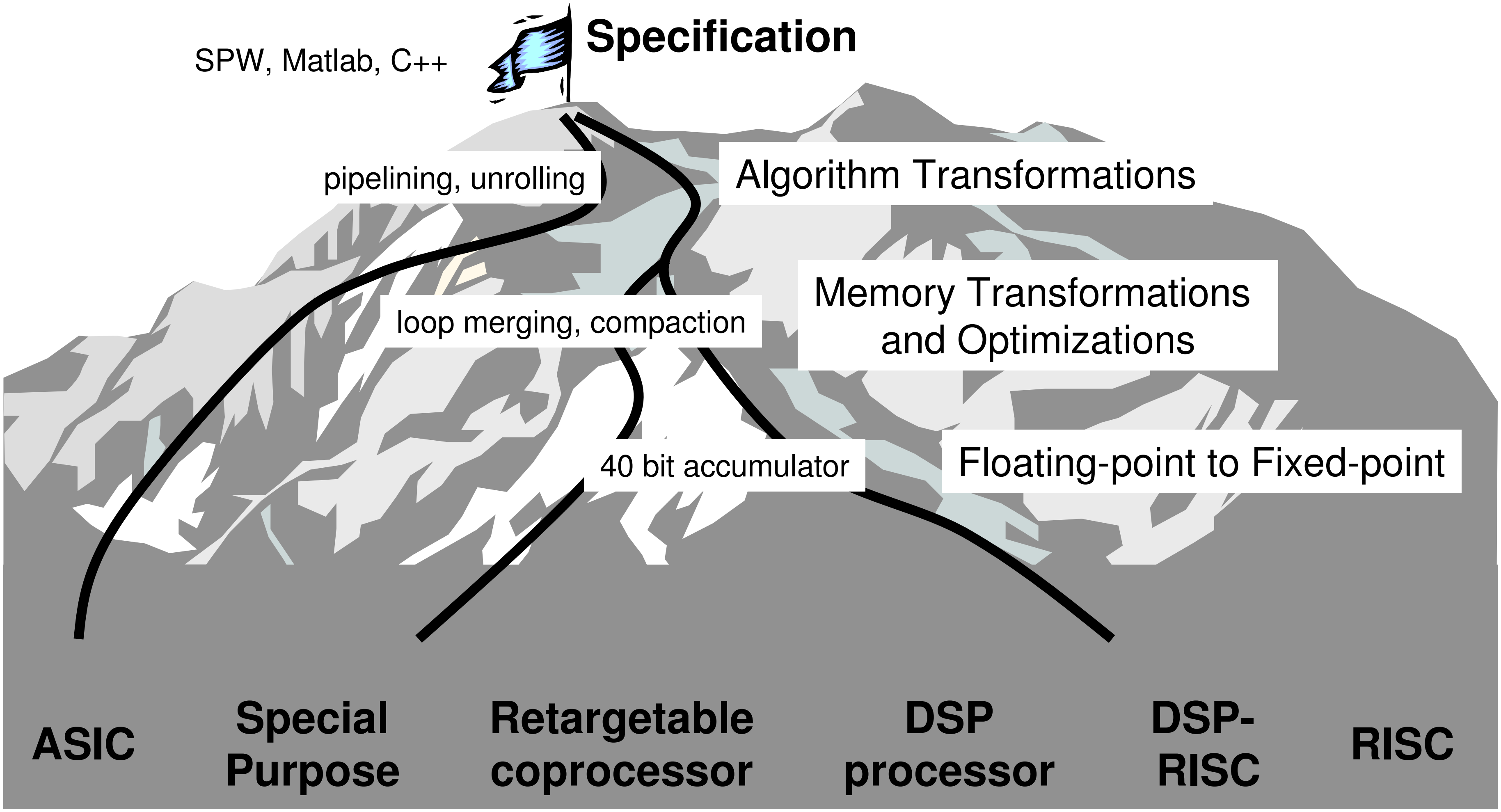
UCLA

Hardware platform design and evaluation using GEZEL

Patrick Schaumont, UCLA

Acknowledgements:
Embedded Security Group (EMSEC) @ UCLA

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

