FPGAs in DBMS

1st Felix Grenzing
Universität Hamburg
Hamburg, Deutschland
felix.grenzing@studium.uni-hamburg.de

***Zusammenfassung*—Abstract here.**
***Index Terms*—FPGAs, DBMS, Hardware Acceleration**

I. Einführung

II. Begriffe

In [1] und [2] werden verschidene Begriffe verwendet, die Erklärung bedürfen.

A. FPGAs

B. Einsatzmöglichkeiten

FPGAs können in Datenbanksystemen auf verschiedene Weisen eingesetzt werden. Klassischer Weise gibt es On-the-Side Beschleuniger, wobei die CPU die Daten besitzt und der FPGA über eine Schnittstelle wie PCIe angebunden ist. Die CPU muss in diesem Szenario die Daten an den FPGA senden und die Ergebnisse wieder empfangen. Diese Architektur wurde häufig auch mit GPUs umgesetzt.

Eine andere Möglichkeit ist die In Datapath Beschleunigung, bei der der FPGA direkt in den Datenpfad eingebunden ist. Der FPGA kann die Daten in Echtzeit verarbeiten und die Ergebnisse mit gleicher Geschwindigkeit an die CPU zurückgeben. Häufig ist das Ziel die Datenmenge zu reduzieren, da nahe der Datenquelle häufig höhere Bandbreiten zur Verfügung stehen. Die Architektur ist somit mit Smart-Storage Lösungen verwandt.

Eine dritte Möglichkeit ist die Koprozessor Architektur, bei der der FPGA als Koprozessor der CPU fungiert. Der FPGA ist auf demselben Chip wie die CPU und hat häufig auch direkten Speicherzugriff. Flaschenhälse durch langsame Schnittstellen können so vermieden werden, was Vorteile gegenüber On-the-Side Beschleunigern bietet.

C. Pipelining und Datenparallelität

Zwei wichtige Arten der Parallelität sind die Pipelineparallelität und die Datenparallelität. Pipelineparalleität beschreibt die Aufteilung der Befehlsabarbeitung in mehrere Stufen, die parallel arbeiten. Jede Stufe bearbeitet dabei Teil des Befehls und gibt die Ergebnisse an die nächste Stufe weiter. Ein Beispiel für eine Pipeline ist in Abbildung **??** zu sehen. Die Befehle werden in vier Stufen aufgeteilt. Bearbeitet die Fetch Stufe den ersten Befehl. Sobald dies geschehen ist, wird der Befehl an die nächste Stufe weitergegeben. Die Fetch Stufe kann nun den nächsten Befehl bearbeiten. Die Befehle werden so stufenweise durch die Pipeline geschoben.

Datenparallelität beschreibt die Aufteilung der Daten in mehrere Teile, die mit mehreren Bearbeitungseinheiten parallel

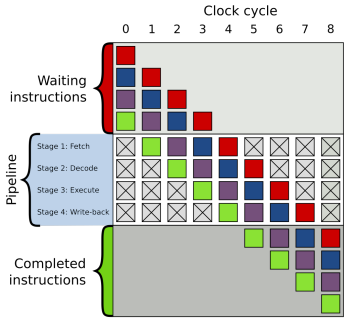

Abbildung 1: Prozessor Pipeline [3]

verarbeitet werden. Beide Ansätze können die Performance von Algorithmen verbessern, und können auch kombiniert werden um die Vorteile beider Ansätze zu nutzen. Um beide Ansätze zu kombinieren, könnten mehrere Pipelines entworfen werden, die jeweils verschidene Datenbearbeiten.

D. Partial Reconfiguration

Partial Reconfiguration (PR) ist ein Konzept, welches aktuelle FPGAs unterstützen. PR bietet die Möglichkeit, Teile des FPGAs zur Laufzeit zu verändern, ohne das gesamte FPGA neu zu konfigurieren oder das FPGA offline nehmen zu müssen. Dies ermöglicht es, verschiedene Funktionen auf einem FPGA zu implementieren und bei Bedarf zu wechseln. Umkonfiguration während der Laufzeit bietet offensichtlich Pozenzial für Beschleunigung, da mehr Algorithmen implementiert werden können, als gleichzeitig auf dem FPGA Platz haben. Eine Problematik ist jedoch, dass die Umkonfiguration Zeit in Größenordnungen von Millisekunden benötigt.

Das Projekt DoppioDB [1] nutzt PR, um verschiedene Algorithmen auf einem FPGA zu implementieren und bei Bedarf zu wechseln.

