A Hierarchical Task Scheduler for Heterogeneous Computing

Hung Quan Vu

Arbeitsbereich Wissenschaftliches Rechnen
Fachbereich Informatik
Fakultät für Mathematik, Informatik und Naturwissenschaften
Universität Hamburg

14.12.2021

Gliederung

- Heterogenes Computing
  - Aufgabenparallelität
  - Herausforderungen beim heterogene Computing
  - Wie können wir die Herausforderungen meistern
  - Derzeitige Arbeit
- RANGER Architektur und Implementierung
  - Baseline-Beschleunigerarchitektur
  - RANGER-Architektur und speicherabgebildete IO-Schnittstelle
  - Top-Level und Low-Level Scheduler
  - Implementierungsdetails von Accelerator Kernels
- Experimentelle Auswertung
- Zusammenfassung
- Literatur

Heterogenes Computing

- In jeder wissenschaftlichen Rechenaufgabe besteht heterogenes Rechnen aus mehreren:
  - Zentraleinheiten (CPUs)
  - Grafikprozessoren (GPUs)
  - Feldprogrammierbares Gate-Array (FPGA)
  - Anwendungsspezifischer integrierter Schaltkreis (ASIC) [3]
- Heterogenes Computing ist eine der zukünftigen Richtungen des High-Performance Computing (HPC) [2]

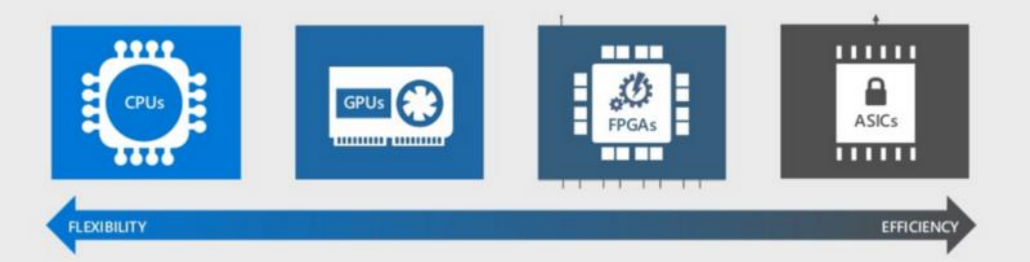


Abb. 1: Überblick über die heterogene Architektur [6]

Heterogenes Computing

- Eine praktikable Lösung für High Performance Computing (HPC)
- Zukunft der “extreme” Heterogenität
- Entscheidend für die Pflege und Verbesserung der Programmierung Portabilität und Produktivität
- Ressourcen verwalten, die von den Heterogene Beschleuniger [2] [3]

Aufgabenparallelität

- Hocheffektives paralleles Programmiermodell
- Laufzeit plant atomare Rechenaufgaben automatisch
- In vielen Programmiersystemen weit verbreitet
- Abhängigkeiten zwischen verschiedenen Tasks können durch einen gerichteten azyklischen Graphen (DAG) dargestellt werden[2]

Herausforderungen beim heterogenen Computing [2]

- Beschleuniger immer vielfältiger werden
  - eine einheitliche Geräteverwaltung im Stil eines Ausstechers möglicherweise nicht optimal
- Mehr Kernel-Beschleuniger mit unterschiedlichen Fähigkeitenverfügbar werden
  - muss sichergestellt werden, dass ein größerer Aufgabenpool vorhanden ist
- Führt ein größerer Aufgabenpool zu einer zunehmenden Dichotomie zwischen der optimalen Verwaltung der Ressourcen.

Wie können wir die Herausforderungen meistern? [2]

- RANGER
  - führt die Aufgabenplanung auf zwei Level durch
    - Top Level
    - Low level
- Es überbrückt die Dichotomie der grobkörnigen Planung, die durch die Schnittstelle zu Programmiermodellen erwünscht ist

Derzeitige Arbeit [2]

- RANGER
- Design und Implementierung im GEM5-Simulator
- Lokale Accelerator-Specific Command Scheduler (ASCS) entwerfen und implementieren
- Wirksamkeit von RANGER zu demonstrieren und quantitativen Flächen- und Rechenaufwand bereitzustellen

RANGER-Architektur und Implementierung [2]

- Baseline-Beschleunigerarchitektur
- RANGER-Architektur und speicherabgebildete IO-Schnittstelle
- Top-Level Scheduler
- Low-Level Scheduler
- Implementierungsdetails von Accelerator Kernels

Baseline-Beschleunigerarchitektur [2]

