The Bit-Nibble-Byte MicroEngine (BnB) for Efficient Computing on Short Data

Dilip Vasudevan  
Department of Computer Science  
University of Chicago  
Chicago, IL, USA  
dilipv@cs.uchicago.edu

Andrew A. Chien  
Department of Computer Science  
University of Chicago  
Chicago, IL, USA  
achien@cs.uchicago.edu

ABSTRACT
Energy is a critical challenge in computing performance. Due to "word size creep" from modern CPUs are inefficient for short-data element processing. We propose and evaluate a new microarchitecture called "Bit-Nibble-Byte" (BnB). We describe our design which includes both long fixed point vectors and as well as novel variable length instructions. Together, these features provide energy and performance benefits on a wide range of applications. We evaluate BnB with a detailed design of 5 vector sizes (128,256,512,1024,2048) mapped into 32nm and 7nm transistor technologies, and in combination with a variety of memory systems (DDR3 and HMC). The evaluation is based on both handwritten and compiled code with a custom compiler built for BnB. Our results include significant performance (19x-252x) and energy benefits (5.6x-140.7x) for short bit-field operations typically assumed to require hardwired accelerators and large-scale applications with compiled code.

Categories and Subject Descriptors
C.1.2 [Processor Architectures]: Multiple Data Stream Architectures (Multiprocessors); C.1.3 [Processor Architectures]: Other Architecture Styles—Heterogeneous (hybrid) systems, Parallel Processors

Keywords
Heterogeneous Computing, SIMD, DDR3, Hybrid Memory Cube, Microengines

1. INTRODUCTION
With the current scaling of technology nodes, energy dissipation has become the prime factor compared to the performance. To address this challenge, several different architectures are proposed exploiting the data-level and instruction level parallelism. Multicore system design with homogeneous architecture was proposed to address this challenge. But due to the general purpose nature of these cores, several applications still incur higher overhead in terms of performance and energy attributed by the instruction issue cost than the actual workload computation. Heterogeneous computing architecture addresses this challenge by designing architectures with multiple heterogeneous cores tailored to specific application.10x10 paradigm [1] is a systematic approach to heterogeneity in computer architecture design. As a first step towards this goal, this paper introduces one of the microengine named Bit-Nibble-Byte Microengine which constitutes one of the microengine of the 10x10 core. Motivation: Many applications require short fixed point or bit-level manipulations, but such operations (like shifting, rotation, shuffling and logical operations) are often not well supported on modern architectures. For traditional RISC ISA’s, even extended by multimedia extensions, these operations incur high instruction count, runtime, and energy overheads. This challenge motivates us to implement efficient bit level operations supporting several data sizes and better instructions and intrinsics to carry out these operations. Thus we propose efficient an ISA with larger vector size and improved instructions to implement the algorithms efficiently: 3D Stacked memory architectures are becoming very popular recently [3],[4]. Given that the proposed micro-engine will require higher bandwidth of data, performance and energy analysis of the BnB MicroEngine with a three-dimensional (3D) stacked memory system is also experimented.

Contributions: The contributions of this paper are as outlined below: 1) Design of BnB, a SIMD instruction set extension and micro-architecture with a scalar and vector (sizes ranging 128,256,512,1024 and 2048) data path that incorporate novel instructions for short data calculations with high efficiency, 2) A detailed empirical study on the impact of the several vector sizes of the BnB micro-engine over Performance and Energy for 6 applications namely GF Multiplication, AES Encryption, AES Decryption, RLE (Run-Length Encoding), 2D convolution and DWT (Discrete Wavelet Transform) is shown, 3) Detailed RTL and Gate Level Implementation of BnB, and evaluation of the performance and energy efficiency on a set of benchmark kernels, using a 32nm and 7nm CMOS process, 4) Integration of BnB microarchitecture implemented at Gate level (Hardware model) with two different memory systems namely DDR and Stacked Memory (software model) which provides knowledge on impact of the change in the memory system model over the performance and energy of the whole system, 5) Demon-
strated the flexibility, programmability of BnB intrinsics and actual C code compiled for the BnB using a BnB compiler over 6 application, 6) Empirical results of running full system models namely BnB-32nm-DDR, BnB-7nm-DDR, BnB-32nm-HMC, BnB-7nm-HMC over 5 different vector sizes (128,256,512,1024,2048) compared with scalar showing clearly how BnB MicroEngine achieves performance and power efficiency by improving 3 aspects of system design namely micro-architectural/ISA enhancement, Memory system upgrade and technology scaling.

