371;
1555
1556 1500;4;writeimage;5010;5387.26;6168.61;14.5037; 3469.5;-35.598;
1557 1500;4;writeimage;5050;5384.98;5783.76;7.4054; 4245.18;-21.1664;
1558 1500;4;writeimage;5010;5375.29;5994.12;11.5124; 4221.21;-21.4702;
1559 1500;4;writeimage;5012;5439.2;6145.62;12.9875; 4219.36;-22.4269;
1560 1500;4;writeimage;5009;5381.23;6120.28;13.7339; 4174.8;-22.4193;
1561
1562 2000;1;writeimage;5006;5323.75;3422.22;-35.7179; 2496.87;-53.0993;
1563 2000;1;writeimage;5006;5173.83;3486.34;-32.6159; 2487.55;-51.9206;
1564 2000;1;writeimage;5007;5204.29;3418.17;-34.3202; 2475.03;-52.4424;
1565 2000;1;writeimage;5008;5054.05;3321.83;-34.2738; 2492.46;-50.6838;
1566 2000;1;writeimage;5006;4844.97;3322.51;-31.4235; 2469.76;-49.0243;
1567
1568 2000;2;writeimage;5005;6158.4;5187.45;-15.7662; 3564.87;-42.1138;
1569 2000;2;writeimage;5010;6225.61;4902.5;-21.2527; 3568.59;-42.6789;
1570 2000;2;writeimage;5009;6147.65;5241.52;-14.7394; 3566.32;-41.9888;
1571 2000;2;writeimage;5004;6060.29;5241.83;-13.5052; 3576.73;-40.9808;
1572 2000;2;writeimage;5008;5903.62;4939.58;-16.3297; 3579.36;-39.3701;
1573
1574 2000;3;writeimage;5004;6630.96;7108.9;7.20774; 4783.77;-27.8571;
1575 2000;3;writeimage;5008;6727.98;6278.49;-6.68085; 4720.96;-29.8309;
1576 2000;3;writeimage;5005;6642.17;6957.6;4.74895; 4714.39;-29.0233;
1577 2000;3;writeimage;5010;6677.31;6878.7;3.01597; 4148.74;-37.8681;
1578 2000;3;writeimage;5010;6785.63;7051.37;3.91619; 4701.29;-30.7169;
1579
1580 2000;4;writeimage;5009;6720.52;8440.36;25.5909; 5723.96;-14.8285;
1581 2000;4;writeimage;5012;6737.06;8545.21;26.8389; 5828.33;-13.4885;
1582 2000;4;writeimage;5011;6952.97;8363.55;20.2874; 5887.19;-15.3284;
1583 2000;4;writeimage;5012;6924.37;8575.02;23.8383; 5875.25;-15.1511;
1584 2000;4;writeimage;5006;6818.46;8354.55;22.5284; 5869.34;-13.9198;
1585
1586 2200;1;writeimage;5003;5837.45;3687.47;-36.8308; 2835.15;-51.4317;
1587 2200;1;writeimage;5003;5588.89;3554.76;-36.3961; 2811.02;-49.7034;
1588 2200;1;writeimage;5005;5343.59;3551.12;-33.5443; 2813.52;-47.3477;
1589 2200;1;writeimage;5006;5491.1;3518.04;-35.932; 2822.74;-48.5942;
1590 2200;1;writeimage;5002;5603.03;3544.75;-36.7351; 2824.12;-49.5966;
1591
1592 2200;2;writeimage;5004;7107.87;5492.52;-22.7262; 4163.68;-41.4216;
1593 2200;2;writeimage;5007;7059.11;5881.99;-16.6751; 4170.8;-40.916;
1594 2200;2;writeimage;5004;6919.3;5967.13;-13.7611; 4193.15;-39.3992;
1595 2200;2;writeimage;5010;7041.79;5543.08;-21.2831; 4204.97;-40.2855;
1596 2200;2;writeimage;5010;7184.88;5853.77;-18.5265; 4196.31;-41.5952;
1597
1598 2200;3;writeimage;5011;7560.07;7833.16;3.61233; 5622.06;-25.6349;
1599 2200;3;writeimage;5011;7720;8080.71;4.67246; 5603.21;-27.4195;
1600 2200;3;writeimage;5003;7497.95;7995.92;6.6415; 5604.72;-25.2499;
1601 2200;3;writeimage;5004;7656.71;7848.59;2.50611; 5632.1;-26.4423;
1602 2200;3;writeimage;5006;7687.69;8022.29;4.35241; 5598.58;-27.1747;
1603
1604 2200;4;writeimage;5015;7002.96;9980.36;42.5163; 6973.25;-0.424221;
1605 2200;4;writeimage;5009;7471.31;9723.53;30.145; 6952.58;-6.94297;
1606 2200;4;writeimage;5017;7674.94;9861.8;28.4935; 7007.01;-8.70269;
1607 2200;4;writeimage;5005;7748.92;9759.21;25.9428; 6979.98;-9.92322;
1608 2200;4;writeimage;5021;7651.79;9831.37;28.4845; 7009.41;-8.39519;
1609
1610 2400;1;writeimage;5006;6306.25;3942.44;-37.4836; 3343.4;-46.9827;
1611 2400;1;writeimage;5004;6169.13;3663.88;-40.6094; 3322.97;-46.1356;
1612 2400;1;writeimage;5002;6001.27;3537.51;-41.054; 3312.05;-44.8109;
1613 2400;1;writeimage;5004;6290.69;3553.29;-43.5152; 3275.97;-47.9236;
1614 2400;1;writeimage;5002;6164.97;3631.55;-41.0938; 3302.39;-46.433;
1615
1616 2400;2;writeimage;5007;7908;6353.38;-19.6589; 4962.66;-37.2451;
1617 2400;2;writeimage;5007;7932.54;6492.8;-18.1498; 4984.99;-37.1577;
1618 2400;2;writeimage;5006;8095.9;6702.95;-17.2057; 4959.11;-38.7455;
1619 2400;2;writeimage;5008;8062.09;6592.48;-18.2286; 4944.61;-38.6684;
1620 2400;2;writeimage;5005;8003.82;6304.62;-21.2298; 4967.08;-37.9411;
1621
1622 2400;3;writeimage;5006;8267.18;9430.42;14.0706; 6548.37;-20.7908;
1623 2400;3;writeimage;5005;8561.41;9278.42;8.3749; 6598.07;-22.9324;
1624 2400;3;writeimage;5008;8620.67;9430.39;9.39274; 6474.73;-24.8929;
1625 2400;3;writeimage;5003;8554.67;9398.79;9.86728; 6635.73;-22.4315;
1626 2400;3;writeimage;5006;8741.18;9655.48;10.4596; 6610.89;-24.3707;
1627
1628 2400;4;writeimage;5008;8882.49;12059.3;35.7653; 8440.1;-4.9805;
1629 2400;4;writeimage;5010;8923.66;11719;31.3252; 8447.5;-5.33596;
1630 2400;4;writeimage;5005;8822.14;12094.5;37.0923; 8451.66;-4.19943;
1631 2400;4;writeimage;5021;8897.98;11972.1;34.5482; 8066.69;-9.34248;
1632 2400;4;writeimage;5022;8460.26;11936;41.0827; 8528.38;0.805131;
1633
1634 2500;1;writeimage;5002;6687.2;4132.78;-38.1987; 3666.98;-45.1641;

