QI 00:
The Architecture and Design of a DATABASE PROCESSING UNIT

Lisa Wu, Andrea Lottarini, Tim Paine, Martha Kim, and Ken Ross

Columbia University, NYC
DPUs are analogous to GPUs

Graphics Workloads

CPU → GPU

Database Workloads

CPU → DPU
Q100: A Stream-Based DPU
Q100: A Stream-Based DPU

- Accelerates analytic queries
Q100: A Stream-Based DPU

• Accelerates analytic queries

The sale of an airline ticket

Sales Projection

Transactional Processing

Analytic Processing
Q100: A Stream-Based DPU

- Accelerates analytic queries
- Direct hardware support for relational operators
Q100: A Stream-Based DPU

- Accelerates analytic queries
- Direct hardware support for relational operators
Q100: A Stream-Based DPU

• Accelerates analytic queries
• Direct hardware support for relational operators
• Processes data as streams
Q100: A Stream-Based DPU

- Accelerates analytic queries
- Direct hardware support for relational operators
- Processes data as streams
Q100: A Stream-Based DPU

- Accelerates analytic queries
- Direct hardware support for relational operators
- Processes data as streams
- Combines spatial and temporal instructions to form a DPU ISA
SELECT s_season, SUM(s_qty) as sum_qty
FROM sales
WHERE s_shipdate >= '2013-01-01'
GROUP BY s_season
ORDER BY s_season
SELECT s_season, SUM(s_qty) as sum_qty
FROM sales
WHERE s_shipdate >= '2013-01-01'
GROUP BY s_season
ORDER BY s_season
SELECT s_season, SUM(s_qty) as sum_qty
FROM sales
WHERE s_shipdate >= '2013-01-01'
GROUP BY s_season
ORDER BY s_season
Q100 Execution and Efficiencies

SALES TABLE

MEMORY

02/27/14

Columbia University

Thursday, March 6, 2014
Q100 Execution and Efficiencies

SALES TABLE

MEMORY

02/27/14 Columbia University
Q100 Execution and Efficiencies

SALES TABLE

MEMORY

02/27/14

Columbia University
Q100 Execution and Efficiencies

SALES TABLE

MEMORY

Partitioned Tables

Temp Column

Thursday, March 6, 2014
Q100 Execution and Efficiencies

SALES TABLE

MEMORY

Partitioned Tables

Temp Column
Q100 Execution and Efficiencies

Read datum once, perform multiple operations
Q100 Execution and Efficiencies

Read datum once, perform multiple operations

Pipeline Parallelism

02/27/14  Columbia University  7
Q100 Execution and Efficiencies

Read datum once, perform multiple operations

Pipeline Parallelism

Data Parallelism
Q100 Execution and Efficiencies

SALES TABLE

MEMORY

Partitioned Tables

Temp Column

Temp Column

Temp Table

Read datum once, perform multiple operations

Pipeline Parallelism

Data Parallelism

Minimize Spills/Fills

02/27/14

Columbia University
Use coarse-grain hardware primitives that operate on coarse-grain data.
How do we implement these operators?

How many tiles should there be and of what type?

How do we generate these query plans?

What kind of interconnect should we use?

Is the Q100 performance and energy efficient?

How do we schedule the plans?
How do we implement these operators?

How many tiles should there be and of what type?

How do we generate these query plans?

What kind of interconnect should we use?

Is the Q100 performance and energy efficient?

How do we schedule the plans?
How do we generate these query plans?

What kind of interconnect should we use?

How do we implement these operators?

How many tiles should there be and of what type?

Is the Q100 performance and energy efficient?

How do we schedule the plans?
How do we generate these query plans?

How do we implement these operators?

How many tiles should there be and of what type?

Is the Q100 performance and energy efficient?

What kind of interconnect should we use?

How do we schedule the plans?
How do we generate these query plans?

How do we implement these operators?

How many tiles should there be and of what type?

How do we schedule the plans?

Bandwidth needs on- and off-chip

What kind of interconnect should we use?

Is the Q100 performance and energy efficient?
How do we schedule the plans?

How do we generate these query plans?

What kind of interconnect should we use?

Bandwidth needs on- and off-chip

How do we implement these operators?

How many tiles should there be and of what type?

Is the Q100 performance and energy efficient?