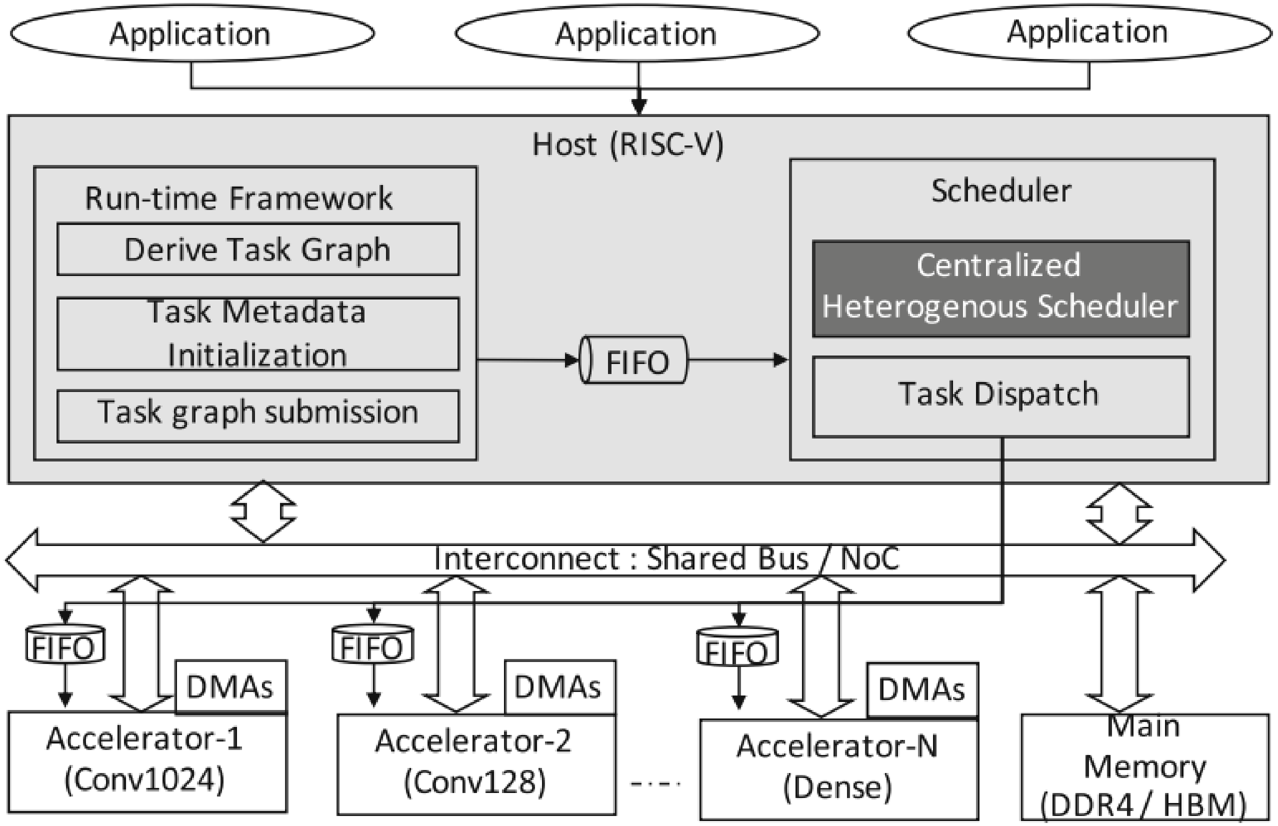


Abb. 2: Baseline-Architekturdesign der heterogenen Beschleunigerplattform [2]

Baseline-Beschleunigerarchitektur [2]

