Hardware benchmarking for HASH\(^3\)
(for non Hardware designers)

Ingrid Verbauwhede
ingrid.verbauwhede-at-esat.kuleuven.be

K.U.Leuven, COSIC
Computer Security and Industrial Cryptography
www.esat.kuleuven.be/cosic

with input from:
Junfeng Fan, Miroslav Knezevic,
Patrick Schaumont
Slides from: own Course notes, Rabaey’s
Digital Integrated Circuit
Outline

• Goal of hardware design
• What is hardware design?
• What are the different options?
• What are the different contexts?
• How to compare hardware design: benchmark
• Where are we now?
HW - SW continuum

When Hardware design?
When Hardware design?

- Fast
- Small
- Low power
- Security
- (Analog, RF)
HW-SW continuum

<table>
<thead>
<tr>
<th>HW</th>
<th>HW-SW</th>
<th>SW</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASIC</td>
<td>Domain specific</td>
<td>VLIW</td>
</tr>
<tr>
<td>FPGA</td>
<td>DSP</td>
<td>General purpose</td>
</tr>
<tr>
<td></td>
<td>Area efficiency</td>
<td></td>
</tr>
<tr>
<td>High</td>
<td></td>
<td>Intel AES-NI</td>
</tr>
<tr>
<td></td>
<td>Performance/Energy unit</td>
<td>Westmere</td>
</tr>
<tr>
<td></td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>Low</td>
<td>Programmability</td>
<td></td>
</tr>
</tbody>
</table>
Design parameters

• Speed or throughput:
  – Gbits/sec or Mbits/sec/slice
  – Cycles/byte (see D. Bernstein)

• Area:
  – mm² (gate or transistor count)
  – Memory

• Power or energy consumption:
  – Power (Watts) for cooling or transmission (RFID)
  – Energy: battery operated devices

• Security:
  – Side channel resistance: special circuits styles
Power density will increase

Power density too high to keep junctions at low temp

[Author: S. Borkar, Intel]
“Immediate need to add 8MW to prepare for 2007 installs of new systems”
“Need total of 40-50 MW for projected systems by 2011.”
“Numbers just for computers, add 75% for cooling.”
Cooling will require 12,000-15,000 tons of chiller capacity.”

Source: ORNL Oak Ridge National Lab, US Dept. of Energy
Heat and parallelism

Reduce power = reduce WASTE!!

Power (Heat)

\[ P_{\text{mono}} = CV^2f \quad \text{(Watt)} \]

\[ 4 \left( \frac{C}{4} \right) V^2 \left( \frac{f}{4} \right) = P_{\text{mono}}/4 \]

but since \( f \sim V \)

can be even \( P_{\text{mono}}/4^3 \)

TREND: MULTI-CORE!!
What can one do with 1 cm\(^3\)?

*Energy Storage*

<table>
<thead>
<tr>
<th>Type</th>
<th>J/cm(^3)</th>
<th>μW/cm(^3)/year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Micro Fuel cell</td>
<td>3500</td>
<td>110</td>
</tr>
<tr>
<td>Primary battery</td>
<td>2880</td>
<td>90</td>
</tr>
<tr>
<td>Secondary battery</td>
<td>1080</td>
<td>34</td>
</tr>
<tr>
<td>Ultra-capacitor</td>
<td>100</td>
<td>3.2</td>
</tr>
</tbody>
</table>

© J. Rabaey - 06

Power-Intro 20
What is hardware design?
Skiing down a mountain

Translation from spec into RTL (Register Transfer Level, e.g. VHDL, Verilog)

Specification: HASHX

Algorithm Transformations

Memory Transformations and Optimizations

Multi-precision arithmetic

C, C++, block diagram

p pipelining, unrolling

loop merging, compaction

40 bit accumulator

ASIC

FPGA

Retargetable coprocessor

DSP processor

DSP-RISC

GPU
From RTL to tape-out or FPGA

• “Back-end”: VHDL, Verilog, synthesis, FPGA

ASIC → FPGA → Retargetable coprocessor → DSP

DSP Extensions to RISC → RISC, VLIW, CPU

Hardware

Software

• Verilog-VHDL
• Synopsys synthesis
• Cadence place&route
• FPGA download

• C-compilation
• Assembly optimization

System-on-a-chip, system in package
Context 1: ASIC design

Standard cell based design
Semicustom Design Flow

- Design Capture
  - HDL
    - Logic Synthesis
      - Floorplanning
        - Placement
          - Routing
            - Tape-out