Columbia University
How do we implement these operators?
Example Tile: Boolean Generator

IN 0

IN 1

==

EN

op

BOOLGEN

OUT
Example Tile: Boolean Generator

IN 0  ---  IN 1  ---  OUT

COLUMN FILTER

BOOLGEN

| op | EN |
|====|----|

IN 2
Example Tile: Boolean Generator

WHERE s_shipdate >= '2013-01-01'
Example Tile: Aggregator

AGG

GRP

DATA

op

sum

EN

OUT

 Thursday, March 6, 2014
Example Tile: Aggregator
Example Tile: Sorter
Example Tile: Sorter

Limitation: number of records
Example Tile: Sorter

PARTITION

GRP DATA

Gamma: number of records

SORT

Columbia University
Q100 Tiles

- **Functional Tiles (7)**
  - Aggregator
  - ALU
  - Boolean Generator
  - Column Filter
  - Joiner
  - Partitioner
  - Sorter
Q100 Tiles

• **Functional Tiles (7)**
  Aggregator
  ALU
  Boolean Generator
  Column Filter
  Joiner
  Partitioner
  Sorter

• **Auxiliary Tiles (4)**
  Table Appender
  Column Selector
  Column Concatenator
  Column Stitcher
Q100 Tiles

- **Functional Tiles (7)**
  - Aggregator
  - ALU
  - Boolean Generator
  - Column Filter
  - Joiner
  - Partitioner
  - Sorter

- **Auxiliary Tiles (4)**
  - Table Appender
  - Column Selector
  - Column Concatenator
  - Column Stitcher

**Tile Characterization Methodology**

Verilog implementation for each tile, synthesized, placed, and routed using Synopsys 32nm Generic Libraries
Tile Characterization

- Area (mm²)
- Power (mW)
- Critical Path (ns)

Bar Graphs for Different Operations (AGG, ALU, BOOLGEN, COLFILTER, JOIN, PART, SORT, APPEND, COLSELECT, CONCAT, STITCH)

02/27/14

Thursday, March 6, 2014
Tile Characterization

- Area (mm²)
- Power (mW)
- Critical Path (ns)

Max Freq 315 MHz

02/27/14

Columbia University

02/27/14 Thursday, March 6, 2014
How many tiles should there be and of what type?
Unbounded Design Space

AGGREGATOR 1 2 3 4 5 6 7 8 9 10 11 12...
ALU 1 2 3 4 5 6 7 8 9 10 11 12...
BOOLEAN GENERATOR 1 2 3 4 5 6 7 8 9 10 11 12...
COLUMN FILTER 1 2 3 4 5 6 7 8 9 10 11 12...
JOINER 1 2 3 4 5 6 7 8 9 10 11 12...
PARTITIONER 1 2 3 4 5 6 7 8 9 10 11 12...
SORTER 1 2 3 4 5 6 7 8 9 10 11 12...
TABLE APPENDER 1 2 3 4 5 6 7 8 9 10 11 12...
COLUMN SELECTOR 1 2 3 4 5 6 7 8 9 10 11 12...
COLUMN CONCATENATOR 1 2 3 4 5 6 7 8 9 10 11 12...
COLUMN STITCHER 1 2 3 4 5 6 7 8 9 10 11 12...
Performance Simulation Methodology

- TPC-H as target workload
- Home-grown C++ simulator, validated against MonetDB
- Completion cycles for each spatial and temporal instructions
- Memory access overheads
- Completion time for each query converted to throughput using the Q100 frequency
Example: Bounding ALU Count

Query Runtime wrt. 1 ALU

Number of ALUs

TPC-H Queries

- Q 1
- Q 2
- Q 3
- Q 4
- Q 5
- Q 6
- Q 7
- Q 8
- Q 9
- Q 10
- Q 11
- Q 12
- Q 13
- Q 14
- Q 15
- Q 16
- Q 17
- Q 18
- Q 19
- Q 20
- Q 21
Example: Bounding ALU Count

![Graph showing query runtime with respect to ALU count.](image)

TPC-H Queries:
- Q 1
- Q 2
- Q 3
- Q 4
- Q 5
- Q 6
- Q 7
- Q 8
- Q 10
- Q 11
- Q 12
- Q 14
- Q 15
- Q 16
- Q 17
- Q 18
- Q 19
- Q 20
- Q 21
Bounded Design Space

AGGREGATOR 1 2 3 4 5 6 7 8 9 10 11 12...

ALU 1 2 3 4 5 6 7 8 9 10 11 12...

BOOLEAN GENERATOR 1 2 3 4 5 6 7 8 9 10 11 12...

COLUMN FILTER 1 2 3 4 5 6 7 8 9 10 11 12...

JOINER 1 2 3 4 5 6 7 8 9 10 11 12...

