An MPSoC for Energy-Efficient Database Query Processing

Sebastian Haas, Oliver Arnold, Benedikt Nöthen, Stefan Scholze, Georg Ellguth, Andreas Dixius, Sebastian Höppner, Stefan Schiefer, Stephan Hartmann, Stephan Henker, Thomas Hocker, Jörg Schreiter, Holger Eisenreich∗, Jens-Uwe Schlußler, Dennis Walter, Tobias Seifert, Friedrich Pauls, Mattis Hasler, Yong Chen, Hermann Hensel, Sadia Moriam, Emil Matúš, Christian Mayr, René Schüffny, Gerhard P. Fettweis
Technische Universität Dresden, Dresden, Germany
{first.last}@tu-dresden.de

ABSTRACT
This paper presents a heterogeneous database hardware accelerator MPSoC manufactured in 28 nm SLP CMOS. The 18 mm² chip integrates a runtime task scheduling unit for energy-efficient query processing and hierarchical power management supported by an ultra-fast dynamic voltage and frequency scaling. Four processing elements, connected by a star-mesh network-on-chip, are accelerated by an instruction set extension tailored to fundamental data-intensive applications. We evaluate the MPSoC with typical database benchmarks focusing on scans and bitmap operations. When the processing elements operate on data stored in local memories, the chip consumes 250 mW and shows a 96x energy efficiency improvement compared to state-of-the-art platforms.

1. INTRODUCTION
Data processing is becoming an ubiquitous challenge as it needs to be implemented everywhere. The continuously growth of data further intensifies resolving technical issues. Applications are spreading into many fields such as sensor nodes [1], Car2X communication [13], tomorrow for realizing edge computing at every base station or WiFi access points [16], and thereafter into mobile devices by the Internet of Things (IoT). This centralized approach creates the advantage to store and process data as close as possible to its sources and where it is needed. Especially, mobile communications systems as well as database systems require minimum latencies to realize features of the IoT and to provide low reaction times for query processing, respectively.

General relational database systems contain a database management system (DBMS) and the database storage. The DBMS acts as the executive component which interacts with the user. It is responsible for data processing, data security, data integrity, and query processing. Thereby, query processing is the most timing-critical task due to a direct interconnection to the user and the storage. Hence, it significantly determines the performance of the entire database system.

Existing database processors typically use general purpose CPUs [15, 19] or GPUs [6, 7] which can achieve high performances. In particular, GPUs perform excellent due to a great thread parallelism while CPUs have advantages on branch intensive codes. However, they always lead to a poor power-to-throughput ratio. Consequently, domain-specific solid-state circuits are required to boost the performance and to offload intensive processing to lower the power consumption over today’s CPUs and GPUs by orders of magnitude. Besides the specialized computing elements, these circuits also benefit from separate control units to handle task scheduling and data management.

For instance, the authors in [12] apply this promising approach to improve query processing performance by integrating specialized hardware. They introduce the SPARC M7 processor with database analytical accelerators to deal with simple queries and basic database algorithms. Another work in [18] shows a Field Programmable Gate Array (FPGA) accelerator next to a relational DBMS. By offloading computations of database analytics queries, they achieve up to an order of magnitude improvement. Further work investigating application-specific hardware processors has been evaluated in [22]. They demonstrate a significant performance gain while operating on less than 15 % of the power of state-of-the-art Intel cores. The authors in [5, 17] have shown that even the usage of the well-known instruction set extensions of general-purpose Intel processors such as SSE improve the performance of important database algorithms. Similarly in [20], research about implementing database query processing on an MPSoC has been presented. They address the challenges on system level while using a low-energy processor with dedicated compute and control units. All these solutions show the excellent usage of custom-made hardware providing the required energy efficiency. However, to the best of our knowledge, they do not offer a complete system that integrates application-specific hardware accelerators to-
gether with runtime task dispatching for parallelization as well as power and data management in a single chip.

In this paper, we present a heterogeneous database hardware accelerator (Tomahawk3). Our solution combines advanced database processing with dynamic task scheduling, resource management, and hierarchical power management to achieve the performance requirements and to minimize the energy consumption. These include the applied power management techniques dynamic voltage and frequency scaling (DVFS) and adaptive voltage scaling (AVS). The database-specific processing elements exhibit an enormous energy-efficiency enabled by the included extended instruction set. Our integrated scheduling unit is optimized for query processing and allows to allocate the processing elements as well as to dispatch tasks to them. We evaluate the Tomahawk3 chip with typical database operators and benchmarks and compare our results with state-of-the-art CPUs and a GPU.

