Optimizing data management on new hardware

Data processing on FPGAs

Motivation

Today's Requirements on data management

- Large data size and variety
- Realtime Analytics
- Utilizing Technological Advances

FPGAs

Integrated Circuits reprogrammable to fit applications needs

- Configurable Logic gates
- Input/Output circuitry
- Programmed using Hardware Definition Languages (HDL)

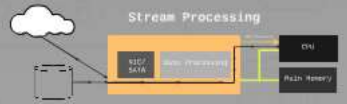
Architecture
High throughput rates
Low latency
High degree of parallelism

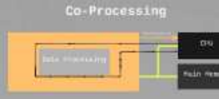
Network Stream Processing

In software: Non-deterministic FA ... considered inefficient

In FPGAs:

Architectural Integration





Data Stream Processing

Glacier

- Component library
- Compiler

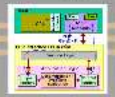
Compiles CQL expressions in VHDL expression in 3 Steps

1) Query in Algebraic Plan

2) Algebraic Plan in VHDL expressions

3) Optimization Heuristics

Co-Processor Architecture



Predicate Evaluation and Row decompression

Performance

Hash Joins

Performance

Sort-Merge Join

Conclusion

FPGAs offer flexible hardware with a high degree of parallelism, low latency and high throughput rates

Suitable for:

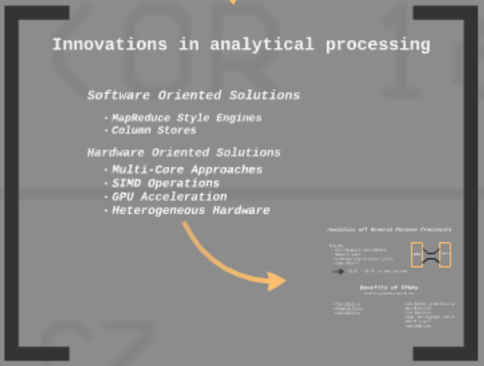
- Network Stream Processing
- Stream Processing
- Co-Processing

Questions?

Motivation

Today's Requirements on data management

- Large data size and variety
- Realtime Analytics
- Utilizing Technological Advances



Innovations in analytical processing

Software Oriented Solutions

- *MapReduce Style Engines*
- *Column Stores*

Hardware Oriented Solutions

- *Multi-Core Approaches*
- *SIMD Operations*
- *GPU Acceleration*
- *Heterogeneous Hardware*

[illegible]

Benefits of FPGAs

[illegible]

Innovations in analytical processing

Software Oriented Solutions

- *MapReduce Style Engines*
- *Column Stores*

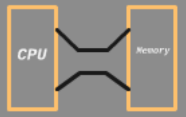
Hardware Oriented Solutions

- *Multi-Core Approaches*
- *SIMD Operations*
- *GPU Acceleration*
- *Heterogeneous Hardware*

Analytics off General Purpose Processors

Avoids:
- Von-Neumann bottleneck
- Memory wall
- Endangering Service Level Agreements

→ OLAT + OLTP in one system



Benefits of FPGAs

Field Programmable Gate Arrays

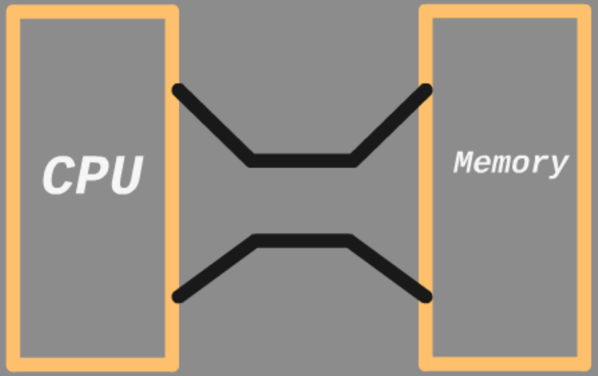
- Flexibility
- Adaptability
- Scalability
- Low-level granularity parallelism
- Low latency
- High throughput rates
- Small power consumption

Analytics off General Purpose Processors

Avoids:

- Von-Neumann bottleneck
- Memory wall
- Endangering Service Level Agreements

→ OLAT + OLTP in one system



Benefits of FPGAs

Field **P**rogrammable **G**ate **A**rrays

- Memory wall
- Endangering Service Level Agreements

➜ OLAT + OLTP in one system

Benefits of FPGAs

Field **P**rogrammable **G**ate **A**rrays

- Flexibility
- Adaptability
- Scalability

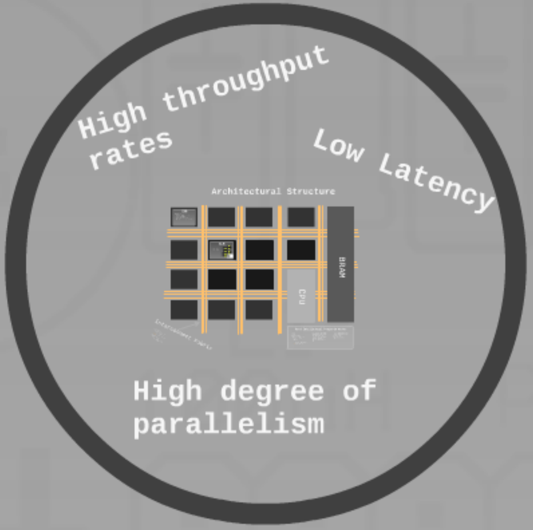
- Low-level granularity parallelism
- Low latency
- High throughput rates
- Small power consumption

FPGAs

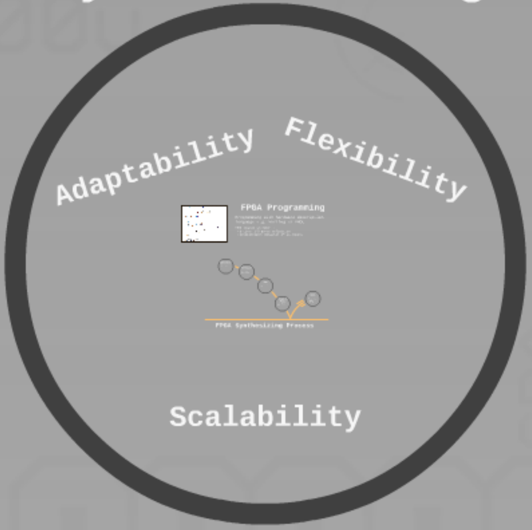
Integrated Circuits reprogrammable to fit applications needs

- Configurable Logic gates
- Input/Output circuitry
- Programmed using Hardware Definition Languages (HDL)

Architecture

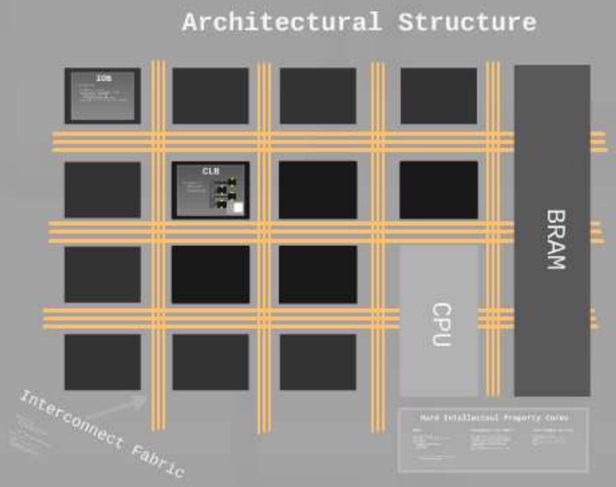


Synthesizing



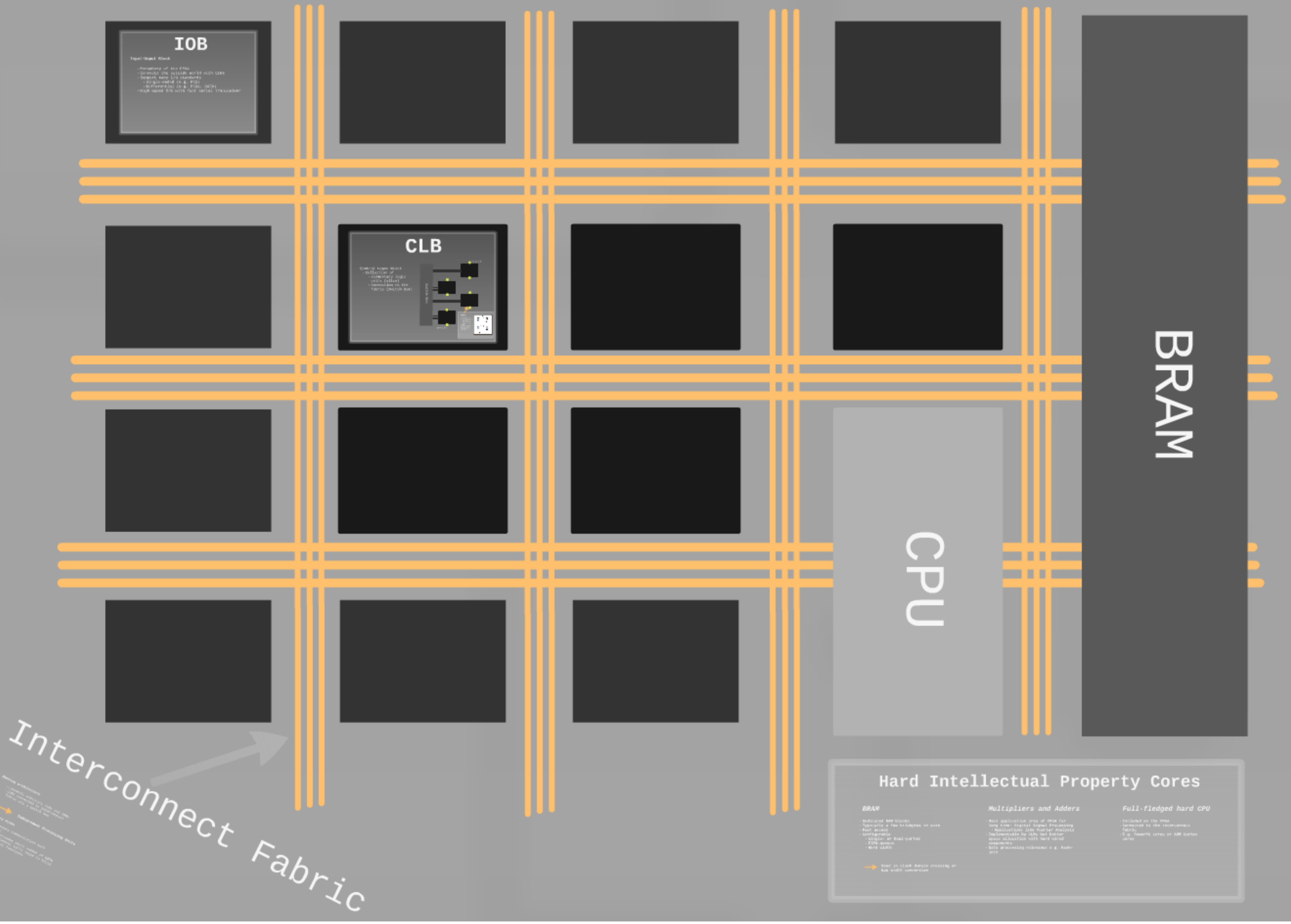
High throughput rates

Low Latency



High degree of parallelism

Architectural Structure


IOB
Input/Output Block
- Periphery of the FPGA
- Connects the outside world with CLBs
- Support many I/O standards
- Single-ended (e.g. PCI)
- Differential (e.g. PCIe, SATA)
- High-speed I/O with fast serial transceiver
CLB
Control Logic Block
- Collection of
- elementary logic units (slices)
- Connections to the fabric (switch box)
Switch Box
CPU
BRAM
Interconnect Fabric
Routing architecture
- [illegible]
- [illegible]
Fast carry paths
Independent Processing Units
- [illegible]

Hard Intellectual Property Cores

BRAM

- Dedicated RAM blocks
- Typically a few kilobytes in size
- Fast access
- Configurable
  - Single- or Dual-ported
  - FIFO queues
  - Word width

→ Used in clock domain crossing or bus width conversion

Multipliers and Adders

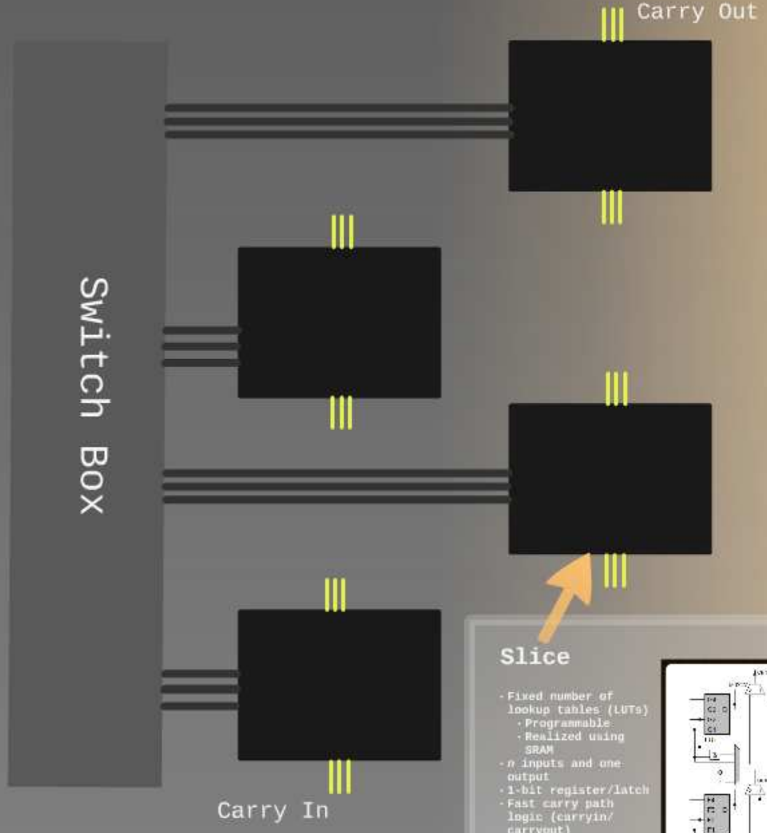
- Main application area of FPGA for long time: Digital Signal Processing
  - Applications like Fourier Analysis
- Implementable by LUTs but better space allocation with hard wired components
- Data processing relevance e.g. hash-join