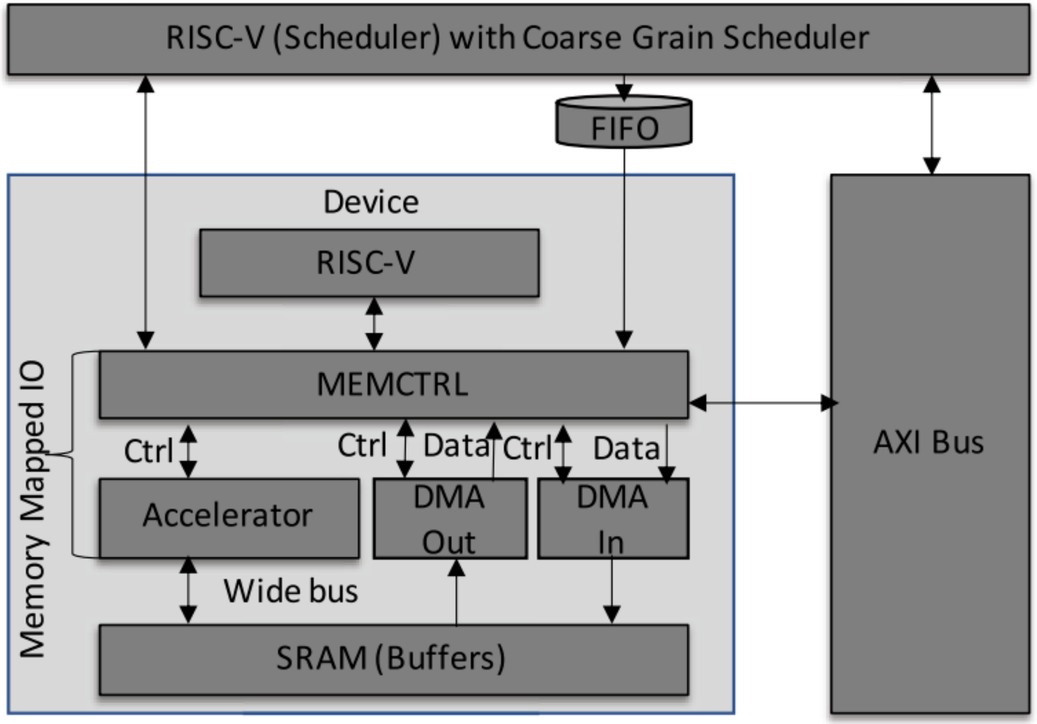


Abb. 3: RANGER-Gerät: RISC-V-, Beschleuniger- und DMA-Kanäle mit Memory Mapped I/O [2]

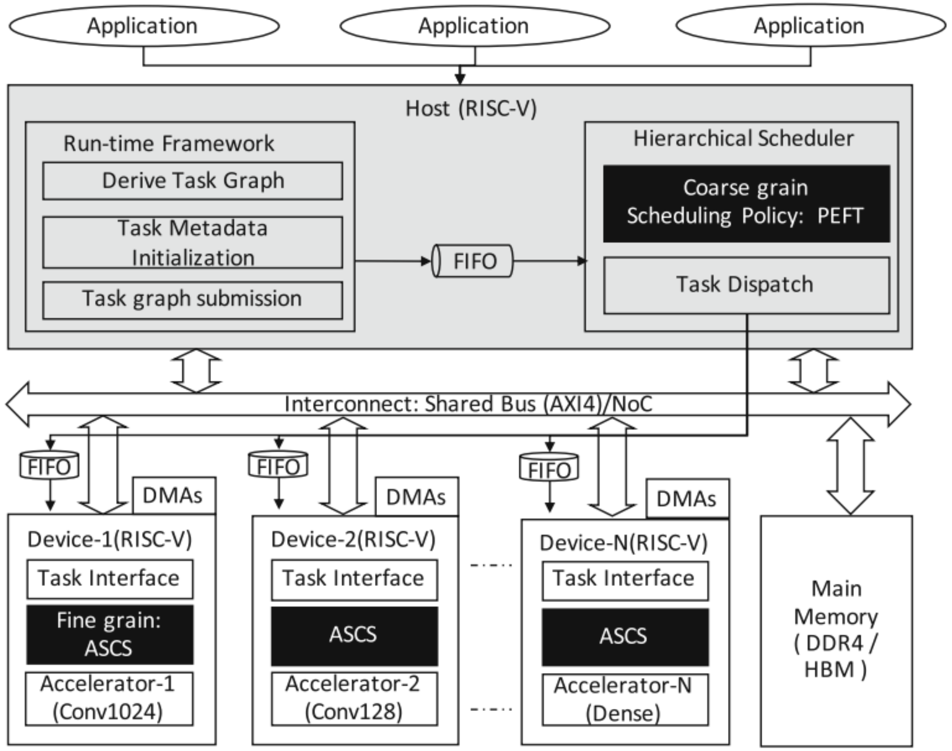


Abb. 4: RANGER-Architekturdesign der heterogenen Beschleunigerplattform [2]