2. THE TOMAHAWK3 CHIP

In this section, we explain the concept as well as the architectural overview of our manufactured Tomahawk3 chip and describe the single components.

2.1 Concept

The general Tomahawk architecture [4] consists, firstly, of multiple heterogeneous application-specific processing elements (PEs) and, secondly, of a so-called control-plane including a control unit named CoreManager (CM), a global memory, and peripherals. The PEs include their own local data and instruction memory and are isolated from the control-plane. The type of execution accomplished by the CM is comparable with the concept of a superscalar processor but on a higher level of abstraction. Similar to the execution of parallel instructions, the common tasks of the CM are to control the platform, to allow an adaptive runtime scheduling with power management capabilities, and to be responsible for the data transfer between the PEs, the global memory, and the peripherals. A data flow graph given by the application is executed by the CM to schedule tasks to the PEs. Therefore, the CM analyzes task and data dependencies to assure the parallelism. Additionally, the individual PE’s supply voltages and frequencies are adapted according to the current workload to ensure an overall low power consumption.

The primary focus of the Tomahawk architecture was on mobile communications systems. However, the concept can be easily mapped to query processing which require high performance and energy-efficient computing as well. Data and power management together with task scheduling can be exploited in a similar way. The parallelization methods of basic database operators such as scan, compression, and sorting are deposited in the program code of the CM. In database systems, typical queries are composed of multiple subsequent such operators. By using this knowledge, the CM is able to execute the entire application control-flow, now defined as the query execution plan.

2.2 Platform Specification

The Tomahawk3 MPSoC is composed of five heterogeneous cores, two low power DDR2 (LPDDR2) interfaces, and an FPGA interface connected by a packet-switched network-on-chip (NoC) as depicted in Figure 1. The chip was fabricated in GLOBALFOUNDRIES 28 nm SLP CMOS technology and occupies 3×6 mm² = 18 mm² with 16.85 M NAND2 gate equivalents and 480 kB SRAM. The die photo is shown in Figure 2.

![Figure 1: Tomahawk3 MPSoC block diagram.](image)

The CM follows our application-specific instruction set processor (ASIP) approach in [14], accelerated with an instruction set extension optimized for query processing. The remaining four cores, the PEs, are based on a Tensilica RISC processor (ASIP) approach in [14], accelerated with an included extended instruction set for frequently used database algorithms [3]. Additionally, two LPDDR2 memory interfaces each with 12.5 Gbit/s data rate provide access to two off-chip memories. Multiple MPSoCs can be connected by a SerDes link that serves as a 5 Gbit/s high-speed connection to the FPGA interface. All domains have a separate local synchronous clock and are globally asynchronously connected via NoC (GALS). This approach enables flexible connections between the components which can run at different clock frequencies generated by their own all-digital phase locked loops (ADPLL) [8] in the range of 83-2000 MHz varied in clock period steps of 0.125 ns.

A NoC architecture with two routers and a star-mesh topology is implemented and minimizes the routing delay. The routers are connected by a serial NoC link that bridges 5 mm on-chip distance on the top metal layers with low voltage swing [10]. It achieves an energy consumption of less than 70 fJ/bit/mm operating at 80 Gbit/s. The NoC allows a high flexibility in terms of the packet structure and the topology. Hence in future, the architecture can be scaled to many more routers and cores for different application scenarios.

![Figure 2: Tomahawk3 die photo.](image)

2.3 CoreManager

The CM serves as the central control unit in Tomahawk3 which configures the platform during runtime. The overall specification is given in Table 1.
In the following, we explain the procedure to be executed by the CM after receiving an incoming query. Previously, the database or a part of it is provided in the global memory of the Tomahawk3. Furthermore, the control-flow for each individual query is stored in the local memory of the CM. Hence at first, the CM analyzes possible partitioning methods and initiates the data transfer from the off-chip memory to the PE’s local data memories. Most of the database algorithms can be fully parallelized among multiple PEs, i.e., there exists no or only a small sequential part, respectively. This leads to a simplified data partitioning and thus to a higher speedup. Next, the CM dispatches tasks dynamically to all cores. The tasks include information about the operation to be executed as well as pointers to the input and output data. Every PE informs the CM after a task has finished. Additionally, the CM estimates the task runtime to configure the DVFS performance levels. PEs executing tasks with comparatively shorter runtimes can run at a lower frequency than PEs processing on longer-lasting tasks. The acceptable reduction of the supply voltage and the throughput, respectively, leads to a decreased energy consumption. Hence, the CM defines the trade-off between speed and power. Finally, the processed data remains in the PEs to be reused for subsequent queries (data locality), is passed among the PEs, or can be transferred back to the global memory.