1635 2500;1;writeimage;5004;6578.5;3697.46;-43.7947; 3602.2;-45.2429;
1636 2500;1;writeimage;5006;6496.27;3577.62;-44.928; 3608.02;-44.4602;
1637 2500;1;writeimage;5007;6236.53;3738.25;-40.0589; 3470.25;-44.356;
1638 2500;1;writeimage;5002;6144.85;3557.03;-42.1136; 3514.85;-42.8002;
1639
1640 2500;2;writeimage;5006;8451.59;6305.33;-25.3948; 5250.15;-37.8797;
1641 2500;2;writeimage;5005;8466.6;6277.37;-25.8573; 5241.81;-38.0883;
1642 2500;2;writeimage;5006;8523.21;6501.77;-23.7169; 5253.15;-38.3666;
1643 2500;2;writeimage;5006;8385.01;6483.97;-22.6719; 5259.74;-37.2721;
1644 2500;2;writeimage;5007;8615.96;6135.06;-28.7942; 5272.74;-38.8026;
1645
1646 2500;3;writeimage;5003;8749.34;9712.92;11.0131; 7194.29;-17.7734;
1647 2500;3;writeimage;5006;9110.21;9932.02;9.02074; 7234.18;-20.5926;
1648 2500;3;writeimage;5010;9225.23;9887.79;7.18208; 7183.05;-22.1369;
1649 2500;3;writeimage;5007;8809.62;9976.79;13.2488; 7219.79;-18.0465;
1650 2500;3;writeimage;5010;9187.67;10017.2;9.02928; 7226.08;-21.3502;
1651
1652 2500;4;writeimage;5012;8847.49;12630.8;42.761; 8948.22;1.13845;
1653 2500;4;writeimage;5004;8688.61;12567.1;44.6392; 8323.94;-4.19713;
1654 2500;4;writeimage;5032;9017.97;12667.4;40.4681; 8959.35;-0.650048;
1655 2500;4;writeimage;5021;8860.04;12722.2;43.5903; 7184.98;-18.9058;
1656 2500;4;writeimage;5009;9165.62;12822.1;39.8932; 8951.49;-2.33624;
1657
1658 3000;1;writeimage;5006;8099.61;3759.11;-53.5891; 4971.99;-38.6145;
1659 3000;1;writeimage;5002;7933.38;3646.89;-54.0311; 4966.01;-37.4036;
1660 3000;1;writeimage;5004;6964.75;3650.86;-47.5809; 4929.19;-29.2266;
1661 3000;1;writeimage;5004;7883.54;3580.11;-54.5875; 4838.38;-38.6268;
1662 3000;1;writeimage;5007;7186.01;3571.22;-50.3032; 4900.37;-31.8068;
1663
1664 3000;2;writeimage;5003;10932.8;5621.46;-48.5815; 7476.69;-31.6121;
1665 3000;2;writeimage;5007;11171.2;7776.1;-30.3916; 7551.39;-32.4031;
1666 3000;2;writeimage;5002;11110.2;6651.57;-40.1307; 7602.19;-31.5744;
1667 3000;2;writeimage;5005;11240.4;6522.93;-41.9689; 7602.1;-32.368;
1668 3000;2;writeimage;5009;10624.8;6918.14;-34.887; 7623;-28.253;
1669
1670 3000;3;writeimage;5006;11392.1;10905.4;-4.27248; 10367.9;-8.99021;
1671 3000;3;writeimage;5006;11749.7;11955.2;1.74938; 10482.7;-10.7831;
1672 3000;3;writeimage;5009;12758.7;11956.6;-6.28646; 10528.5;-17.4794;
1673 3000;3;writeimage;5009;12363.3;12211.6;-1.22641; 10584;-14.3917;
1674 3000;3;writeimage;5009;12403.5;13410.8;8.12077; 10632.8;-14.2757;
1675
1676 3000;4;writeimage;5017;12871.7;17010.7;32.1561; 13152.9;2.18518;
1677 3000;4;writeimage;5018;11378.8;17655.2;55.1582; 13133.6;15.4212;
1678 3000;4;writeimage;5005;12778.4;17603.9;37.7636; 13102.3;2.53536;
1679 3000;4;writeimage;5009;12395.1;17592.2;41.9294; 12766.9;2.99995;
1680 3000;4;writeimage;5006;12680.6;16264.9;28.2659; 13285.3;4.76858;
1681
1682 3500;1;writeimage;5004;10786.7;4116.48;-61.8373; 7093.02;-34.2427;