Full-fledged hard CPU

- Included on the FPGA
- Connected to the interconnect fabric
- E.g. PowerPC cores or ARM Cortex cores

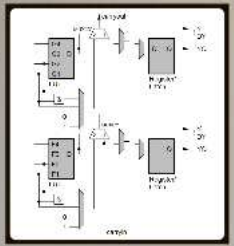
CLB

Control Logic Block
- Collection of
  - elementary logic units (slice)
  - Connection to the fabric (switch box)



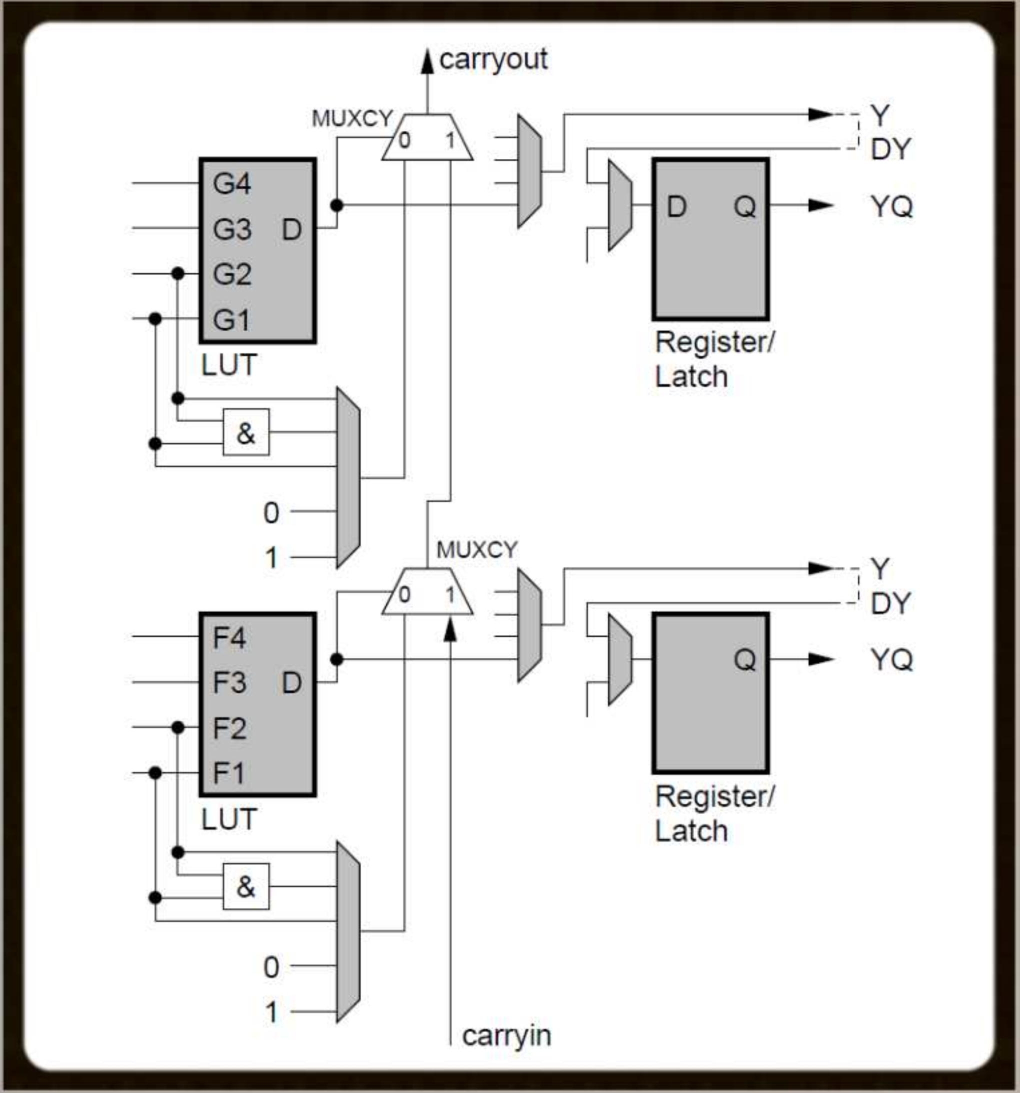
Slice

- Fixed number of lookup tables (LUTs)
  - Programmable
  - Realized using SRAM
- *n* inputs and one output
- 1-bit register/latch
- Fast carry path logic (carryin/carryout)

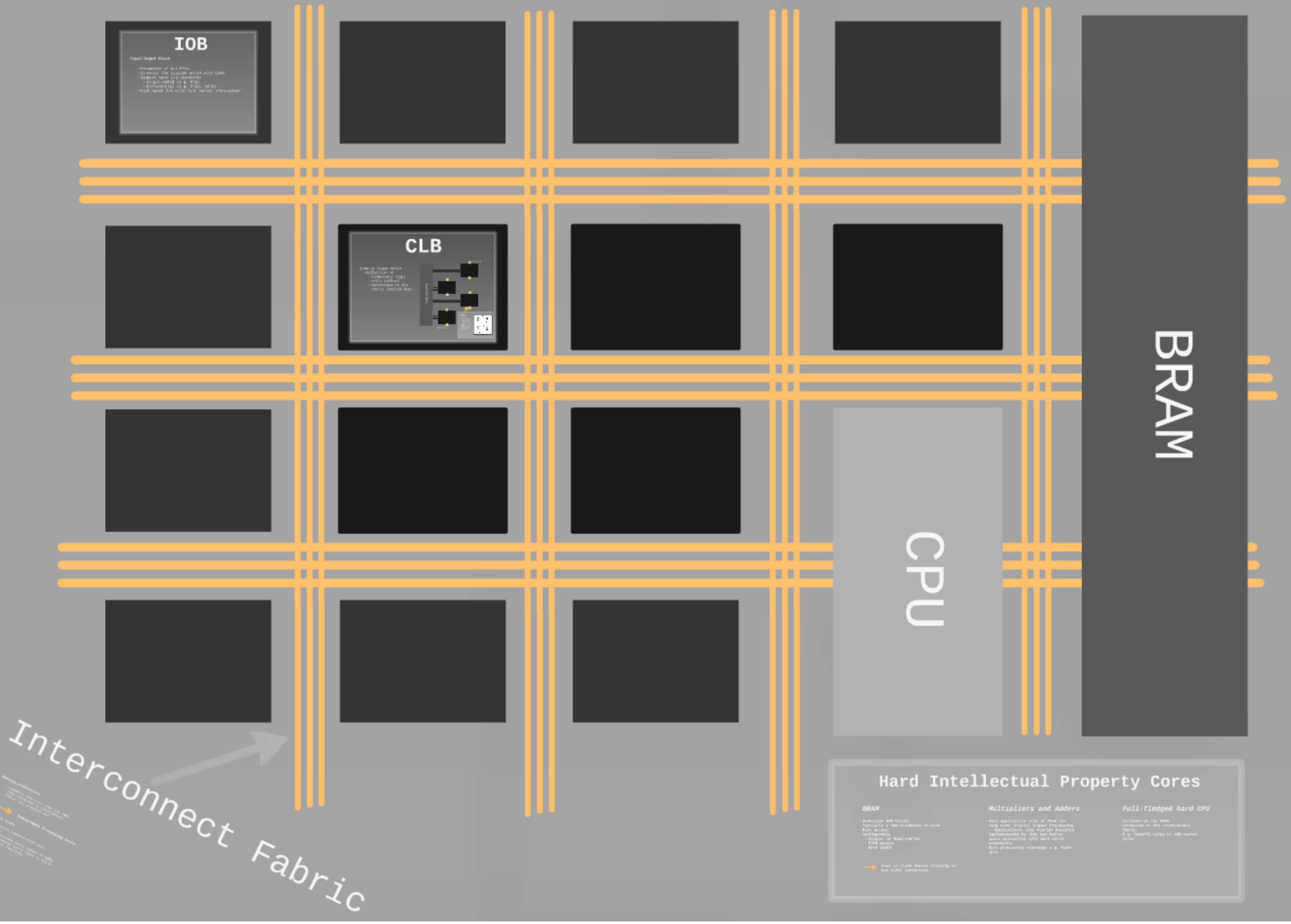


Slice

- Fixed number of lookup tables (LUTs)
  - Programmable
  - Realized using SRAM
- *n* inputs and one output
- 1-bit register/latch
- Fast carry path logic (carryin/ carryout)



Architectural Structure



IOB

Input/Output Block

- Periphery of the FPGA
- Connects the outside world with CLBs
- Support many I/O standards
  - Single-ended (e.g. PCI)
  - Differential (e.g. PCIe, SATA)
- High-speed I/O with fast serial transceiver

CLB

Control Logic Block
- Collection of
  - elementary logic units (slice)
  - Connections to the fabric (Switch Box)

Routing architecture
- [illegible]
- CLBs connected to [illegible]
- [illegible]

Fast Carry Paths

[illegible] Processing Units

- [illegible]
- [illegible]
- [illegible]
- [illegible]

Hard Intellectual Property Cores

BRAM

- Dedicated RAM blocks
- Typically a few kilobytes in size
- Fast access
- Configurable
  - Single- or Dual-ported
  - FIFO-queues
  - Word width

Used in clock domain crossing or bus width conversion

Multipliers and Adders

- Main application area of FPGA for long time: Digital Signal Processing
  - Applications like Fourier Analysis
- Implementable by LUTs but better space allocation with hard wired components
- Data processing relevance e.g. hash-join

Full-fledged hard CPU

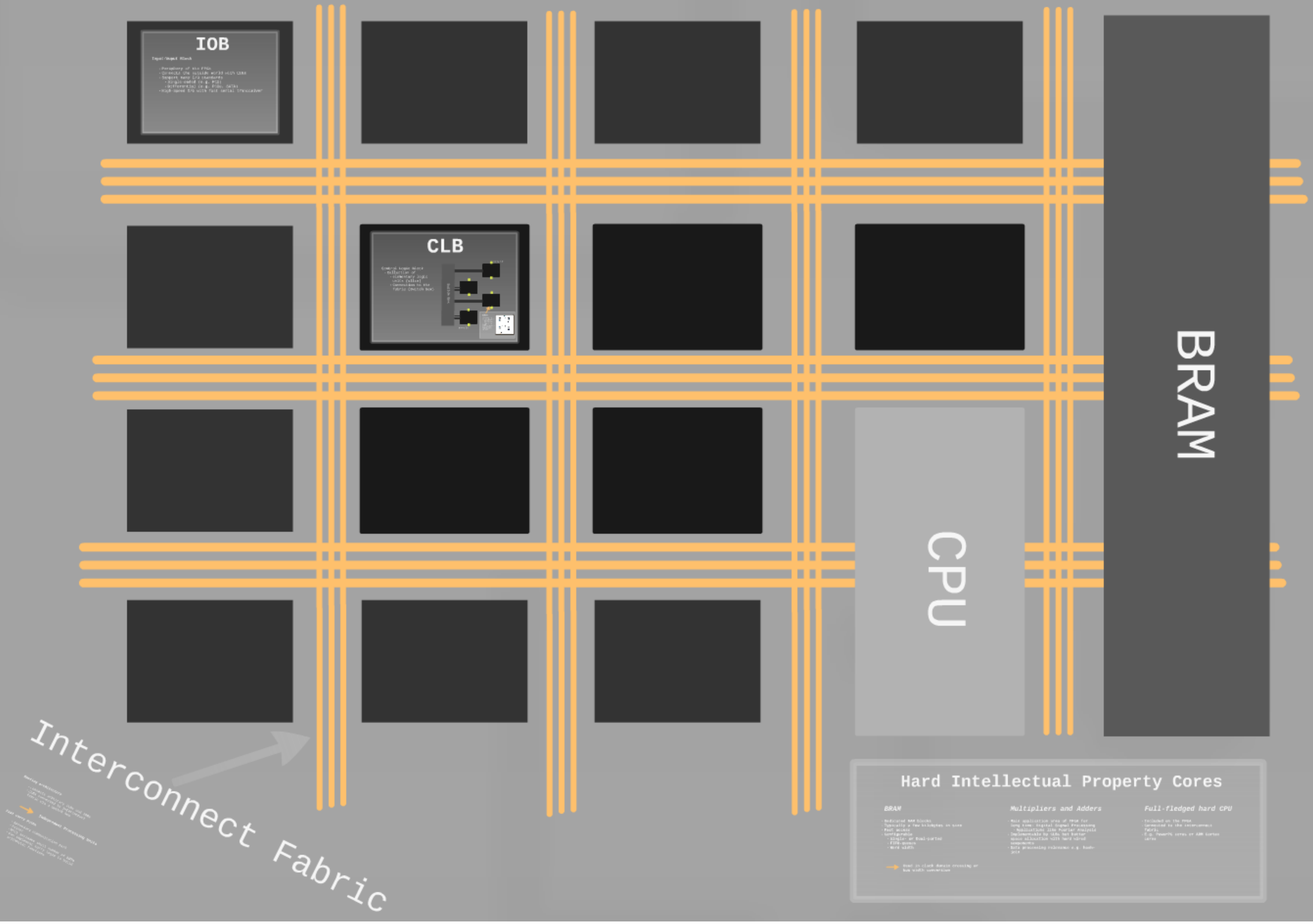
- Included on the FPGA
- Connected to the interconnect fabric
- E.g. PowerPC cores or ARM Cortex cores