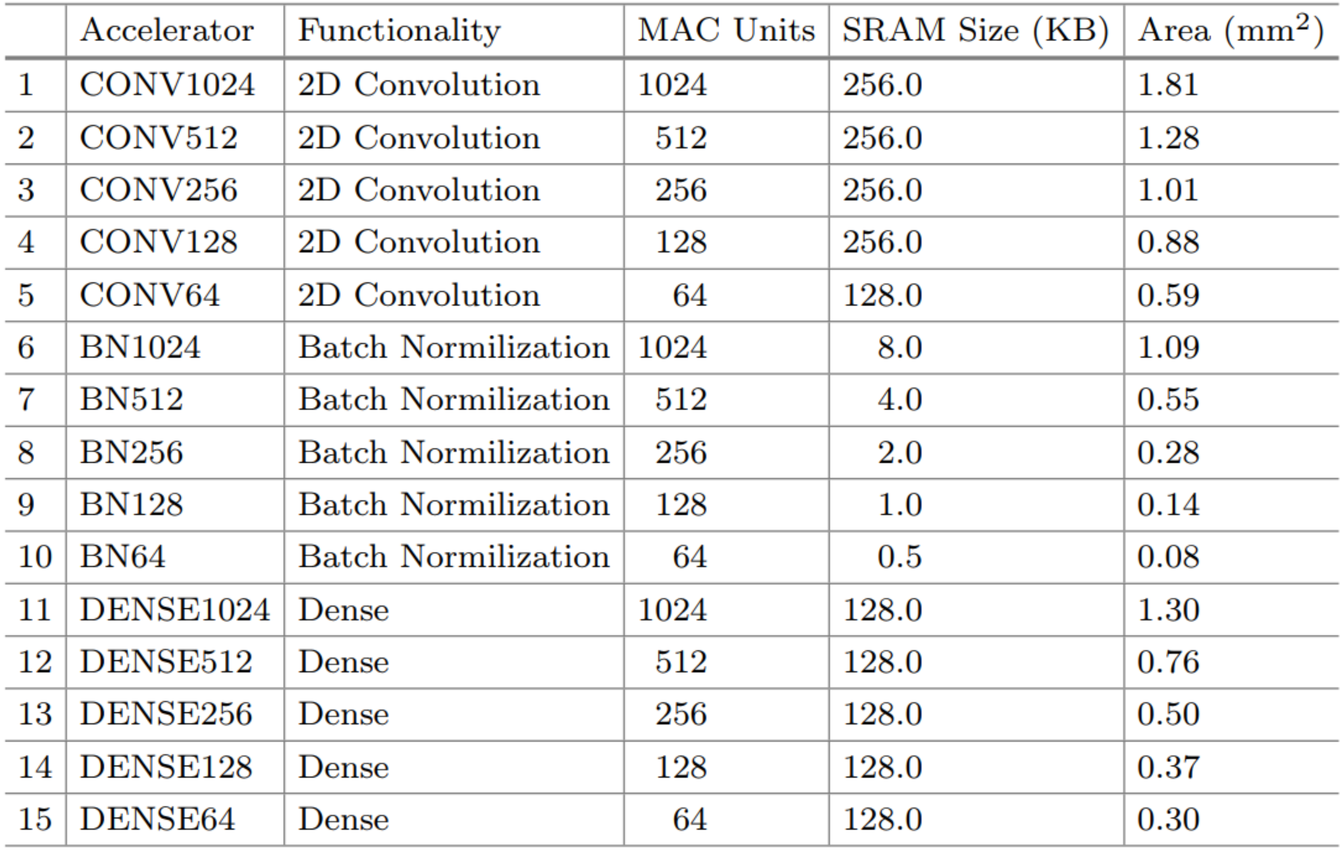
Top-Level- und Low-Level-Scheduler [2-5]

- Der grobkörnige globale Scheduler der Top-Level läuft auf dem Host (Abb. 4).
- PEFT als globaler Zeitplan implementiert
- Der Low-Level-Scheduler ist für die weitere Unterteilung der gegebenen Aufgabe in eine feinere Granularität verantwortlich
- Die Aufteilung und Reihenfolge der Unteraufgaben und ihrer Abhängigkeiten kann von einem Kernel-Beschleuniger zum anderen variieren.

Implementierungsdetails von Accelerator Kernels [2]

- Studie von drei Arten von Beschleunigern:
  - 2D-KONV,
  - BN, und
  - Vollständig verbundene dichte Schicht (DENSE)
- Mehrere Arten: Arten von CONV
  - CONV1024,
  - CONV512,
  - CONV256,
  - CONV128, und
  - CONV64.
- Größeren Beschleuniger haben
  - höhere Leistung
  - benötigen größere Fläche.

Implementierungsdetails von Accelerator Kernels



	Accelerator	Functionality	MAC Units	SRAM Size (KB)	Area (mm^2)
1	CONV1024	2D Convolution	1024	256.0	1.81
2	CONV512	2D Convolution	512	256.0	1.28
3	CONV256	2D Convolution	256	256.0	1.01
4	CONV128	2D Convolution	128	256.0	0.88
5	CONV64	2D Convolution	64	128.0	0.59
6	BN1024	Batch Normilization	1024	8.0	1.09
7	BN512	Batch Normilization	512	4.0	0.55
8	BN256	Batch Normilization	256	2.0	0.28
9	BN128	Batch Normilization	128	1.0	0.14
10	BN64	Batch Normilization	64	0.5	0.08
11	DENSE1024	Dense	1024	128.0	1.30
12	DENSE512	Dense	512	128.0	0.76
13	DENSE256	Dense	256	128.0	0.50
14	DENSE128	Dense	128	128.0	0.37
15	DENSE64	Dense	64	128.0	0.30

Abb. 5: Liste der Kernel-Beschleuniger [2]

Experimentelle Auswertung [2]

- Bewertung des RANGER
  - Anwendungsbenchmarks
  - Skalierbarkeitsstudie
  - Overhead der lokalen Scheduler

Anwendungsbenchmarks [2]

- DNN Benchmarking
  - Einführung V3
  - ResNet-50
  - UNet
  - Vgg16
- RANGER ist ein Allzweck-Scheduler