1683 3500;1;writeimage;5007;8952.69;3748.4;-58.1311; 6814.69;-23.8811;
1684 3500;1;writeimage;5004;8083.84;3687.77;-54.381; 6807.86;-15.7843;
1685 3500;1;writeimage;5004;5874.52;3697.58;-37.0574; 6782.29;15.4526;
1686 3500;1;writeimage;5008;8983.39;3658.91;-59.2703; 6792.2;-24.3916;
1687
1688 3500;2;writeimage;5006;14748.8;6889.35;-53.2886; 10621.3;-27.9849;
1689 3500;2;writeimage;5002;14651.2;5589.66;-61.8484; 10629.2;-27.4517;
1690 3500;2;writeimage;5005;14475.6;6530.14;-54.8886; 10766.9;-25.6205;
1691 3500;2;writeimage;5003;12753;6715.28;-47.3437; 10767;-15.5734;
1692 3500;2;writeimage;5008;14035.6;6908.78;-50.7768; 10769.8;-23.2682;
1693
1694 3500;3;writeimage;5005;17045.5;12271.2;-28.0088; 14565.4;-14.5497;
1695 3500;3;writeimage;5002;16918.4;12912.6;-23.6771; 14736.8;-12.8949;
1696 3500;3;writeimage;5007;17705;12514.3;-29.318; 14761.2;-16.6269;
1697 3500;3;writeimage;5006;17017.2;13752.9;-19.1822; 15047.6;-11.5743;
1698 3500;3;writeimage;5004;17895.7;9402.15;-47.4614; 14900;-16.7398;
1699
1700 3500;4;writeimage;5008;13220.2;16009.6;21.099; 16324.4;23.4805;
1701 3500;4;writeimage;5003;10313.3;21299.1;106.521; 15817.1;53.3661;
1702 3500;4;writeimage;5010;11560.4;13482.9;16.6299; 19085.7;65.0951;
1703 3500;4;writeimage;5006;9697.98;12007.3;23.8126; 17900.8;84.5827;
1704 3500;4;writeimage;5026;17022.6;21331.9;25.3155; 18940.4;11.2661;
1705
1706 4000;1;writeimage;5005;12634.5;3877.06;-69.3138; 9679.13;-23.3915;
1707 4000;1;writeimage;5003;9753.56;3736.52;-61.6907; 9514.1;-2.45504;
1708 4000;1;writeimage;5006;5868.98;3621.07;-38.3015; 9399.73;60.1595;
1709 4000;1;writeimage;5004;11326.2;3715.91;-67.1918; 9290.84;-17.9701;
1710 4000;1;writeimage;5002;10575.7;3703.79;-64.9783; 9266.49;-12.3795;
1711
1712 4000;2;writeimage;5002;15230.8;6506.02;-57.2839; 14417.6;-5.33951;
1713 4000;2;writeimage;5003;17871.3;9101.48;-49.0722; 14551.2;-18.5782;
1714 4000;2;writeimage;5002;17062.2;6852.02;-59.8411; 14784.5;-13.3498;
1715 4000;2;writeimage;5006;14379.4;5621.02;-60.9093; 14799.3;2.92018;
1716 4000;2;writeimage;5003;17595.4;6049;-65.6217; 14790.6;-15.9406;
1717
1718 4000;3;writeimage;5006;20256.5;13553.6;-33.09; 18089.6;-10.6973;
1719 4000;3;writeimage;5003;19695.5;12763.6;-35.1953; 18140.7;-7.89392;
1720 4000;3;writeimage;5004;20089.6;13210.6;-34.2414; 18321.5;-8.80094;
1721 4000;3;writeimage;5004;18501.9;9234.34;-50.0897; 14817.9;-19.9117;

```
1722 4000;3;writeimage;5006;20165.1;10743.4;-46.7227; 18138.9;-10.0481;  
1723  
1724 4000;4;writeimage;5015;18951.8;15582.1;-17.7808; 18238.7;-3.76272;  
1725 4000;4;writeimage;5004;16241.7;16745.1;3.09946; 23436.4;44.2974;  
1726 4000;4;writeimage;5011;10798.5;18256.2;69.0618; 18566.4;71.9348;  
1727 4000;4;writeimage;5006;16781.7;18148;8.14161; 23464;39.8191;  
1728 4000;4;writeimage;5029;12750.4;16181.1;26.907; 20702.3;62.3662;
```

