Heterogeneous Computing: Different Concepts with different Results

Wladimir Assmann*, Fabian Peddinghaus**
Embedded Systems Architecture Group
Technische Universität Berlin
10587 Berlin, Germany

***Zusammenfassung*—Multikernprozessoren bieten Leistung- und Energieeffizienzsteigerung im Vergleich zu Einzelkernprozessoren. Bei besonders energiekritischen Anwendungen bieten homogene Multikernprozessoren jedoch weiterhin nicht die nötige Energieeffizienz. In solchen Fällen kann durch Spezialisierung einzelner Komponenten oder Kerne die Effizienz des Multikernprozessors deutlich gesteigert werden. Es Bedarf sogenannter "heterogener Architekturen". Diese können, entsprechend ihrer jeweiligen Anwendung, gezielt entworfen werden. Wir präsentieren auf den folgenden Seiten drei unterschiedliche Ansätze einen heterogenen Multikernprozessoren zu designen. Dabei geben wir zunächst eine kurze Einführung in Heterogeneous Computing und einen Überblick über die drei Paper. Anschließend vergleichen wir die drei Paper unter Verschiedenen Gesichtspunkten miteinander. Wir stellen fest, dass Leistungssteigerungen von mehr als 60% und Energy Delay Produkt (EDP) Reduktionen von bis zu 50% möglich sind. Abschließend nennen wir mögliche Verbesserungen und Ideen für weitere, zukünftige Forschung.**

***Index Terms*—Multikernprozessor, Heterogene Systeme, Embedded Computing**

I. Einführung

Im Bereich des Embedded Computing wird hohe Energieeffizienz benötigt. Dabei können sich Chip Designer heute nicht mehr auf Moores Law verlassen. Auch Dennard Scaling ist nicht mehr gültig und erschwert somit die Entwicklung moderner und effizienter Prozessoren. Hinzu kommt, dass Amdahls Law die Effektivität großer Multikernprozessoren beschränkt [4]. Moderne heterogene Anwendungen erreichen auf generischen homogenen Multikernprozessoren nicht mehr die für Eingebettete Systeme essentielle Energieeffizienz. So ist es für Energieeffizienz kontraproduktiv, wenn z.B. Superskalare Prozessoren für Multimedia-Verarbeitung genutzt werden [7]. Dies zeigt für eingebettete Systeme eine Notwendigkeit spezialisierter Prozessoren auf, welche anwendungsspezifische Paradigmen verfolgen.

Die Motivation hinter heterogenen Architekturen besteht letztlich darin, dass unterschiedliche Anwendungen bzw. Prozesse unterschiedliche Ressourcen benötigen [17]. Heterogene Architekturen bieten im Idealfall genau die Ressourcen, die eine jeweilige Anwendung bzw. ein jeweiliger Prozess benötigt. So wird im Vergleich zu homogenen Architekturen weniger Energie benötigt, da keine unnötigen Ressourcen zur Verfügung gestellt werden. Gleichzeitig kann je nach Parallelisierung ein erhöhte Leistung erzielt werden.

Heterogene Architekturen bieten eine Vielzahl von Designmöglichkeiten. So können z.B. unterschiedliche Instruction Set Architekturen (ISAs) auf einem Prozessor vereint werden. Auch die Hardware bzw. die Mikroarchitektur kann direkt angepasst und z.B. durch dedizierte Hardwarebeschleuniger [10] ergänzt werden. Diese stellen spezialisierte und sehr effiziente Hardware zur Verfügung. Sie bieten die Möglichkeit z.B. spezielle, wiederkehrende Algorithmen effizient und schnell auszuführen [15]. Durch heterogene Prozessoren kann z.B. die Instruction Level Parallelität (ILP) der Anwendungen besser genutzt und so Leistung und Energieeffizienz eines Multikernprozessors gesteigert werden. Heterogenität ermöglicht Prozessoren ihre Ressourcen besser zu nutzen und ein deutlich größeres Spektrum an Anwendungen effizient auszuführen.

Heterogenität birgt allerdings eine Reihe von "Design Challenges". So ist z.B. die Zahl der variablen Parameter in einer heterogenen Architektur um ein vielfaches größer als in einer vergleichbaren homogenen Architektur. Die Suche nach der optimalen Parameterkombination gestalten sich deutlich komplexer und Bedarf u.U. spezieller Verfahren. Außerdem bietet die Heterogenität der Multikernprozessoren besondere Herausforderungen im Bezug auf "Prozess-Scheduling". Prozesse müssen entsprechend ihrer Resourcenanforderungen einem geeigneten, freien, heterogenen Kern zugeteilt werden. Prozesse können nicht einfach, wie bei homogenen Multikernprozessoren, von einem heterogenen Kern auf einen anderen, womöglich komplett anders aufgebauten, heterogenen Kern migrieren. Auch hier Bedarf es besonderer Verfahren und Algorithmen. Verwandt mit dieser Problematik ist die Frage nach Binärer-Kompatibilität. So können sich nicht nur Kerne sondern auch ganze heterogene Multikernprozessoren grundlegend unterscheiden. Es Bedarf also u.U. gesonderter Compiler bzw. Toolchains und Binärdateien. Diese "Challenges" werden durch die drei Paper in den folgenden Abschnitten analysiert und mögliche Lösungsansätze präsentiert.

Zunächst geben wir in Abschnitt II einen Überblick über verwandte Arbeiten. Anschließend präsentieren wir in Abschnitt III drei unterschiedliche Ansätze Heterogenität in Multikernprozessoren zu realisieren. In Abschnitt IV vergleichen wir die unterschiedlichen Ansätze unter verschiedenen Gesichtspunkten miteinander. Es folgen in Abschnitt V eigene Ideen zur Anwendung und Verbesserung der Paper und zuletzt fassen wir in Abschnitt VI unsere Erkenntnisse zusammen.