IOB

Input/Ouput Block

- Periphery of the FPGA
- Connects the outside world with CLBs
- Support many I/O standards
  - Single-ended (e.g. PCI)
  - Differential (e.g. PCIe, SATA)
- High-Speed I/O with fast serial transceiver

Architectural Structure


IOB
Input/Output Block
Periphery of the FPGA
Connects the outside world with CLBs
Support many I/O standards
Single-ended (e.g. PCI)
Differential (e.g. PCIe, SATA)
High-speed I/O with fast serial transceiver
CLB
Control Logic Block
Collection of
elementary logic units
Connection to the fabric (Switch Box)
Switch Box
BRAM
CPU
Interconnect Fabric
Routing architecture
Fast Carry Paths
Independent Processing Units

Hard Intellectual Property Cores

BRAM

- Dedicated RAM blocks
- Typically a few kilobytes in size
- Fast access
- Configurable
  - Single- or Dual-ported
  - FIFO queues
  - Word width

→ Used in clock domain crossing or bus width conversion

Multipliers and Adders

- Main application area of FPGA for long time: Digital Signal Processing
  - Applications like Fourier Analysis
- Implementable by CLBs but better space allocation with hard wired components
- Data processing relevance e.g. hash-join

Full-fledged hard CPU

- Included on the FPGA
- Connected to the interconnect fabric
- E.g. PowerPC cores or ARM Cortex cores

Routing architecture

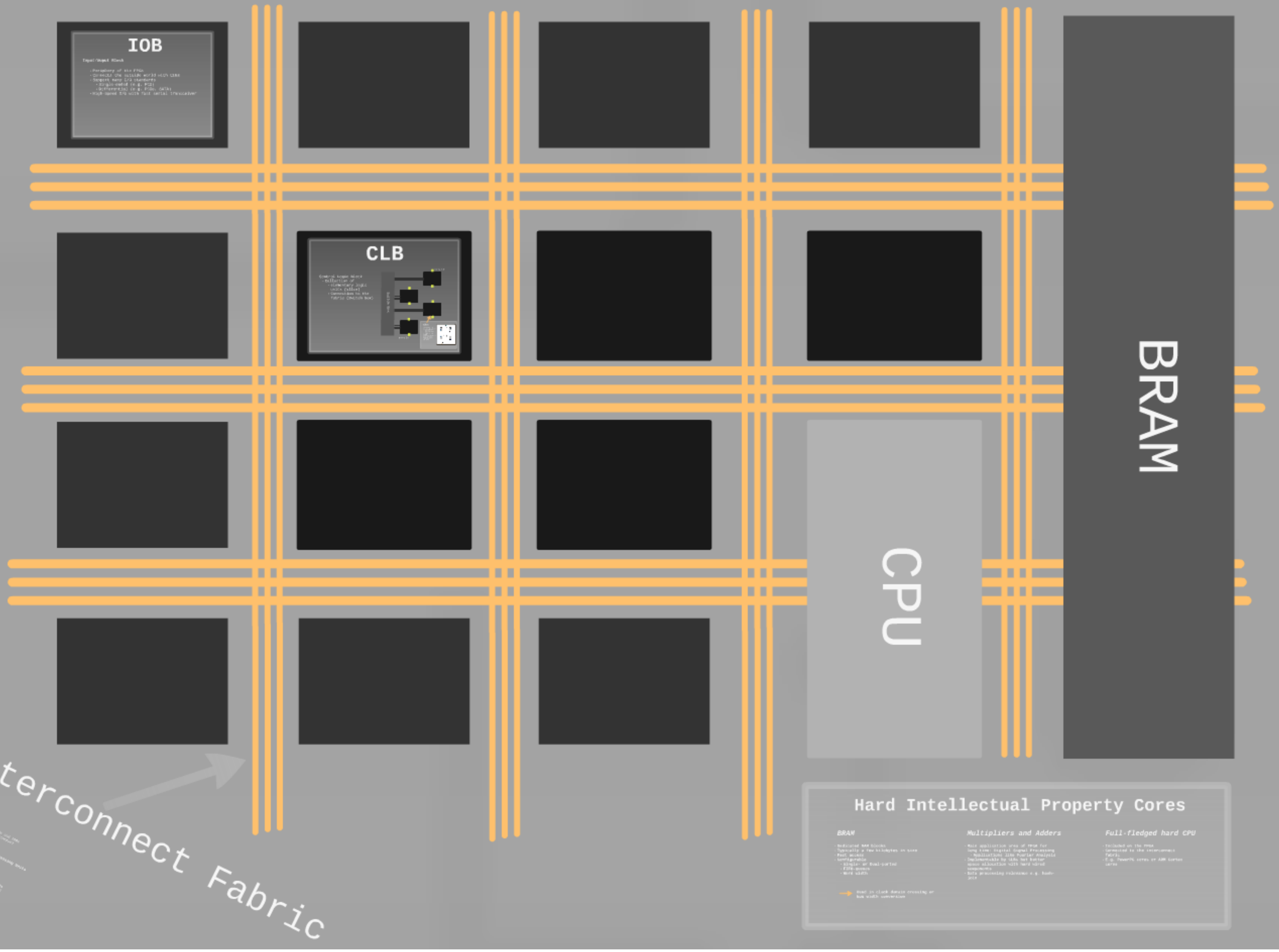
- Connects arbitrary CLBs and IOBs
- CLBs connected to interconnect fabric via a switch box

➔ **Independent Processing Units**

Fast Carry Paths

- Secondary communication path
- Faster
- Only includes small number of **LUTs**
- Can implement carry logic to build arithmetic functions

Architectural Structure


IOB
Input/Output Block
Periphery of the FPGA
Connects the outside world with CLBs
Support many I/O standards
Single-ended (e.g. PCI)
Differential (e.g. PCIe, SATA)
High-speed I/O with fast serial transceiver
CLB
Configurable Logic Block
Collection of
elementary logic units [illegible]
Connections to the fabric (Switch Box)
Switch Box
BRAM
CPU
Interconnect Fabric
Routing architecture
Connects arbitrary CLBs and IOBs
CLBs connected to interconnect fabric via a switch box
Fast carry paths
Independent Processing Units
[illegible]

Hard Intellectual Property Cores

BRAM

- Dedicated RAM blocks
- Typically a few kilobytes in size
- Fast access
- Configurable
  - Single- or Dual-ported
  - FIFO queues
  - Word width

→ Used in clock domain crossing or bus width conversion

Multipliers and Adders

- Main application area of FPGA for long time: Digital Signal Processing
  - Applications like Fourier Analysis
- Implementable by LUTs but better space allocation with hard wired components
- Data processing relevance e.g. hash-join

Full-fledged hard CPU

- Included on the FPGA
- Connected to the interconnect fabric
- E.g. PowerPC cores or ARM Cortex cores

Hard Intellectual Property Cores

BRAM

- Dedicated RAM blocks
- Typically a few kilobytes in size
- Fast access
- Configurable
  - Single- or Dual-ported
  - FIFO-queues
  - Word width

→ Used in clock domain crossing or bus width conversion

Multipliers and Adders

- Main application area of FPGA for long time: Digital Signal Processing
  - Applications like Fourier Analysis
- Implementable by CLBs but better space allocation with hard wired components
- Data processing relevance e.g. hash-join

Full-fledged hard CPU

- Included on the FPGA
- Connected to the interconnect fabric
- E.g. PowerPC cores or ARM Cortex cores

BRAM

- Dedicated RAM blocks
- Typically a few kilobytes in size
- Fast access
- Configurable
  - Single- or Dual-ported
  - FIFO-queues
  - Word width

→ Used in clock domain crossing or bus width conversion

Multipliers and Adders

- **Main application area of FPGA for long time: Digital Signal Processing**
  - **Applications like Fourier Analysis**
- **Implementable by CLBs but better space allocation with hard wired components**
- **Data processing relevance e.g. hash-join**

Full-fledged hard CPU

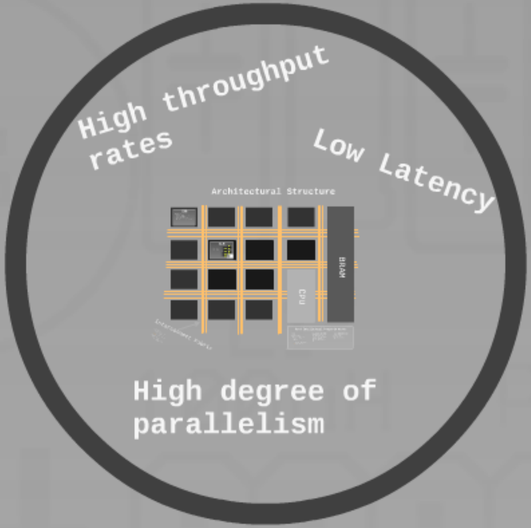
- Included on the FPGA
- Connected to the interconnect fabric
- E.g. PowerPC cores or ARM Cortex cores

FPGAs

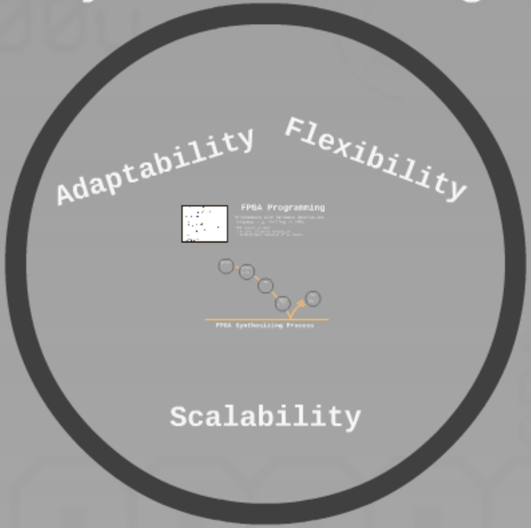
Integrated Circuits reprogrammable to fit applications needs

- Configurable Logic gates
- Input/Output circuitry
- Programmed using Hardware Definition Languages (HDL)

Architecture



Synthesizing



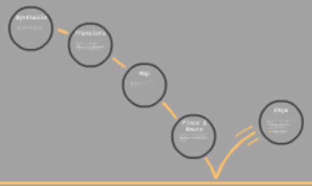
Adaptability

Flexibility

FPGA Programming

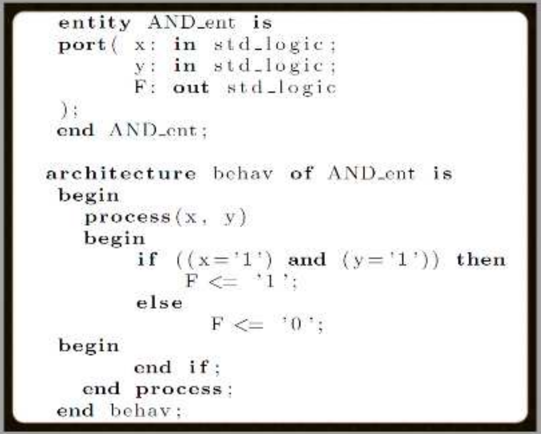
Programming with hardware description language e.g. Verilog or VHDL

VHDL design statement:
- Entity: I/O ports definition
- Architecture: Behavior of an entity



FPGA Synthesizing Process

Scalability



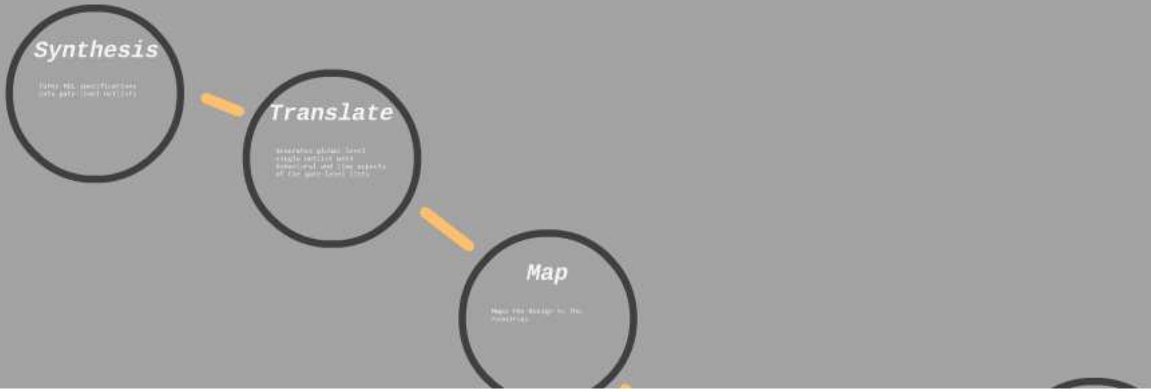
```
entity AND_ent is
port( x: in std_logic;
      y: in std_logic;
      F: out std_logic
);
end AND_ent;

architecture behav of AND_ent is
begin
  process(x, y)
  begin
      if ((x='1') and (y='1')) then
          F <= '1';
      else
            F <= '0';
begin
      end if;
  end process;
end behav;
```