The rest of the paper is organized as follows. In section 2, BNB microarchitecture is introduced. In Section 3, the ISA and intrinsics of BnB are proposed. In Section 4, the performance and Energy results of the BnB Micro-engine are presented and compared with the results from scalar version for DDR3 and HMC memory hierarchy. Future work and summary is provided in section 5.

2. BNB ARCHITECTURE

A 10x10 architecture (Figure 1) exploits deep workload analysis to drive co-design of a federated heterogeneous architecture that exploits customization for energy efficiency, but federates a set of customized engines to achieve general-purpose coverage. The BnB micro-engine is one of the 10x10 micro-engines proposed to accelerate short data type based applications.

2.1 BnB Features

BnB’s ISA includes sixteen 32-bit general purpose registers, sixteen 2048-bit vector registers. The vector registers are 16x wider than the mainstream 128-bit wide SSE. For table lookup operation, with 128-bit vectors, 16 elements of size 8 bit can be indexed. BnB’s 2048 bit vector allows 256 elements of size 8 bits, a 16x table lookup at each instruction. Further, while SSE limits element size to 8, 16, 32 bits for most of the instructions, BnB supports 4, 8, 16, 32, 64, 128 bits. Benefits obtained by the proposed BnB architecture over SSE is that there is a 10x improvement obtained by the increase in the vector register size. And novel custom instructions for bit operations provide another 3.75 improvement for a total of 37.5x instruction count improvement. The size of 2048 bits is especially chosen for the reason that this size is sufficient for the applications targeted. For example this size matches the size of the SBOX (256 bytes) in AES algorithm.

2.2 BNB ISA

The BnB microengine is a 6-stage pipelined architecture, comprising Prefetch (PF), Instruction Fetch (IF), Decode (DE), Execute (EX), Memory (MEM) and Writeback (WB). All the new BnB instructions execute in the execute stage as shown in 2. Figure 2 shows the block diagram for BnB with memory hierarchy. It uses 2 level cache and 2GB memory. The memory hierarchy was modeled as either DDR or HMC. Each BnB instruction and accompanying intrinsic (for direct software access) is shown in the Table. 1. The columns are: intrinsic definition, machine instruction, and description of the functionality.

3. METHODOLOGY AND EXPERIMENT

Design Configurations: BNB micro-architecture was implemented using the Synopsys CAD flow. For cache and memory simulation, DRAMSim2 [6], Cacti [2] and MARSSim [5] are used. We integrate the Synopsys tool flow with the memory hierarchy simulation tool to do full system online simulation. First the C++ model of the micro-engine was generated for functional simulation and then the RTL of the designed microarchitecture was synthesized to gate level using this CAD flow. Two models are evaluated namely Baseline and BnB. All these models are synthesized to gate-level designs using 32nm and 7nm technology and totally 4 models are evaluated namely BnB-DDR-32nm, BnB-DDR-7nm, BnB-HMC-32nm, BnB-HMC-7nm for performance, and energy metrics. Table 2 shows the configuration of the system implemented. Benchmarks were written in ANSI C + intrinsics, and compiled using a custom compiler generated for the BnB MicroEngine based on the BnB architecture description. Thus any benchmark written in ANSI C language is supported. Benchmarks (short data, and long vectors): Benchmarks used for the experiments are GF Multiplication, RLE, AES, 2D Convolution and DWT. For GF multiplication, RLE and AES the input size is 128KB. For 2D convolution and DWT the input size is 640x480 Byte size pixels. For BnB designs, vectorized code of these benchmarks were implemented using BnB intrinsics and large vector data types are compiled using the BnB compiler and run on the BnB microengines.