*wladimir.assmann@campus.tu-berlin.de
**peddinghaus@campus.tu-berlin.de

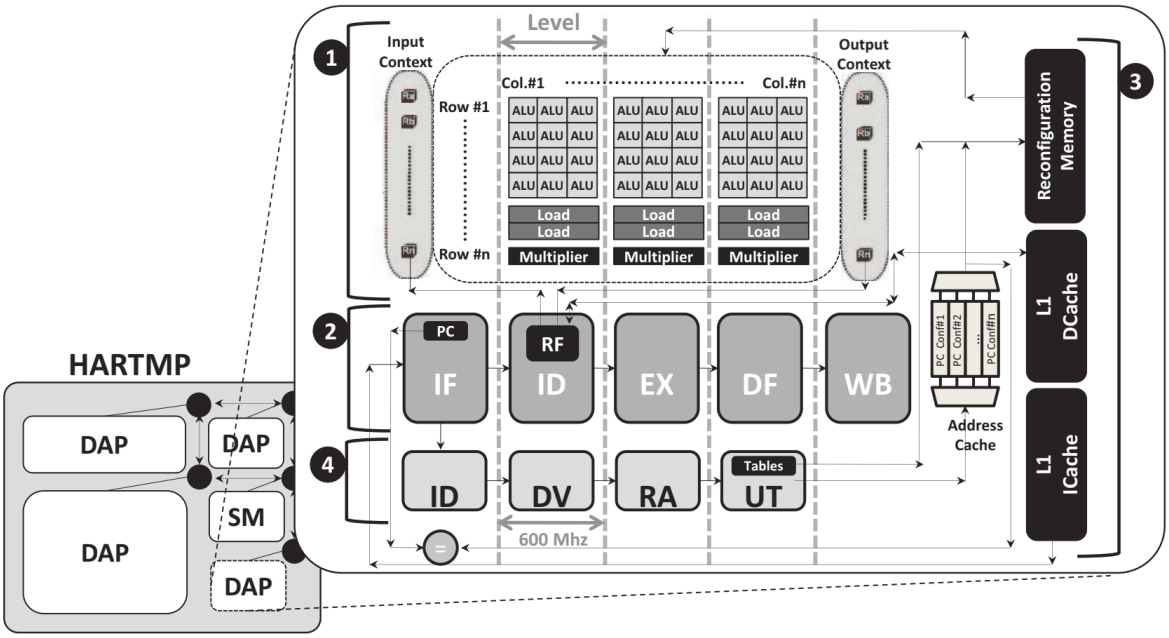

Abbildung 1. HARTMP: Architektur [15]

II. Related Work

Heterogenous Computing ist keine neu Idee. Bereits 2003 präsentierten R. Kumar, et al. [8] einen heterogenen Single-ISA Multikernprozessor. Ziel der Wissenschaftler war es vor allem die Leistungsabgabe der Prozessoren zu verringern. Sie legten dabei wichtige Grundsteine für weitere Arbeiten auf dem Gebiet. 2004 folgte ein Paper der selbigen Wissenschaftler zu Scheduling Algorithmen auf heterogenen Multikernprozessoren [9].

Auch Projekte wie der Warp [12] und der Cell [6] Prozessor nutzten bereits heterogene Ansätze. Dabei wurden im Fall des Warp Prozessors rekonfigurierbare Hardwarebeschleuniger auf FPGAs zur Beschleunigung gewisser Algorithmen genutzt. Der Cell Prozessor hingegen nutze, ähnlich wie Grafikkarten, eine Vielzahl von spezialisierten Processing Elements (PEs), um z.B. schnell Fourier Transformationen errechnen zu können.

Mit ARMs big.LITTLE Architektur [5] sind heterogene Architekturen auch in der Kommerziellen Welt vertreten. big.LITTLE nutzt unterschiedliche Kerne mit identischer ISA um eine bessere Energieeffizienz zu erzielen.

Die hier präsentierten Paper greifen die erwähnten Wissenschaftlichen Arbeiten auf, augmentieren sie um neue Ansätze und Ideen, und Entwickeln auf Grundlage dieser auch komplett neuartige Konzepte.

III. Präsentation der Paper

A. Paper 1

”A reconfigurable heterogeneous Multicore with a homogeneous ISA” [15] ist das von J. D. Souza et. al. auf der DATE 2016 veröffentlichte Paper, in dem der Ansatz von heterogenen Prozessoren mit identischer Instrution Set Architektur (ISA) mithilfe ihrer eigens entwickelten Architektur „Heterogeneous Arrays for Reconfigurable and Transparent Multicore Processing“ (HARTMP), vorgestellt wird.

1) Architektur: Es wird die Fähigkeit der heterogenen Kerne Instruction Level Parallelität (ILP) nutzen zu können variiert. Der dafür verantwortliche Baustein des sogenannten Dynamic Adaptive Processor Kernes (DAP), ist das “heterogene Array“ (Abb. 1). Dabei handelt es sich um einen zur Laufzeit rekonfigurierbaren Ausführungspfad, bestehend aus ALUs, Multiplizierern und Load/Store-Einheiten. Die einzelnen Functional Units sind durch Auswahlschaltungen miteinander verbunden und können z.B. kaskadiert oder parallelisiert werden. So sind sie in der Lage beliebige Befehle auszuführen. Die Konfiguration der Functional Units wird dabei durch eine parallel laufende Pipeline bestimmt. Diese arbeitet aus Sicht der Software komplett transparent. So kann der Prozessor wie ein gewöhnlicher homogener Single-ISA-Prozessor programmiert werden. Zur Vermeidung redundanter Berechnung durch die parallel laufende Pipeline ist HARTMP mit einem dedizierten Konfigurations-Cache ausgestattet. In diesem werden bereits errechnete Arraykonfigurationen gespeichert. Er wird durch den PC indiziert und lädt bei Bedarf eine bereits gespeicherte Konfiguration in das Array.