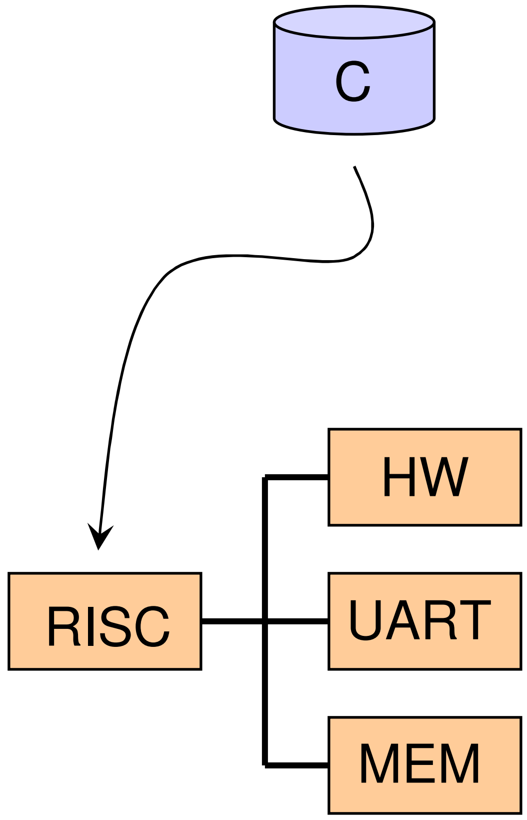
DSP processor

DSP-RISC

RISC

Programs driving MPSOC evolution

The SOC Model



C

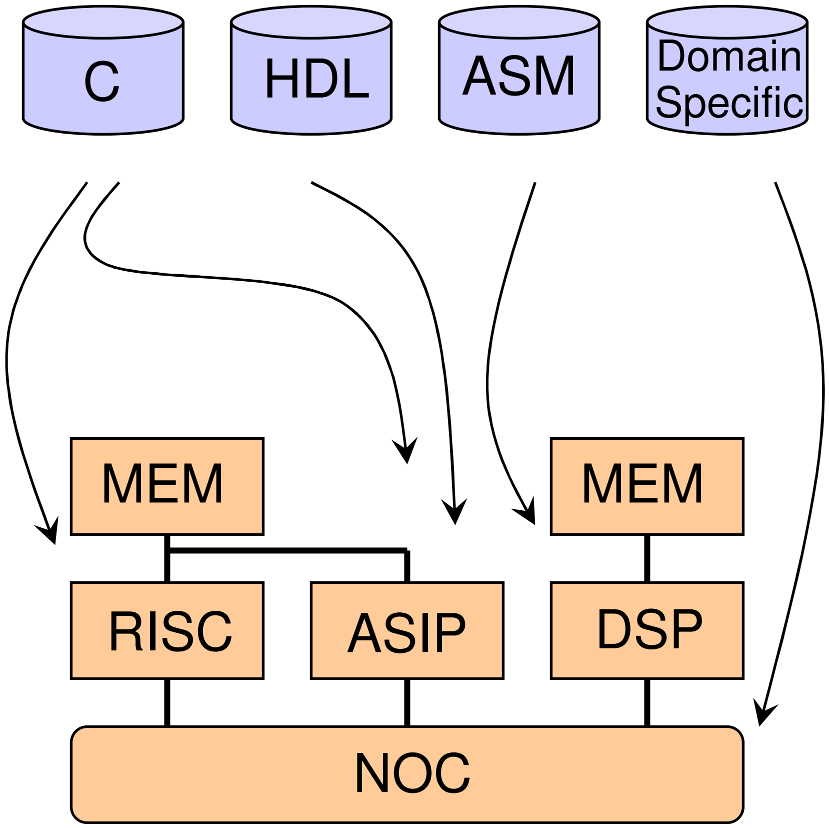
RISC

HW

UART

MEM