4. EXPERIMENTAL RESULTS

4.1 Performance

Performance metrics is detailed in this section by evaluating the instruction count, cycle count and execution time of the BnB designs against scalar version.
Table 1: BnB ISA

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>shuffle_v2k(dst,src,size)</td>
<td>SHUFFLE2K Do shuffle between two 2048 bit vectors (size = element size to shuffle)</td>
</tr>
<tr>
<td>v2k(dst,src)</td>
<td>VXOR Do bitwise XOR between dst, src and store result in dst</td>
</tr>
<tr>
<td>v2k(dst,src)</td>
<td>VAND Do bitwise AND between dst, src and store result in dst</td>
</tr>
<tr>
<td>v2k(dst,src)</td>
<td>VOR Do bitwise OR between dst, src and store result in dst</td>
</tr>
<tr>
<td>v2k(dst,src)</td>
<td>VMUL Do 64 integer multiplication of src and dst and store result in dst</td>
</tr>
<tr>
<td>v2k(dst,src)</td>
<td>VADD Do 64 integer addition of src and dst and store result in dst</td>
</tr>
<tr>
<td>v2k(dst,src)</td>
<td>VSUB Do 64 integer subtraction of src and dst and store result in dst</td>
</tr>
<tr>
<td>shift_v2k(dst,src)</td>
<td>VSHIFT Shift right each element of size &quot;size&quot; in &quot;dst&quot; &quot;count&quot; times</td>
</tr>
<tr>
<td>vcntlz(dst,src)</td>
<td>VCNTLZ Count number of leading zeros in src and store the count in dst</td>
</tr>
<tr>
<td>svxor(dst,src,size)</td>
<td>SVXOR Do XOR of &quot;src&quot; over each element of &quot;dst&quot; vector of size 32</td>
</tr>
<tr>
<td>rvsmov(dst,src,index)</td>
<td>RVSMOV Store value in “src” register indexed by “index” in “dst” vector.</td>
</tr>
<tr>
<td>vrsmov(dst,src,index)</td>
<td>VRSMOV Store 32 bit indexed by “index” in “src” vector to “dst” register.</td>
</tr>
<tr>
<td>lm2vec(dst,src), vec2lm(det,src)</td>
<td>L M2VR, Move data to and from local memoryVR2LM</td>
</tr>
</tbody>
</table>

Table 2: Experimental System - Configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core type</td>
<td>In-order, 6-stage pipeline</td>
</tr>
<tr>
<td>ISA</td>
<td>MIPS-like ISA</td>
</tr>
<tr>
<td>Vector register file</td>
<td>2048b x 16 registers</td>
</tr>
<tr>
<td>Cache hierarchy</td>
<td>L1-I: 32KB, 2-cyc latency, L1-D: 24KB, 2-cyc latency, Shared L2: 512KB, 10-cyc latency</td>
</tr>
<tr>
<td>Main memory DDR3 (Model I)</td>
<td>2GB/4-rank/16-device DDR3</td>
</tr>
<tr>
<td>Main memory HMC (Model II)</td>
<td>4GB/4-rank/8-device</td>
</tr>
<tr>
<td>Scratch Pad/Local Memory</td>
<td>4 MB</td>
</tr>
</tbody>
</table>

Instruction Count: Figure 3.a shows the comparison of relative instruction count for the six applications mentioned earlier. For GF, with input size of 128KB, scalar required as much as 45 million instructions and BnB required only 512K - 565K instructions. For GF, 79.5X - 87.75X reduction in instruction count was noticed. It should be noted that the instruction count for GF multiplication for all vector sizes are in the range around 550K. The reason for the minor change in the count is that the data movement to the vector accounted for more instructions compared to the actual multiplication operations. Alternatively, for the convolution, the effect of vector size is evident from the fact that the instruction count reduction increases from 25x to 237x, when we move from vector size of 128 to 2048 bits. For all the 6 applications instruction count reduction was between 9-237x.