E. BitWeaving

TODO: Faktcheck BitWeaving ist ein Algorithmus, der in [1] vorgestellt wird. Der Algorithmus stellt eine Methode dar, Spaltenscanoperationen durchzuführen, indem die Daten mehrerer Zeilen als Bitvektor codiert in einem Prozessorwort gespeichert und verarbeitet werden. Zwei Varianten des Algorithmus werden vorgestellt, BitWeaving H und BitWeaving V, die sich in der Art und Weise unterscheiden, wie die Daten in den Prozessorworten gespeichert werden. BitWeaving H speichert die Daten zeilenweise, BitWeaving V spaltenweise.

III. Einordnung der Situation

Die Rolle von FPGAs in Datenbanksystemen ist laut [4] aktuell einem Wandel unterzogen. Während bei FPGAs in der Vergangenheit Herausforderungen die Potenziale überwogen haben, bieten aktuelle Entwicklungen neue Möglichkeiten für den Einsatz von FPGAs in Datenbanksystemen.

[4] stellt diese Entwicklung als Gegensätze von Pessimismus der Vergangenheit und Optimismus der Gegenwart und Zukunft dar.

A. Pessimismus

Der Pessimismus der Vergangenheit begründete sich fundamental in drei Probelemen.

Erstens war die Anbindung der On-the-Side Beschleuniger an die CPU ein Flaschenhals, da die Kommunikation in beiden Richtungen über einen Bus erfoglen musste, welcher zu hohe Latenzen und zu geringe Bandbreiten bot. TODO: Quelle On-the-Side Beschleuniger waren die gängige Architektur, um Datenbankoperationen zu beschleunigen, welche nicht im Datenpfad durchgeführt werden können. Koprozessor Architekturen wurden von Chipherstellern nicht wirklich angeboten und waren somit nicht weit verbreitet.

Zweitens profitieren nicht alle Algorithmen von der Beschleunigung durch FPGAs. Iterative Algorithmen, die auf den Ergebnissen vorheriger Iterationen aufbauen, können nicht von der hohen Parallelität von FPGAs profitieren. TODO: Quelle CPUs sind in diesen Fällen performanter, da die hohen Taktraten die itertiven Instruktionen abarbeiten können. Auch weit verzweigte Algorithmen, mit vielen If-Else Anweisungen sind nicht für FPGAs geeignet, da jeder Pfad im FPGA implementiert werden muss, was zu hohen Ressourcenverbrauch führt. Hoher Ressourcenverbrauch wiederrum führt zu geringerer Parallelität, da weniger paralelle Pipelines implementiert werden können. Die Inkompatibilität von Algorithmen wird durch den ersten Punkt noch verstärkt, da die schwierige Kommunikation zwischen CPU und FPGA die Entwickler dazu zwingt, gesamte Algorithmen auf dem FPGA zu implementieren, anstatt nur die Teile, die von der Beschleunigung profitieren, um die Kommunikation zu minimieren.

Drittens ist die direkte Konkurrenz von CPUs ein Hinderniss. CPUs sind sehr flexible, sie können theoretisch jeden Algorithmus ausführen. Hinzu kommt, dass die Leistung der CPUs stetig steigt (Moores's Law) und die Taktraten immer höher werden. Durch den hohe Zeit- und Ressourcenaufwand, welcher die Entwicklung von FPGA Beschleunigern mit sich bringt, war es häufig nicht praktikabel, FPGAs anstelle von CPUs für Datenbankbeschleunigung zu nutzen.

B. Optimismus

Der Optimismus der Gegenwart und Zukunft basiert auf drei wesentlichen Entwicklungen.

Erstens haben sich die Architekturen verändert. Moderne FPGAs sind häufig als Koprozessoren direkt auf dem gleichen Chip wie die CPU integriert, was die Latenzen und Bandbreitenprobleme der On-the-Side Beschleuniger, durch direkten Speicherzugriff des FPGAs, löst. TODO: Quelle Diese enge Integration ermöglicht granulare Beschleunigung, bei der nur die Teile des Algorithmus, die von der Beschleunigung profitieren, auf dem FPGA implementiert werden. Die erhöhte Verfügbarkeit von Koprozessor Architekturen bestärkt nun auch die Erforschung und Entwicklung von FPGA Beschleunigern was zu einer Vielzahl von neuen Anwendungen führt (siehe [2], [5]).

Zweitens haben sich die Workloads verändert.

