A Co-Design Approach for Accelerated SQL Query Processing via FPGA-based Data Filtering

Andreas Becher, Daniel Ziener, Klaus Meyer-Wegener and Jürgen Teich
Department of Computer Science

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

design

Principle

Accelerate query processing by maximizing utilization of available I/O rate together with smart Hardware/Software Co-Processing.

```
select sum(store_sales.ss_net_profit)  from date_dim,
 store_sales where ((date_dim.d_year = 2000 and date_dim.d_moy
  = 12) or (date_dim.d_year = 2001 and date_dim.d_moy = 1 ))
 and store_sales.ss_sold_date_sk = date_dim.d_date_sk;
```

Listing 1: Example query (Query 1)

- Hardware modules reduce the amount of data
  - Calculate results of arithmetic operations and trim no longer needed attributes
  - Filter based on *where* clause restrictions
  - Filter based on HASH table (JOIN) entries
- Software: Use Multiprocessor and cache system for fast HASH table lookups and result aggregation

Hardware Module Library

Query accelerators can be freely composed from 3 types of modules:

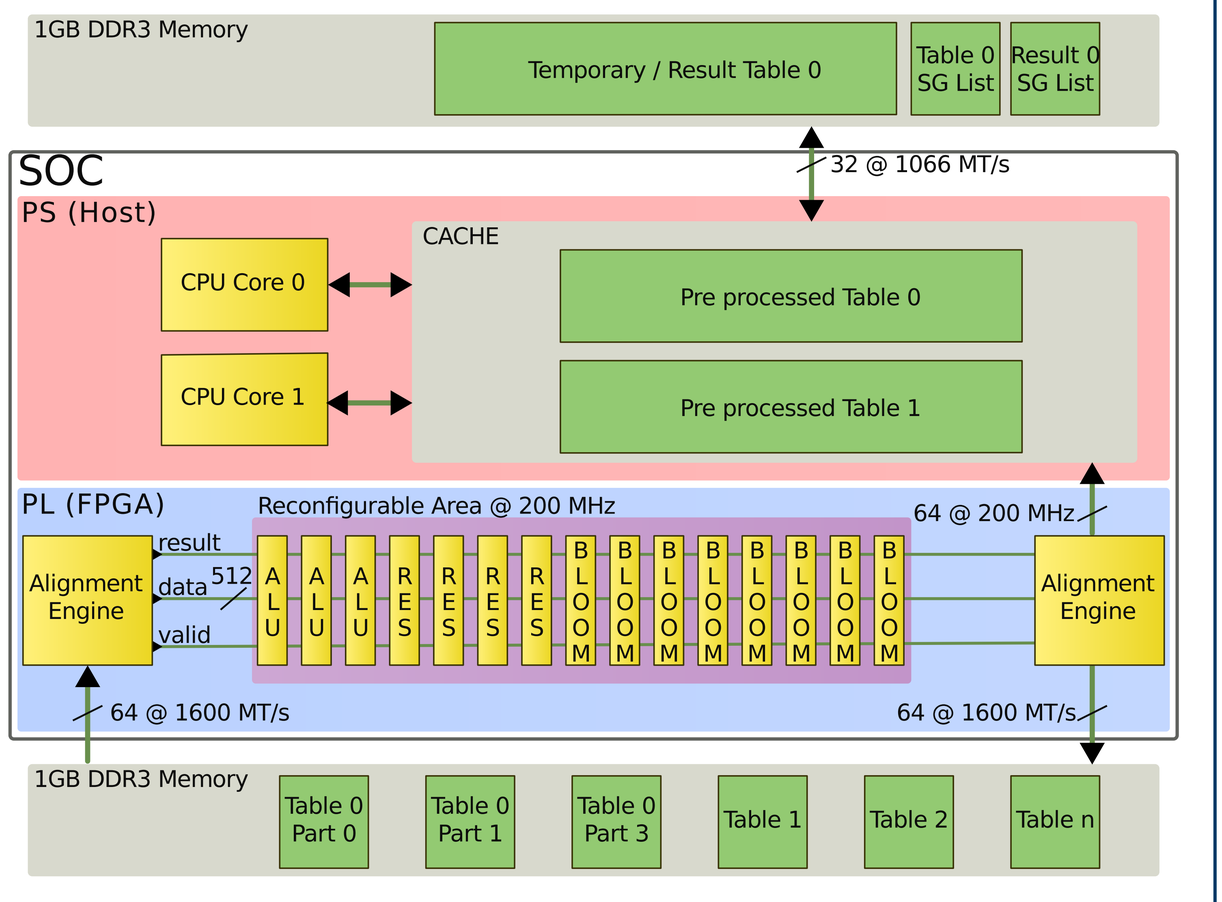
Arithmetic (ALU) $+, -, \times, \div$

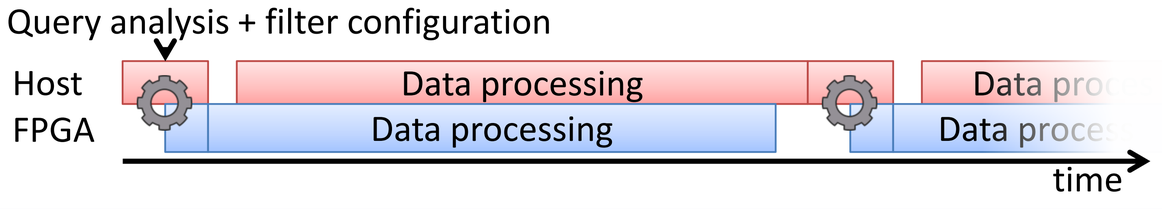
Restriction (RES) $=, \neq, \leq, \geq, <, >$

Bloom Filter (BLOOM) JOIN (*if key* $\in \mathbb{HT}$)

The concatenated modules form an accelerator data path. Multiple accelerators can be synthesized, each suiting a specific query type, and loaded to the reconfigurable area during runtime within milliseconds.

Hardware/Software Co-Processing





Bloom Filter

*JOIN*s in relational databases are used to merge two tables based on common attributes called *keys*. A HASH JOIN uses a HASH function and a HASH table to check for certain *keys* to accelerate the search problem. Bloom Filters may be used to accelerate the *JOIN* operation by removing as much tuples as possible which are, definitely, not contained in the HASH table. The probability E_B of a wrongly passed tuple of our approach is influenced by a number of variables:

- **m** Number of places in one BLOOM module (fixed during synthesis)
- **g** Number of grouped BLOOM modules sharing a HASH function
- **b** Number of groups implementing a Bloom Filter (independent HASH functions)
- **n** Number of entries to be inserted into the Bloom Filter

Our highly parametrizable Bloom Filter modules allow for variation of b and g without hardware reconfiguration based on expected input size n at hand. This leads to a lower probability of a wrongly passed Tuple (E_B) which can be calculated as follows:

$$E_B = \prod_{i=0}^{b-1} \left(1 - \left(1 - \frac{1}{g_i \cdot m}\right)^n\right)$$