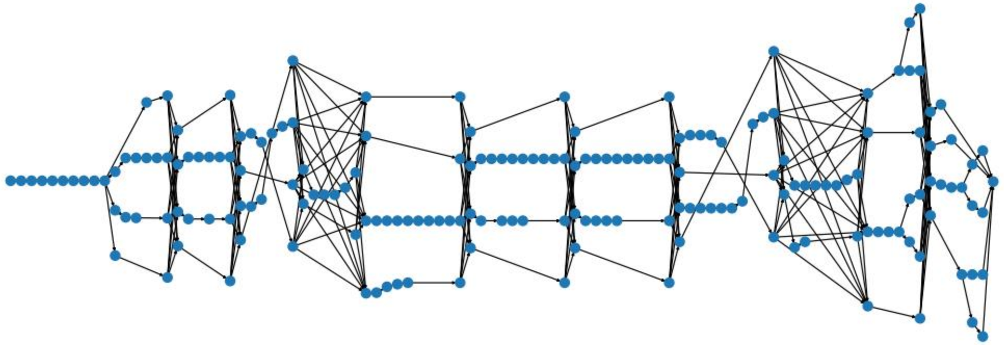
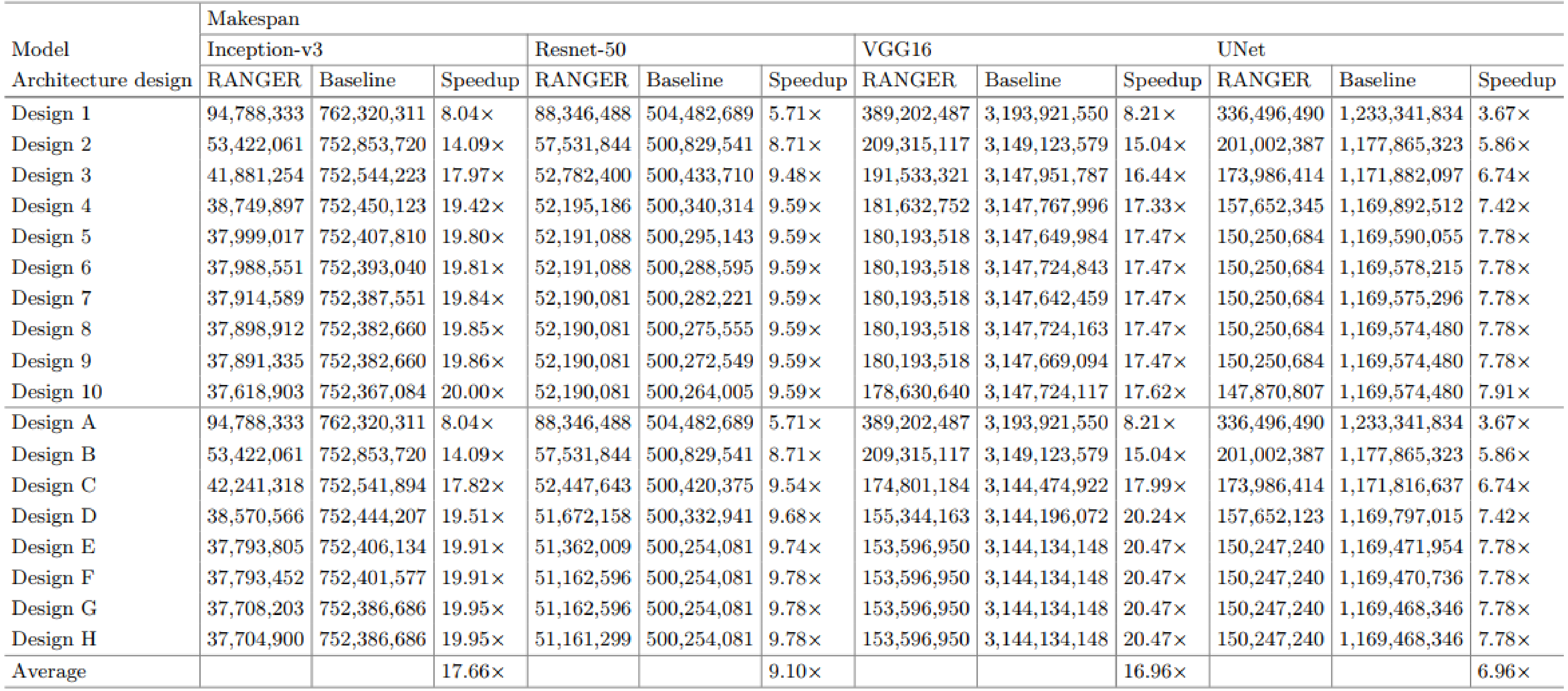


Abb. 6: Aufgaben-DAG von Inception-v3. Der Knoten ganz links ist der Quellknoten der DAG, und der Senkenknoten befindet sich ganz rechts [2]