The MPSOC Model



C

HDL

ASM

Domain Specific

MEM

MEM

RISC

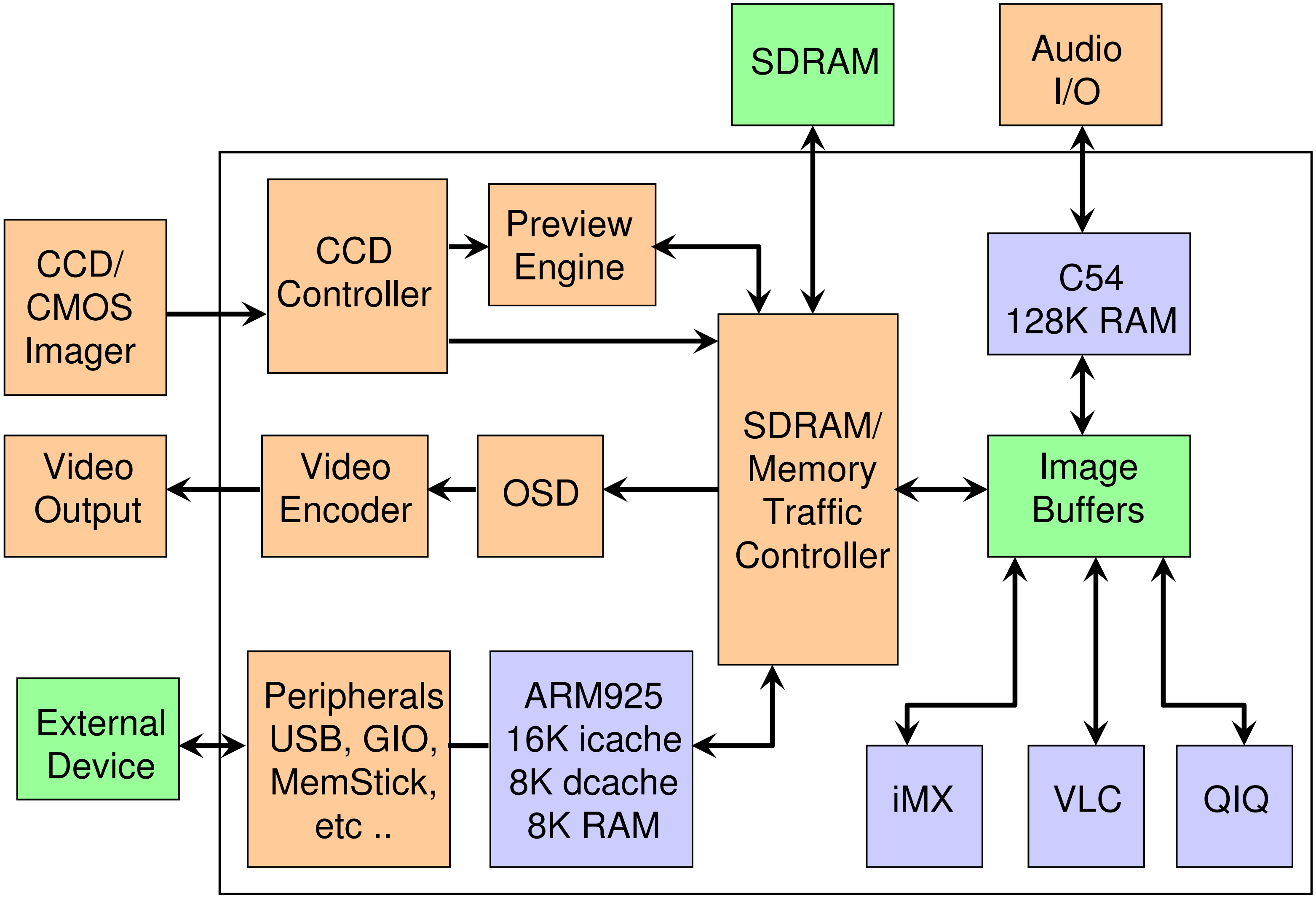
ASIP

DSP

NOC

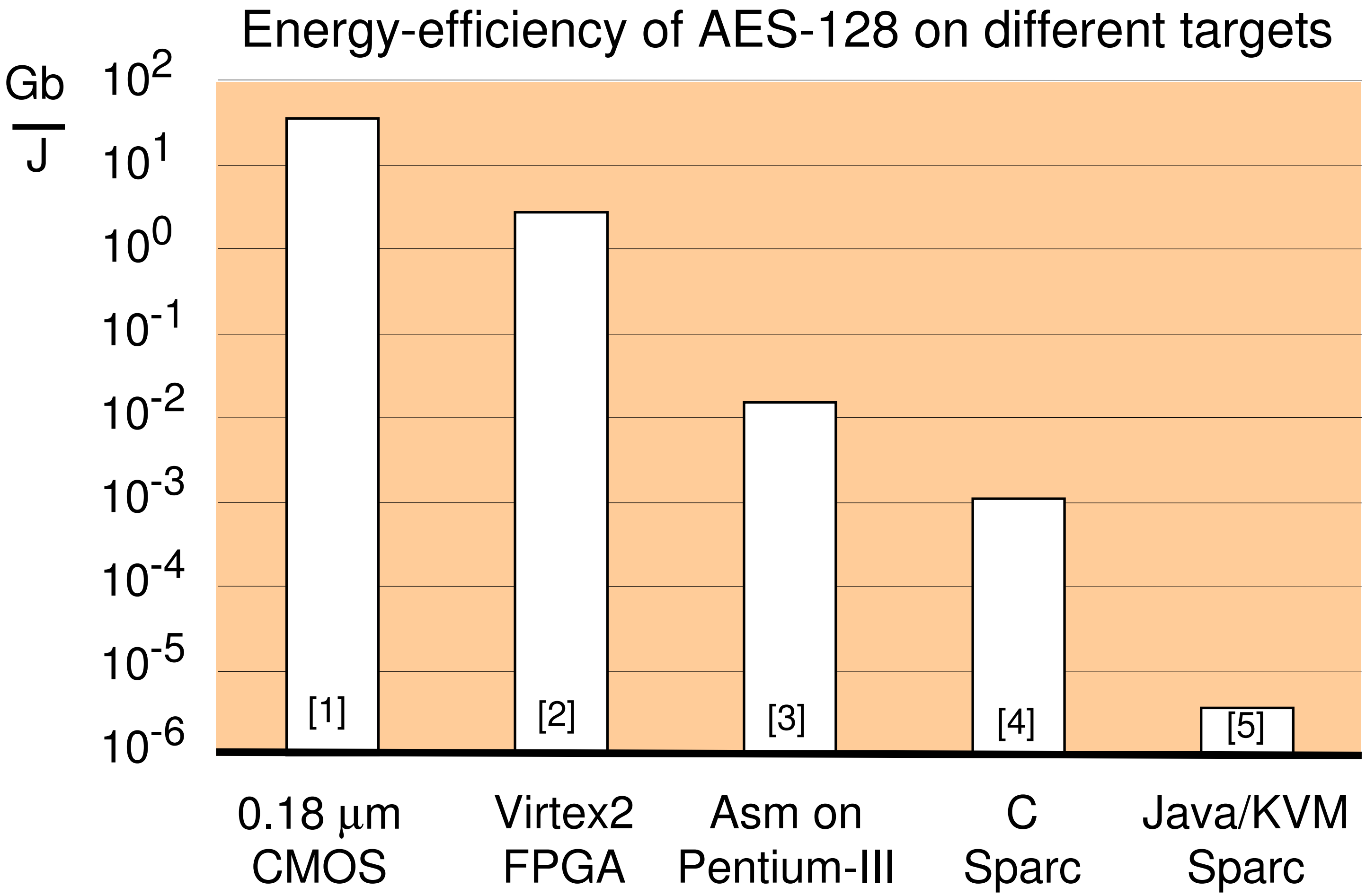
+ FPGA, Coarse-grain,
Stream-Architectures, ..