Drittens ermöglichen hybride Ansätze eine bessere Nutzung der Stärken von FPGAs und CPUs. Durch die Kombination beider Technologien können Algorithmen so aufgeteilt werden, dass die parallelisierbaren Teile auf dem FPGA und die sequentiellen Teile auf der CPU ausgeführt werden. TODO: Quelle

IV. Anderes Paper

Im Kontext veränderter Architekturen ordnet sich auch [2] ein. Dort wurde auf einem FPGA ein Column Store implementiert, der den BitWeaving H Algorithmus nutzt. Es wurden verschidenen Hardwareansätze verglichen, um die beste Performance zu erzielen.

A. Architektur

Die Forschung von [2] baut auf der Zynq Ultrascale+ Architektur von Xilinx auf. Die Plattform ist aus zweiteilig aus Steuersystem mit 4 ARM Cortex-A53 Kernen und 4GB DDR4 Speicher und Logikbereich mit FPGA und 500MB DDR4 Speicher aufgebaut [2].

B. Grundlegendes Vorgehen

Pipeline
Processing Elements
Combiner

C. Verschiedene Ansätze

Mehrere paralelle Pipelines -¿ Datenparallelität
Limitation des DDR4-Controlers
Optimale Nutzung von Combiner
Hybridansatz

D. Ergebnisse

V. Andere Ansätze

A. IBEX

B. REGEXP_LIKE

Ein vorgestelltes Projekt befasst sich mit der Optimierung eines Operators, welcher reguläre Ausdrücke auf Datenbanken anwendet. So können beispielsweise alle Zeilen einer Tabelle selektiert werden, die in einer Spalte einen bestimmten regulären Ausdruck erfüllen. Der Operator wurde auf einem FPGA implementiert und zeigt die Nutzung von hybriden Ansätzen, indem die relevanten Daten so weit wie möglich auf dem FPGA verarbeitet werden und nur falls nötig auf der CPU nachbearbeitet werden [5]. So können alle REGEXP_LIKE-Anfragen durch die Beschleunigung profitieren,

da nicht vollständig passende reguläre Ausdrücke nicht abgewiesen und damit vollständig auf der CPU verarbeitet werden müssen.

Der Reguläre Ausdruck wird an an einer Wildcard-Position aufgeteil, so dass zwei unabhängige reguläre Ausdrücke entstehen. Der FPGA gibt dann, zusammen mit einem Index alle Zeilen zurück, die den ersten regulären Audruck erfüllen. Die CPU überprüft dann, ob die Zeilen vom Index aus auch den zweiten regulären Ausdruck erfüllen. Der verwendete Ausdruck in [5] ist in Listing 1 zu sehen. Der FPGA bearbeitet den Ausdruck bis zur zweiten Wildcard und die CPU muss dann nur noch nach 'delivery' suchen.

```
SELECT count(*)
FROM address_tables
WHERE REGEXP_LIKE(address_string,
'(Strasse|Str\.).*(8[0-9]{4}).*delivery');
```

Listing 1: Regulärer Ausdruck aus [5]

VI. The Road that lies ahead

Literatur

[1] Y. Li and J. M. Patel, "BitWeaving: fast scans for main memory data processing," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '13. New York, NY, USA: Association for Computing Machinery, Jun. 2013, pp. 289–300. [Online]. Available: https://doi.org/10.1145/2463676.2465322

[2] N. J. Lisa, A. Ungethüm, D. Habich, W. Lehner, T. D. Nguyen, and A. Kumar, "Column Scan Acceleration in Hybrid CPU-FPGA Systems." in *ADMS@ VLDB*, 2018, pp. 22–33.

[3] Cburnett, "Generic 4-stage pipeline," https://en.wikipedia.org/wiki/Instruction_pipelining#/media/File:Pipeline,_4_stage.svg, zugriff: 24-Jan-2025.

[4] Z. István, "The Glass Half Full: Using Programmable Hardware Accelerators in Analytics." *IEEE Data Eng. Bull.*, vol. 42, no. 1, pp. 49–60, 2019.

[5] D. Sidler, Z. István, M. Owaida, and G. Alonso, "Accelerating Pattern Matching Queries in Hybrid CPU-FPGA Architectures," in *Proceedings of the 2017 ACM International Conference on Management of Data*, ser. SIGMOD '17. New York, NY, USA: Association for Computing Machinery, May 2017, pp. 403–415. [Online]. Available: https://doi.org/10.1145/3035918.3035954