Cell Broadband Engine

- 1 PowerPC core
  - Based on the PowerPC 970
  - 128-bit AltiVec/VMX SIMD unit
- Currently up to 8 “synergistic processors”
- Runs at ~3.2 GHz

- A Core2 core has three 128-bit SIMD units with just 16 registers.
Running DES on the Cell

- Bitsliced implementation of DES
  - 128-way parallelism per SPU
  - S-boxes optimized for SPU instruction set
- 4 Gbit/sec = $2^{26}$ blocks/sec per SPU
- 32 Gbit/sec per Cell chip
- Can be used as a cryptographic accelerator (ECB, CTR, many CBC streams)
Breaking DES on the Cell

• Reduce the DES encryption from 16 rounds to the equivalent of ~9.5 rounds, by shortcircuit evaluation and early aborts.

• Performance:
  - $108M=2^{26.69}$ keys/sec per SPU
  - $864M=2^{29.69}$ keys/sec per Cell chip
Comparison to FPGA

Expected time to break:

• COPACOBANA
  - ~9 days
  - €8,980
  - A year to build

• 52 PlayStation 3 consoles
  - ~9 days
  - €19,500 (at US$500 each)
  - Off-the-shelf

• Divide by two if you get $E_K(X)$ and $E_K(\overline{X})$. 
DreamHack 2004 LAN Party
5852 connected computers
Under 1 hour for a real-time DES break.
Synergistic Processing Unit

- 256KB of fast local memory
- 128-bit, 128-register SIMD
- Two pipelines
- In-order execution
- Explicit DMA to RAM or other SPUs
SPU memory

• Single-ported

• 6-cycle load-to-use latency

• Read or write 16 or 128 bytes each cycle

• DMA & instruction fetch use 128-byte interface

• Prioritized: DMA > load/store > instruction fetch
SPU registers

- 128 registers
- Up to 77 register parameters and return values according to calling convention
SPU instruction set

- RISC (similar to PowerPC)
- Fixed 32-bit size
- Always aligned on 4-byte boundary
- Most operations are SIMD
## SPU pipelines and latencies

<table>
<thead>
<tr>
<th>Unit</th>
<th>Instructions</th>
<th>Execution Pipe</th>
<th>Unit Pipeline Depth</th>
<th>Instruction Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple Fixed</td>
<td>word arithmetic, logicals, count leading zeros, selects, and compares</td>
<td>Even</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Simple Fixed</td>
<td>word shifts and rotates</td>
<td>Even</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Single Precision</td>
<td>multiply-accumulate</td>
<td>Even</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Single Precision</td>
<td>integer multiply-accumulate</td>
<td>Even</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Byte</td>
<td>pop count, absolute sum of differences, byte average, byte sum</td>
<td>Even</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Permute</td>
<td>Quadword shifts, rotates, gathers, shuffles as well as reciprocal estimate</td>
<td>Odd</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Local Store</td>
<td>Load and store</td>
<td>Odd</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Channel</td>
<td>Channel Read/Write</td>
<td>Odd</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Branch</td>
<td>Branches</td>
<td>Odd</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
SPU limitations

• Fetches 8-byte aligned pairs of instructions
  - Dual issue happens only if first is even-pipe instruction and second is odd-pipe instruction
• Only 16x16->32 integer multiplication
• No hardware branch prediction
Special SPU instructions

- select bits
- gather bits
- carry/borrow generate
- sum bytes
- generate controls for insertion
- shuffle bytes
- form select mask
- add/sub extended
- or across
- count leading zeros
- count ones in bytes
64-bit addition

• 2-way SIMD:
  - carry generate
  - add
  - shuffle bytes
  - add

• 4-way SIMD:
  - carry generate
  - add
  - add extended
64-bit rotate

• 2-way SIMD:
  - rotate words
  - shuffle bytes
  - select bits

• 4-way SIMD:
  - 2 * rotate words
  - 2 * select bits
selb

• Bitwise version of “a = b ? c : d”

• Also known as a multiplexer (mux)

• Very useful for bitslice computations
  - DES S-box average less than 40 instructions
  - Matthew Kwan: 51, without using selb
## Comparison to Core2 for bitslice

<table>
<thead>
<tr>
<th></th>
<th>CPU</th>
<th>SPU</th>
<th>Core2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registers</td>
<td></td>
<td>128</td>
<td>16</td>
</tr>
<tr>
<td>Register width</td>
<td></td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>Registers/instruction</td>
<td>3</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Boolean operations</td>
<td>*+select</td>
<td>and, or, xor, andn</td>
<td></td>
</tr>
<tr>
<td>Instruction parallelism</td>
<td>1</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>Cores per chip</td>
<td>6-8</td>
<td></td>
<td>2-4</td>
</tr>
</tbody>
</table>
shufb

• Concatenate two input registers to form a 32-byte lookup table

• Each byte in the third register selects either a constant value (0x00/0x80/0xFF) or a location in the lookup table

• => 16 table lookups per cycle
AES Table lookups in registers

• 5->8 bit lookups directly supported by shufb

• For the remaining 3 input bits we need to isolate and replicate them, and then use selb to select between 8 different shufb outputs

• High latency, but also high throughput with 4-way interleaving
Cache attack resistance

• SPUs currently immune
  - no address-dependent variability in memory access

• Architecture allows cache in SPU

• In-register lookups should be future-proof
Branch prediction

- Calculate branch address
- Give branch target hint
- ...
- Branch without penalty
Optimization summary

• Do vector (SIMD) processing
• Large number of registers allows interleaving several computations, hiding latencies
• Balance pipeline usage
• Pre-compute branches in time to give hint
• For very memory-intensive code, ensure instruction fetch by using hbrp
Running MD5 on the Cell

- 32-bit addition and rotation, boolean functions
  - Directly supported with 4-way SIMD
  - Bitslice is slow: 128 adds require 94 instructions

- Many streams in parallel hide latencies

- Calculated compression function performance: Up to 15.6 Gbit/s per SPU
Running AES on the Cell

• > 2.1 Gbit/s per SPU (~3.8 GHz Pentium 4)
• ~17 Gbit/s for full Cell, almost 13 Gbit/s for PS3
• CBC implementation only a little slower.
• Bitslice would be very interesting
Other cryptographic applications for the Cell Broadband Engine

• Limited by SPU microarchitecture and memory
• Good match for low-memory, straight-path computation over small operands
• Some promising applications:
  - Stream cipher cryptanalysis
  - Sieving for the Number Field Sieve
  - Hash collisions
The future of the Cell

- More SPUs on a chip
- Internal cache in SPUs
- Fast double precision float
- Different size of local memory?
- New instructions?