Example: Portable Digital Media Processor



Portable Digital Media Processor (after Talla, Micro 04)

Domain-specialization = Energy-efficiency

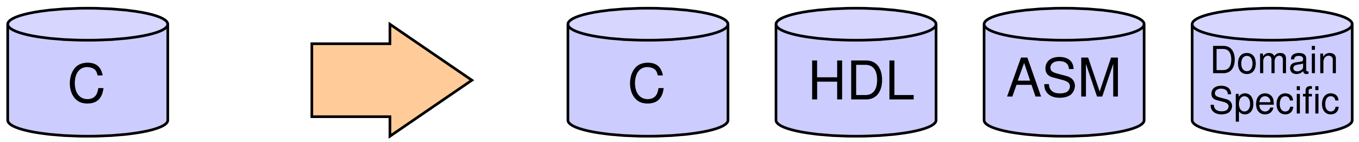


[1] Verbauwhede et al. "Design and performance testing of a 2.29 Gb/s Rijndael Processor,“ IEEE JSSC, Mar 03.
[2] Amphion CS5230 on Virtex2 + Xilinx Virtex2 Power Estimator
[3] Helger Lipmaa PIII assembly handcoded + Intel Pentium III (1.13 GHz) Datasheet
[4] gcc, 1 mW/MHz @ 120 Mhz Sparc – assumes 0.25 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

The MPSOC Model



C

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)	HDL
• Instruction driven • regular time progression	• Event driven • irregular time progression
• Deterministic by design	• Non-deterministic • concurrency + global var • races, ‘X’
• Model = implementation	• Simulation model • processes • hardware inference

GEZEL: A Hardware Programming Language

C (sequential software)

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

GEZEL:

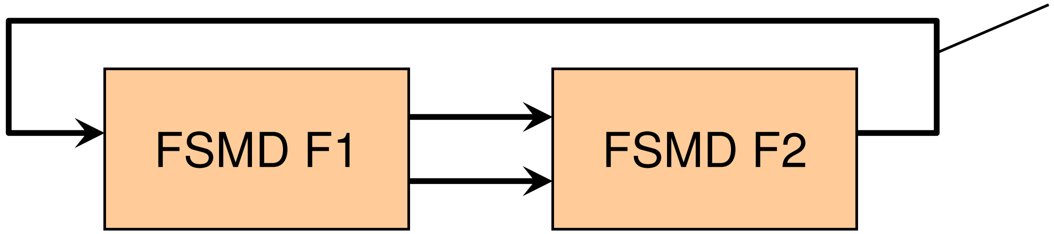
- Cycle-true
  - regular time progression
- Deterministic by design
  - verified by parser/simulator
  - no ‘X’ nor ‘U’
- Model ~ implementation
  - FSMD
  - explicit RT modeling

An FSMD in GEZEL

```
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;
}
```