2.4 Processing Elements

Each PE consists of a Tensilica RISC core with database-specific instructions (Database Accelerator, DBA), two dual-port DMEMs and a dual-port IMEM (details in Table 2). The memories enable the access from both core and NoC side. The core has two 128-bit data interfaces and two load/store units. Hence, it provides SIMD capabilities on 128-bit registers and a simultaneous access to both data memories, respectively. The IMEM interface exhibits a width of 64 bit to allow VLIW features, i.e., the core fetches and executes up to three instructions within one clock cycle. The block diagram is shown in Figure 3.

Further, each PE includes a DMA controller (DMAC), controlled by the CM, responsible for data and task management. During task execution, the DMAC allows data prefetching for consecutive tasks. Hence, data can be transferred from the off-chip memory to the local data memories. This ensures a continuous data availability while operating on high cardinalities of the data sets which would exceed the sizes of the local memories.

The power management controller (PMC), configured by the CM as well, enables ultra-fast per-core DVFS with supply level switching times of less than 50 ns [9]. The low switching times shorten the time the core has to remain in the off-state which arises from the necessary power precharge scheme. In Tomahawk3, each PE domain is connected to two power rails (VDD1, VDD2) variable in the range between 0.6 – 1.1 V. Both supply rail voltages are controlled by an AVS scheme [14] based on hardware performance monitors (HPMs). The closed AVS loop slowly tracks process and temperature variations during system operation and allows for operation at the minimum required levels of VDD1 and VDD2 for timing error free operation with highest energy efficiency.

The identified several potential and performance critical database operators according to their importance and complexity in query processing for acceleration. In particular, we implemented instructions for sorted-set operations, sorting, hashing operators, searching, and bitmap operations. Details about instructions on hashing and sorted-set operations are presented in our previous publications [2, 3].

An overview about popular application workloads and their sizes of the local memories.

The power management controller (PMC), configured by the CM as well, enables ultra-fast per-core DVFS with supply level switching times of less than 50 ns [9]. The low switching times shorten the time the core has to remain in the off-state which arises from the necessary power precharge scheme. In Tomahawk3, each PE domain is connected to two power rails (VDD1, VDD2) variable in the range between 0.6 – 1.1 V. Both supply rail voltages are controlled by an AVS scheme [14] based on hardware performance monitors (HPMs). The closed AVS loop slowly tracks process and temperature variations during system operation and allows for operation at the minimum required levels of VDD1 and VDD2 for timing error free operation with highest energy efficiency.

We identified several potential and performance critical database operators according to their importance and complexity in query processing for acceleration. In particular, we implemented instructions for sorted-set operations, sorting, hashing operators, searching, and bitmap operations. Details about instructions on hashing and sorted-set operations are presented in our previous publications [2, 3].

An overview about popular application workloads and their need in database systems is provided in [11]. For developing the extended instruction set, we obtain, in a first step, information about time consuming parts in the algorithms by profiling the C/C++ code. Afterwards, the hot spots are replaced by additional instructions forming a complete functional database hardware unit for each algorithm.

Figure 4 shows the first four stages of the 5-stage RISC pipeline and the developed application-specific hardware units operating next to the ALU within the execute stage. The functional units include the customized instructions which are composed of load, process, and store instructions. They are connected by 128-bit wide interfaces to the data memory to allow performing single instructions on multiple data (SIMD).

Figure 4 also depicts one detailed instruction including the pipelined data path incorporated into the functional database unit to perform a logical operation (AND, OR, XOR) on compressed bitmaps. The before applied com-

### Table 1: CoreManager specification.

<table>
<thead>
<tr>
<th>Component</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>Tensilica LX5 RISC with extended instruction set for query processing</td>
</tr>
<tr>
<td>Local Memory Area</td>
<td>64 KB DMEM, 32 KB IMEM</td>
</tr>
<tr>
<td>Max. clock frequency</td>
<td>2.33 MTasks/s</td>
</tr>
<tr>
<td>Peak Performance</td>
<td>667 MHz at VDD = 1.1 V</td>
</tr>
<tr>
<td>Power consumption</td>
<td>42.6 mW at 667 MHz</td>
</tr>
</tbody>
</table>

1) Averaged over several benchmarks.