Heterogenität unter den DAPs resultiert aus einer variablen Anzahl an Einheiten, welche den einzelnen Arrays zur Verfügung stehen. Diese werden zwischen den Kernen unterschiedlich dimensioniert und können so für Programme mit unterschiedlich hoher ILP optimiert werden. Dabei wird ohne sogenannte “Design Space Exploration“ ein heterogenes Design festgelegt. In diesem sind die Ausbaustufen der DAPs des Multikernprozessors bezogen auf die Anzahl der Kerne $^1/_4$ groß, $^1/_4$ mittel und $^2/_4$ klein. Diese Referenzdesigns werden für weitere Simulationen verwendet. Die Wahl der Kerngrößen durch die Wissenschaftler bleibt in ihrem Paper unbegründet.

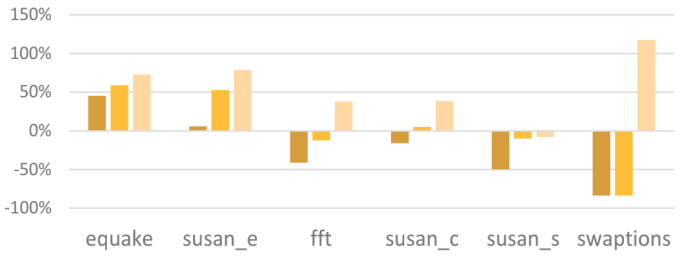

Abbildung 2. HARTMP: Performance unter Area Parity [15]

2) Leistung: Die in der Simulation erfolgten Benchmarks spiegeln unterschiedliche Szenarien in Bezug auf ILP und Thread Level Parallelität (TLP) wider. Dabei werden unterschiedlich dimensionierte homogene Implementierungen der gleichen Architektur als Referenz verwendet und unter dem Aspekt der Area-Parity, also homogene Multikernprozessoren mit identischer Fläche, mit ihren heterogenen Versionen verglichen.

Dabei wird deutlich, dass die homogenen Implementierung vor allem niedrige TLP bzw. hohe ILP bevorzugen, wie in Abbildung 2 mit nach TLP aufsteigend sortierten Benchmarks erkenntlich ist.

Vor allem im Benchmark *equake* zeigt die heterogene Variante einen Anstieg um 50% des EDP aufgrund gut balancierter Threads. Die homogene Version übertrifft dies jedoch teils bei hoher TLP, mit 80% besserem EDP. Diese profitiert von der doppelten Anzahl an Kernen, während durch geringere ILP auf den größeren homogenen Kernen wahrscheinlich eine weniger optimale Ausnutzung der verfügbaren Einheiten (ALUs, L/S, etc.) vorliegt.

B. Paper 2

”Harnesing ISA Diversity: Design of a Heterogenous-ISA Chip Multiprocessor” [17] von Ashish Venkat und Dean Tullsen von der University of California at San Diego präsentiert einen neuartigen Ansatz unterschiedliche Instruktionssatz-Architekturen (ISAs) in einem Prozessor zu vereinen. Die Wissenschaftler haben dabei drei bekannte ISAs, die ARM-Thumb, die x86-64 und die Alpha Architektur in einem Multikernprozessor vereint. Sie konnten zeigen, dass durch Heterogene ISAs in einem einzelnen Multiprozessor bis zu 21% mehr Leistung, 23% weniger Energie und ein 32% geringeres Energy Delay Product (EDP) im Vergleich zu homogenen ISA Multikernprozessoren erzielt werden kann.

1) Motivation: Die unterschiedlichen ISAs besitzen intrinsische Heterogenität. Diese intrinsische Heterogenität wirkt sich sowohl auf die für die jeweilige ISA kompilierte Software, als auch auf die Mikroarchitektur, also die Implementierung der ISA in Hardware, aus. So unterscheiden sich die ISAs z.B. elementar in Codedichte, Decode- und Instruktionskomplexität, Registerpressure, native Floating-Point Arithmetik und SIMD Hardware. Diese Heterogenität führt dazu, dass für unterschiedliche Prozesse, und sogar für Prozessabschnitte unterschiedliche ISA Kerne hinsichtlich Geschwindigkeit und Energie vorzuziehen sind. So ist es z.B. vorteilhaft einen sehr vektorlastigen Prozess auf einem Kern mit x86-64 ISA auszuführen, da die x86-64 ISA leistungsstarke Vektorinstruktionen zur Verfügung stellt. In energie-kritischen Situationen, z.B. im Batteriebetrieb, kann es hingegen Sinnvoll sein, Prozesse auf einem Kern mit der Energieeffizienten Thumb-ISA auszuführen.

2) Architektur: Damit die Heterogenität des Multikernprozessors ideal ausgenutzt werden kann, muss ein Prozess während seiner Ausführung zwischen unterschiedlichen heterogenen Kernen migrieren können. Die Migration gestalten sich allerdings äußerst komplex, da sich der Zustand des Prozesses in einem architekturspezifischen Format befindet. Der Prozess kann also, anders als bei homogenen ISA Multikernprozessoren, nicht einfach von einem Kern zu einem anderen migrieren. Der Prozess muss ab dem Migrationszeitpunkt auf dem Ziel ISA Kern zunächst von der ”alten” in die ”neue” ISA binär Echtzeit-Übersetzt werden. Diese Übersetzung geschieht solange, bis ein sogenannter ”Äquivalenzpunkt” in der kompilierten Binärdatei erreicht wird. An diesem kann mit Hilfe sogenannter ”State-Transformationen” der Prozesszustand von der ”alten” ISA in die ”neue” ISA übertragen werden. Das Übersetzen wird durch besondere Binärdateien, sogenannte ”Fat-Binarys”, ermöglicht. Diese besitzen mehr Informationen als herkömmliche Binärdateien, z.B. die State-Transformationen. Um Fat-Binarys generieren zu können haben die Wisschenschaftler eine eigene Cross-Compiler Toolchain entwickelt. Dafür nutzten die Wissenschaftler das Modulare LLVM Compiler Framework und das damit kompatible Clang Frontend.