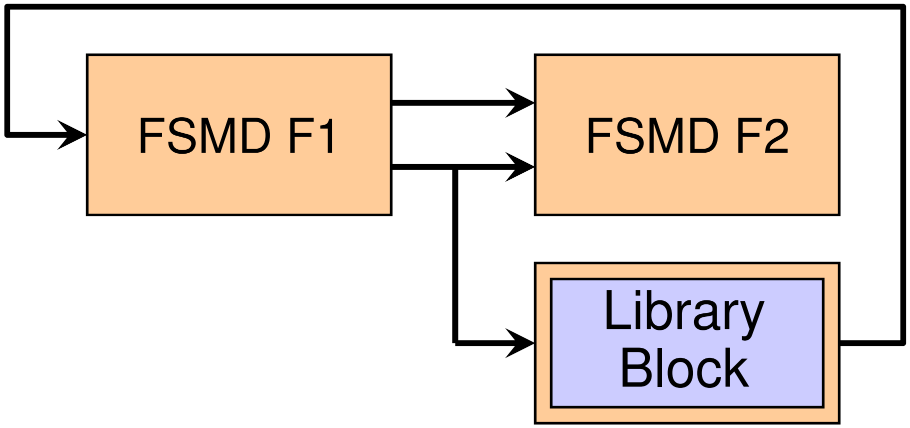
FSMD networks

(Closed) FSMD networks



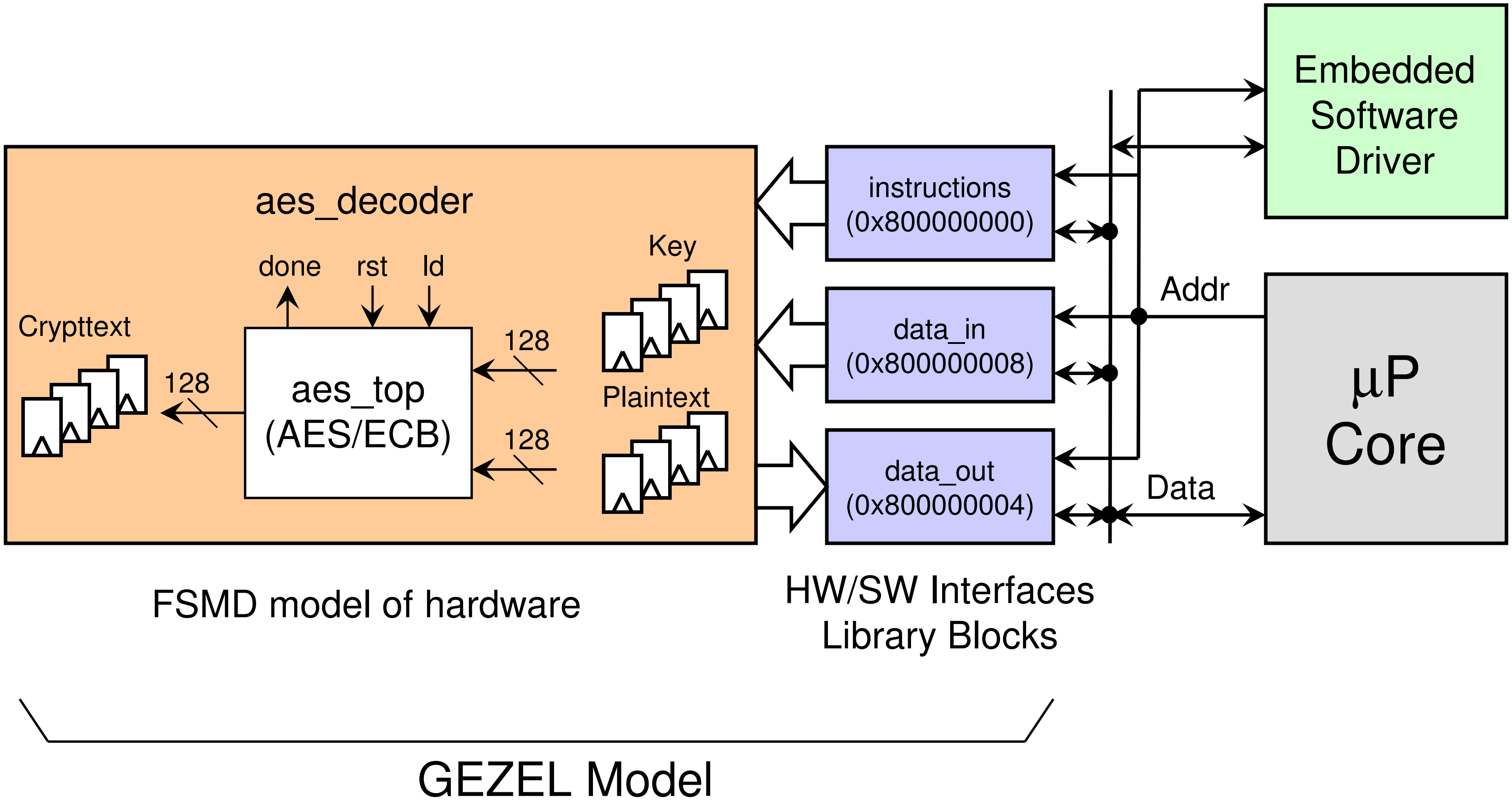
wire (= input is instantaneously defined by conn. output)

