Hardware platform design and evaluation using GEZEL

Patrick Schaumont, UCLA

Acknowledgements:
Embedded Security Group (EMSEC) @ UCLA

Schaumont, Penn State, 6/23/05
Skiing the embedded systems mountain

- **Specification**
  - SPW, Matlab, C++
  - pipelining, unrolling

- **Algorithm Transformations**
  - loop merging, compaction

- **Memory Transformations and Optimizations**
  - 40 bit accumulator

- **Floating-point to Fixed-point**

- **ASIC**
- **Special Purpose**
- **Retargetable coprocessor**
- **DSP processor**
- **DSP-RISC**
- **RISC**

Schaumont, Penn State, 6/23/05
Programs driving MPSOC evolution

The SOC Model

- C
- HW
- RISC
- UART
- MEM

The MPSOC Model

- C
- HDL
- ASM
- Domain Specific

- MEM
- NOC
- RISC
- ASIP
- DSP
- + FPGA, Coarse-grain, Stream-Architectures, ..

Schaumont, Penn State, 6/23/05
Example: Portable Digital Media Processor

Portable Digital Media Processor (after Talla, Micro 04)
Domain-specialization = Energy-efficiency

Energy-efficiency of AES-128 on different targets

[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
Programs driving MPSOC evolution

The SOC Model

C

The MPSOC Model

C  HDL  ASM  Domain Specific

• MPSOC: An explosion in programming paradigms!
• Effective codesign requires that programming paradigms look alike
  • E.g. C + ASM. But C and classic HDL? No way.
  • It’s not sufficient to throw it all in a single language (like C++)
• In this talk: GEZEL: Hardware ‘Programming Language’
  • With application examples from embedded security, network-on-chip, design classes, multiprocessor-system-on-chip
C and (V)HDL paradigms do not mix well

⇒ Differences become an issue when application designer needs to program both (‘ski both sides of the slope’)

C (sequential software)

• Instruction driven
  • regular time progression

• Deterministic by design

• Model = implementation

HDL

• Event driven
  • irregular time progression

• Non-deterministic
  • concurrency + global var
  • races, ‘X’

• Simulation model
  • processes
  • hardware inference
GEZEL: A Hardware Programming Language

C (sequential software)

- Instruction driven
  - regular time progression
- Deterministic by design
- Model = implementation

GEZEL:

- Cycle-true
  - regular time progression
- Deterministic by design
  - verified by parser/simulator
  - no ‘X’ nor ‘U’
- Model ~ implementation
  - FSMD
  - explicit RT modeling
An FSMD in GEZEL

```plaintext
dp updown(out a : ns(4)) {
    reg c  : ns(4);
    sfg inc { c = c + 1;
      a = c;     }
    sfg dec { c = c - 1;
      a = c;     }
}

fsm ctl_updown(updown) {
    initial s0;
    state  s1;
    @s0 if (c < 10) then (inc) -> s0;
        else (dec) -> s1;
    @s1 if (c > 0) then (dec) -> s1;
        else (inc) -> s0;
}
```

(Closed) FSMD networks

wire (= input is instantaneously defined by conn. output)

GEZEL models Extended FSMD networks

Library Block:
- Interface in GEZEL
- Body in C++
- IO, Cosimulation, IP
Codesign with GEZEL

FSMD model of hardware

HW/SW Interfaces
Library Blocks

GEZEL Model

Schaumont, Penn State, 6/23/05
Platform Simulators with GEZEL

GEZEL Kernel
(C++ Library)

EmSW

Application
(by designer)

Platform Simulator (by tool builder)

parser

VHDL
codegen

RT
codegen

executable
object
hierarchy

user-defined
ipblock impl.