3) Experimenteller Aufbau: Die Wissenschaftler präsentieren als Referenzimplementierung einen idealen heterogenen ISA 4-Kern-Prozessor. Durch die starke Heterogenität der ISA und der damit verbundenen heterogenen Mikroarchitekturen ist der sogenannte ”Design-Space” sehr groß. Design-Space bezeichnet das Kartesische Produkt aller variablen Parameter, die in einen heterogenen ISA Kern variiert werden können. Die Wissenschaftler nutzen daher ein eigens entwickelt Verfahren zum ”Design-Space-Pruning”. Dabei werden zunächst alle offensichtlich nicht kompatiblen Parameter-Kombinationen ausgeschlossen. Anschließend werden alle heterogenen ISA-Kombinationen einzeln Simuliert und anschließend zu einem Prozessor vereint. So können die benötigten Simulationskombinationen drastisch reduziert und optimale heterogene Multikernprozessoren nach angemessener Zeit gefunden werden. Der Design Space wurde effektiv von 128.3 Milliarden unterschiedlichen Multikernprozessoren auf 600 unterschiedliche Einzelkernprozessoren reduziert. Die Wissenschaftler nutzten gem5 [2] um die CPU Performance zu simulieren. Um Flächen- und Energieverbrauch zu analysieren wurde McPAT [11] genutzt.

4) Leistung: Heterogene ISA Multikernprozessoren liefern auch unter engen Design-Limitierungen, wie z.B, einem festen Leistungs- oder Flächenbudget, im Vergleich zu homogenen ISA-Multikernprozessoren, noch erhebliche Energie- und Leistungsverbesserungen (siehe Abb. 3). So ist der heterogene ISA-Multikernprozessor bei festem Energiebudget bis zu 18,8 % Leistungsstärker als ein vergleichbarer homogener ISA-Multikernprozessor. Bei sehr großzügigen Design-Limitierungen ist der Performance-Unterschied deutlich ge-

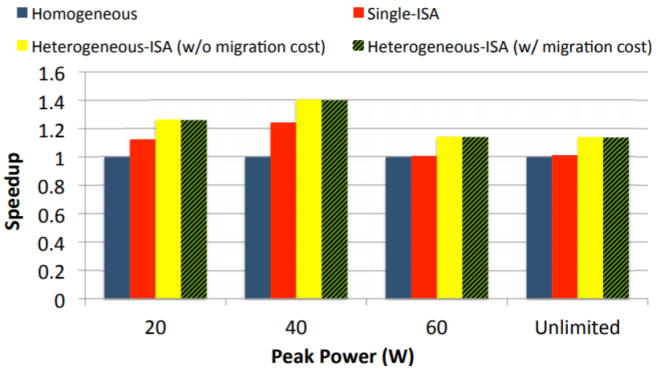

Abbildung 3. Speedup unter Peak-Power-Beschränkung [17]

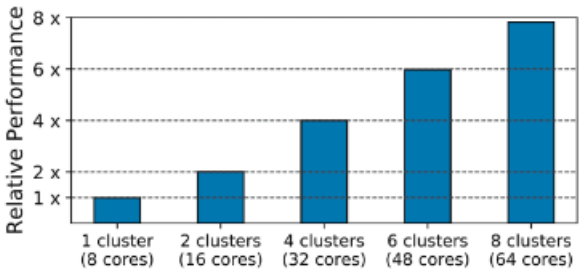

Abbildung 4. Relativer Speedup verschiedener Clustergrößen [10]

ringer, da die Multikernprozessoren dann immer mehr homogenen Architekturen entsprechen. Außerdem konnten die Wissenschaftler zeigen, dass sowohl die größeren Binärdateien, die Fat-Binarys, als auch die Run-Time-Migration zwischen den heterogenen ISA-Kernen eine vernachlässigbare Leitungsverschlechterung bewirkt. Der Overhead durch Fat-Binarys war für die Wissenschaftler nicht messbar. Der Overhead durch die Run-Time-Migration lag im schlimmsten Fall bei 0,7%.

C. Paper 3

Das Paper ”Hero: Heterogenous Embedded Research Platform for Exploring RISC-V Manycore Accelerators on FPGA” [10] von Wissenschaftlern der ETH Zürich und der Universität Bologna stellt einen sogenannten Heterogenous Embedded System on Chip (HESoC) vor. Dabei handelt es sich um einen normalen Host Multikernprozessor der durch einen sogenannten Pogrammable Manycore Accelerator (PCMA) ergänzt wird. Der PCMA besteht aus vielen modifizierbaren Kernen, sogenannten Processing Elements (PEs). HERO vereint General-Purpose Computing mit Domain-Specific Computing. Der PMCA ist bei HERO in Form von 64 RISC-V Kernen als Softcores auf einem FPGA realisiert und mit einem ARM Cortex-A Multikern Host Prozessor augmentiert.

1) Architektur: Bei HERO handelt sich um eine Research-Platform die als Grundlage für weitere Forschung dienen soll. HERO stellt eine komplette Entwicklungssuite, bestehend aus einem vollständigem Softwarestack, heterogenous Crosscompilation-Toolchain mit OpenMP [3] Support, Linux Treibern und diversen Runtime-Bibliotheken, zur Verfügung. Außerdem bietet HERO automatisierte Hard- und Software Builds, sowie automatisierte Test einzelner Komponenten oder des gesamten Systems. So wird die Zeit für Entwicklung und Validierung auf ein Minimum reduziert. HERO vereint die in der Industrie stark etablierte ARMv8 Architektur mit der aufstrebenden RISC-V [18] Architektur. RISC-V ist eine moderne und modulare RISC Architektur. Sie ist offen (im Sinne von Open-Source) und wird von der Non-Profit RISC-V Foundation spezifiziert. RISC-V ist modular und bietet zur der sogenannten Base-ISA eine Vielzahl von custom Extensions. Z.B. gibt es Floating-Point und Vektor Erweiterungen. RISC-V ist durch ihre modulare Architektur ideal für heterogene Multikernprozessoren geeignet. In HERO wird RISC-V durch PULP [14], einer Prozessorfamilie der ETH Zürich mit Fokus auf Embedded- und Energy-Efficent-Computing, realisiert. Dabei stellt PULP u.a. Digital Signal Processing (DSP), parallel Execution und skalierbare Kerngrößen zur Verfügung.