PARTITIONER 1 2 3 4 5 6 7 8 9 10 11 12...

SORTER 1 2 3 4 5 6 7 8 9 10 11 12...

TABLE APPENDER 1 2 3 4 5 6 7 8 9 10 11 12...

COLUMN SELECTOR 1 2 3 4 5 6 7 8 9 10 11 12...

COLUMN CONCATENATOR 1 2 3 4 5 6 7 8 9 10 11 12...

COLUMN STITCHER 1 2 3 4 5 6 7 8 9 10 11 12...
Bounded Design Space

AGGREGATOR 1 2 3 4 5 6 7 8 9 10 11 12...
ALU 1 2 3 4 5 6 7 8 9 10 11 12...
BOOLEAN GENERATOR 1 2 3 4 5 6 7 8 9 10 11 12...
COLUMN FILTER 1 2 3 4 5 6 7 8 9 10 11 12...
JOINER 1 2 3 4 5 6 7 8 9 10 11 12...
PARTITIONER 1 2 3 4 5 6 7 8 9 10 11 12...
SORTER 1 2 3 4 5 6 7 8 9 10 11 12...
TABLE APPENDER 1 2 3 4 5 6 7 8 9 10 11 12...
COLUMN SELECTOR 1 2 3 4 5 6 7 8 9 10 11 12...
COLUMN CONCATENATOR 1 2 3 4 5 6 7 8 9 10 11 12...
COLUMN STITCHER 1 2 3 4 5 6 7 8 9 10 11 12...
Bounded Design Space

AGGREGATOR 1 2 3 4 5 6 7 8 9 10 11 12...
ALU 1 2 3 4 5 6 7 8 9 10 11 12...
BOOLEAN GENERATOR 1 2 3 4 5 6 7 8 9 10 11 12...
COLUMN FILTER 1 2 3 4 5 6 7 8 9 10 11 12...
JOINER 1 2 3 4 5 6 7 8 9 10 11 12...
PARTITIONER 1 2 3 4 5 6 7 8 9 10 11 12...
SORTER 1 2 3 4 5 6 7 8 9 10 11 12...
TABLE APPENDER 1 2 3 4 5 6 7 8 9 10 11 12...
COLUMN SELECTOR 1 2 3 4 5 6 7 8 9 10 11 12...
COLUMN CONCATENATOR 1 2 3 4 5 6 7 8 9 10 11 12...
COLUMN STITCHER 1 2 3 4 5 6 7 8 9 10 11 12...

2.9 Million Designs!!
Bounded Design Space

AGGREGATOR: 1 2 3 4 6 7 8 9 10 11 12...
ALU: 1 2 3 4 5 7 8 9 10 11 12...
BOOLEAN GENERATOR: 1 2 3 4 5 6 8 9 10 11 12...
COLUMN FILTER: 1 2 3 4 5 6 8 9 10 11 12...
JOINER: 1 2 3 4 6 7 8 9 10 11 12...
PARTITIONER: 1 2 3 4 5 7 8 9 10 11 12...
SORTER: 1 2 3 4 5 6 8 9 10 11 12...
TABLE APPENDER: 1 2 3 4 5 6 7 8 10 11 12...
COLUMN SELECTOR: 1 2 3 4 5 6 7 9 10 11 12...
COLUMN CONCATENATOR: 1 2 3 4 5 6 7 5 6 7 8 9 10 11 12...
COLUMN STITCHER: 1 2 4 5 6 7 8 9 10 11 12...

Explore tiles that consume >= 5mW

2.9 Million Designs!!
Bounded Design Space

AGGREGATOR
ALU
BOOLEAN GENERATOR
COLUMN FILTER
JOINER
PARTITIONER
SORTER
TABLE APPENDER
COLUMN SELECTOR
COLUMN CONCATENATOR
COLUMN STITCHER

Explore tiles that consume >= 5mW

150 Designs

02/27/14
Columbia University
Q100 Designs for Further Evaluation

TPC-H Runtime (milliseconds)

Power (Watts)

02/27/14

Columbia University
Q100 Designs for Further Evaluation

TPC-H Runtime (milliseconds) vs. Power (Watts)

- Low Power
- 1 ALU
- 1 Partitioner
- 1 Sorter

02/27/14
Columbia University
Q100 Designs for Further Evaluation

TPC-H Runtime (milliseconds) vs. Power (Watts)

- Low Power
  - 1 ALU
  - 1 Partitioner
  - 1 Sorter

- Pareto
  - 4 ALUs
  - 2 Partitioners
  - 1 Sorter

02/27/14
Columbia University
Q100 Designs for Further Evaluation

TPC-H Runtime (milliseconds) vs. Power (Watts)