Listing B.4: Ergebnisdatei vom 24.06.2024

Anhang C

Ergänzungen zu den Implementierungen

C.1 Dateistruktur

Die Abbildung C.1 auf Seite 125 zeigt die Dateistruktur aus dem Repository. Die umgesetzten Artefakte sind anhand eines numerischen Präfixes erkennbar. Neben diesen Artefakten werden auch die Anwendungen `edgedetection` und `epEBench` benötigt. Der Quelltext von `edgedetection` wurde bereitgestellt und ist daher aufgrund einiger Anpassungen in diesem Repository versioniert. Der Ordner `sampleapp` enthält eine Beispielanwendung, die im Rahmen eines Exposés recherchiert wurde.

Die Datei `README.md` enthält einige Informationen, wie die Artefakte gebaut und verwendet werden.

Vor Ausführung der Anwendungen muss noch das sog. Init-Skript ausgeführt werden, dies wird im nachfolgenden Abschnitt C.2 erläutert.

C.2 Init-Skript

Das Listing C.1 auf Seite 126 zeigt den Inhalt des Init-Skriptes. Dieses Skript bereitet das Testsystem für die Durchführung des Experiments vor. Im wesentlichen werden:

- `make` installiert
- der Benchmark `epEBench` geklont, gebaut und vorbereitet
- die Anwendung `cpupower` installiert
- die Anwendung `edgedetection` vorbereitet und gebaut
- die Anwendung `cmake` für den Build des `Tasktypeestimator` installiert
- der Build der drei Artefakte ausgeführt

```
pp8
├── 1_tasktype_analyzer
│   ├── main
│   └── run_all_tasks.sh
├── 2_apptask_analyzer
│   ├── apptasks
│   └── run_all_tasks.sh
├── 3_tasktype_estimator
│   ├── main
│   ├── run_build.sh
│   ├── run_checkpreconditions.sh
│   └── run_estimation.sh
├── edgedetection
│   ├── edgedetection
│   └── run.sh
├── epEBench
│   └── taskmap.txt
├── sampleapp
├── init.sh
└── README.md
```

Abbildung C.1: Dateistruktur von PP8

```

1 #make installieren
2 sudo apt-get update
3 sudo apt-get -y install make
4
5 # epEBench auschecken und bauen
6 git -C epEBench pull || git clone https://github.com/RobertMueller/epEBench.git
7 cd epEBench
8
9 # Im Benchmark epEBench lässt sich der Task-Typ m4x4smul nicht mehr ausführen, da ein cach-Fehler provoziert wird, der zum Abbruch
  führt.
10 # Dies wurde auskommentiert, da dieser Fehler für das Experiment nicht relevant ist.
11 sed -i '/idx = (idx + 16) % MEM_SIZE-20;/c\idx = (idx + 16) % MEM_SIZE/*-20*/;' eblloads.cpp
12
13 # In der Konfigurationsdatei von epEBench fehlt das Tasktyp ssub32_SIMD als einzelnes Modell. Dieses wird hier eingefügt.
14 echo "model=ssub32_SIMD" >> ebmodels.ini
15 echo " ssub32_SIMD=1" >> ebmodels.ini
16 echo "end_model" >> ebmodels.ini
17 make
18 cd ..
19
20 ## Die Quelldatei für den GDBInstructionScanner heißt ci.py und wurde bereits mit den Artefakten eingecheckt. Die Quelldatei muss
  daher nicht mehr ausgecheckt werden.
21 #git -C GDBInstructionScanner pull || git clone https://gitlab.abo.fi/sholmbac/GDBInstructionScanner.git
22
23 #cp GDBInstructionScanner/ci.py sampleapp/ci.py
24 #git -C uarch-configure pull || git clone https://github.com/deater/uarch-configure.git
25
26
27 #Installation von cpupower
28 #uname -r Prüfen des Kernels, dann folgende Version installieren
29 sudo apt install linux-tools-6.5.0-27-generic
30
31 #edgedetection bauen und vorbereiten
32 cd edgedetection
33 make
34 ./getfiles.sh
35 sudo apt install python3-pil
36 sudo apt install python3-numpy
37 cd ..
38
39 #cmake installieren
40 sudo apt-get install cmake
41
42 #Artefakte bauen
43 cd l_tasktype_analyzer
44 make
45 cd ..
46
47 cd 2_apptask_analyzer
48 make
49 cd ..
50
51 cd 3_tasktype_estimator
52 ./run_build.sh

```

Listing C.1: init.sh

C.3 Vorbedingungen für Experiment prüfen

Die Abbildung A.1 auf Seite 77 zeigt das Auswahlmenü des Tasktypeestimators. Damit die Schätzung der Leistungsaufnahme anhand von Tasktypen mit dem Tasktypeestimator ausgeführt werden kann, gibt es eine Reihe von Vorbedingungen, die zu beachten sind. Um sicherzustellen, dass diese Vorbedingungen erfüllt sind, besteht die Möglichkeit den Menüpunkt I (=Init - Vorbedingungen für Experiment prüfen!) auszuführen.

Im wesentlichen werden folgende Bedingungen überprüft:

- Konfigurationsdatei experiment.config muss vorhanden sein
- Zu den hinterlegten Task-Typen und Anwendungstasks müssen Skripte generiert worden sein (ggfls. Menüpunkt C ausführen)

- Zu den Anwendungstasks müssen 1:N-Skripte vorhanden sein (ggfls. Menüpunkt C ausführen)
- Für den Taskmapper müssen Sequenzdateien vorhanden sein (ggfls. tasktypeanalyzer oder apptaskanalyzer ausführen)
- Das Ergebnis des Taskmappers (results/taskmap.txt) muss vorhanden sein und für jeden Anwendungstask eine 1:1 und eine 1:N-Abbildung enthalten (ggfls. Taskmapper ausführen)
- Sudoberechtigungen für den Zugriff auf den Energiezähler müssen vorhanden sein
- Rootberechtigungen für den Zugriff auf cpupower muss vorhanden sein
- epEBench muss vorhanden und kompiliert sein, die Datei epEBench/bin/Release/ebmodels.ini muss die Modelle der 1:N-Abbildung enthalten (ggfls. Transfer to epebench ausführen)
- edgedetection muss vorhanden und kompiliert sein, das Skript testEdgedetection.sh war erfolgreich durchgelaufen

Da beim Ausführen dieser Überprüfung sehr viel Bildschirmtext ausgegeben wird, besteht die Möglichkeit das Skript `run_checkpreconditions.sh` auszuführen. Bei Ausführung dieses Skriptes wird die Bildschirmausgabe zusätzlich in den Ordner `logs/output` geschrieben.

Anhang D

Exposé für Masterarbeit

D.1 Zielsetzung und Erkenntnisinteresse

Aus der Problemstellung ergeben sich folgende Ziele für die Masterarbeit:

- Herleitung und Definition geeigneter prototypischer Tasks
- Erhebung der Leistungsaufnahme der prototypischen Tasks auf einem Testsystem unter verschiedenen CPU-Frequenzen
- Evaluierung und Auswahl (ggfls. Implementierung) einer geeigneten Anwendung
- Evaluierung eines Vorgehens, um Tasks aus einer konkreten Anwendung zu bestimmen
- Schätzung der Leistungsaufnahme auf Grundlage der ermittelten Tasks (Soll-Wert)
- Ermittlung der Leistungsaufnahme und Laufzeit einer konkreten Anwendung (Ist-Wert)
- Überprüfung der Schätzungen mit den gemessenen Werten

D.2 Visualisierung

Die Abbildung D.2 dient zur Übersicht des geplanten Experiments. Das Experiment wurde in vier Phasen unterteilt, die nachfolgend kurz beschrieben wurde.

