

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

felix.grenzing@studium.uni-hamburg.de

**Index Terms**—FPGAs, DBMS, Hardware Acceleration

## II. BEGRIFFE

### B. Einsatzmöglichkeiten

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.

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

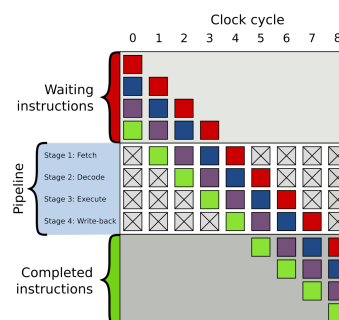


Abbildung 1: Prozessor Pipeline [3]

#### D. Partial Reconfiguration

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 Problemen.

Erstens war die Anbindung der On-the-Side Beschleuniger an die CPU ein Flaschenhals, da die Kommunikation in beiden Richtungen über einen Bus erfolgen 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 iterativen 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 wiederum führt zu geringerer Parallelität, da weniger parallele 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 Hindernis. CPUs sind sehr flexibel, sie können theoretisch jeden Algorithmus ausführen. Hinzu kommt, dass die Leistung der CPUs stetig steigt (Moore's Law) und die Taktraten immer höher werden. Durch den hohen 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 für die Gegenwart und Zukunft, den der Autor motiviert, 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. Bisher wurden hauptsächlich klassische SQL-Operatoren in Datenbank durchgeführt, welche auf der verfügbaren Hardware häufig Memory-bound waren, also durch die Geschwindigkeit der Datentransfers limitiert. Durch neue Anwendungen wie Maschinelles Lernen und Künstliche Intelligenz sind neue Workloads entstanden, die häufig Compute-bound sind, also durch die Geschwindigkeit der Berechnungen limitiert. Da diese Anwendungen viele Daten benötigen, liegt es nahe, die Anwendungen als neue Datenbank Operatoren einzuführen. Für Compute-bound Workloads werden FPGAs wieder interessant, da sie durch ihre hohe Parallelität und Rekonfigurierbarkeit die Berechnungen beschleunigen können.

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. Ein Beispiel dafür ist REGEXP\_LIKE [5]. **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 verschiedenen 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]. **TODO: Mehr**

#### B. Grundlegendes Vorgehen

Pipeline  
Processing Elements  
Combiner

#### C. Verschiedene Ansätze

Mehrere parallele Pipelines -> Datenparallelität  
Limitation des DDR4-Controllers  
Optimale Nutzung von Combiner  
Hybridansatz

## D. Ergebnisse

### V. ANDERE ANSÄTZE

#### A. IBEX

IBEX ist ein FPGA-basierter Beschleuniger für Datenbanken, der in [4] vorgestellt wird. IBEX ist ein In-Data-Path Beschleuniger, der direkt in den Datenpfad eingebunden ist, also zwischen SSD und der CPU. Die Daten werden mit gleicher Bandbreite verarbeitet, wie sie von der SSD ankommen. Der FPGA führt SQL-Operatoren aus, welche die Datenmenge reduzieren, also beispielsweise Filter und Aggregationen, nicht aber Operatoren, die die Datenmenge erhöhen können, wie Joins. IBEX verwendet auch einen hybriden Ansatz, um Aggregate zu berechnen. Die Aggregatfunktionen werden auf dem FPGA mit einem Hashtable berechnet, welcher mit fester Größe im RAM verortet ist, was gleiche Laufzeiten, unabhängig von den Daten ermöglicht, da der Hashtable nicht dynamisch vergrößert werden muss, aber dadurch die Anzahl der Gruppen bei Aggregatbildung limitiert. Um dieses Limit zu umgehen, gibt der FPGA Teilaggregate weiter, sobald der Hashtable keinen Platz mehr für neue Gruppen hat. Die CPU vereinigt dann die Teilaggregate zu einem Endergebnis auf eine Weise, das die Bearbeitung von der Arbeit des FPGAs in jedem Fall profitiert. Der FPGA kann so die Datenmenge reduzieren und die CPU kann die reduzierte Datenmenge effizienter verarbeiten.

#### B. Caribou

#### C. DoppioDB

Das Projekt DoppioDB zeigt einen solchen Ansatz, indem die Datenbank die FPGA-Steuerung übernimmt und Hardware-Threads nutzt, die wie Software-Threads angesprochen werden. Dabei wurde über SQL hinausgehend Funktionalität für maschinelles Lernen integriert, wie z. B. stochastische Gradientenabstiege und Entscheidungsbaum-Inferenz. **TODO: FIXME**

#### D. REGEXP\_LIKE

Ein vorgestelltes Projekt befasst sich mit der Optimierung des REGEXP\_LIKE 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 aufgeteilt, so dass zwei unabhängige reguläre Ausdrücke entstehen. Der FPGA gibt dann, zusammen mit einem Index alle Zeilen zurück, die den ersten regulären Ausdruck 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

Ein zentraler Aspekt der zukünftigen Datenbankentwicklung ist die Integration programmierbarer Hardware wie FPGAs, die zwar neu konfigurierbar, jedoch nicht so flexibel wie Software sind. Die Kernfrage hierbei ist, wer die Kontrolle über die Beschleunigungsfunktionen übernimmt – das Betriebssystem bzw. der Hypervisor oder die Datenbank selbst.

#### A. Betriebssystem- oder Datenbankverwaltung?

Wenn das Betriebssystem die Kontrolle behält, muss die Datenbank Strategien entwickeln, um von hardwarebasierten Beschleunigungsmöglichkeiten zu profitieren, die vom Infrastruktur- oder Cloudanbieter bereitgestellt werden. Hierbei könnten bereits etablierte Techniken zur Code-Optimierung für spezifische CPU-Features, wie SIMD-Einheiten, als Grundlage dienen.

Übernimmt hingegen die Datenbank die Kontrolle, eröffnen sich größere Gestaltungsspielräume. Die Datenbank könnte benutzerdefinierte Hardware-Beschleuniger entwickeln, die exakt auf ihre Bedürfnisse abgestimmt sind, und diese sogar zur Laufzeit synthetisieren (Siehe DoppioDB)

### 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](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>