GEZEL models Extended FSMD networks

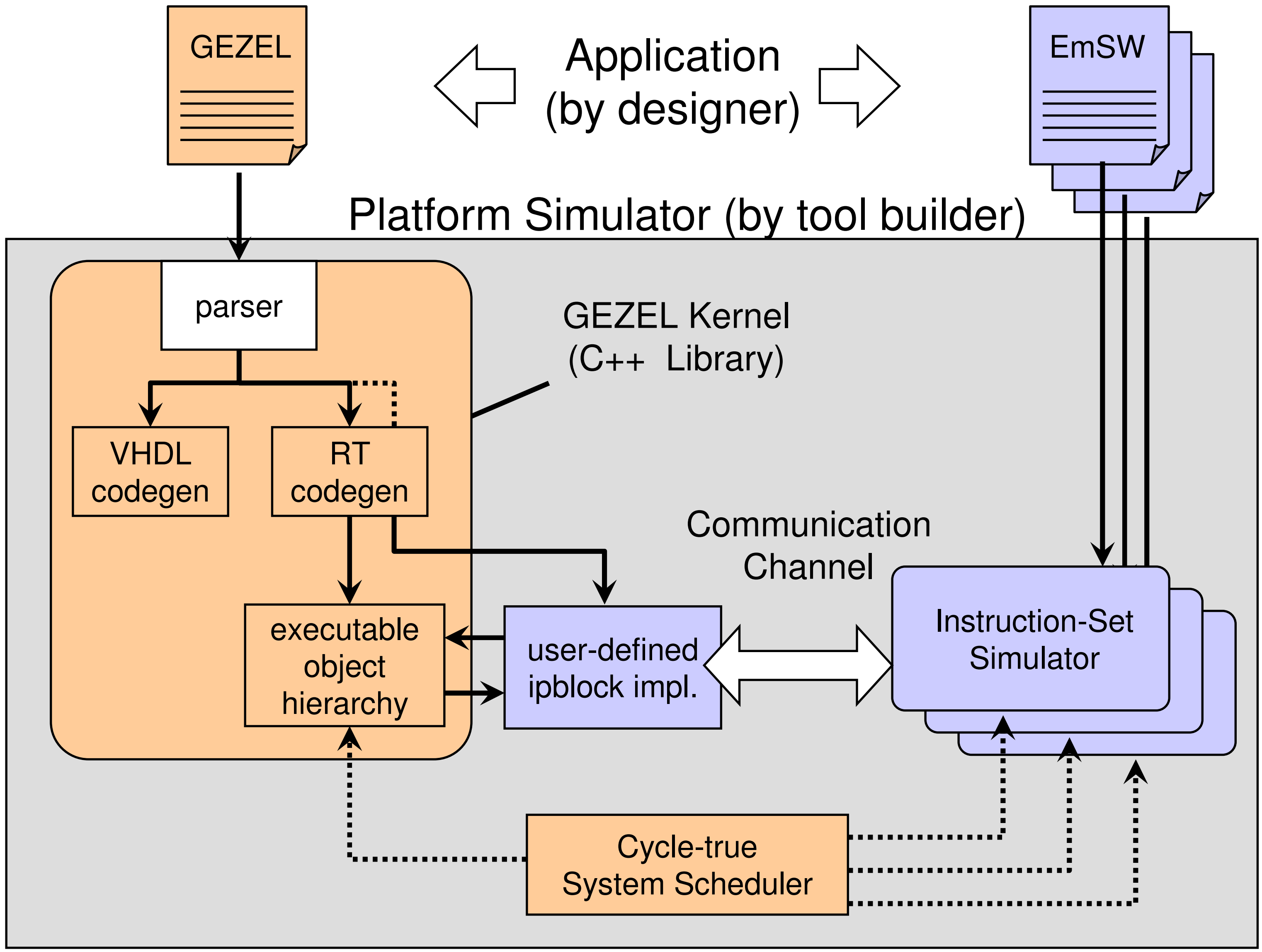


Library Block:
- Interface in GEZEL
- Body in C++
- IO, Cosimulation, IP

Codesign with GEZEL



Platform Simulators with GEZEL



GEZEL Platform Simulator Examples

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

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

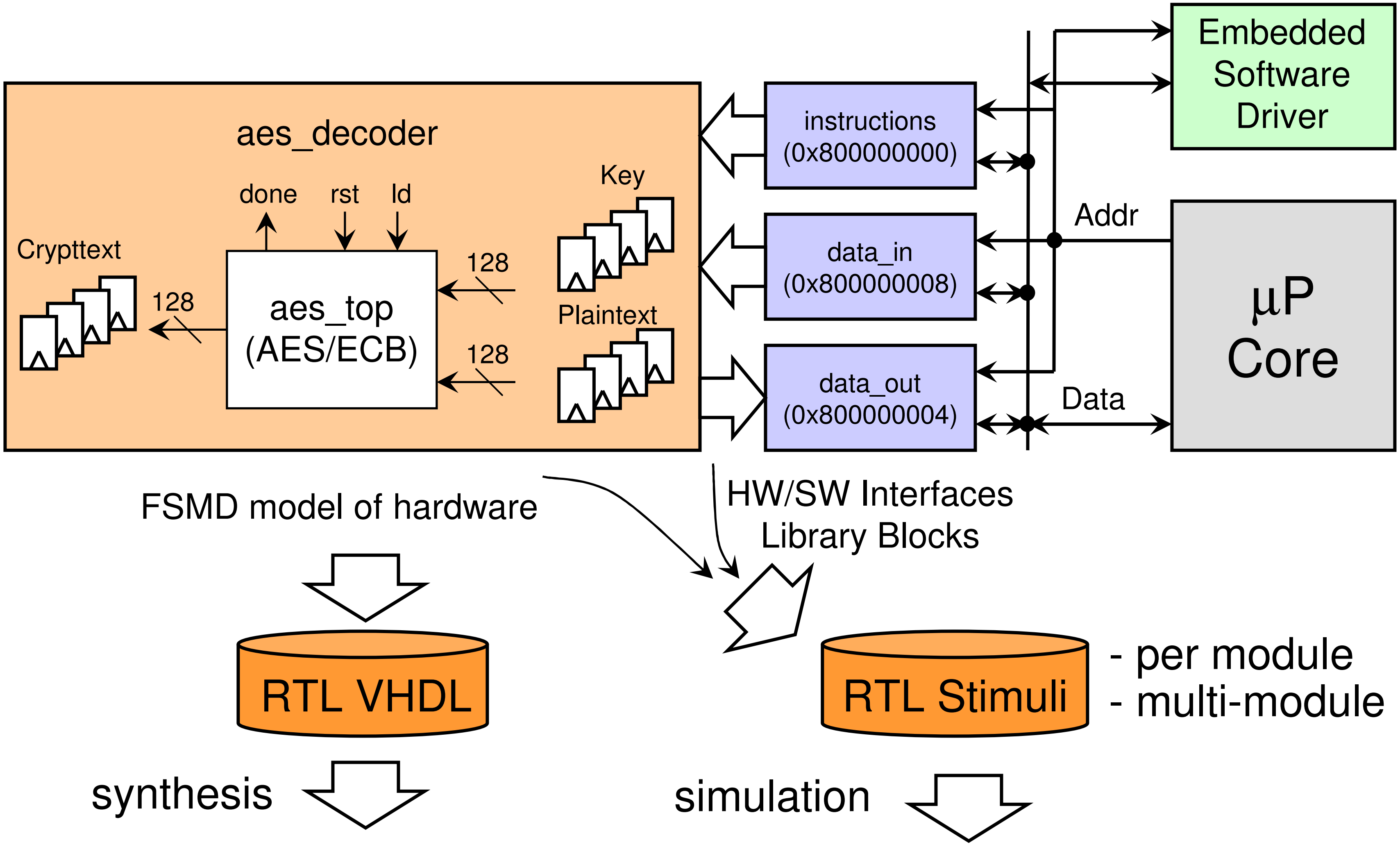
[2] TSIM 2.1 by Gailser research http://www.gaisler.com