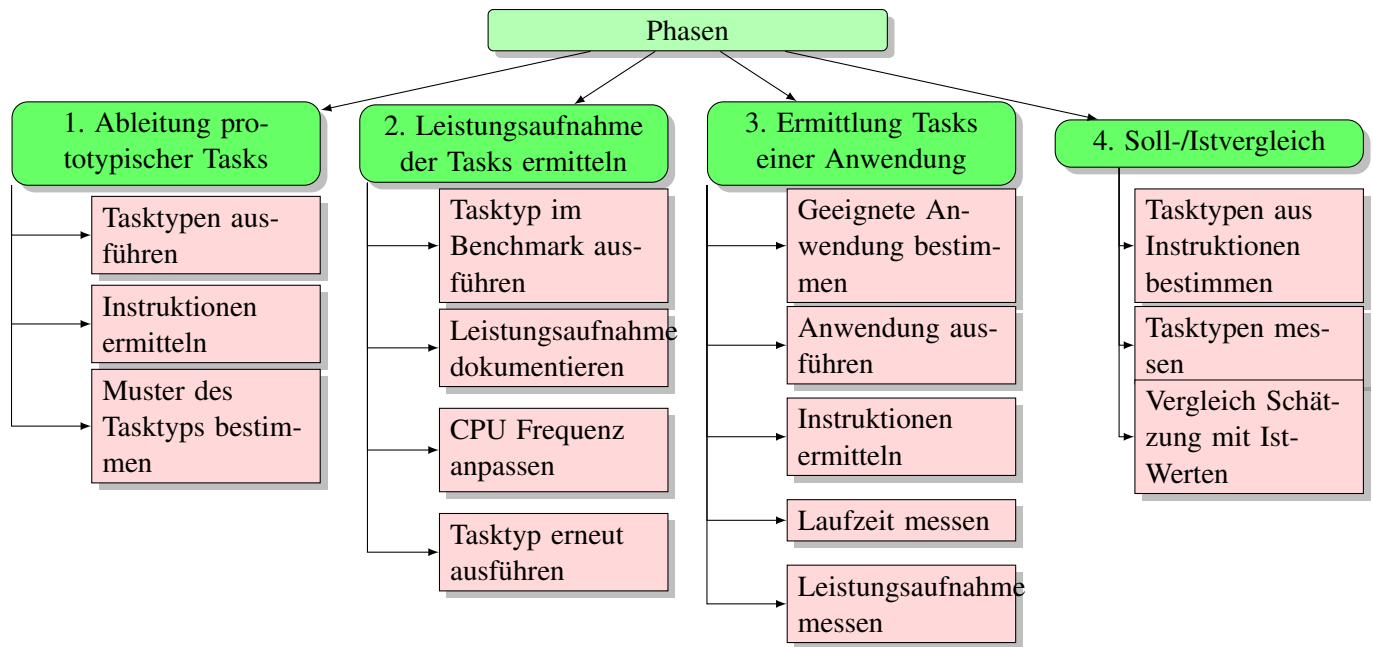


Abbildung D.1: Visualisierung

D.3 Phasen

D.3.1 Ableitung prototypischer Tasks

Die Software epEBench verwendet zur Berechnung der Leistungsaufnahme sog. Workloads, die einen Mix aus Modellen ausführen[7], die sich wie konkrete Anwendungen (vidplay, gzip) verhalten.

Dieser Benchmark ermöglicht außerdem die Ausführung einzelner Operationen in Form von Modellen. Das nachfolgende Listing entstammt aus der Schnittstelle eblloads.h und zeigt die einzelnen Modelle, die im epEBench zur Verfügung stehen:

```

1 // SIMD
2 void *run_m4x4smul_SIMD(int);
3 void *run_v1x4smul_SIMD(int);
4 void *run_dm164_SIMD(int);
5 void *run_sm132_SIMD(int);
6 void *run_ssub32_SIMD(int);
7 void *run_dsub64_SIMD(int);
8 void *run_dmem64_SIMD(int);
9 void *run_smem32_SIMD(int);
10 void *run_vmov_SIMD(int);
11 void *run_vconvert_SIMD(int);
12
13 // ALU / FP
14 void *run_m4x4smul(int);
15 void *run_dm164(int);
16 void *run_dadd(int);
17 void *run_iadd(int);
18 void *run_imul(int);
19 void *run_icompare(int);
20 void *run_logic(int);
21 void *run_branch(int);
22 void *run_imem(int);
23 void *run_dmem(int);
24 void *run_imov(int);
25 void *run_shift(int);
  
```

```

26 void *run_bitbyte(int);
27 void *run_nop(int);

```

Listing D.1: Prototypische Tasks aus epEBench ebloads.h[17]

Mithilfe des Benchmarks kann ein solches Modell einmalig oder für einen bestimmten Zeitraum ausgeführt und gemessen werden. Dies zeigt das nachfolgende Listing D.2. Als Beispiel wird das Modell dadd behandelt, welches in mehreren Schleifenläufen drei double Variablen summiert.

```

1 ./epebench -m dadd -t 1 -a 1 -n 1
2
3 *****
4 * epEBench - Energy Benchmark *
5 * Version 1.1.2 *
6 * build Dec 2 2023 *
7 * Robert Mueller *
8 * 2016 *
9 *****
10 > !Too less allocated CPU cores to reach 30% CPU load!
11
12 Initial settings:
13   CPU load: 12%
14   cores# : 8
15   threads#: 1
16   model : dadd
17   runtime : 1.0s
18
19 Returned pthread affinity and core load:
20   used cores: 1
21   CPU 0: 100.0%
22
23
24 [1Acpu usage [12.72% / 12.72%] max [ 0.00%] min [ 0.00%]
25 Run: 1010.96 ms
26 Counts: 1326.447140 Mio
27
28 Benchmarking finished.