Instruction-Set
Simulator

Communication
Channel

Cycle-true
System Scheduler

Schaumont, Penn State, 6/23/05
## GEZEL Platform Simulator Examples

<table>
<thead>
<tr>
<th>Simulator</th>
<th>Instruction Set Simulator</th>
<th>Multi/Single Processor</th>
<th>Cross Compiler</th>
<th>Codesign Interfaces</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>armcosim</td>
<td>SimIt-ARM ¹</td>
<td>S</td>
<td>arm-linux-gcc</td>
<td>Memory &amp; CP bus</td>
<td>coprocessors</td>
</tr>
<tr>
<td>armthreads</td>
<td>SimIt-ARM</td>
<td>M</td>
<td>arm-linux-gcc</td>
<td>Memory &amp; CP bus</td>
<td>SMP</td>
</tr>
<tr>
<td>fdltsim</td>
<td>LEON-2 SPARC ²</td>
<td>S</td>
<td>sparc-rtems-gcc</td>
<td>Memory &amp; CP bus</td>
<td>emSW accelerators</td>
</tr>
<tr>
<td>gezelsh</td>
<td>SH3-DSP ³</td>
<td>S</td>
<td>sh-elf-gcc</td>
<td>Memory</td>
<td>emSW accelerators</td>
</tr>
<tr>
<td>gezel51</td>
<td>Dalton 8051 ⁴</td>
<td>S</td>
<td>SDCC</td>
<td>Ports</td>
<td>Sensor nodes</td>
</tr>
<tr>
<td>libsysc</td>
<td>SystemC 2.0.1 (HW)</td>
<td></td>
<td></td>
<td>Ports</td>
<td>Legacy integration</td>
</tr>
<tr>
<td>gplatform</td>
<td>8051 + ARM</td>
<td>M</td>
<td>arm-linux-gcc</td>
<td>Memory &amp; CP bus</td>
<td>MPSOC</td>
</tr>
<tr>
<td>avrora</td>
<td>Atmel AVR ⁵</td>
<td>S</td>
<td>avr-gcc</td>
<td>Memory</td>
<td>emSW accelerators</td>
</tr>
</tbody>
</table>

¹ [sourceforge.net/projects/simit-arm/](http://sourceforge.net/projects/simit-arm/); arm-linux-gcc v. 2.95.2 from [http://www.lart.tudelft.nl](http://www.lart.tudelft.nl)

² TSIM 2.1 by Gaisler research [http://www.gaisler.com](http://www.gaisler.com)

³ RENESAS SH3DSP Simulator/Debugger; sh-elf-gcc v3.3 from [http://www.kpitgnutools.com](http://www.kpitgnutools.com)


⁵ Avrora project, [http://compilers.cs.ucla.edu/avrora/](http://compilers.cs.ucla.edu/avrora/)
VHDL Code generation

**aes_decoder**
- Inputs: done, rst, ld
- Outputs: 128-bit

**aes_top** (AES/ECB)
- Inputs: Key, Plaintext
- Outputs: 128-bit

**Cipher Text**
- Inputs: 128-bit
- Outputs: 128-bit

**μP Core**
- Inputs: Addr, Data
- Outputs: instructions, data_in, data_out

**Embedded Software Driver**
- Inputs: 64-bit
- Outputs: 64-bit

**RTL VHDL**
- Synthesis: Synopsys dc_shell, Synplicity Synplify, Xilinx XST

**RTL Stimuli**
- Simulation: Modeltech Modelsim

**FSMD model of hardware**

**HW/SW Interfaces Library Blocks**
- per module
- multi-module
Example Applications with GEZEL

- Embedded security applications
  - ThumbPod-1: Embedded fingerprint authentication prototype on FPGA
  - ThumbPod-2: Side-channel-resistant fingerprint authentication processor in 0.18um CMOS
- Multiprocessor applications
  - Network-on-chip design: Topology and protocol-stack evaluation
  - Energy-scaled multiprocessors: Voltage-scaled Symmetric Multi-processor
- Teaching codesign and coprocessor design
The ThumbPod Project

Challenge/response

Bank

ThumbPod

Embedded electronics

Fingerprint sensor
Crypto coprocessor for Embedded JAVA

Performance on KVM+LEON2 (clock cycles)