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

[4] UCR Dalton project http://www.cs.ucr/edu/~dalton/i8051; SDCC from http://sdcc.sourceforge.net

[5] Avrora project, http://compilers.cs.ucla.edu/avrora/

VHDL Code generation



aes_decoder

done rst ld

Key

Crypttext

128

aes_top
(AES/ECB)

128

Plaintext

128

instructions
(0x800000000)

data_in
(0x800000008)

data_out
(0x800000004)

Embedded
Software
Driver

Addr

µP
Core

Data

FSMD model of hardware

HW/SW Interfaces
Library Blocks

RTL VHDL

RTL Stimuli

- per module
- multi-module

synthesis

simulation

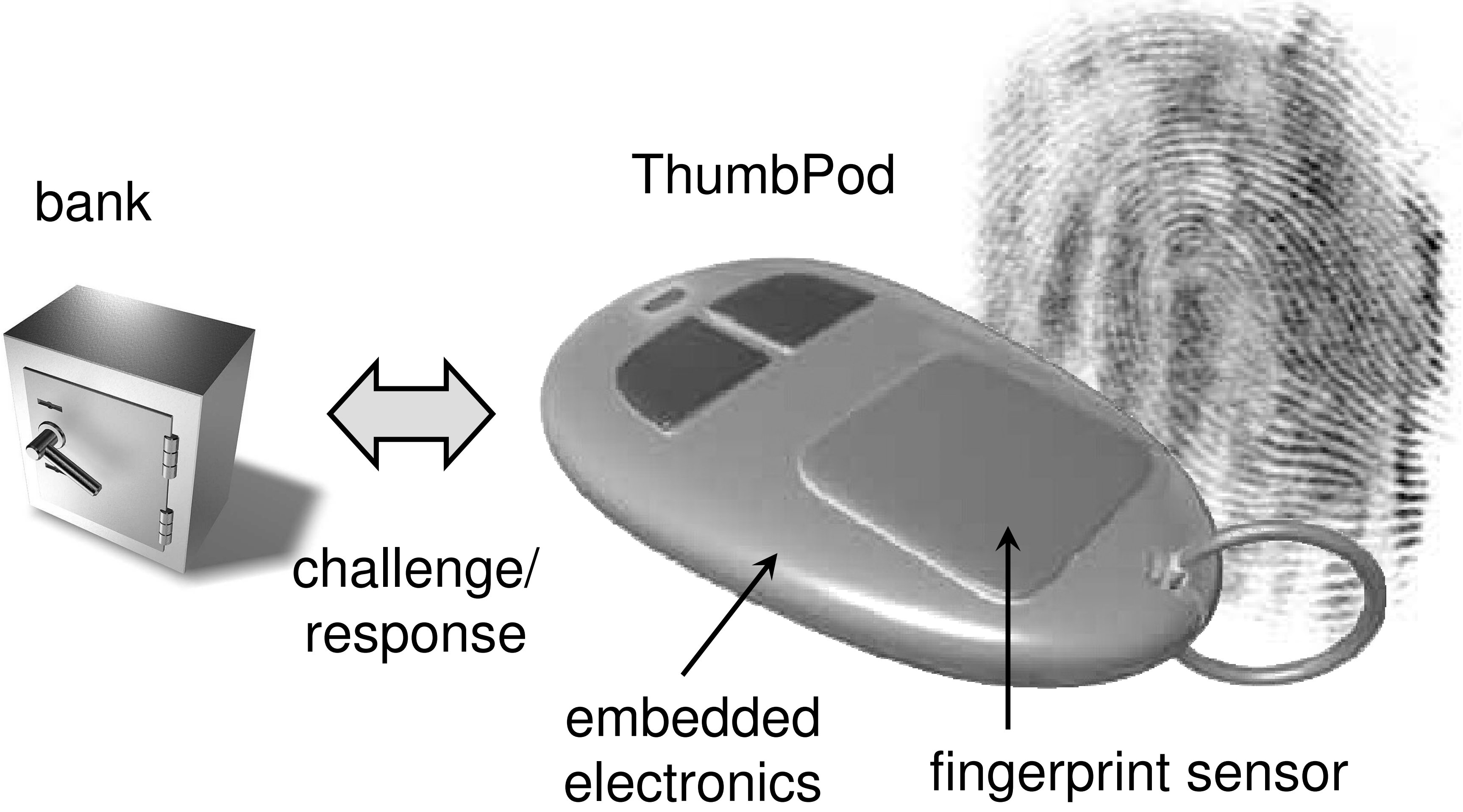
Synopsys dc_shell,
Synplicity Synplify, Xilinx XST