```

Listing D.2: Ausführung des Tasks dadd

Das Ergebnis dieser Ausführung ist aus der Logdatei zu entnehmen:

```

1 # Testrun settings:
2 #   Model : dadd
3 #   Cpu# : 8
4 #   Thread# : 1
5 time usage avg_usage
6 0 0.1195 0.1272

```

Listing D.3: Ergebnis Leistungsaufnahme von dadd

Die Tasks aus epEBench liegen als C-Quelltext vor. Im nachfolgenden Listing wird der Quelltext des Modells dadd vorgestellt:

```

1 void* run_dadd(int counts)
2 {
3     volatile double x = 3.21, y = 0.2, z = 1.34;
4
5     instcnt += counts*CNT_DIV;
6     while ((counts--) && !done0) {
7         x+=1.121;
8         y+=1.122;
9         z+=1.123;
10        x+=1.124;
11        y+=1.125;
12        z+=1.126;
13        x+=1.127;
14        y+=1.128;
15        z+=1.129;
16        x+=1.120;
17        y+=1.121;
18        z+=1.122;
19        x+=1.123;
20        y+=1.124;
21        z+=1.125;
22        x+=1.126;
23        y+=1.127;
24        z+=1.128;
25        x+=1.129;
26        y+=1.120;
27    }
28    return NULL;

```

Listing D.4: eblloads.cpp[7]

Damit im weiteren Verlauf eine Untersuchung möglich ist, wird diese Funktion ausgeführt und mithilfe des Instruktionsscaners von Holmbacka, in Assembler-Instruktionen konvertiert[5].

Um den Assembleranweisungen auf die wesentliche Ausführung zu reduzieren, können einerseits Breakpoints gesetzt werden, damit nur bestimmte Bereiche konvertiert werden. Andererseits ist es auch möglich, zunächst den Assemblercode eines leeren Programms zu generieren und mit dem tatsächlichen Programm zu vergleichen. Das nachfolgende Listing zeigt den Assemblercode der dadd-Operation.

```

1  call    0x55555555129 <run_dadd>
2  endbr64
3  push    %rbp
4  mov     %rsp,%rbp
5  mov     %edi,-0x24(%rbp)
6  movsd   0xec0(%rip),%xmm0          # 0x555555556008
7  movsd   %xmm0,-0x18(%rbp)
8  movsd   0xec7(%rip),%xmm0          # 0x555555556010
9  movsd   %xmm0,-0x10(%rbp)
10 movsd   0xec2(%rip),%xmm0          # 0x555555556018
11 movsd   %xmm0,-0x8(%rbp)
12 jmp     0x55555555318 <run_dadd+495>
13 mov     -0x24(%rbp),%eax
14 lea     -0x1(%rax),%edx
15 mov     %edx,-0x24(%rbp)
16 test    %eax,%eax
17 jg      0x55555555160 <run_dadd+55>
18 movsd   -0x18(%rbp),%xmm1
19 movsd   0xeb3(%rip),%xmm0          # 0x555555556020
20 addsd   %xmm1,%xmm0
21 movsd   %xmm0,-0x18(%rbp)
22 movsd   -0x10(%rbp),%xmm1
23 movsd   0xea5(%rip),%xmm0          # 0x555555556028
24 addsd   %xmm1,%xmm0
25 movsd   %xmm0,-0x10(%rbp)
26 movsd   -0x8(%rbp),%xmm1
27 movsd   0xe97(%rip),%xmm0          # 0x555555556030
28 addsd   %xmm1,%xmm0
29 movsd   %xmm0,-0x8(%rbp)
30 movsd   -0x18(%rbp),%xmm1
31 movsd   0xe89(%rip),%xmm0          # 0x555555556038
32 addsd   %xmm1,%xmm0
33 movsd   %xmm0,-0x18(%rbp)
34 movsd   -0x10(%rbp),%xmm1
35 movsd   0xe7b(%rip),%xmm0          # 0x555555556040
36 addsd   %xmm1,%xmm0
37 movsd   %xmm0,-0x10(%rbp)
38 movsd   -0x8(%rbp),%xmm1
39 movsd   0xe6d(%rip),%xmm0          # 0x555555556048
40 addsd   %xmm1,%xmm0
41 movsd   %xmm0,-0x8(%rbp)
42 movsd   -0x18(%rbp),%xmm1
43 movsd   0xe5f(%rip),%xmm0          # 0x555555556050
44 addsd   %xmm1,%xmm0
45 movsd   %xmm0,-0x18(%rbp)
46 movsd   -0x10(%rbp),%xmm1
47 movsd   0xe51(%rip),%xmm0          # 0x555555556058
48 addsd   %xmm1,%xmm0
49 movsd   %xmm0,-0x10(%rbp)
50 movsd   -0x8(%rbp),%xmm1
51 movsd   0xe43(%rip),%xmm0          # 0x555555556060
52 addsd   %xmm1,%xmm0
53 movsd   %xmm0,-0x8(%rbp)
54 movsd   -0x18(%rbp),%xmm1
55 movsd   0xe35(%rip),%xmm0          # 0x555555556068
56 addsd   %xmm1,%xmm0
57 movsd   %xmm0,-0x18(%rbp)
58 movsd   -0x10(%rbp),%xmm1
59 movsd   0xdd7(%rip),%xmm0          # 0x555555556020
60 addsd   %xmm1,%xmm0
61 movsd   %xmm0,-0x10(%rbp)
62 movsd   -0x8(%rbp),%xmm1
63 movsd   0xdc9(%rip),%xmm0          # 0x555555556028
64 addsd   %xmm1,%xmm0
65 movsd   %xmm0,-0x8(%rbp)
66 movsd   -0x18(%rbp),%xmm1
67 movsd   0xdbb(%rip),%xmm0          # 0x555555556030

