Sixth ACM-IEEE International Conference on Formal Methods and Models for Codesign
(MEMOCODE'2008)
June 5-7, Anaheim, CA, USA

MEMOCODE ‘08
Design Contest - Submitted by
Nanyang Technological University, NTU
Singapore

Resources:

- Performance Normalization Factor : 1.0
- Xilinx XUP Virtex™-II Pro Development System
- 100MHz System Clock
- 300MHz PPC Clock
- EDK 9.1
- ISE 9.1, ChipScope Pro 9.1i, ModelSIM
- + 2 weeks of Design/System Exploration
- + 2 weeks of Implementation
Result:

```
<table>
<thead>
<tr>
<th>Case</th>
<th>Elapsed</th>
<th>Speed up</th>
<th>Correct</th>
</tr>
</thead>
<tbody>
<tr>
<td>(rand, pwr= 6)</td>
<td>193</td>
<td>27.145079</td>
<td>correct</td>
</tr>
<tr>
<td>(rot,  pwr= 6)</td>
<td>201</td>
<td>28.427860</td>
<td>correct</td>
</tr>
<tr>
<td>(rand, pwr= 10)</td>
<td>4520</td>
<td>39.267700</td>
<td>correct</td>
</tr>
<tr>
<td>(rot,  pwr= 10)</td>
<td>4377</td>
<td>40.223896</td>
<td>correct</td>
</tr>
<tr>
<td>(rand, pwr=14)</td>
<td>94300</td>
<td>46.767433</td>
<td>correct</td>
</tr>
<tr>
<td>(rot,  pwr=14)</td>
<td>70858</td>
<td>60.684029</td>
<td>correct</td>
</tr>
<tr>
<td>(rand, pwr=18)</td>
<td>1960018</td>
<td>49.089465</td>
<td>correct</td>
</tr>
<tr>
<td>(rot,  pwr=18)</td>
<td>2078183</td>
<td>47.133228</td>
<td>correct</td>
</tr>
</tbody>
</table>

Relative Geometric Mean: 41.094964
Sorting completed
```

MAX SPEED UP: 60X, AVERAGE: 41X
Approach:

Reference C code → C-level Optimization

Algorithm Level Exploration → Accelerators modeled in C

Performance analysis → Test Vector

Accelerators modeled in C → Accelerators modeled in HDL

Hardware Simulation ModelSIM → Synthesized HW/SW System

Verification
Concept:
QUIKOM SORT (Combine Quick and Comb Sort)

- Increase System’s Parallelism by Performing Quick Sort first.
- Hardware Comb Sorters sort the partitioned intervals independently
  - Quikom Sort (Pseudo Code): On Power PC

```c
Quikom_sort(l,r)
{
    If (l<r)
    {
      If ((r-l)< COMB_SIZE) Do_HW_Comb_Sort(l,r);
    else {
      j = Partition(l,r);
      Quikom_sort (l,j-1);
      Quikom_sort (j+1,r);
    }
}
```

- Software Quick Sort is accelerated by Hardware AES Core
- Do_HW_Comb_Sort function feeds the data to HW sorters and returns
System Architecture:

**AES Engine:**
- Average 320 ns per AES compared to 2400 ns in SW
- Speed up 8x
- AES Core taken from [www.opencores.org](http://www.opencores.org)

**PPC:**
- Quikom_sort(l, r)
  
  ```c
  Quikom_sort(l, r)
  {
    If (l<r)
    {
      If ((r-l)< COMB_SIZE)
        Do_HW_Comb_Sort(l, r);
      else {
        j = Partition(l, r);
        Quikom_sort (l, j-1);
        Quikom_sort (j+1, r);
      }
    }
  }
  ```

**Comb Sort Co-processor:**
- 4K records local memory
- Sort 4k of records in ~ 17ms
- 2 Sorters in final version
- **Attached PLB-DMA Controller is unstable**
Performance:
FPGA Resources Used:

Comb sort with HW AES:
- 50% SLICES
- 10% BRAM
- **Speedup - 6X**

Quick sort with HW AES:
- 50% SLICES
- 10% BRAM
- **Speedup - 10X**

1 comb sorter:
- 80% of SLICES
- 30% of BRAM
- **Speedup - 35X**

2 comb sorters:
- 100% of SLICES
- 52% of BRAM
- **Speedup - 41X**
Execution Time Distribution:
- Quick Sort’s Partitioning: 40%
- Data Transfer: 30% (10% for during Quick Sort and 20% during Comb Sort)
  - EDK’s DMA controller: Max 160MB/sec
  - When data > 8kB → transfer fails :-/
- Comb Sort: 30%
  - Bigger local memory of Comb-Sorter will reduce Partitioning time
  - Comb Sort Time is almost constant for each MAXRECORD

Investigations and Bottlenecks:
- EDK’s PLB-DMA set-up did not help much
- More Comb Sort Co-Processors would help
- Better Scheduler would help in performance increase
THANK YOU!