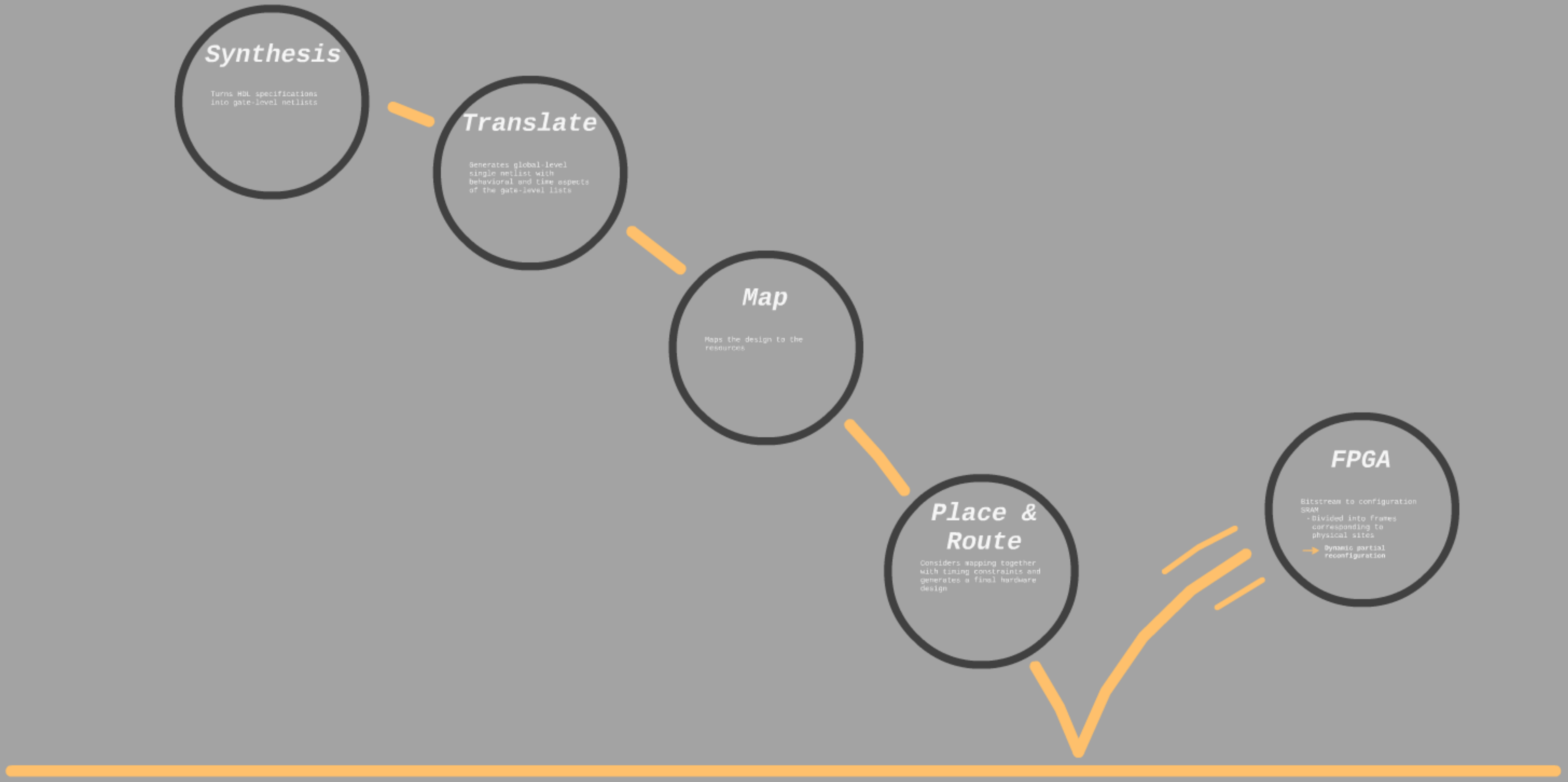
FPGA Programming

Programming with hardware description language e.g. Verilog or VHDL

VHDL design statement:

- Entity: I/O ports definition
- Architecture: Behavior of an entity




Synthesis
Turns HDL specifications into gate-level netlists
Translate
Generates global-level single netlist with behavioral and time aspects of the gate-level lists
Map
Maps the design to the resources
Place & Route
Considers mapping together with timing constraints and generates a final hardware design
FPGA
Bitstream to configuration SRAM
- Divided into frames corresponding to physical sites
→ Dynamic partial reconfiguration

FPGA Synthesizing Process

Synthesis

Turns HDL specifications
into gate-level netlists

Translate

Generates global-level single netlist with behavioral and time aspects of the gate-level lists

Map

Maps the design to the resources

Place & Route

Considers mapping together with timing constraints and generates a final hardware design