- Pre-Layout Simulation
- Post-Layout Simulation
- Circuit Extraction

Design Iteration

Behavioral
- Technology/library/manufacturer input

Structural
- Physical

Timing closure!
Cell-based Design (or standard cells)

Routing channel requirements are reduced by presence of more interconnect layers.
Standard Cell — Example

[Brodersen92]
Standard Cell – The New Generation

Cell-structure hidden under interconnect layers
The “Design Closure” Problem

Iterative Removal of Timing Violations (white lines)

Courtesy Synopsys
Synthesis *together* with Physical Design

- RTL (Timing) Constraints
- Physical Synthesis
- Macromodules
  - Fixed netlists
- Place-and-Route Optimization
- Netlist with Place-and-Route Info
- Technology/library manufacturer input
- Artwork
Benchmark on gate count??

• Gate count (GE) depends on library and tools!

• Definition of one GATE?

• Example:
  – PRESENT[20] contains 1,000 GE in 0.35 \(\mu\)m technology – 53,974 \(\mu\)m\(^2\).
  – PRESENT[20] contains 1,169 GE in 0.25 \(\mu\)m technology – 32,987 \(\mu\)m\(^2\).
  – PRESENT[20] contains 1,075 GE in 0.18 \(\mu\)m technology – 10,403 \(\mu\)m\(^2\).

• Comparison is fair ONLY if the SAME library, SAME tools, and SAME settings are used.
Benchmark on synthesis settings??

• Same VHDL design synthesized with different constraints will result in different performance.
• Benchmark on “area-time” product ??

<table>
<thead>
<tr>
<th>Cipher</th>
<th>Area Constrained</th>
<th>Speed Constrained</th>
<th>Speed-Up</th>
<th>Area Increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>KATAN32</td>
<td>802GE @ 720MHz</td>
<td>853GE @ 2.7GHz</td>
<td>3.75</td>
<td>1.06</td>
</tr>
<tr>
<td>KATAN48</td>
<td>927GE @ 720 MHz</td>
<td>967GE @ 2.7GHz</td>
<td>3.75</td>
<td>1.04</td>
</tr>
<tr>
<td>KATAN64</td>
<td>1054GE @ 720 MHz</td>
<td>1110GE @ 2.7 GHz</td>
<td>3.75</td>
<td>1.05</td>
</tr>
<tr>
<td>KTANTAN32</td>
<td>462GE @ 720 MHz</td>
<td>631GE @ 1.3 GHz</td>
<td>1.80</td>
<td>1.36</td>
</tr>
<tr>
<td>KTANTAN48</td>
<td>588GE @ 720 MHz</td>
<td>674GE @ 1.3 GHz</td>
<td>1.80</td>
<td>1.15</td>
</tr>
<tr>
<td>KTANTAN64</td>
<td>688GE @ 720 MHz</td>
<td>845GE @ 1.3 GHz</td>
<td>1.80</td>
<td>1.23</td>
</tr>
</tbody>
</table>

Note: 2.7GHz is ‘synthesis’ report: NOT FEASIBLE in practice!

[source: M. Knezevic]
Context 2: FPGA design
Late-Binding Implementation

Array-based

- Pre-diffused (Gate Arrays)
- Pre-wired (FPGA's)
Look-up Table Based Logic Cell

```
<table>
<thead>
<tr>
<th>In</th>
<th>Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
</tr>
</tbody>
</table>
```
LUT-Based Logic Cell

Courtesy Xilinx

Xilinx 4000 Series
Not most up to date

Multiplexer Controlled by Configuration Program

Courtesy Xilinx
RAM-based FPGA

Xilinx XC4000ex

Courtesy Xilinx
Xilinx Virtex-II Pro FPGA

- Conexant 3.125Gb Serial
- IBM PowerPC® RISC CPU
- XtremeDSP™
- Synchronous Dual-Port RAM
- SelectIO-Itra™
- SystemIO™ & XCITE™

Tenerife, Hash³ – 28

KULeuven - COSIC

Nov 2009
Multi-Pass Place-and-Route Analysis
GMU SHA-512, Xilinx Virtex 5

100 runs for different placement starting points

No clock requested

~ 20%

The smaller the better

[Courtesy: Kris Gaj]
Dependence of Results on Requested Clock freq.