- **Low Power**
  - 1 ALU
  - 1 Partitioner
  - 1 Sorter

- **Pareto**
  - 4 ALUs
  - 2 Partitioners
  - 1 Sorter

- **High Perf**
  - 5 ALUs
  - 3 Partitioners
  - 6 Sorters

02/27/14
Columbia University
Bandwidth needs on- and off-chip
Interconnect (Network on Chip) Bandwidth Needs

Runtime Normalized to IDEAL vs NoC BW Limit (GB/s)

High Perf
Interconnect (Network on Chip) Bandwidth Needs

Runtime Normalized to IDEAL

NoC BW Limit (GB/s)

High Perf

02/27/14

Columbia University
Interconnect (Network on Chip) Bandwidth Needs

Runtime Normalized to IDEAL

NoC BW Limit (GB/s)

High Perf
Pareto

Columbia University
Interconnect (Network on Chip) Bandwidth Needs

Runtime Normalized to IDEAL

NoC BW Limit (GB/s)

High Perf
Pareto
Low Power

02/27/14
Columbia University

02/27/14
Columbia University
Interconnect (Network on Chip) Bandwidth Needs

NoC Limit @ 6.3 GB/s
Scaled down from Intel TeraFlop

Low Power
Pareto
High Perf

NoC BW Limit (GB/s)

Runtime Normalized to IDEAL
Bandwidth to/from Memory

Low Power

- Blue circles: Read
- Blue triangles: Write

Bandwidth (GB/s)

02/27/14  Columbia University
Bandwidth to/from Memory

Low Power

- Read
- Write

Pareto

- Read
- Write

High Perf

- Read
- Write

Bandwidth (GB/s)

Thursday, March 6, 2014
Bandwidth to/from Memory

Low Power
- Read
- Write

Pareto
- Read
- Write

High Perf
- Read
- Write

BW Read Limit @ 20 or 30 GB/s

BW Write Limit @ 10 GB/s
Is the Q100 performance and energy efficient?
Software Comparison Methodology

• MonetDB on Sandybridge server
• Energy Measurements:
  • Intel’s Running Average Power Limit (RAPL) energy meters
  • Core domain only
  • Sample energy counters at 10ms intervals
  • Exclude machine idle power
Comparison with Software (MonetDB)

Relative Runtime:
- 0% - LowPower
- 10% - Pareto
- 20% - HighPerf

Relative Power:
- 0% - LowPower
- 5% - Pareto
- 15% - HighPerf

Relative Energy:
- 0% - LowPower
- 5% - Pareto
- 100% - HighPerf

02/27/14

Columbia University
Comparison with Software (MonetDB)

37X-70X Better Performance

Relative Runtime
30%
20%
10%
0%

Relative Power
15%
10%
5%
0%

Relative Energy
1.00%
0.75%
0.50%
0.25%
0%

LowPower
Pareto
HighPerf

02/27/14
Columbia University
Comparison with Software (MonetDB)

- 37X-70X Better Performance
- 1/1000th Energy Consumption
Comparison with Software (MonetDB)

**Relative Runtime**
- 37X-70X Better Performance

**Relative Power**
- 5%
- 0.25%
- 0.50%
- 0.75%
- 1.00%

**Relative Energy**
- 1/1000th Energy Consumption

**100X Input Data Size**
- LowPower
- Pareto
- HighPerf

Columbia University

02/27/14

Thursday, March 6, 2014
Comparison with Software (MonetDB)

- Relative Runtime
  - 37X-70X Better Performance
  - 10X Performance

- Relative Power
  - 0%
  - 5%
  - 10%
  - 15%

- Relative Energy
  - 0%
  - 0.25%
  - 0.50%
  - 0.75%
  - 1.00%

1/1000th Energy Consumption

100X Input Data Size

Columbia University
Comparison with Software (MonetDB)

- **Relative Runtime**:
  - 37X-70X Better Performance
  - 10X Performance

- **Relative Power**
  - LowPower
  - Pareto
  - HighPerf

- **Relative Energy**
  - 1/1000th Energy Consumption
  - < 1/100th Energy Consumption

10X Performance

Columbia University

02/27/14
Conclusions

• Q100 is a highly efficient domain-specific accelerator for analytical database workloads

• ISA exploits parallelism and streaming efficiencies

• At < 15% area and power of a Xeon core, a Q100 device gets exceptional performance and energy efficiency

• Exciting research opportunities for DPU