FPGA

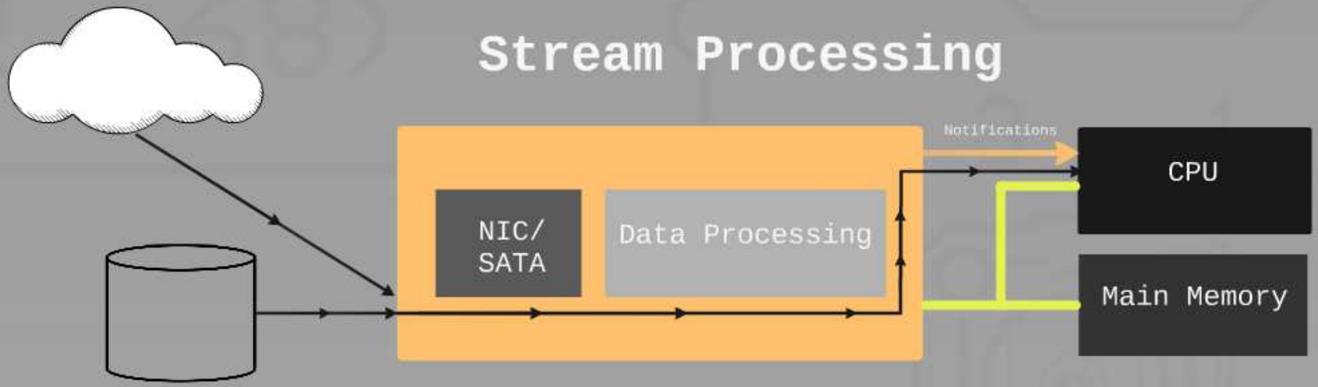
Bitstream to configuration SRAM

- Divided into frames corresponding to physical sites

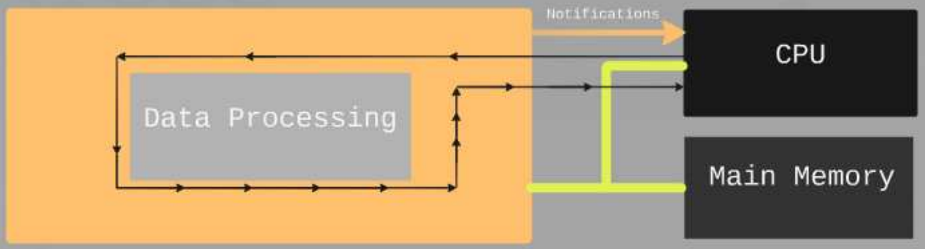
→ **Dynamic partial reconfiguration**

Architectural Integration

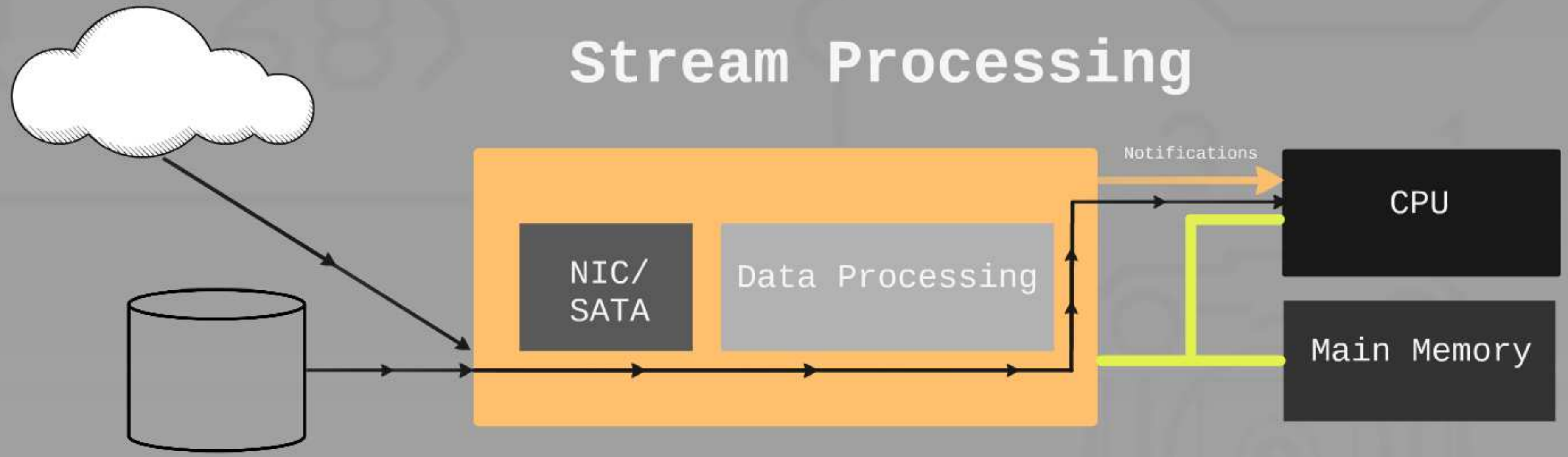
Stream Processing



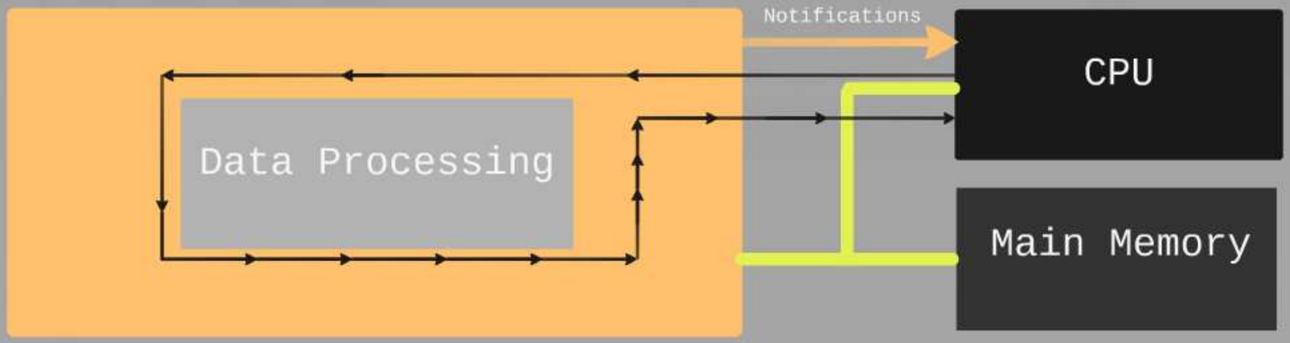
Co-Processing



Stream Processing

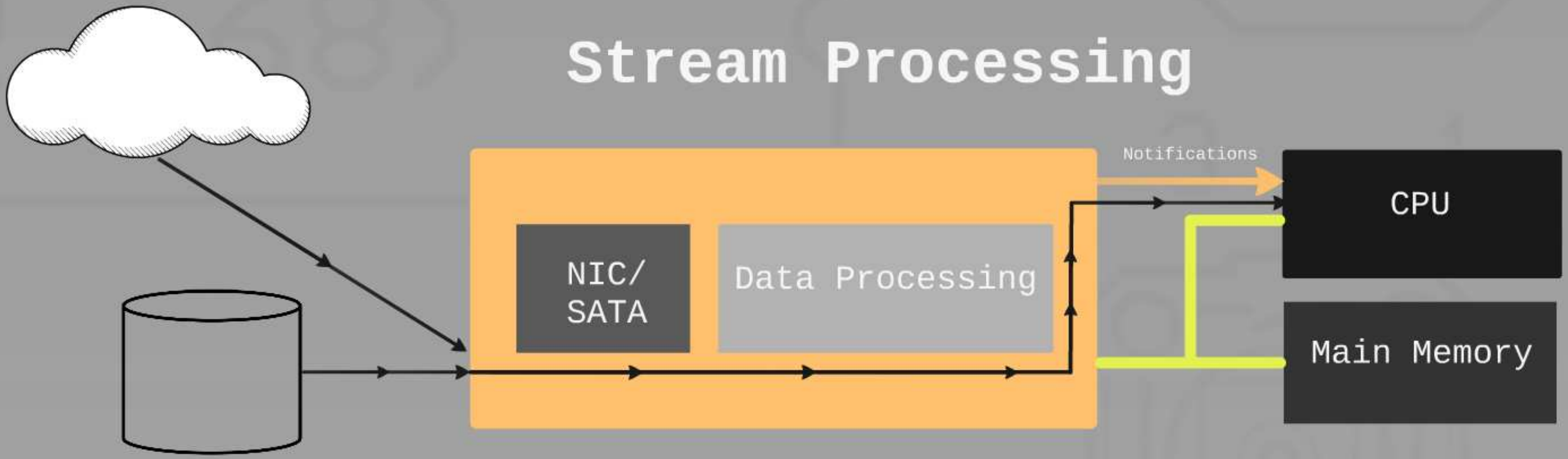


Co-Processing

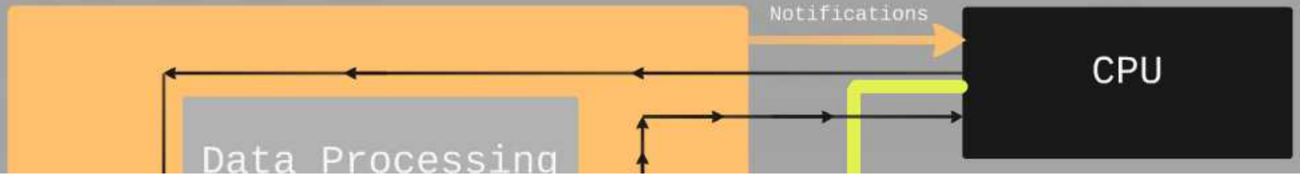


Architectural Integration

Stream Processing

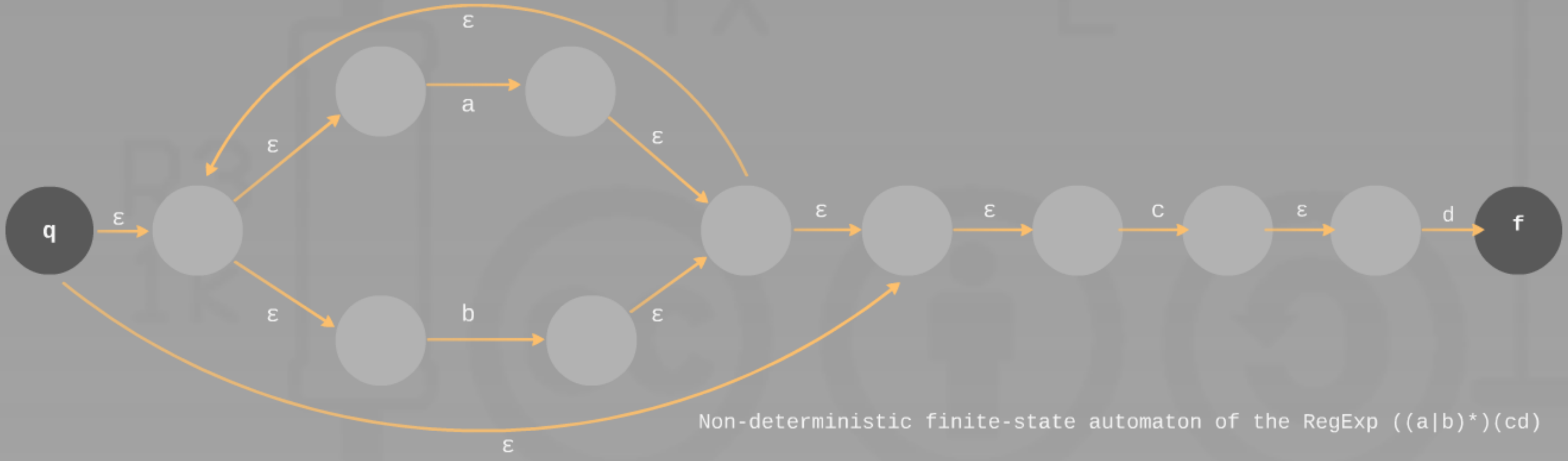


Co-Processing



Network Stream Processing

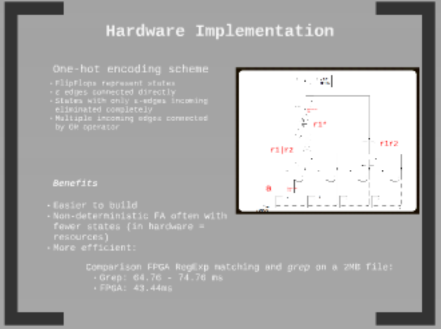
Application Scenario: Network Intrusion Detection
- Suspicious patterns detectable via Regular Expression Matching
  - Compare a sequence of characters and meta characters to all strings in the network stream



Non-deterministic finite-state automaton of the RegExp ((a|b)*)(cd)

In software: Non-deterministic FS automaton considered inefficient
- Every candidate state and transition considered iteratively

On FPGA:
- States & transitions considered in parallel



Hardware Implementation

One-hot encoding scheme
- FlipFlops represent states
- ε edges connected directly
- States with only ε-edges incoming eliminated completely
- Multiple incoming edges connected by OR operator

Benefits

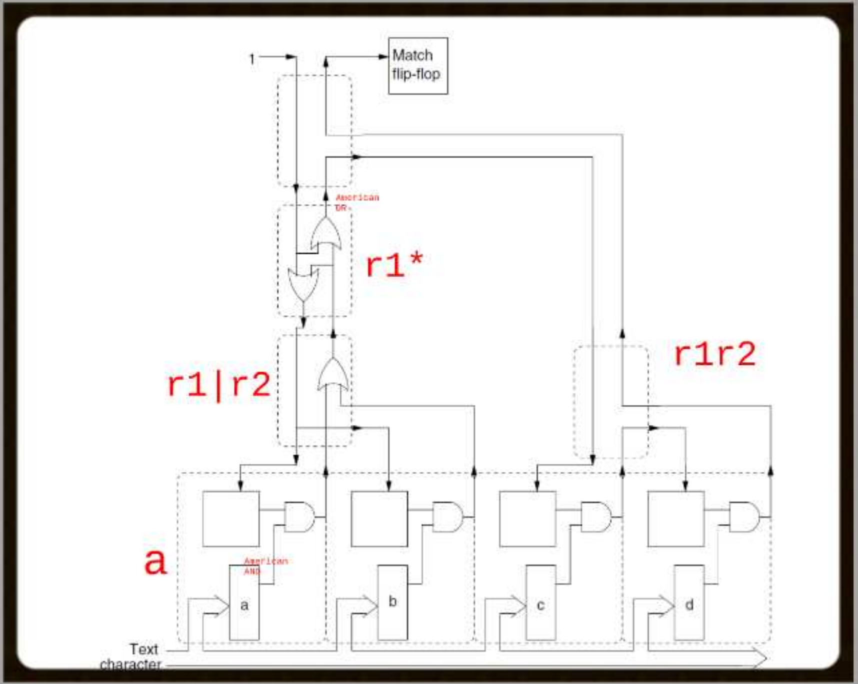
- Easier to build
- Non-deterministic FA often with fewer states (in hardware = resources)
- More efficient:

Comparison FPGA RegExp matching and grep on a 2MB file:
- Grep: 64.76 - 74.76 ms
- FPGA: 43.44ms

Hardware Implementation

One-hot encoding scheme

- FlipFlops represent states
- ε edges connected directly
- States with only ε-edges incoming eliminated completely
- Multiple incoming edges connected by OR operator



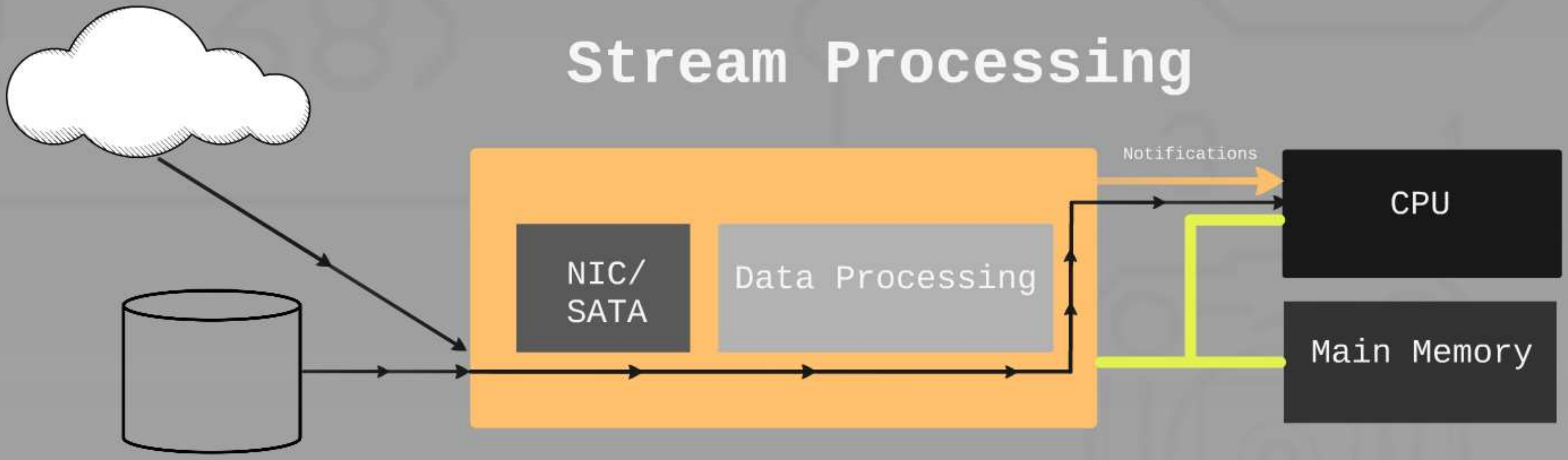
Benefits

- Easier to build
- Non-deterministic FA often with fewer states (in hardware = resources)
- More efficient:

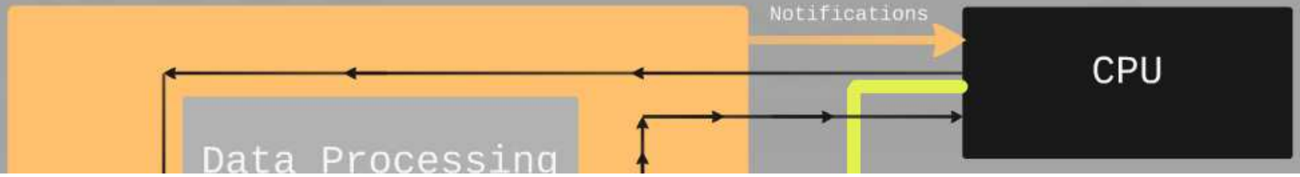
Comparison FPGA RegExp matching and *grep* on a 2MB file:
- Grep: 64.76 - 74.76 ms
- FPGA: 43.44ms

Architectural Integration

Stream Processing



Co-Processing



Data Stream Processing

Glacier

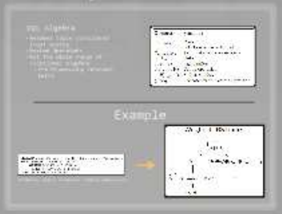
- Component library
- Compiler

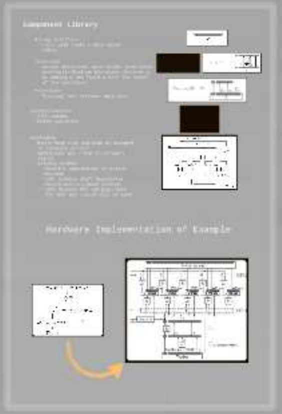
Compiles CQL expressions to VHDL expression in 3 Steps

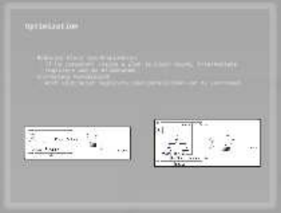
1) Query to Algebraic Plan

2) Algebraic Plan to VHDL expressions

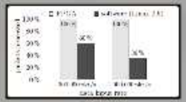
3) Optimization Heurisitics







Performance



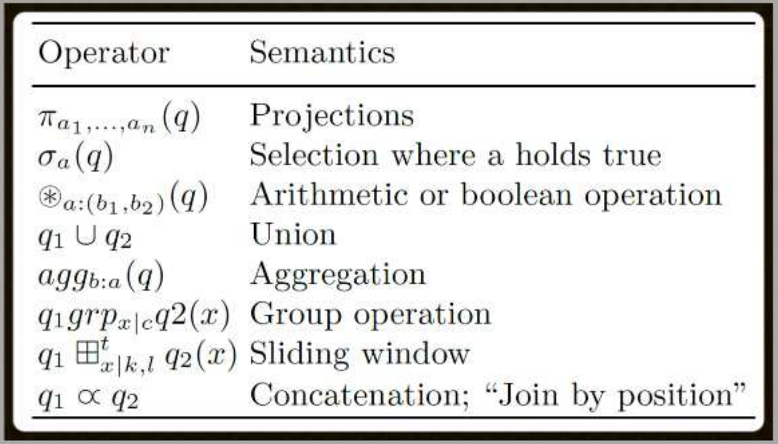
Network Data:
- High package rates
- Actual applications suffer from intra-host communication

In lab: High package rates difficult

→ FPGA not saturated

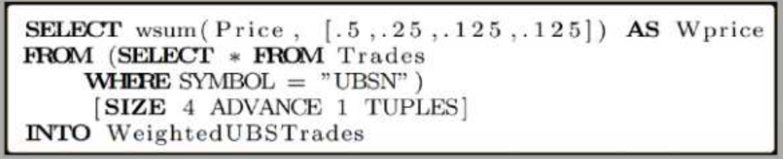
CQL Algebra

- Assumes tuple structured input events
- Nested Operators
- Not the whole range of relational algebra
  - Pre-Processing relevant parts