Häufig werden Accelerators für HESoC getrennt vom Hauptprozessor und dem restlichen System entwickelt. Das Zusammenspiel wird lediglich durch Simulationen untersucht. Dabei müssen allerdings viele relevante Parameter, wie Memory-Hierarchy und Peripherie, bedingt durch ihre hohe Komplexität, vernachlässigt werden. HERO ermöglicht komplexe, real-world Benchmark Evaluation auf Systemebene. HEROs bis zu 64 RISC-V Kerne können mit mehr als 30 Mhz und 1.9 GMIPS Durchsatz arbeiten. So werden selbst schnelle, zyklengenaue softwarebasierte Simulatoren um Größenordnungen überboten.

2) Kommunikation und Speicher: HERO kann auf einer Vielzahl von Hardwareplattformen realisiert werden. Dabei gibt es eine Reihe von elementaren Konfigurationsparametern. So kann z.B. zwischen Bus- oder Networkinterconnect, zwischen verschiedenen Speicher- oder Clustergrößen, usw., gewählt werden. Die Wissenschaftler nutzen in ihrem Paper eine Referenzimplementierung auf der Juno ARM Development Platform. Auf allen Plattformen teilen sich der Hauptprozessor und der PCMA Adressraum und Arbeitsspeicher kohärent durch ein AXI Bus Interface. Die PULP RISC-V Kerne in dem PCMA nutzen anstelle eines Caches, Software managend multibanked Scratchpad-Memorys (SPMs) und multichannel Direct Memory Access (DMA). Zugriffe auf den gemeinsame Hauptspeicher werden durch ein Remapping Address Block (RAB) von virtuellen auf physische Adressen übersetzt. Der Host ARM Prozessor nutzt Multilevel-Caches und eine Memory Managment Unit (MMU) für Hauptspeicherzugriffe. Hostprozessor und PCMA können so effizient Speicher und virtuelle Adresspointer teilen.

3) Simulation und Leistung: Zur Performanceanalyse wurde u.a. ein Matrix-Multiplikations-Benchmark genutzt. Die Matrizen werden durch sogenanntes ”tiling” auf die Cluster des PCMA aufgeteilt. Der Abbildung 4 kann der mit entsprechender Clusteranzahl erzielte Speedup entnommen werden. Man erkennt, dass sich das PCMA sehr gut skalieren lässt. Der Speedup könnte durch den Wechsel von einem Bus- zu einem Network-Interconnect ggf. noch besser und bis deutlich über 8 Cluster skalieren.

Die PEs können dank der modularen RISC-V Architektur einfach angepasst und modifiziert werden. So kann

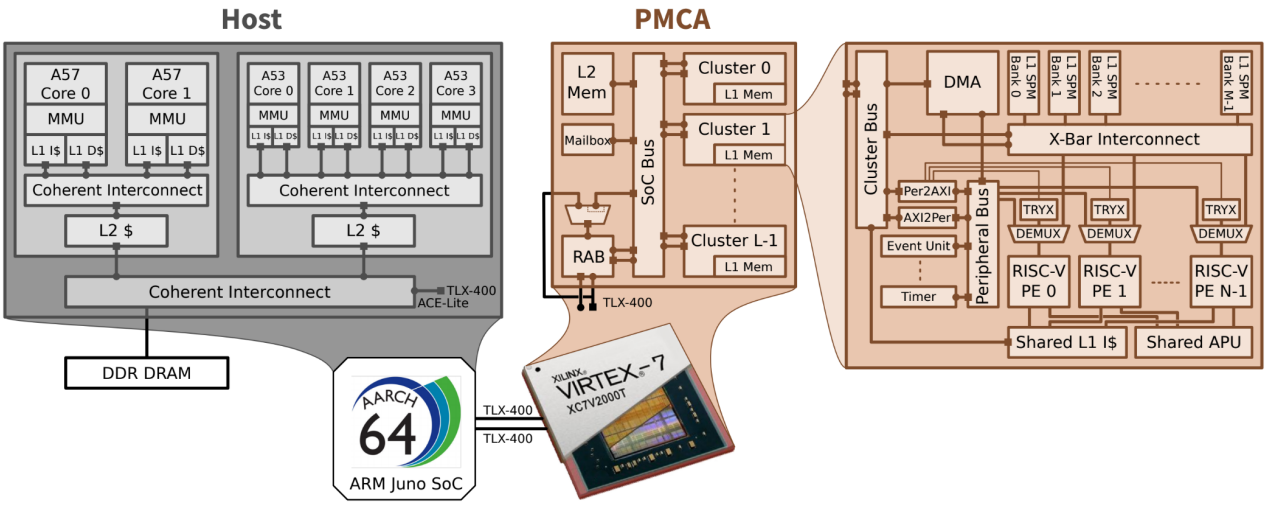

Abbildung 5. Architektur der HERO-Plattform [10]

nach Bedarf z.B. jedem PE eine eigene Multiplikations-, Divisions- oder Floatingpointeinheit zugewiesen werden. Alternativ könne sich mehrere PEs Hardware durch eine cluster-interne Auxiliary-Processing-Element-Unit (APU) teilen. Hardwareaufwand und Computing Performance werden so ideal abgestimmt.

HERO wird mit OpenMP 4.5 programmiert. Anschließend kann durch den eigens entwickelten Softwarestack, basierend auf GCC 5.2, das Offloading auf den PCMA transparent durch das OpenMP Runtime-Environment (RTE) durchgeführt werden.

Durch lokale Puffer in dem PCMA können Daten während der Laufzeit, ohne Beeinflussung des PCMA, gesammelt und anschließend über eine dedizierte Schnittstelle bereitgestellt werden. Diese wird durch einen eigenen Debugger Softwarestack erweitert. So ist es möglich Informationen über den Zustand des PCMA zur Laufzeit zu extrahieren.

IV. Vergleich der Paper

Im Folgenden werden die drei Paper unter den vier wichtigsten Gesichtspunkten: *Ansatz, Architektur und Hardware, Software* und *Leistung* miteinander verglichen. Die in den Papern vorgestellten Architekturen seien mit HARTMP, Heterogeneous-ISA (He-ISA) sowie HERO abgekürzt.