Figure 3: Relative Instruction Count Reduction

Cycle Count/Execution Time: Since we are using multi cycle instructions, the execution time varies compared to instruction count. The performance results for the 6 applications are shown in Figure 4 in ms. For the 2D-Convolution and DWT, the bars pass above the graph scale and the actual values are printed next to the bars in “ms”. With the technology scaling, the execution time is improved as 32nm systems run at 1GHz and 7nm systems are running at 4GHz. Most of the benefit on cycle count reduction for convolution and DWT comes by operating on the image pixels stored in local memory. From Figure 4, for 2d convolution, scalar version in 32nm technology took around 830 ms and the fastest BnB design was “BnB-2048” executing 2d convolution on a 640x480 pixel image in 3.3 ms. For AES-ENC and AES-DEC, it should be noted that the execution time for scalar, BnB-128,BnB-256,BnB-512 and BnB 1024 are same as they run scalar code.

Figure 4: Performance in "ms"

Speedup: The relative performance speedup is shown in Figure 5 ranges from 19 - 248x for BnB designs with DDR3 memory and 20 - 252 x for the BnB designs with HMC memory hierarchy. Convolution achieved the highest speed up of 248x for DDR3 based design and 252x for HMC based designs. For GF application, BnB with 256 bit vector size had highest speedup of 115x for DDR3 based system and 130x for the HMC based system. For AES encoding and decoding BnB-2048 bit system achieved highest benefit of 19x and 21x for DDR3 and HMC based systems respectively. Cycle count reduction due to the computation on data from local memory and the large vector size had high impact on the speedup of AES algorithms. For DWT and RLE the speed up was scaling with vector size with 21x-41x and 7x-61x respectively.

Figure 5: Relative Performance/cycle count
4.2 Energy

System Energy: The component level energy distribution of the scalar and BnB of different vector sizes is shown in Figure 6.

Operating on the data from the local memory and wider vector datapath influenced the reduction in DRAM energy. For the convolution application, the energy is scaling linearly with the increase in vector size.

However, for the DWT application, the energy was scaling up to the vector size 1024 and for BnB 2048, the energy increased. This is due to the fact that, the balance between larger instruction count reduction with wider vector datapath (dissipating higher power) reaches the equilibrium. Further improvement with BnB-2048 design can be achieved if this data movement is carried out efficiently. Similar trend is seen for the RLE and GF. Nevertheless, there are applications for which most of the computation are vectorized and the energy benefit scales with vector size. An example for this case is the AES application. For AES, the size of S-Box used in the AES algorithm is suitable to the vector size BnB 2048, which is 256 Bytes. As shown in last two graphs of Figure. 6, the energy benefit is higher for BnB-2048 compared to other sizes. The reason for this benefit is of two folds. First, BnB-2048 runs vectorized code and second is that lower vector size designs run the scalar version of the AES.

Energy Benefit: The relative energy benefit of BnB with different vector sizes over scalar version is shown in Figure 7. It ranges from 7.7x-140.7 x for BnB designs with DDR3 memory and 5.6x - 138.7 x for the BnB designs with HMC memory hierarchy. Convolution achieved the highest benefit of 140.69 x for DDR3 based design and 129x for the HMC based designs. For GF application, BnB with 257 bit vector size had highest speedup of 126x for DDR3 based system and 5.6x-138.7 x for the BnB designs with HMC memory hierarchy. Next step is to do physical design of the BnB microarchitecture at 32nm and obtained the energy and performance results at layout level.

5. SUMMARY AND FUTURE WORK

A new microarchitecture for accelerating the lower bit-level operations was proposed in this paper. With the Gate level BnB model integrated with DDR and HMC memory hierarchy, BnB achieved performance benefit ranging from 19x-248x for designs with DDR3 and 20-252 x for the BnB designs with HMC compared to the baseline RISC core with similar memory hierarchies. Energy benefit achieved was from 7.7x-140.7 x for BnB designs with DDR3 memory and 5.6x-138.7 x for the BnB designs with HMC memory hierarchy. Next step is to do physical design of the BnB microarchitecture at 32nm and obtained the energy and performance results at layout level.

6. ACKNOWLEDGMENTS

This work is being supported in part by the Defense Advanced Research Projects Agency under award HR0011- 13-2-0014 and the generous Synopsys Academic program.

7. REFERENCES