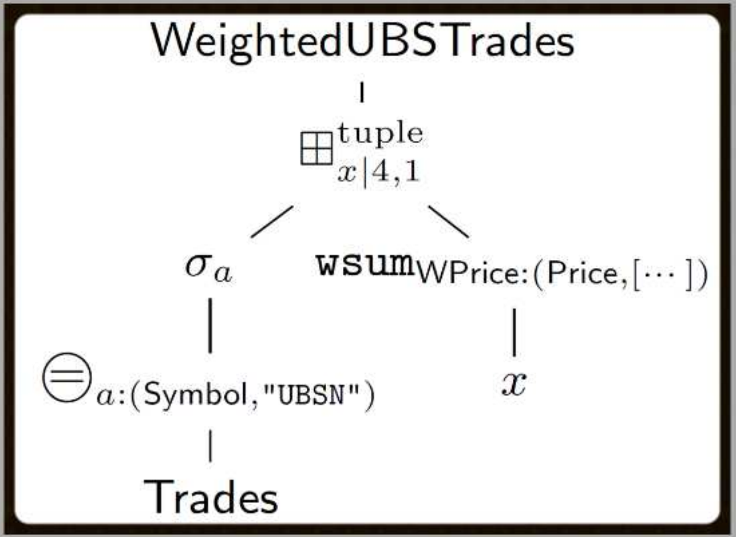
Operator	Semantics
$\pi_{a_1,\ldots,a_n}(q)$	Projections
$\sigma_a(q)$	Selection where a holds true
$\circledast_{a:(b_1,b_2)}(q)$	Arithmetic or boolean operation
$q_1 \cup q_2$	Union
$agg_{b:a}(q)$	Aggregation
$q_1 grp_{x\|c} q2(x)$	Group operation
$q_1 \boxplus^t_{x\|k,l} q_2(x)$	Sliding window
$q_1 \propto q_2$	Concatenation; "Join by position"

Example



```
SELECT wsum(Price, [.5,.25,.125,.125]) AS Wprice
FROM (SELECT * FROM Trades
     WHERE SYMBOL = "UBSN")
      [SIZE 4 ADVANCE 1 TUPLES]
INTO WeightedUBSTrades
```

Examplary Query: Financial trading application


WeightedUBSTrades
$\boxplus^{\text{tuple}}_{x|4,1}$
σ_a
$\texttt{wsum}_{\text{WPrice:(Price,}[\cdots])}$
$\ominus_{a:(\text{Symbol},\texttt{"UBSN"})}$
x
Trades

Component Library

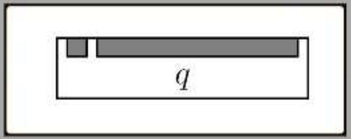
Wiring interface
- n-bit wide tuple + *data_valid* signal

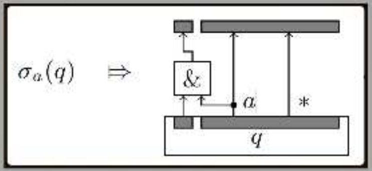
Selection
- Actual Selection: data_valid= true/false
- Arithmetic/Boolean Operation: Extends *q* by adding a new field *a* with the result of the operation

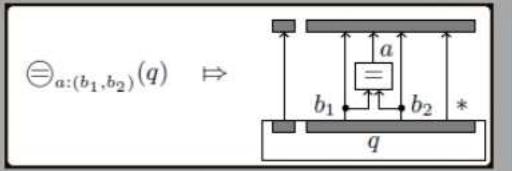
Projection
- "Cutting" non-relevant data bits

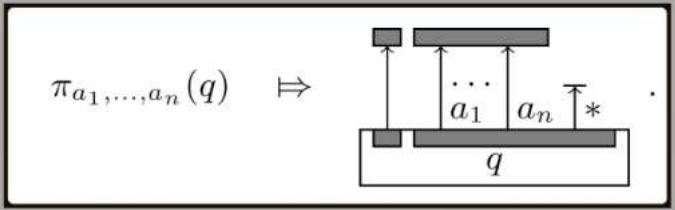
Synchronization
- FIFO queues
- Delay operators

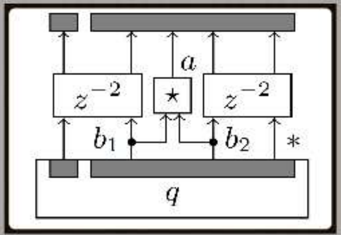
Windowing
- Right-hand side sub-plan q2 wrapped in template circuit
- Additional eos ("end of stream")

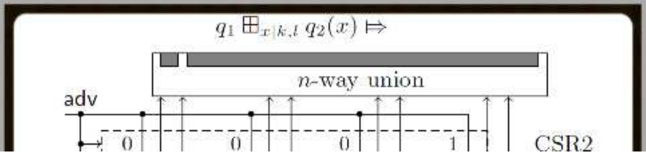






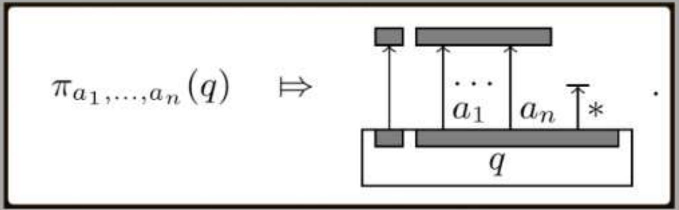




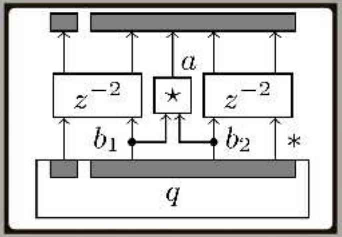


- Arithmetic/Boolean Operation: Extends *q* by adding a new field *a* with the result of the operation

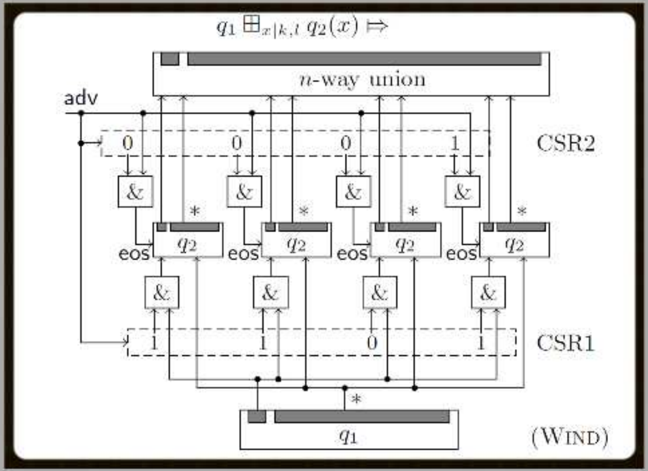
Projection
- "Cutting" non-relevant data bits



Synchronization
- FIFO queues
- Delay operators

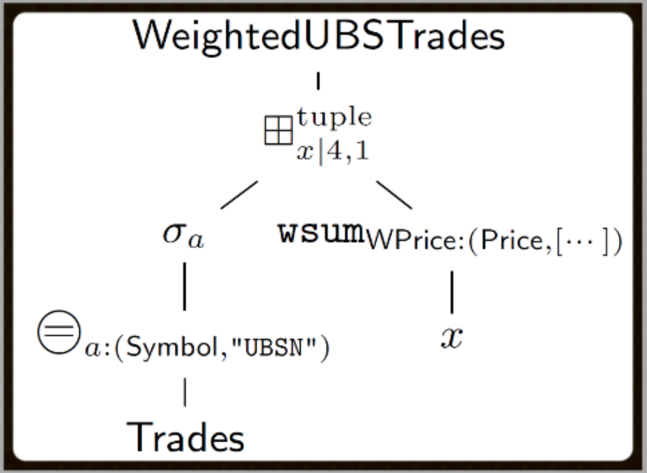


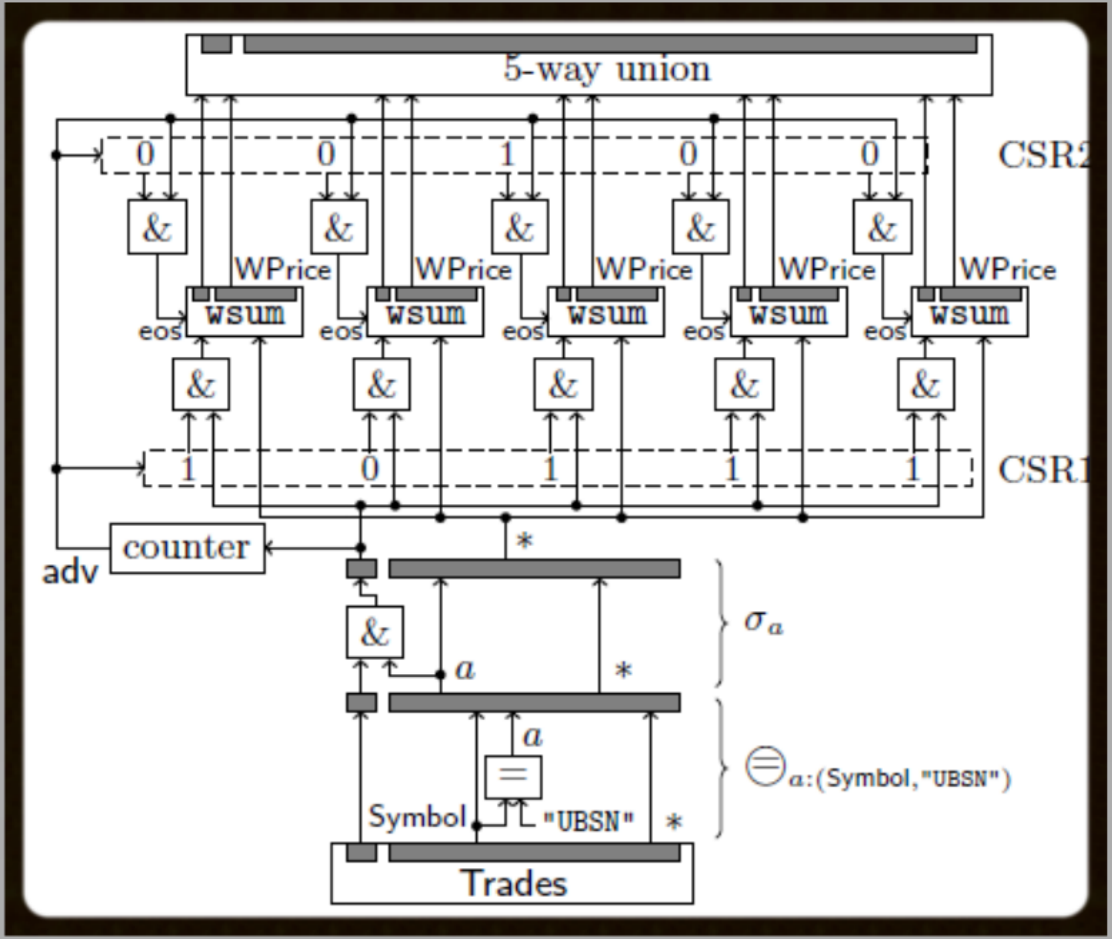
Windowing
- Right-hand side sub-plan q2 wrapped in template circuit
- Additional *eos* ("end of stream") signal
- Sliding window:
  - Parallel computation of active windows
  - CSR1 (Cyclic Shift Registers) denote active/closed windows
  - CSR2 denotes the sub-plan were the next *eos* signal will be send



Hardware Implementation of Example

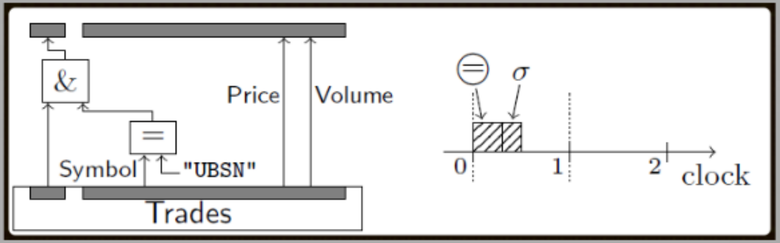
Hardware Implementation of Example

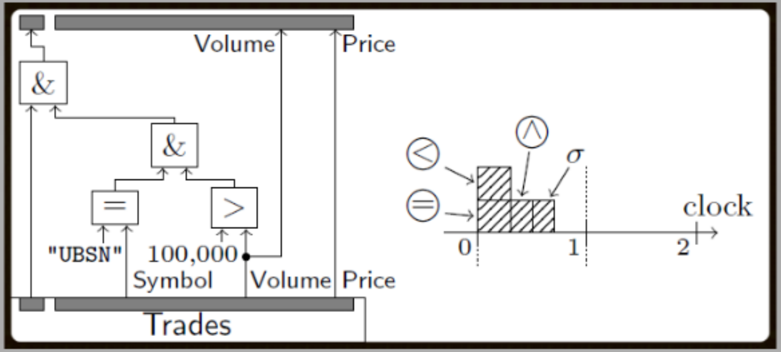




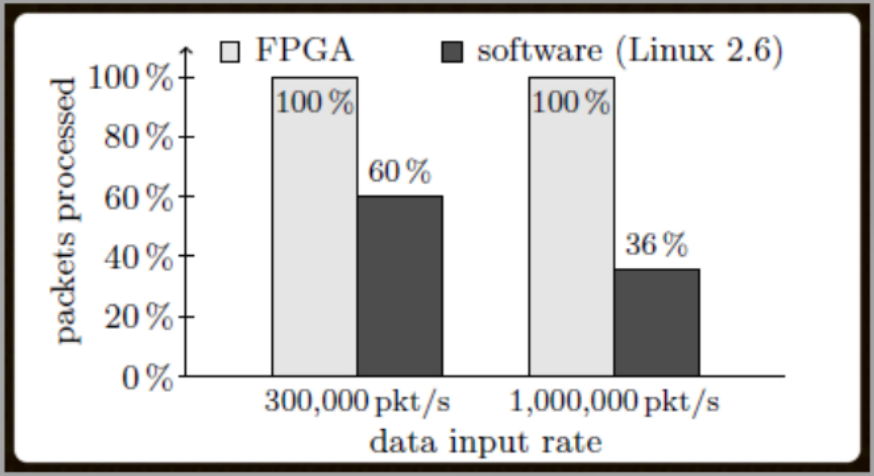
Optimization

- Reducing Clock synchronization
  - If no component inside a plan is clock bound, intermediate registers can be eliminated
- Increasing Parallelism
  - With eliminated registers task parallelism can be increased