### Table 2: Processing Element specification.

<table>
<thead>
<tr>
<th>Component</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core</td>
<td>Tensilica LX5 RISC with extended instruction set for database operators</td>
</tr>
<tr>
<td>Local Memory Area</td>
<td>2x 32 KB DMEM, 32 KB IMEM</td>
</tr>
<tr>
<td>Max. clock frequency</td>
<td>255 GOPS1</td>
</tr>
<tr>
<td>Peak performance</td>
<td>61.1 mW at 500 MHz</td>
</tr>
</tbody>
</table>

1) For arithmetic logic operations per PE.

2) Averaged over several algorithms.
The actual bitmap operation (chosen by BitOp_Sel) is performed on the pipelined data path. This results in a 4-fold SIMD approach. Each composed of four 32-bit code words, are loaded from the local data memory and represent the input data for the operation. The pipelined data path includes a code word detection, the actual bitmap operation (chosen by BitOp_Sel), a code word composition, and a controller which ensures storing the available output vector Bitmap_2.

We exploit the 64-bit interface to the instruction memory of our DBA by fetching up to three instructions simultaneously. For instance, it allows to execute two load as well as one process instructions within one clock cycle. The load instructions provide access to the data of the two local data memories while the process instruction operates on the data (cf. bitmap operation in Figure 4). Apart from the RISC pipeline, both instructions perform in an additional pipelined fashion.

3. EXPERIMENTS

In this section we firstly introduce our measurement set-up. Next, we consider the execution time of the selected algorithms and the consumed power of a single processing element. Finally, we evaluate the results on system-level with two typical database benchmarks in Section 3.3.

3.1 Measurement Set-Up

The Tomahawk3 measurement set-up consists of the MPSoC module with two SDRAMs, situated on the power supply board, and a Xilinx Virtex7 FPGA evaluation board as shown in Figure 5. The FPGA incorporates the IP stack for communication via Ethernet between the MPSoC and the host-PC. The host-PC is used to configure the chip and to start the applications. We determine the execution time with hardware performance counters integrated in the PEs. The power consumption is measured via a PC oscilloscope.

3.2 Single-Core Performance

The DBA core contains several functional units as described in Section 2.4. In a first run, we measure the time and power consumption of selected database operators with different data widths. For these tests, we execute the algorithms on a single PE with (DBA) and without (RISC) additional instructions. The single-core performance demonstrates the potential of the extended instruction set. The input data is randomly generated and fits into the local data memories. We choose the maximum frequency of 500 MHz which can be achieved at a supply voltage of 1.1 V. Table 3 summarizes the results.

The intersection is one well-known algorithm to perform set operations on sorted lists containing 16-bit or 32-bit values in our case. The result set includes all list elements that appear in both input sets. The preceding sorting step is realized by a merge sort algorithm. The speedups are achieved by a 4-fold SIMD approach and a simultaneous loading and storing of the input and output sets by taking the advantage of two separated local data memories.

Our database-specific instruction set includes a sampling method and a bit extraction operator performing on 32-bit values. Sampling is applied to determine a hash function dedicated to the input data set. The bit extraction operator is permanently used while mapping values into hash tables. The customized instructions lead to a huge speedup of more than 1000x due to the expensive bitwise operations such as bit selections or partial bit shifts in a pure C/C++ code. Additionally and in contrast to the sorting algorithm which needs several accesses per value, the hashing operators load every value only once.

We implemented instructions for the equality search operator. The scan-like operation compares all values of the given data set with a reference element. The result of a comparison of two values is represented by one bit. Greater data widths lead to higher throughputs of the RISC implementation due to less memory accesses. The DBA uses the extended instructions indicating a constant number of memory accesses that results in independent throughputs for different data widths and SIMD factors, respectively.

The ten additional database units, included in each DBA, consume 76.4% of the core. The added comparators and multipliers performing on 128-bit wide instruction-specific registers used by every database operator mainly increase the size of the core area. In general, instructions for algorithms using smaller data widths lead to greater area consumptions due to an increased parallelization degree and thus to more active logical cells. In contrast, the instructions for the equality search operation include counter registers with increasing sizes for greater data widths.

Generally, the usage of the functional units increases the power consumption by an average of 17%. However, we always obtain a higher energy efficiency due to a further increased throughput. The DBA shows speedups from 8x to...
Table 3: Comparison of functional database units measured on a single DBA core.