[courtesy: Kris Gaj]
Saar Drimer, Figure 5.2
Ph.D. thesis
Distribution max achievable clock frequency for Place&Route with 100 different PAR seeds.
1 & 2: for 1 or 4 AES instances
3 & 4: same on different platform
5: different speed grade
FPGA benchmarks??

• Easier than ASIC
  – Tools are (almost) free (at least at universities)
  – Options: similar to software

• Trend getting worse: FPGA becomes heterogeneous machine
  – Report with/without “block-Rams”
  – Report with/without DSP multipliers
  – Report with/without high speed IO
FPGA benchmarks??

• Area numbers:
  – Slices, LUT’s, CLB’s, …
  – Xilinx application engineer: “The number of CLB’s inside LUT’s changes from generation to generation.” (or was it LUT’s inside CLB’s?)

• Speed: accurately reported by tools

• Power:
  – Poorly reporting by tools
  – Hard to measure on board
Context 3: HW-SW interface

Dan would call this the API?
Intro: SHA3-ZOO

• 3 types of “Hardware” reporting, but no interface!

Integration of Hash module??
Integration of the Hash module: options for HW/SW co-design

• Option 1: instruction set extension

  - Tightly coupled
  - Reuse of busses
  - Reuse of registers
  - Define instruction
  - Usually: C-intrinsic or pragma

Example: AES-NI off Intel (see Shay’s presentation!)
Example: Build your own extension to embedded processor
  see e.g. Xtensa or Target Compiler Technologies
• Option 2: Memory mapped

- Memory-mapped coprocessor
- Loosely coupled
- Typical for DSP and other embedded processors
- No need to change compiler
- Check latency of co-processor & memory consistency!
• Option 3: novel forms of co-operation

- Custom HW or Network on Chip (NOC)
- Loosely coupled
- Flexible interconect
- Popular for large multi-core designs (80 or 100 cores)

One of many other cores
Can have different forms in on System-on-chip (SOC)
AES acceleration for SH3-DSP

- AES Co-processor
  - For 128bit key
  - Using GEZEL
  - Communicate with the SH3-DSP ISS via the memory mapped interface

[Ref: Y. Matsuoka et al, CASES04]
AES Optimization results

- Number of lock cycles per AES encryption
  (Key scheduling + Block encryption)
  - Starting from Java function call in user application
  - KNI overhead limits the overall performance gain

[Ref: Y. Matsuoka et al, CASES04]
Context 4: Bandwidth
Adapt HW platform to application

Simple example: Key Schedule for secret key

Two options:
- On the “fly” = just in time processing
- Pre-compute and store in memory

Typical for Hardware

Typical for Software
Key schedule on the fly

- The cost of fast key context switching in SW
- Example for IPSEC router
  - one 128 bit key = 1408 bits round keys (10 rounds + initial key)
  - half of internet packets are only 64 bytes in length (512 bits)

![Data at 1Gbps](chart.png)

Context bandwidth (Gbps)

Record Size (bytes)

[source: J. Goodman]
Benchmark??

Cost of HW module (minimum minimorum):

- **Key storage**
  - assume sub-keys on the fly

- **State storage:**
  - Does all state need to be alive all the time?
  - Wide pipe - narrow pipe
  - Windowing?
  - Think context switching

- **Input block / output block**
  - Can I process input already before the complete input block and/or padding is present?
  - Same for output: can I send output, or do I have to wait for the complete output block
Context 5: gap between application and architecture
Match between algorithm & architecture

Close the gap:
- Dedicated HW: ASIC
- Programmable HW: FPGA
- Custom instructions, hand-coded assembly
- Compiled code
- JAVA on virtual machine, compiled on a real machine

\[ \begin{align*}
\text{Power} & \quad \text{Cost} \\
\text{Application} & \quad \text{ASIC} \\
\text{Platform} & \quad \text{Fixed}
\end{align*} \]
## Throughput – Energy numbers