Performance

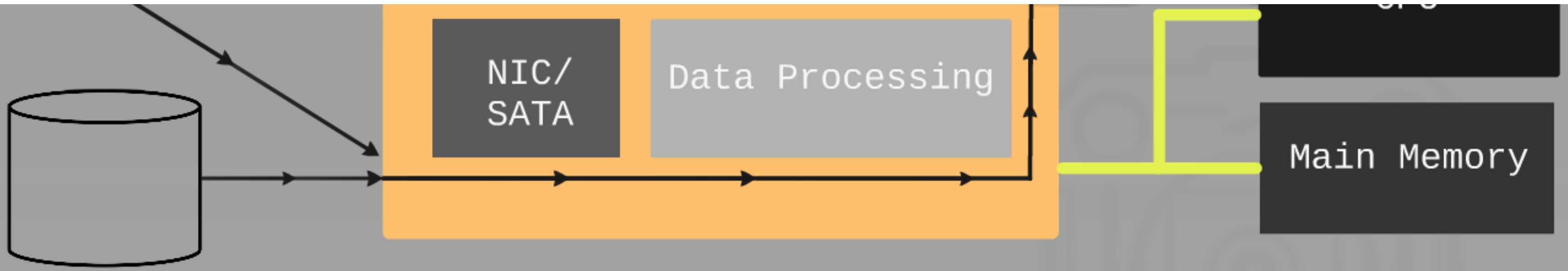


Network Data:

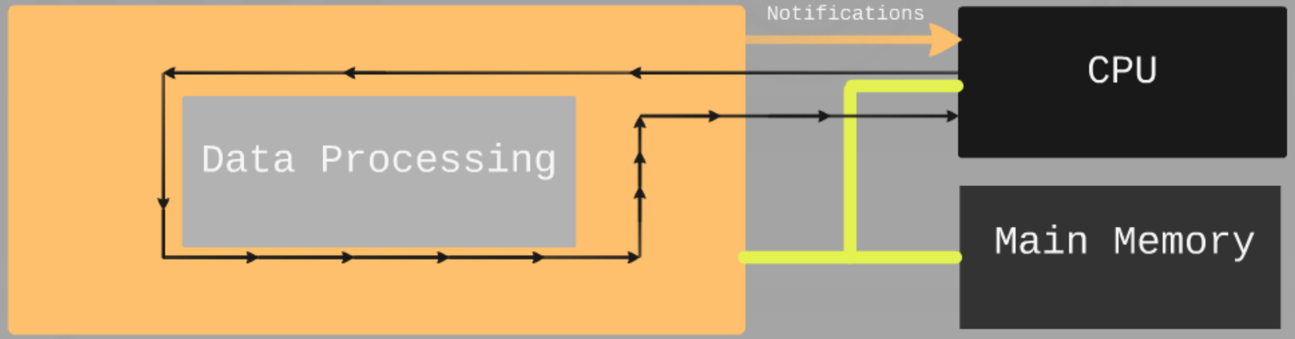
- High package rates
- Actual applications suffer from intra-host communication

In lab: High package rates difficult

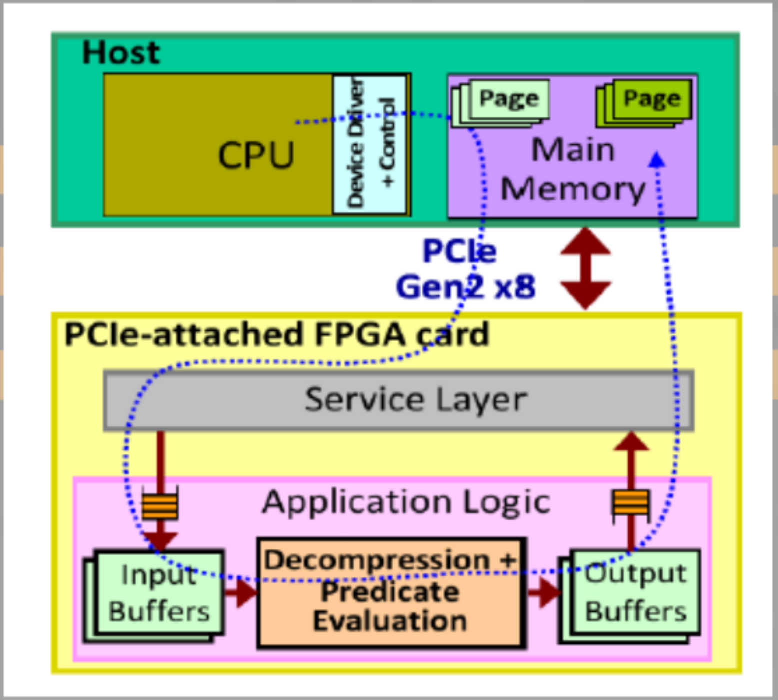
$\longrightarrow$ FPGA not saturated



Co-Processing



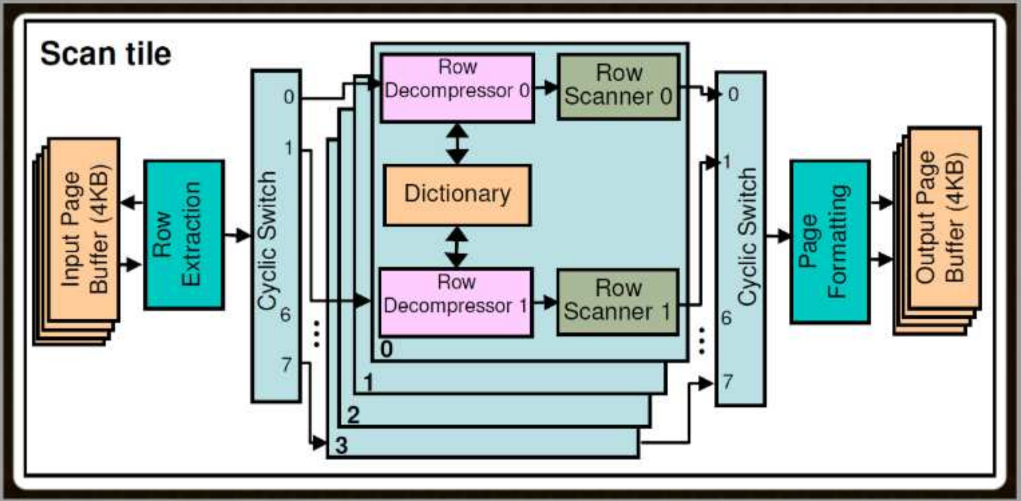
Co-Processor Architecture



- Service Layer
  - DMA Management
  - PCIe Management
  - Job Management
- Application Logic
  - Implementation of required functionality

→ Assumption: In-Memory Database

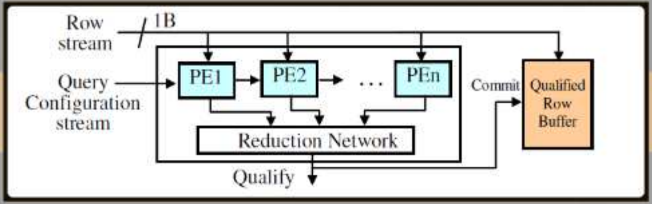
Predicate Evaluation and Row decompression



Central Element for query execution:

- Decompressors
- Row Scanners
- Page buffer
- Logic for extraction of single rows within pages

Row Scanner



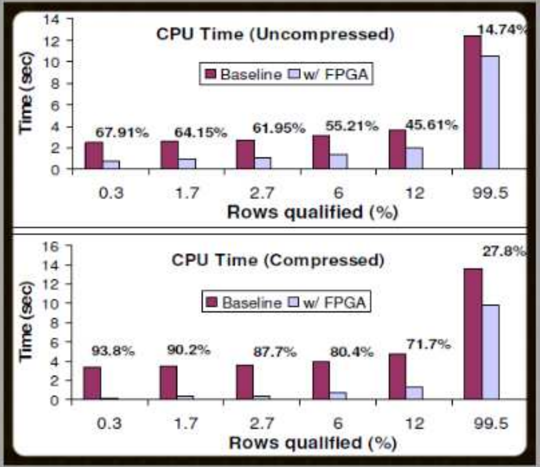
- 1 Predicate Evaluation Unit per Single Predicate
- Speculative Writes into Qualified Row Buffer
- Reduction Network: Tree of reducer units performing logical operations

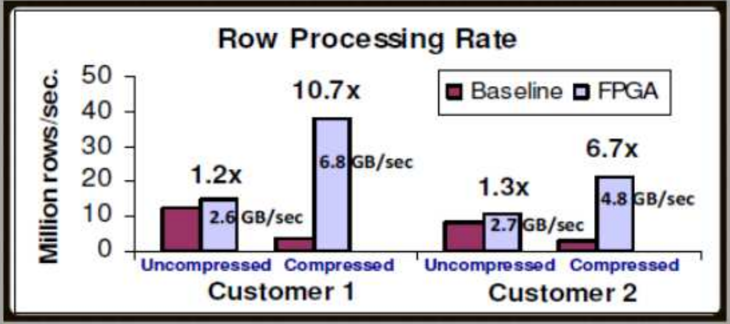
Row Scanners

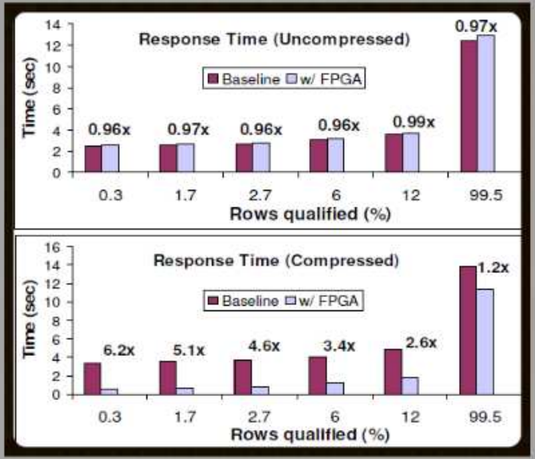
- Page buffer
- Logic for extraction of single rows within pages

- 1 Predicate Evaluation Unit per Single Predicate
- Speculative Writes into Qualified Row Buffer
- Reduction Network: Tree of reducer units performing logical operations

Performance





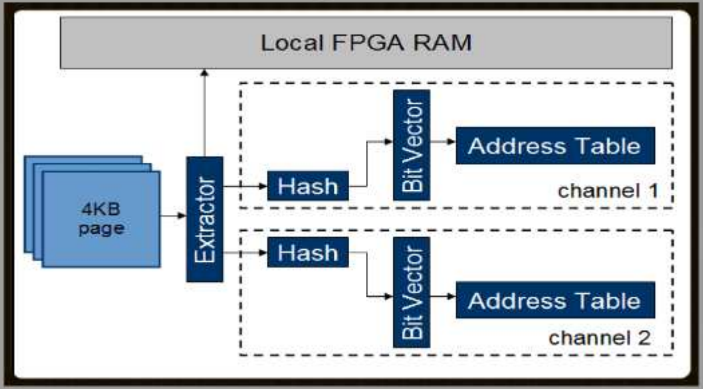


Significant improvement with compressed data

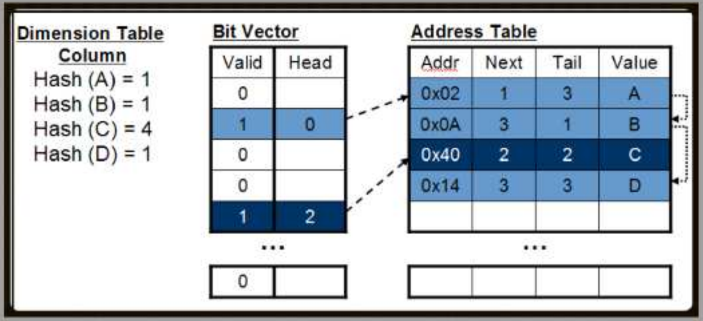
- The lesser qualified records the better

Hash Joins

Build Phase Logic



Address Table



Dimension Table Column

Hash (A) = 1
Hash (B) = 1
Hash (C) = 4
Hash (D) = 1

Bit Vector

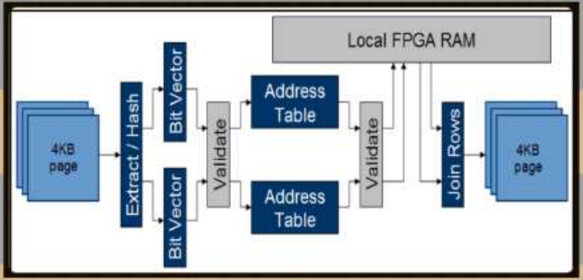
Valid	Head
0	
1	0
0	
0	
1	2
…	
0	

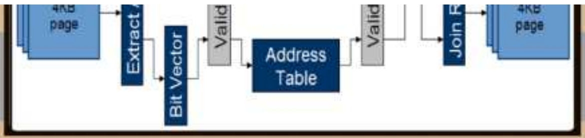
Address Table

Addr	Next	Tail	Value
0x02	1	3	A
0x0A	3	1	B
0x40	2	2	C
0x14	3	3	D
…			

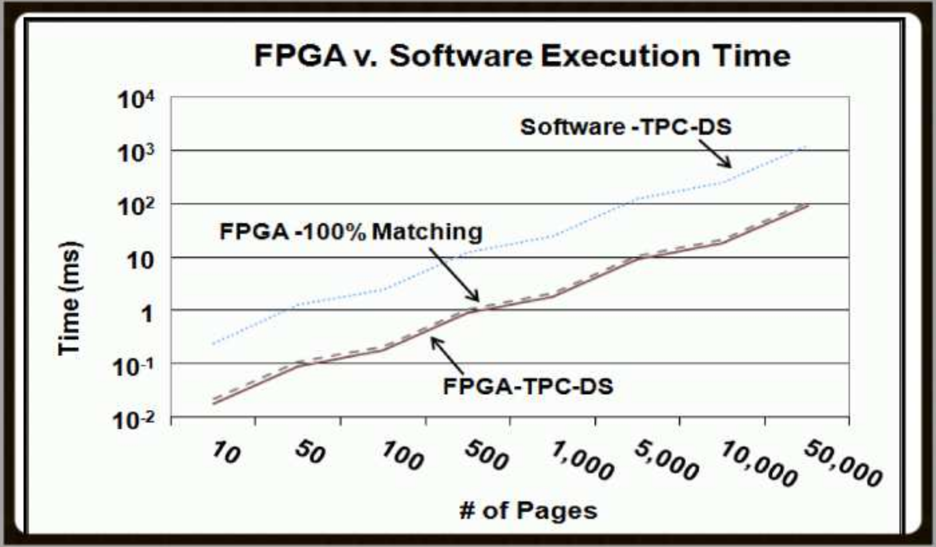
- Classical Hash Join
  - Dimension table must fit into FPGA memory
- Build Phase:
  - Hashed + Pointer to Actual Value stored in address table /(also chaining scheme values)
  - One Dimension Table or join column per channel
- Probe Phase:
  - Fact table streamed through the FPGA
  - Actual Values in the address table prevent false matches due to imperfect hashing