```



```

68 addsd %xmm1,%xmm0
69 movsd %xmm0,-0x18(%rbp)
70 movsd -0x10(%rbp),%xmm1
71 movsd 0xdad(%rip),%xmm0          # 0x555555556038
72 addsd %xmm1,%xmm0
73 movsd %xmm0,-0x10(%rbp)
74 movsd -0x8(%rbp),%xmm1
75 movsd 0xd9f(%rip),%xmm0         # 0x555555556040
76 addsd %xmm1,%xmm0
77 movsd %xmm0,-0x8(%rbp)
78 movsd -0x18(%rbp),%xmm1
79 movsd 0xd91(%rip),%xmm0         # 0x555555556048
80 addsd %xmm1,%xmm0
81 movsd %xmm0,-0x18(%rbp)
82 movsd -0x10(%rbp),%xmm1
83 movsd 0xd83(%rip),%xmm0         # 0x555555556050
84 addsd %xmm1,%xmm0
85 movsd %xmm0,-0x10(%rbp)
86 movsd -0x8(%rbp),%xmm1
87 movsd 0xd75(%rip),%xmm0         # 0x555555556058
88 addsd %xmm1,%xmm0
89 movsd %xmm0,-0x8(%rbp)
90 movsd -0x18(%rbp),%xmm1
91 movsd 0xd67(%rip),%xmm0         # 0x555555556060
92 addsd %xmm1,%xmm0
93 movsd %xmm0,-0x18(%rbp)
94 movsd -0x10(%rbp),%xmm1
95 movsd 0xd59(%rip),%xmm0         # 0x555555556068
96 addsd %xmm1,%xmm0
97 movsd %xmm0,-0x10(%rbp)
98 mov -0x24(%rbp),%eax
99 lea -0x1(%rax),%edx
100 mov %edx,-0x24(%rbp)
101 test %eax,%eax
102 jg 0x555555555160 <run_dadd+55>
103 mov $0x0,%eax
104 pop %rbp

```

Listing D.5: Assemblercode von dadd[5]

Es ist nun zu überprüfen, ob die einzelnen Modelle aus epEBench als prototypische Tasks in Frage kommen. Dabei werden diese Modelle ausgeführt und die Assembler-Instruktionen nach Auffälligkeiten untersucht. Dazu gehören u. A. die Häufigkeit (bspw. der prozentuale Anteil) der einzelnen Assembler-Befehle.

D.3.2 Leistungsaufnahme prototypischer Tasks in einem Testsystem

Wenn die Tasktypen definiert sind, dann finden Messungen dieser Modelle mit epEBench statt. Die Messungen werden unter unterschiedlichen CPU-Frequenzen durchgeführt und dokumentiert, um zu überprüfen, wie die Frequenz sich auf Energieverbrauch und Laufzeit auswirken.

Für die Messungen ist ein Notebook Dell Latitude 5500 mit Intel CPU i5-8365 vorgesehen. Das Notebook wurde mit einem Ubuntu Betriebssystem eingerichtet.

D.3.3 Ermittlung der Task einer konkreten Anwendung

Zunächst ist eine geeignete Anwendung zu bestimmen. Die Anwendung sollte taskbasiert und parallelisierbar sein. Der Umfang dieser Anwendung sollte geeignet sein, um nach Ausführung die Assembler-Instruktionen untersuchen zu können. Dabei wird geprüft, ob eine Zuordnung der Anwendung zu den definierten prototypischen Task möglich ist.

Das nachfolgende Listing zeigt ein mögliches C-Programm welches, hinsichtlich verwendeter Tasks, untersucht werden könnte.

```

38 #include <pthread.h>
39 #include <stdlib.h>
40
41 typedef struct {
42     int size, row, column;
43     double (*MA)[8], (*MB)[8], (*MC)[8];
44 } matrix_type_t;
45
46 void *thread_mult (void *w) {
47     matrix_type_t *work = (matrix_type_t *) w;
48     int i, row = work->row, column = work->column;
49     work->MC[row][column] = 0;
50     for (i = 0; i < work->size; i++)
51         work->MC[row][column] += work->MA[row][i] * work->MB[i][column];
52     return NULL;
53 }
54
55 int main() {
56     int row, column, size = 8, i;
57     double MA[8][8], MB[8][8], MC[8][8];
58     matrix_type_t *work;
59     pthread_t thread[8 * 8];
60     for (row = 0; row < size; row++)
61         for (column = 0; column < size; column++) {
62             work = (matrix_type_t *) malloc(sizeof(matrix_type_t));
63             work->size = size;
64             work->row = row;
65             work->column = column;
66             work->MA = MA;
67             work->MB = MB;
68             work->MC = MC;
69             pthread_create(&thread[column + row * 8], NULL,
70                 thread_mult, (void *) work);
71         }
72     for (i = 0; i < size * size; i++)
73         pthread_join(thread[i], NULL);
74 }

```

Listing D.6: Thread-Programm zur Realisierung einer Matrixmultiplikation[21][s.S. 291

D.3.4 Berechnung und Ergebnisbestimmung

Wenn zur Anwendung mehrere Tasks ermittelt wurden, so wird die Leistungsaufnahme und die Laufzeit dieser Tasks summiert um zu einer Abschätzung zu gelangen. Diese Abschätzung wird nun bewertet, in dem das Ergebnis der Schätzung, mit der tatsächlichen Leistungsaufnahme und Laufzeit der Anwendung gegenübergestellt wird.