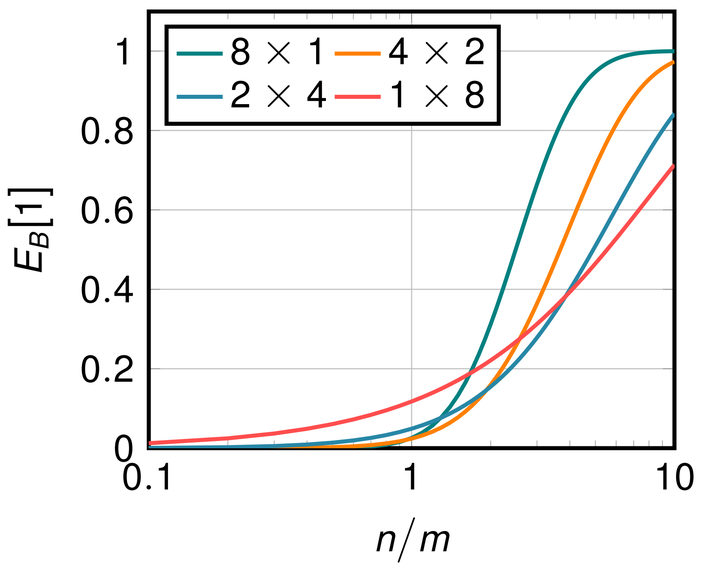


Figure 1: Bloom Error E_B depending on the ratio of inserted items $\left(\frac{n}{m}\right)$ and different groups sizes g and number of concatenated filters b for $m = 16384$ and a total of 8 instances. The legend reads $b \times g$.

Experimental Results

The performance gains of our approach come with the better exploitation of available I/O. While the fastest I/O interface is used to store the tables row-based, only a fraction of data remains as the final result. Taking this into account, our accelerator acts as a *smart* filter to remove attributes and tuples not needed. Results have been obtained with a 1 GB DB scale factor and queries based on TPC-DS test suite [1].

Attribute Trimming The Alignment Engines trim superfluous attributes from the stream and change the attribute order in a suited way (e.g., restriction operands)

Restrictions Compare attributes of tuples and mark them as *valid*

Bloom After inserting *valid* tuples into the Bloom Filters, tuples can be probed

		Approach		Improvement Factors			
		exec. time	energy	ARM		x86	
	BLOOM (b×g)	[ms]	[J]	exec. time	energy	exec. time	energy
Query 1	(3×1)	41.45	0.17	200.95	9.94	10.13	29.83
Query 2	(8×1)	89.10	0.38	98.43	4.87	6.40	18.83
	(2×4)	57.40	0.24	152.80	7.56	9.93	29.23
Average (All tested Queries)		50.87	0.22	149.36	7.39	8.55	25.17

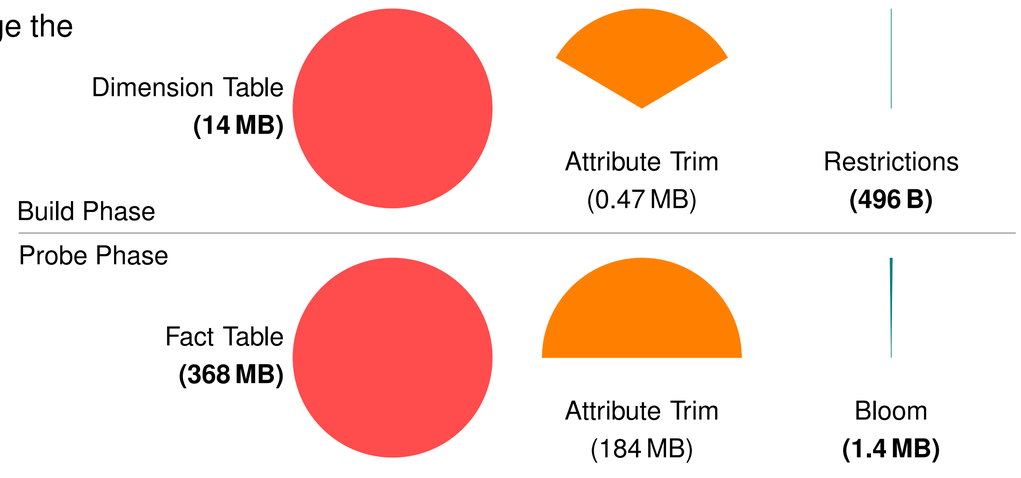


Figure 2: Query 1, Remaining number of tuples after each step in percentages and corresponding data size

[1] *TPC Benchmark™DS (TPC-DS): The New Decision Support Benchmark Standard*. [accessed 03-December-2015]. URL: http://www.tpc.org/tpcds/.