![Graph showing energy per clock cycle over varying clock frequencies for selected supply voltages.](image)

(a) Energy per clock cycle over varying clock frequencies for selected supply voltages normalized to energy consumption at 500 MHz, $V_{DD} = 1.1$ V.

1) Relative Area Consumption of the functional database units regarding to the complete DBA processor.

The first benchmark is a scan algorithm which performs the equality search operator on 8-bit integer values. As shown in Table 4, the bandwidth of the LPDDR2 memory of 2x 12.5 Gbit/s limits the total throughput of our Tomahawk3. Hence, as long as the data transfer time is lower than the data processing time, the throughput is defined by the database operators. Additionally, we measured an almost 3x higher power consumption compared to the entire on-chip performance. This is mainly caused by the DRAM controllers which consume almost 40% of the total power. In addition due to a more intensive data transfer, the DMA controllers included in each PE as well as the NoC increase the power consumption. Psaroudakis et al. [15] run the same benchmark on two 8-core Intel Xeon E5-2690 processors with 32 threads in parallel. They identified the best energy efficiency at a clock frequency of 2.0 GHz and distinguish between different data sizes: the data fits into the cache, or the data is stored in the DDR3 DRAM. For the former, we obtain an up to 96x higher energy efficiency even though the Intel doubles its performance while operating on the cache-conscious data set. When comparing the DRAM approaches, our Tomahawk3 still consumes 10x less energy. The two less advanced quad core Intel Xeon E5430 processors, equipped with DDR2 memory interfaces, with only eight parallel threads used by Tsirogiannis et al. [19] show the worst performance for the scan benchmark. In comparison, the Tomahawk3 achieves a 5.6x higher performance and a more than 1400x better energy efficiency when the data is initially located in the global memories.

The second benchmark, WAH indexing, processes an essential query such as “SELECT * WHERE data=const” as in [6]. The four DBAs perform an equality search on $n$ 8-bit integer values. The resulting bitmap index is a binary matrix with $n$ rows and $2^n = 256$ columns representing the 256 distinct values of an 8-bit number. This is followed by 256 column-wise WAH compressions of $n$ bits. Fusco et al. [6] run this benchmark on an NVIDIA GTX-670 and on an Intel i7-2600K. The CPU uses a single thread per core whereas the GPU operates with 1344 hardware threads. In comparison with Tomahawk3, the Intel is 7.5x and the GPU is 1.8x slower despite of a 336x higher parallelism of the cores. The cores run at their maximum clock frequency at a supply voltage of 1.1 V.

The second benchmark, WAH indexing, processes an essential query such as “SELECT * WHERE data=const” as in [6]. The four DBAs perform an equality search on $n$ 8-bit integer values. The resulting bitmap index is a binary matrix with $n$ rows and $2^n = 256$ columns representing the 256 distinct values of an 8-bit number. This is followed by 256 column-wise WAH compressions of $n$ bits. Fusco et al. [6] run this benchmark on an NVIDIA GTX-670 and on an Intel i7-2600K. The CPU uses a single thread per core whereas the GPU operates with 1344 hardware threads. In comparison with Tomahawk3, the Intel is 7.5x and the GPU is 1.8x slower despite of a 336x higher parallelism of the
GPU. Power measurements are not provided by the reference.

4. CONCLUSION

High throughputs and low latencies are two main requirements of query processing in a database system. Today’s modern computer architectures mostly keep up with the required performances but do not address the major issue – energy efficiency. Various research is focused on application-specific hardware in customized processors as well as advanced scheduling algorithms for parallelization in general-purpose CPUs to overcome this problem.

In this paper, we present an MPSoC to accelerate query processing in database systems by combining RISC processors extended with a database-specific instruction set as well as a dynamic task scheduling unit to enable system performance scalability. It includes a hierarchical power management composed of AVS and ultra-fast DVFS to minimize the power consumption for selected database queries to only several 100 mW. In summary for a scan benchmark, the MPSoC shows a 96x higher energy efficiency compared to an Intel Xeon processor.

5. ACKNOWLEDGMENTS

This work has been supported in part by the state of Saxony under grant of ESF 100098198 (IMData), the German Research Foundation (DFG) via EXC1056, cfaed and via ESF 100098198 (IMData), the German Research Foundation (DFG) via EXC1056, cfaed and via ESF 100098198 (IMData). Furthermore, we would like to thank Synopsys and Cadence for providing software and IP.

6. REFERENCES


Table 4: System level benchmark comparisons.