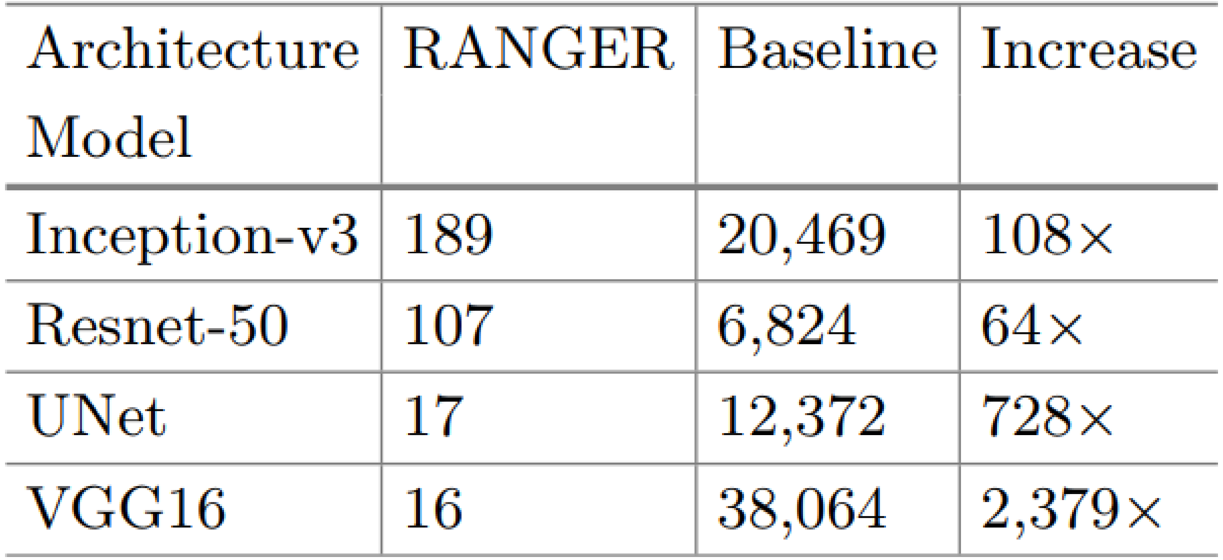
Vergleich der Makespans für verschiedene RANGER [2]



	Makespan											
Model	Inception-v3			Resnet-50			VGG16			UNet		
Architecture design	RANGER	Baseline	Speedup	RANGER	Baseline	Speedup	RANGER	Baseline	Speedup	RANGER	Baseline	Speedup
Design 1	94,788,333	762,320,311	8.04×	88,346,488	504,482,689	5.71×	389,202,487	3,193,921,550	8.21×	336,496,490	1,233,341,834	3.67×
Design 2	53,422,061	752,853,720	14.09×	57,531,844	500,829,541	8.71×	209,315,117	3,149,123,579	15.04×	201,002,387	1,177,865,323	5.86×
Design 3	41,881,254	752,544,223	17.97×	52,782,400	500,433,710	9.48×	191,533,321	3,147,951,787	16.44×	173,986,414	1,171,882,097	6.74×
Design 4	38,749,897	752,450,123	19.42×	52,195,186	500,340,314	9.59×	181,632,752	3,147,767,996	17.33×	157,652,345	1,169,892,512	7.42×
Design 5	37,999,017	752,407,810	19.80×	52,191,088	500,295,143	9.59×	180,193,518	3,147,649,984	17.47×	150,250,684	1,169,590,055	7.78×
Design 6	37,988,551	752,393,040	19.81×	52,191,088	500,288,595	9.59×	180,193,518	3,147,724,843	17.47×	150,250,684	1,169,578,215	7.78×
Design 7	37,914,589	752,387,551	19.84×	52,190,081	500,282,221	9.59×	180,193,518	3,147,642,459	17.47×	150,250,684	1,169,575,296	7.78×
Design 8	37,898,912	752,382,660	19.85×	52,190,081	500,275,555	9.59×	180,193,518	3,147,724,163	17.47×	150,250,684	1,169,574,480	7.78×
Design 9	37,891,335	752,382,660	19.86×	52,190,081	500,272,549	9.59×	180,193,518	3,147,669,094	17.47×	150,250,684	1,169,574,480	7.78×
Design 10	37,618,903	752,367,084	20.00×	52,190,081	500,264,005	9.59×	178,630,640	3,147,724,117	17.62×	147,870,807	1,169,574,480	7.91×
Design A	94,788,333	762,320,311	8.04×	88,346,488	504,482,689	5.71×	389,202,487	3,193,921,550	8.21×	336,496,490	1,233,341,834	3.67×
Design B	53,422,061	752,853,720	14.09×	57,531,844	500,829,541	8.71×	209,315,117	3,149,123,579	15.04×	201,002,387	1,177,865,323	5.86×
Design C	42,241,318	752,541,894	17.82×	52,447,643	500,420,375	9.54×	174,801,184	3,144,474,922	17.99×	173,986,414	1,171,816,637	6.74×
Design D	38,570,566	752,444,207	19.51×	51,672,158	500,332,941	9.68×	155,344,163	3,144,196,072	20.24×	157,652,123	1,169,797,015	7.42×
Design E	37,793,805	752,406,134	19.91×	51,362,009	500,254,081	9.74×	153,596,950	3,144,134,148	20.47×	150,247,240	1,169,471,954	7.78×
Design F	37,793,452	752,401,577	19.91×	51,162,596	500,254,081	9.78×	153,596,950	3,144,134,148	20.47×	150,247,240	1,169,470,736	7.78×
Design G	37,708,203	752,386,686	19.95×	51,162,596	500,254,081	9.78×	153,596,950	3,144,134,148	20.47×	150,247,240	1,169,468,346	7.78×
Design H	37,704,900	752,386,686	19.95×	51,161,299	500,254,081	9.78×	153,596,950	3,144,134,148	20.47×	150,247,240	1,169,468,346	7.78×
Average			17.66×			9.10×			16.96×			6.96×