<table>
<thead>
<tr>
<th></th>
<th>Throughput</th>
<th>Power</th>
<th>Figure of Merit (Gb/s/W = Gb/J)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>0.18μm CMOS</strong></td>
<td>3.84 Gbits/sec</td>
<td>350 mW</td>
<td>11 (1/1)</td>
</tr>
<tr>
<td><strong>FPGA [1]</strong></td>
<td>1.32 Gbit/sec</td>
<td>490 mW</td>
<td>2.7 (1/4)</td>
</tr>
<tr>
<td><strong>ASM StrongARM [2]</strong></td>
<td>31 Mbit/sec</td>
<td>240 mW</td>
<td>0.13 (1/85)</td>
</tr>
<tr>
<td><strong>Asm Pentium III [3]</strong></td>
<td>648 Mbits/sec</td>
<td>41.4 W</td>
<td>0.015 (1/800)</td>
</tr>
<tr>
<td><strong>C Emb. Sparc [4]</strong></td>
<td>133 Kbits/sec</td>
<td>120 mW</td>
<td>0.0011 (1/10.000)</td>
</tr>
<tr>
<td><strong>Java [5] Emb. Sparc</strong></td>
<td>450 bits/sec</td>
<td>120 mW</td>
<td>0.0000037 (1/3.000.000)</td>
</tr>
</tbody>
</table>

[1] Amphion CS5230 on Virtex2 + Xilinx Virtex2 Power Estimator
[4] gcc, 1 mW/MHz @ 120 Mhz Sparc – assumes 0.25 u CMOS
[5] Java on KVM (Sun J2ME, non-JIT) on 1 mW/MHz @ 120 MHz Sparc – assumes 0.25 u CMOS
Context 6: transformations
Data Flow Graph representation

- Illustrate with RIPEMD
- Indicate loops, operations, and delays
Iteration Bound

\[ T_\infty = \max_{l \in L} \left\{ \frac{t_l}{w_l} \right\} \]

- \( t_l \) – loop calculation time
- \( w_l \) – number of algorithmic delays (marked with \( T_D \)) in the \( l \)-th loop
Iteration Bound

\[ T_\infty = \max_{i \in L} \left\{ \frac{t_i}{w_i} \right\} \]

\[ T_\infty = 2 \times \text{Delay}(\oplus) + \text{Delay}(F) + \text{Delay}(\text{rol}) \]
Critical path

- The longest path between any two storage elements.
- $4 \times \text{Delay}(\oplus) + \text{Delay}(\text{rol})$. Determines the clock frequency!
- Problem: Critical Path $> \text{Iteration Bound}$!
Retiming transformation

- Transformation technique that changes the locations of unit-delay elements in a circuit without affecting the input/output characteristic.
- After retiming: Critical Path = Iteration Bound!
Hardware tricks

For speed:
- Parallelism
- Pipelining
- Loop unrolling
- FPGA: Block RAM instead of Logic
- ...

For area:
- Multiplexing
- Composite field instead of Sbox

For power/energy:
- Parallelism
- Pipelining
Algorithm properties

As they affect HW realization

- Internal state
- Block size
- Initialization cost
- Iterative, sequential, ...
- Parallelism
Benchmark efforts

Benchmarks on FPGA, ASIC
API efforts
Open questions
See his presentation for the “context”
Brian Baldwin

- FPGA: CubeHash, Grostl, Shabal, SIMD, JH, Hamsi and Fugue
- Core functionality & compression function
- See his presentation for ‘context’
Christian Wenzel-Benner

• eXternal Benchmarking eXtension
Miroslav Knezevic

- Illustration of transformations: applied to Luffa and others
- More observations
Patrick Schaumont: API for HW

- **INIT & GETCONFIG**: initialization, type of I/O, etc
- **IDATA & ODATA**: parameter
- 16, 32 bit: low end processor
- 64, 128 (256): high end processors

Figure 1: Overview of the interface signals
ATHENa Major Features

- synthesis, implementation, and timing analysis in the batch mode
- support for devices and tools of multiple FPGA vendors:
  - XILINX®, ALTERA®, Actel®
- generation of results for multiple families of FPGAs of a given vendor
  - VIRTEX®, VIRTEX-II PRO, SPARTAN®, SPARTAN-III
- automated choice of a best-matching device within a given family
Open questions

• Area comparisons
• Throughput comparisons
• Power/Energy comparisons
• Sets of environments
Conclusions

• Results depend on:
  – ASIC set-up
  – FPGA set-up
  – Hardware API
  – Bandwidth
  – Transformations

• Need:
  – Set of ‘contexts’
  – area and speed, but also POWER and ENERGY!