A. Ansätze

Alle drei Architekturen haben grundverschiedene Ansätze um Heterogenität zu erzielen. Während HARTMP Heterogenität durch eine Variation der pro Kern verfügbaren Ausführungseinheiten erreicht, sind bei der He-ISA die einzelnen Kerne auf ISA-Ebene verschieden. Prozesse können zwischen den verschieden ISA-Kernen wechseln. HERO hingegen stellt eine FPGA-basierte Forschungsplattform mit hartem ARM-Cortex Hostprozessor für Design Exploration dar und wird von spezialisierten RISC-V Softcores für programmierbare Hardwarebeschleunigung erweitert. Es werden, wie bei der He-ISA, verschiedene ISAs zusammengeführt, wobei zu beachten ist, dass bei HERO die RISC-V Kerne auch untereinander durch Spezialisierungen verschieden sind. Gerade diese Spezialisierungen der HERO-Beschleuniger weisen konzeptuelle Ähnlichkeiten zu dem unterschiedlich dimensionierten Ausführungspfad in HARTMP auf.

Für HARTMP wird sich aus unbalancierten Prozessen eine Verbesserung des EDP durch heterogenen Kerne erhofft. Bei He-ISA wiederum wird mit ISA-Affinität, also der Beobachtung wie effizient bzw. performant Anwendungen auf den verschiedenen ISAs ausgeführt werden, argumentiert. Bei HERO werden Ansätze aus HARTMP und He-ISA kombiniert. So wird die Forschung und Entwicklung heterogenen Multikernprozessoren deutlich erleichtert.

B. Architektur und Hardware

Die He-ISA nutzt keine zur Laufzeit veränderliche Hardware. Die einzelnen Prozessorkerne stellen für sich genommen herkömmliche ”off-the-shelf” Kerne dar, wie sie in einem beliebigen homogenen Single- oder Multikernprozessor zu finden sind. Erst durch das zusammenbringen unterschiedlicher Prozessorkerne mit heterogenen ISAs wird He-ISA zu einer heterogenen Architektur. Dabei bieten die unterschiedlichen ISAs Heterogenität, die sich auf die Mikroarchitektur auswirkt. Gleichzeitig können Parameter der Mikroarchitektur selber gezielt verändert werden, wie weiter oben bereits erläutert. Die He-ISA bietet, durch ihre heterogenen ISAs, Herausforderungen im Bezug auf Softwarekompatibilität, wie sie bei HARTMP nicht auftreten.

HARTMP nutzt hingegen eine homogene SPARC-basierte ISA. Die Heterogenität kommt durch Variation der Mikroarchitektur auf zwei unterschiedlichen Ebenen zustande. So können auf hoher Ebene die Größen und Ressourcen der einzelnen Dynamic Adaptive Processors (DAPs), also der Kerne von HARTMP, zur Design-Time des Prozessor gewählt werden. Z.B. kann die Pipeline des eigentlichen Prozessors oder die Größe der sogenannten CReAMS Struktur sowie die Verteilung der Functional Units variiert werden. Auf

einer tieferen Ebene kann sich während der Laufzeit die eigentliche CReAMS Architektur dynamisch umstrukturieren. Dabei verändert sich der spezialisierte Datenpfad und passt sich an die Bedürfnisse der jeweiligen Anwendung an. Durch die zur Run-Time variable Hardware unterscheidet HARTMP grundlegend von den anderen vorgestellten Architekturen.

HERO vereint Ansätze aus den beiden zuvor verglichenen Arbeiten, bringt also eine heterogene ISA mit spezialisierter heterogener Hardware zusammen. Im Gegensatz zur He-ISA ist die heterogene ISA bei HERO allerdings nicht elementarer Bestandteil der im Paper präsentierten Arbeit. HERO widmet sich viel mehr dem Rekonfigurierbarem Array, dem PCMA, und dem Zusammenspiel zwischen Host-Prozessor und Rekonfigurierbarem Array. Das PCMA besteht, im Gegensatz zu HARTMP, nicht aus einem spezialisierten Datenpfad, sondern aus vielen kleinen, vollwertigen Prozessoren, den sogenannten Processing Elements (PEs). Die PEs werden, anders als bei HARTMP, zur Compile-Time von der spezialisierten Toolchain auf Grundlage der mit OpenMP gegebenen Anweisungen und der vom Compiler gefundenen intrinsischen Parallelität im Code, konfiguriert. HERO kann, im Unterschied zu HARTMP und He-ISA, auf beliebigen, durch die Toolchain unterstützen "off-the-shelf" Prozessor und FPGA Kombinationen realisiert werden, muss also weder auf komplette FPGA Implementierungen noch auf spezielle ASICs zurückgreifen. HERO ist, im Gegensatz zu den anderen Ansätzen, eine Forschungsplattform. Deswegen bietet HERO ein Reihe von architektonischen Eigenschaften, die bei den anderen Plattformen nicht zu finden sind. So nutzt HERO z.B. "Lightweight Tracer Hardware Blocks" um beliebige Signale auf dem PCMA mit Zeitstempeln zu versehen und zwischenzuspeichern. Diese nutzen die vorhandenen Blockrambausteine auf dem FPGA und können nach Bedarf über das PCMA verteilt werden. Über entsprechende Treiber können die Lightweight Tracer Hardware Blocks von dem Host Prozessor ausgelesen werden und Runtime Informationen über den PCMA Liefern. Vergleichbare Systeme, so z.B. das J-TAG Protokoll [1], werden auch in herkömmlichen Prozessoren, wie z.B. den ARM-Cortex Prozessoren, genutzt.