Abb. 7:Vergleich der Fabrikspannen für verschiedene RANGER- und Basisausführungen.
Im Durchschnitt erreicht RANGER eine 12,7-fache Beschleunigung [2]

Aufgaben für den globalen Host in RANGER [2]



Architecture Model	RANGER	Baseline	Increase
Inception-v3	189	20,469	108×
Resnet-50	107	6,824	64×
UNet	17	12,372	728×
VGG16	16	38,064	2,379×

Abb. 8: Die Anzahl der Aufgaben, die der globale Host in der RANGER- und Baseline-Architektur berücksichtigen muss [2]

Skalierbarkeitsstudie [2]

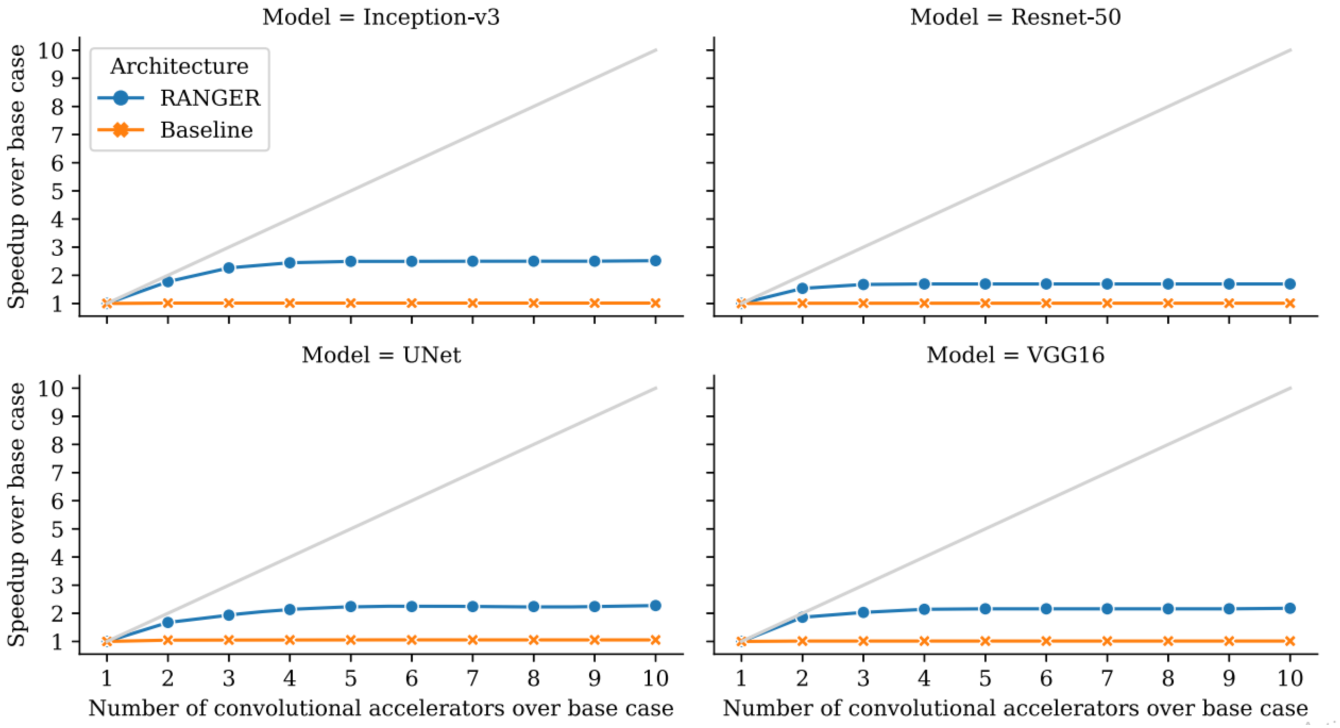


Abb. 9: Anzahl der Faltungsbeschleuniger im Basisfall [2]

Overhead der lokalen Scheduler [2]