<table>
<thead>
<tr>
<th>Host</th>
<th>AES implementation in</th>
<th>JAVA</th>
<th>C</th>
<th>GEZEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAVA</td>
<td>194K</td>
<td>10K</td>
<td>1.7K</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>9.2K</td>
<td></td>
<td>790</td>
<td></td>
</tr>
<tr>
<td>GEZEL</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Speedup: 162
Overhead: 109
Secure Authenticated Communication

Minutiae Extraction

Matching Algorithm

Accept

Load Master

Reject

Load Bogus

Random (from server)

E

Session Key $S_k$

Master Key

Template
Secure Authenticated Communication

Minutiae Extraction

Matching Algorithm

Accept

Load Master

Template

Reject

Load Bogus

Master Key

Session Key $S_k$

Secure Authenticated Communication

C (insecure)

GEZEL (DPA-safe HW)

random (from server)

Schaumont, Penn State, 6/23/05
ThumbPod-2: DPA-resistant matching

GEZEL Coprocessor

VHDL code generation

Logic Synthesis

WDDL Conversion

WDDL Netlist

Under DPA attack, key disclosure in 3 minutes

No full key disclosure under similar attack
Network-on-chip design in GEZEL

GEZEL Model

Schaumont, Penn State, 6/23/05
Energy-scaled embedded multiprocessor

V/f scaling under control of the application

e.g. StrongARM (LART)
  High V/f  1.65/251
  Low V/f  0.79/59
  V^2f ratio  18.5
  f ratio  4.25
  switching  140 us

GEZEL is used for system integration of ARM ISS
int main( ) {
    create(my_thread);
    start();
}

int slave_main {
    .. getprocid();
}

void my_thread() {
    // user thread
    while (1)
        yield();
    abort();
}

Quickthreads-based library
(350 lines C + 25 lines asm, 1600 bytes obj)
Thread-parallel Minutiae Detection

4 threads + main thread
Energy Scaling Results

(*) 2_HL = two-processors: one high-power, one low-power
(**) GEZEL MPSOC Model runs at 400 KHz (4-ARM on 3GHz-PIII/512 MB)
GEZEL for Teaching

VLSI Design Methods and Arch
UCLA

Spec (C)

TI C54
AD Blackfin
ARM+ GEZEL
LEON2+ GEZEL

Hands-on Projects: JPEG encoder, Embedded web server

Introduction to Codesign
DTU (Denmark)

Spec (C)

basic blocks control flow graphs

SW organization (files, stack,..)
HW organization (FSMD=dp+ctl)

GEZEL

Hands-on Project: MIC-1 Microcontroller with coprocessor
Modeling issues: Non-determinism

- Verilog, VHDL, SystemC are non-determinate
  - ‘X’, race-resolution function
- Non-determinism can be useful at high level
  - StateCharts, CSP, ..
- But it is undesirable for RTL design (races)
  - Sneaks in as a side effect
  - Challenge for verification & comprehension of code
  - May give simulator-dependent behavior
GEZEL yields deterministic HW

The Kahn Principle: systematic determinism

- System = $\Sigma$ (deterministic processes)
- Applicable to different process semantics
  - Kahn Process Networks [Kahn 74]
  - I/O Automata [Lynch 88]
  - Synchronous Languages [Potop 03]
  - GEZEL-type FSMD
4 rules yield deterministic FSMD

‘Proper FSMD’ can be enforced using only 4 *verifiable* rules

1. Single-assignment over a single clock cycle
2. No dangling (undefined) signals over any clock cycle
3. No combinatorial loops over any clock cycle
4. All FSMD outputs defined over any clock cycle

Result is deterministic hardware for an arbitrary network of FSMD (guaranteed by Kahn Principle)
Conclusions

Architectures
- CPU+Coprocessor,
- Heterogeneous multiprocessor
- DPA-resistant blocks

Applications
- Embedded Security
- Biometric Authentication

Teaching Projects
- Making it matter
- Structures & Generalize

Methods + Tools
- GEZEL
- WDDL

Tune & Specialize