C. Software

Die unterschiedlichen Architekturen bieten bei der Entwicklung auf der jeweiligen Plattform gewisse Vorteile. So ist bei HARTMP durch die homogene ISA eine hohe Software-Kompabilität vorhanden. Die He-ISA, im Vergleich dazu, benötigt, aufgrund der Verschiedenen ISAs, spezielle Binärdateien (Fat-Binarys). Diese enthalten Maschinencode jeder, auf dem Multikernprozessor vorkommenden ISA. Außerdem beinhalten die Binärdateien State-Transforms, um ISA-Spezifische Zustandsüberführungen realisieren zu können, sowie weitere, prozessorabhängige Informationen. HERO nutzt, ähnlich wie He-ISA, spezielle Binärdatein. Diese beinhalten sowohl die Software für den ARM-Hostprozessor und die RISC-V PEs, als auch die synthetisierte FPGA-Konfiguration für das PCMA. HERO und He-ISA bieten somit nur sehr schlechte hardwareübergreifende, binäre Kompatibilität.

HERO bietet für die Design Space Exploration ein breites Repertoire an Werkzeugen und ist als Forschungsplattform damit schwer vergleichbar mit HARTMP und He-ISA in Hinsicht auf Leistung und Effizienz, da HERO auf anwendungsspezifische Forschung hinaus zielt.

Durch den optimierten Workflow von HERO durch Toolchains etc., ist nicht auszuschließen, dass HERO potentiell anwendungsspezifisch effizientere bzw. schnellere Designs ermöglichen kann, als HARTMP bzw. He-ISA, u.A. weil HERO, wie bereits erwähnt, beide Ansätze vereint.

D. Leistung

Aus o.g. Grund sei im Folgenden lediglich HARTMP und He-ISA in Bezug auf Leistung verglichen. Zu HERO sei dennoch erwähnt, dass im Matrix-Multiplikations-Benchmark (Abb. 4) ein spezifischer Anwendungsfall getestet wurde und damit das Potential an Skalierbarkeit durch HERO illustriert.

Da bei HARTMP die Ergebnisse ausschließlich mit einer homogenen Version der zugrundeliegenden DAPs verglichen werden, ist HARTMP nicht unmittelbar mit der He-ISA vergleichbar, zudem bei beiden unterschiedliche Benchmarks verwendet werden.

HARTMP zeigt vor allem bei hoher ILP bzw. niedriger TLP einen Vorteil über homogenen Architekturen, was sich in einem 50% geringerem EDP widerspiegelt.

Bei der He-ISA wurde leider nicht zwischen ILP und TLP unterschieden und stattdessen der Fokus auf den Vergleich mit homogenen ISA Prozessoren und einzelner, optimaler ISA Implementierungen, sowie den durch die Migration erzeugten Overhead gelegt. Der Overhead ist dabei marginal im Vergleich zu dem erzielten Speedup. Dieser beträgt bis zu 20%. Der EDP-Vorteil beläuft sich auf ca. 30%, wodurch sich folglich der durch die Migration hervorgerufene Overhead amortisiert.

V. Anwedung und Ausblick

Um das Problem der TLP Ausnutzung in HARTMP zu adressieren, könnte mithilfe von Simultaneous Multithreading (SMT) [16] TLP zu ILP transformiert werden. So kann eine effizientere Nutzung der Kerne in vergleichbaren Szenarien realisiert werden. Dabei müsste wiederum mittels umfangreicher Design Space Exploration untersucht werden, ob dabei der Vorteil der heterogenen Kernen, im Zuge des zusätzlichen Aufwandes, bestehen bliebe. Des weiteren entspräche dies einer Optimierung für den Allzweckeinsatz, welcher für die, meist spezialisierten, Eingebetten Systeme nicht notwendig seien könnte.

He-ISA, wie in Paper 2 präsentiert, nutzt lediglich drei unterschiedliche ISAs. Interessant wäre eine Ausführung mit deutlich mehr ISAs. So könnten die Vorteile heterogener Architekturen womöglich nochmals verstärkt werden. Auch interessant wäre eine Implementierung mit sehr hoher Kernanzahl, wie sie z.B. in Grafikprozessoren zu finden sind. Hier könnten die heterogenen ISA Kerne zu erheblichen Speedups und EDP-Verringerungen führen. Man könnte womöglich den benötigten relativen Transformationsaufwand, um von einer auf eine andere ISA zu migrieren, deutlich verringern, in dem man viele Prozesse parallel, also nach dem SIMD Prinzip, von einem ISA Cluster auf ein anderes ISA Cluster migriert.

Andererseits könnten viele, unterschiedliche ISAs den für die Migration benötigten Overhead deutlich steigern und verkomplizieren. Eine Idee wäre z.B. auf eine rekonfigurierbare und modulare ISA zurückzugreifen. RISC-V eignet sich als offene, modulare und rapide wachsende ISA hierfür wahrscheinlich sehr gut. So könnte man heterogene Kerne, basierend auf einer gemeinsamen Basis-ISA, mit vielen, verschiedenen und womöglich hochspezialisierten Ausführungen Implementieren. Die Migration würde sich deutlich vereinfachen, man könnte deutlich gezielter heterogene Kerne generieren und im Idealfall allgemein eine bessere Performance erzielen. Gleichzeitig wäre das Problem der binären Kompatibilität und der komplexen Toolchains deutlich einfacher zu lösen.

HERO wird als reine Forschungsplattform präsentiert. Es gibt allerdings vergleichbare kommerzielle Produkte wie z.B. das Catapult Projekt von Microsoft [13]. Dabei handelt es sich um eine FPGA Plattform die als Hardwarebeschleuniger für Server in Datenzentren konzipiert ist. Bei Catapult werden entsprechend konfigurierte FPGAs über PCI-Express an einen beliebigen Host-Server angeschlossen und dienen diesem als schnelle Netzwerkschnittstelle oder als rekonfigurierbare Hardwarebeschleuniger. Microsoft nutzt Catapult um diverse Machine-Learning (vor allem Convolutional Neural Networks) und Search Anwedungen zu beschleunigen. Da die FPGAs über HDLs Programmiert werden, sind sie nicht sonderlich skalierbar. Außerdem ist Catapult, und die damit verbundene Architektur, bis auf einzelne Teile, nicht öffentlich. HERO könnte eine interessante Open-Source Alternative bilden. Der Schritt zu einem fertigen Produkt ist, bedingt durch die bereits vorhandenen Toolchains, nicht besonders groß. HERO wäre, u.A. dank der Modularen RISC-V Architektur, nicht nur auf Server beschränkt, sondern könnte auch in Eingebetteten Systemen zum Einsatz kommen.