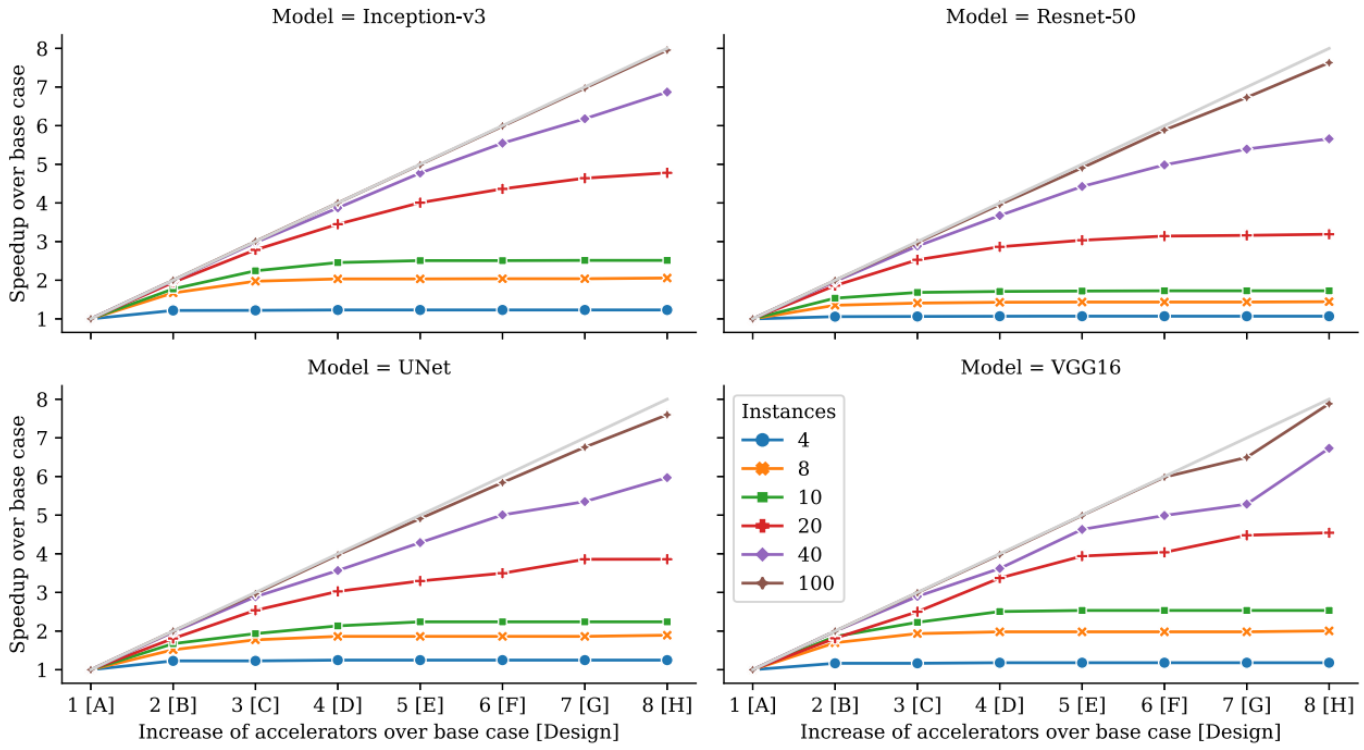


Abb. 10: Erhöhung der Beschleuniger gegenüber dem Basisfall [2]

Zusammenfassung

- RANGER
  - ein Framework und Architekturdesign für hierarchisches Task-Scheduling im extrem heterogenen Computing.
  - entscheidender Vorteil des hierarchischen Schedulings
  - Portabilität und Produktivität der Programmierung im heterogenen Computing einfacher aufrechtzuerhalten.
- Lokalisierter Low-Level-Scheduler
  - ermöglicht die Bereitstellung ausgefeilterer
  - beschleunigerspezifischer Scheduling-Lösungen
  - RANGER verwendet angepasste RISC-V-Kerne, um den Rechenaufwand der Aufgabenplanung zu verringern.
- RANGER-Architektur
  - 12,7-fache Leistungssteigerungen
  - Spannweite bei 2,7 % Flächen-Overhead in einer 16-nm-Technologie.

Literatur

- [1] https://www.adlinktech.com/en/heterogeneous-computing
- [2] A Hierarchical Task Scheduler for Heterogeneous Computing
- [3] https://www.arrow.com/en/research-and-events/articles/fpga-vs-cpu-vs-gpu-vs-microcontroller
- [4]https://www.sciencedirect.com/science/article/abs/pii/B9780124202320000052
- [5] https://www.teach-ict.com/as_as_computing/ocr/H447/F453/3_3_1/scheduling/miniweb/pg2.htm#:~:text=The%20'Low%20Level%20scheduler%20(LLS,between%20primary%20and%20secondary%20memory.
- [6] https://medium.com/grovf/heterogeneous-computing-as-a-next-generation-architecture-for-scaling-data-centers-trends-ae5c5f5725f9