Probe Phase





Performance



Tested with cycle accurate simulator

- TPC-DS query: Multiple dimension tables
- 1 Channel for each table

→ Software implementation: 1.6 million rows/second

FPGA implementation: 18 million rows/second

Worst Cases:

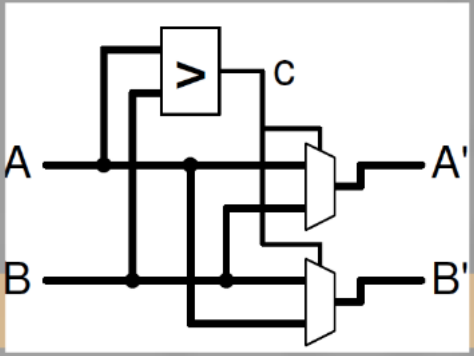
- Every column hashes to the same location
- Every fact table tuple matches and must be joined
  - Merge operation bottleneck

Sort-Merge Join

Initial Sorting in practice most expensive part

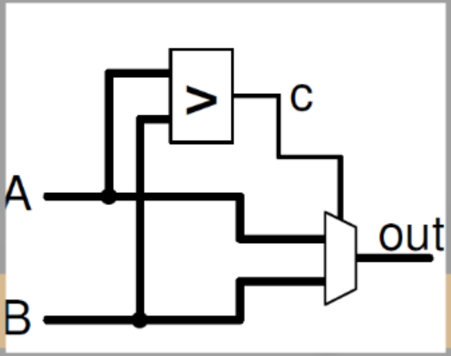
→ External Sorting

Hardware Sorting essential elements



Compare-Swap Element

Compares two inputs and swaps depending on configuration



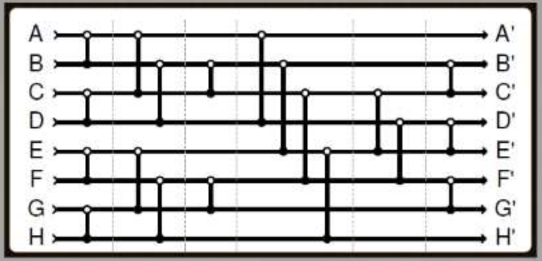
Select-Value Element

Compares two inputs and selects depending on configuration

Compare-Swap Element

Compares two inputs and swaps depending on configuration

Sorting networks



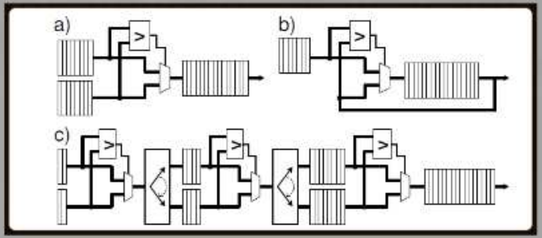
Mathematical Model for sorting; here Even-odd sorting network
- Vertical lines represent compare-swap elements
- Usually only suited for smaller sorting problems
  - On FPGA: Large amount of parallel compare-swap elements available

→ Larger Sorting Problems solvable in parallel

Select-Value Element

Compares two inputs and selects depending on configuration

FIFO-based merge sorter

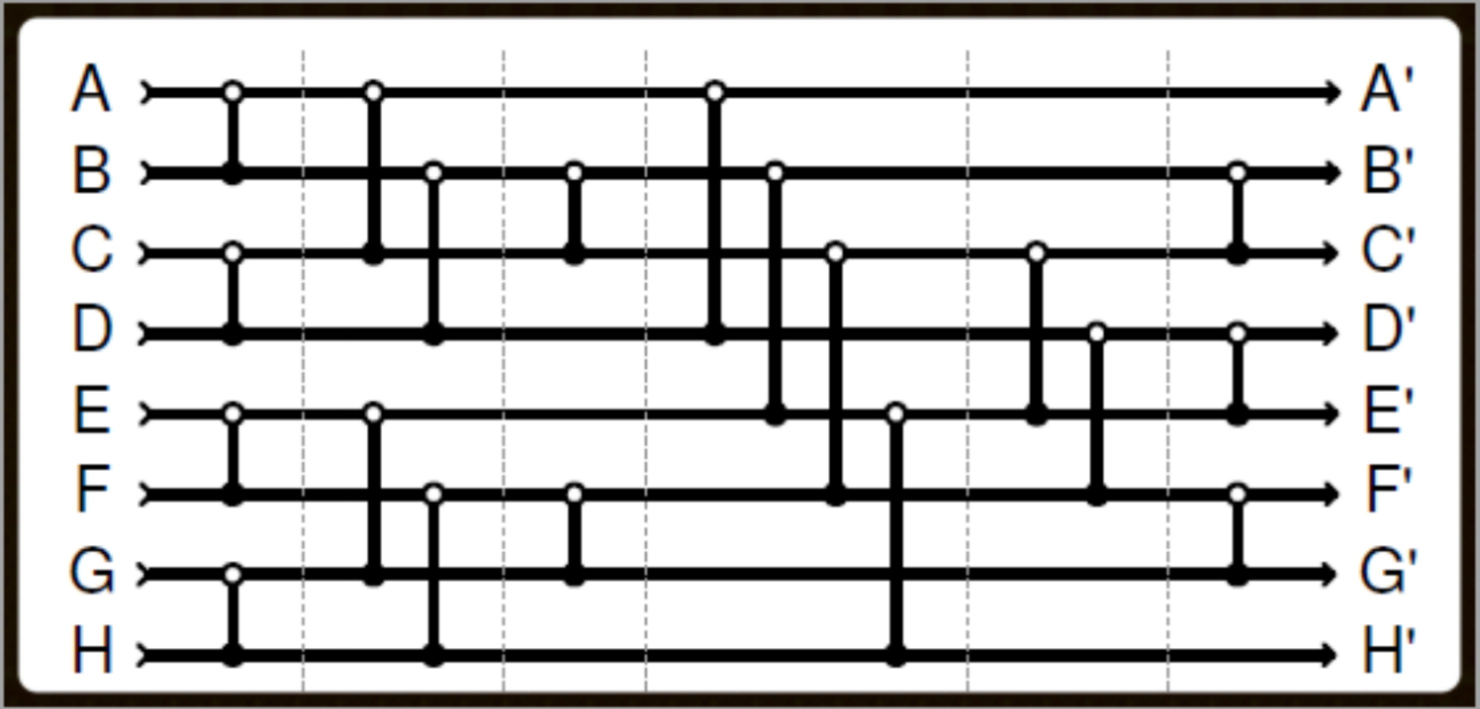


BRAM based:
- BRAM elements used to implement FIFO structures with sorted runs

Select value elements:
- Smaller (bigger) value forwarded to output
- Unselected Value kept for next cycle comparison

→ Cascade of FIFO merge sorters solves bigger sorting problems in parallel

Sorting networks

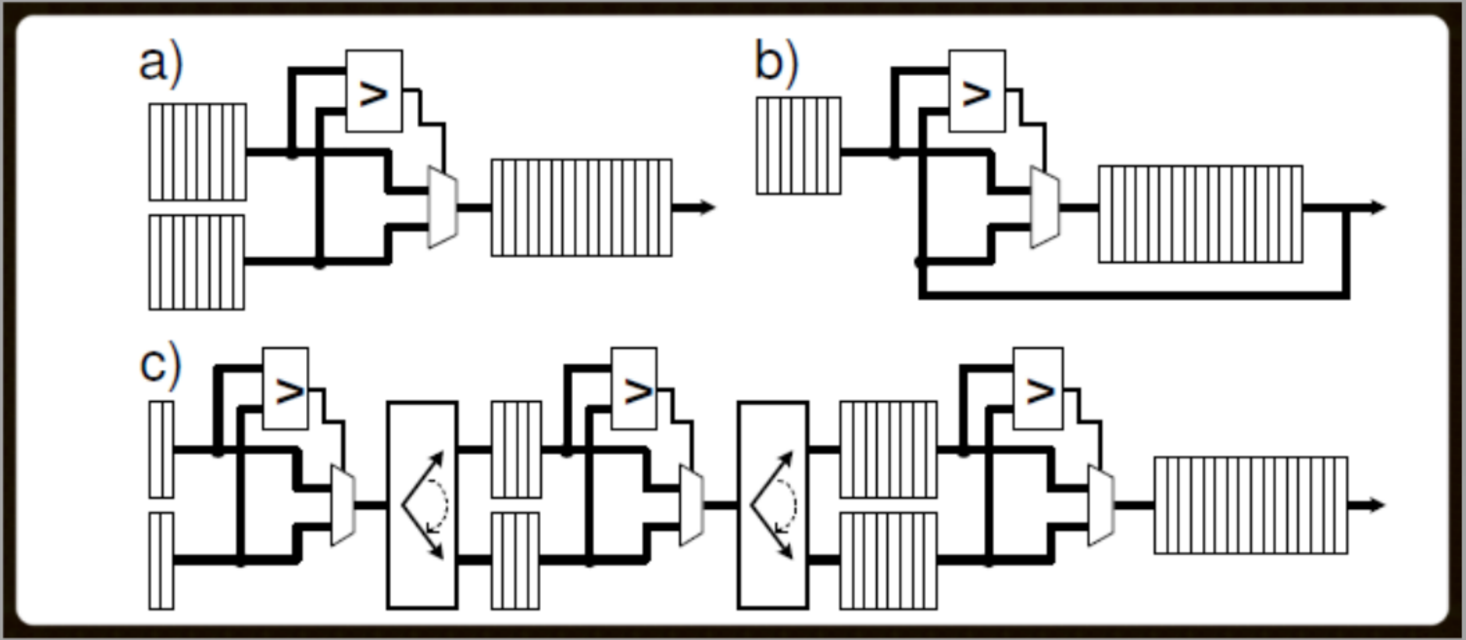


Mathematical Model for sorting; here Even-odd sorting network

- Vertical lines represent compare-swap elements
- Usually only suited for smaller sorting problems
  - On FPGA: Large amount of parallel compare-swap elements available

→ Larger Sorting Problems solvable in parallel

FIFO-based merge sorter



BRAM based:

- BRAM elements used to implement FIFO structures with sorted runs

Select value elements:

- Smaller (bigger) value forwarded to output
- Unselected Value kept for next cycle comparison

→ Cascade of FIFO merge sorters solves bigger sorting problems in parallel

Conclusion

FPGAs offer flexible hardware with a high degree of parallelism, low latency and high throughput rates

Suitable for:

- Network Stream Processing
- Stream Processing
- Co-Processing

Existing Industrial Applications

Netezza

- IBM appliance utilizing FPGAs
- Streaming architecture
- Decompression, Projection and Row Selection

Kickfire

- Column Store
- Co-Processor Architecture
- coupled with MySQL DBMS
- ACID compliant
- Large fraction of SQL processing

DBx

- XtremeData's data warehouse appliance
- Co-Processor Architecture
- PostgreSQL DB engine
- Cluster with Infiniband support
- Large portion of operators implementable on FPGA

Further Research

Concentrates on utilizing re-configurability of FPGAs:

- Partial Reconfiguration
  - On-the-fly composition of FPGA-based SQL query accelerators
- Parameterized circuits
  - Netezza's FAST engine
  - Skeleton automata
    - E.g. XPath -> Implementable using FS automata
    - Generates a parameterized skeleton that can be initialized along transition edges

Existing Industrial Applications

Netezza
- IBM appliance utilizing FPGAs
- Streaming architecture
- Decompression, Projection and Row Selection

Kickfire
- Column Store
- Co-Processor Architecture
- coupled with MySQL DBMS
- ACID compliant
- Large fraction of SQL processing

DBx
- XtremeData's data warehouse appliance
- Co-Processor Architecture
- PostgreSQL DB engine
- Cluster with Infiniband support
- Large portion of operators implementable on FPGA

Further Research

Concentrates on utilizing re-configurability of FPGAs:

- Partial Reconfiguration
  - On-the-fly composition of FPGA-based SQL query accelerators
- Parameterized circuits
  - Netezza's FAST engine
  - Skeleton automata
    - E.g. XPath -> Implementable using FS automata
    - Generates a parameterized skeleton that can be initialized along transition edges

Thank you for your attention!

Questions?