VI. Schlusswort

Wir haben drei grundlegend unterschiedliche Ansätze heterogene Architekturen zu realisieren vorgestellt und miteinander Verglichen. Die vorgestellten Arbeiten unterscheiden sich sowohl auf Software- als auch Hardwareebene und bieten unterschiedliche Vor- und Nachteile im Bezug auf Softwarekompatibilität, Konfigurierbarkeit und Entwicklungkomplexität. Keiner der Vorgestellten Ansätze kann, bei Berücksichtigung aller relevanten Aspekte, den anderen vorgezogen werden. Alle haben erheblich Leistungs- und Energieeffizienzsteigerungen im Vergleich zu ihren homogenen Counterparts aufweisen können. So muss je nach Anwendung individuell entschieden werden, welcher der drei Ansätze zu bevorzugen ist.

Literatur

[1] 1149.1-2013 - ieee standard for test access port and boundary-scan architecture.

[2] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt. The m5 simulator: Modeling networked systems. *IEEE Micro*, 26(4):52–60, July 2006.

[3] Leonardo Dagum and Ramesh Menon. Openmp: An industry-standard api for shared-memory programming. *IEEE Comput. Sci. Eng.*, 5(1):46–55, January 1998.

[4] John L. Hennessy and David A. Patterson. *Computer Architecture, Sixth Edition: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 6th edition, 2017.

[5] Brian Jeff. Big.little system architecture from arm: saving power through heterogeneous multiprocessing and task context migration. In Patrick Groeneveld, Donatella Sciuto, and Soha Hassoun, editors, *DAC*, pages 1143–1146. ACM, 2012.

[6] J. A. Kahle, M. N. Day, H. P. Hofstee, C. R. Johns, T. R. Maeurer, and D. Shippy. Introduction to the cell multiprocessor. *IBM J. Res. Dev.*, 49(4/5):589–604, July 2005.

[7] C. Kozyrakis and D. Patterson. Vector vs. superscalar and vliw architectures for embedded multimedia benchmarks. In *35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002. (MICRO-35). Proceedings.*, pages 283–293, Nov 2002.

[8] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen. Single-isa heterogeneous multi-core architectures: the potential for processor power reduction. In *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, pages 81–92, Dec 2003.

[9] R. Kumar, D. M. Tullsen, P. Ranganathan, N. P. Jouppi, and K. I. Farkas. Single-isa heterogeneous multi-core architectures for multithreaded workload performance. In *Proceedings. 31st Annual International Symposium on Computer Architecture, 2004.*, pages 64–75, June 2004.

[10] Andreas Kurth, Pirmin Vogel, Alessandro Capotondi, Andrea Marongiu, and Luca Benini. Hero: Heterogeneous embedded research platform for exploring risc-v manycore accelerators on fpga. In *Proceedings of Computer Architecture Research with RISC-V Workshop (CARRV' 17)*, 2017-10. First Workshop on Computer Architecture Research with RISC-V (CARRV 2017); Conference Location: Boston, MA, USA; Conference Date: October 14, 2017.

[11] S. Li, J. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 2009)*, Los Alamitos, CA, USA, dec 2009. IEEE Computer Society.

[12] Roman Lysecky, Greg Stitt, and Frank Vahid. Warp processors. *ACM Trans. Des. Autom. Electron. Syst.*, 11(3):659–681, June 2004.

[13] Andrew Putnam, Adrian M. Caulfield, Eric S. Chung, Derek Chiou, Kypros Constantinides, John Demme, Hadi Esmaeilzadeh, Jeremy Fowers, Gopi Prashanth Gopal, Jan Gray, Michael Haselman, Scott Hauck, Stephen Heil, Amir Hormati, Joo Young Kim, Sitaram Lanka, James Larus, Eric Peterson, Simon Pope, Aaron Smith, Jason Thong, Phillip Yi Xiao, and Doug Burger. A reconfigurable fabric for accelerating large-scale datacenter services. In *41st Annual International Symposium on Computer Architecture, ISCA 2014 - Conference Proceedings*, Proceedings - International Symposium on Computer Architecture, pages 13–24. Institute of Electrical and Electronics Engineers Inc., 1 2014.

[14] D. Rossi, F. Conti, A. Marongiu, A. Pullini, I. Loi, M. Gautschi, G. Tagliavini, A. Capotondi, P. Flatresse, and L. Benini. Pulp: A parallel ultra low power platform for next generation iot applications. In *2015 IEEE Hot Chips 27 Symposium (HCS)*, pages 1–39, Aug 2015.

[15] Jeckson Dellagostin Souza, Luigi Carro, Mateus Beck Rutzig, and Antonio Carlos Schneider Beck. A reconfigurable heterogeneous multicore with a homogeneous isa. In *Proceedings of the 2016 Conference on Design, Automation & Test in Europe*, DATE '16, pages 1598–1603, San Jose, CA, USA, 2016. EDA Consortium.

[16] Dean M. Tullsen, Susan J. Eggers, and Henry M. Levy. Simultaneous multithreading: Maximizing on-chip parallelism. In *Proceedings of the 22Nd Annual International Symposium on Computer Architecture*, ISCA '95, pages 392–403, New York, NY, USA, 1995. ACM.

[17] Ashish Venkat and Dean M. Tullsen. Harnessing isa diversity: Design of a heterogeneous-isa chip multiprocessor. In *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ISCA '14, pages 121–132, Piscataway, NJ, USA, 2014. IEEE Press.

[18] Andrew Waterman, Yunsup Lee, David A. Patterson, Krste Asanovic, Volume I User level Isa, Andrew Waterman, Yunsup Lee, and David Patterson. The risc-v instruction set manual, 2014.