Bei Intel-Systemen besteht die Möglichkeit, die Leistungsaufnahme mithilfe der RAPL-Schnittstelle auszulesen [25].

```

1 root@allan-Latitude-5500:/home/allan/git/uarch-configure/rapl-read# ./rapl-plot
2
3 RAPL read -- use -s for sysfs, -p for perf_event, -m for msr
4
5 Found Kaby Lake Processor type
6 0 (0), 1 (0), 2 (0), 3 (0), 4 (0), 5 (0), 6 (0), 7 (0)
7
8 Detected 8 cores in 1 packages
9
10
11 Using sysfs powercap interface to gather results
12
13 Time (s)  Package0(W) Cores(W) GPU(W) DRAM(W) Psys(W) |
14 0.500160  18.228521 17.099118 0.025626 0.507044 0.000000
15 1.000436  18.318819 16.902001 0.042215 0.526685 0.000000

```

Listing D.7: Leistungsaufnahme der CPU über RAPL[25]

Die Messung der konkreten Anwendung und der Vergleich wird unter unterschiedlichen CPU-Frequenzen ausgeführt und dargestellt.

D.4 erwartete Ergebnisse

epEBench enthält bis zu 24 Modelle die daraufhin zu untersuchen sind, ob diese abgrenzbar voneinander zu unterscheiden sind. Es kann daher sein, dass einige Modelle sich zu ähnlich sind und nicht eindeutig einer konkreten Anwendung zuzuordnen wären. Sollten jedoch tatsächlich 24 prototypische Tasks zur Verfügung stehen, dann würde die Zuordnung zur konkreten Aufwändig etwas aufwändig werden.

Dies wird im Rahmen dieser Arbeit untersucht werden.

D.5 vorläufige Gliederung

1. Einleitung
2. Theorie
3. Methodik
4. Ergebnisse und Analyse
5. Diskussion und Interpretation
6. Fazit

A Literaturverzeichnis

B Anhang

D.6 Zeitplan

Der Zeitplan ist in der nachfolgenden Tabelle dargestellt.

Von	Bis	Aufgaben
01.11.2023	30.11.2023	Erstellung Exposé, Literaturarbeit
01.12.2023	15.12.2023	Abstimmung und Überarbeitung Exposé
15.12.2023	31.12.2023	Anmeldung Masterarbeit und Aushändigung
01.01.2024	31.01.2024	Ableitung und Untersuchung der prototypischen Tasks
01.02.2024	29.02.2024	Erhebung der Leistungsaufnahme von Tasks auf Testsystem
01.03.2024	31.03.2024	Evaluierung eines Vorgehens, um Tasks aus konkreter Anwendung abzuleiten
01.04.2024	30.04.2024	Berechnung der Soll/Ist-Werte sowie Abweichung
01.05.2024	30.06.2024	Schreiben der Master-Thesis
-	30.06.2024	Abgabe

Tabelle D.1: Zeitplan

Anhang E

In eigener Sache

E.1 Aufwandsstatistik

Mein Studium begann am 01.04.2014 mit einem Probelauf. Die offizielle Immatrikulation erfolgte dann zum 01.10.2014. Neben der Möglichkeit mich weiterzubilden wollte ich auch herausfinden, ob ich einen universitären Studienabschluss neben einer hohen Arbeitsbelastung, familiären Verpflichtungen und einer Vielzahl persönlicher Interessen erreichen würde. Um währenddessen den Studienfortschritt sicherzustellen, hatte ich sämtliche zeitlichen Aufwände für das Studium mithilfe einer Zeiterfassungsapp erfasst und regelmäßig überprüft. Diese Erfassungen hatte ich dokumentiert, aggregiert und teile ich nachfolgend mit:

Mein Bachelorstudium in Wirtschaftsinformatik endete am 17.12.2018. Für diesen Studiengang hatte ich insgesamt 3.175,5 Stunden erfasst.

Mein Masterstudium in Wirtschaftsinformatik begann am 01.10.2022. Für diesen Studiengang hatte ich bis zur Abgabe dieser Masterarbeit etwa 1.211,5 Stunden erfasst.

Meiner Erfahrung nach lässt sich ein solches Studium, trotz hoher Arbeitsbelastung und einer Vielzahl persönlicher Interessen, gut in die Arbeitswochen integrieren. Aber aufwändig ist ein solches Studium schon.

E.2 Danksagungen

Ich bedanke mich bei

- der Fakultät M+I, insbesondere dem Lehrgebiet Parallelität und VLSI, für die Möglichkeit eine Masterarbeit schreiben zu dürfen und für die Betreuung,
- meiner Familie und allen mir nahestehenden Personen für die kontinuierliche Unterstützung,
- allen die mich kennen und mich trotz so einiger Abwesenheiten nicht vergessen haben.

Anhang F

Erklärung

Name: Kaufmann, Allan
Matrikel-Nr.: 9502998
Fach: Wirtschaftsinformatik

Ich erkläre, dass ich die Abschlussarbeit selbstständig und ohne unzulässige Inanspruchnahme Dritter verfasst habe. Ich habe dabei nur die angegebenen Quellen und Hilfsmittel verwendet und die, aus diesen, wörtlich, inhaltlich oder sinngemäß entnommenen Stellen, als solche den wissenschaftlichen Anforderungen entsprechend, kenntlich gemacht. Die Versicherung selbstständiger Arbeit gilt auch für Zeichnungen, Skizzen oder graphische Darstellungen. Die Arbeit wurde bisher in gleicher oder ähnlicher Form weder derselben noch einer anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Mit der Abgabe der elektronischen Fassung der endgültigen Version der Arbeit nehme ich zur Kenntnis, dass diese mit Hilfe eines Plagiatserkennungsdienstes auf enthaltene Plagiate überprüft und ausschliesslich für Prüfungszwecke gespeichert wird.

Lehrte, 27.06.2024

Allan Kaufmann