Modeltech Modelsim

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



bank

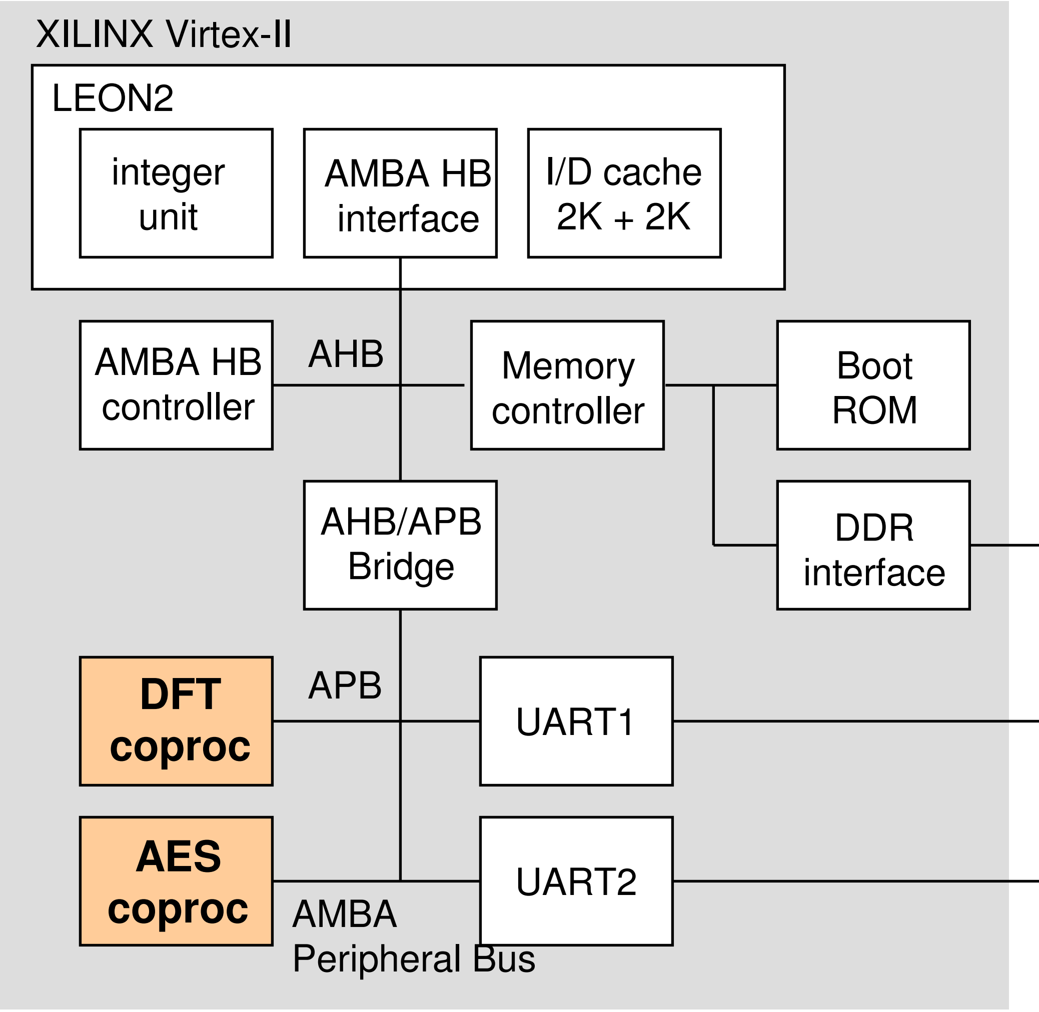
ThumbPod

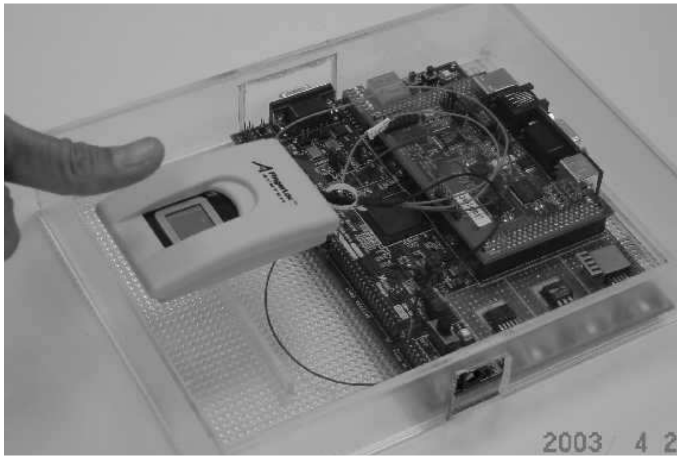
challenge/ response

embedded electronics

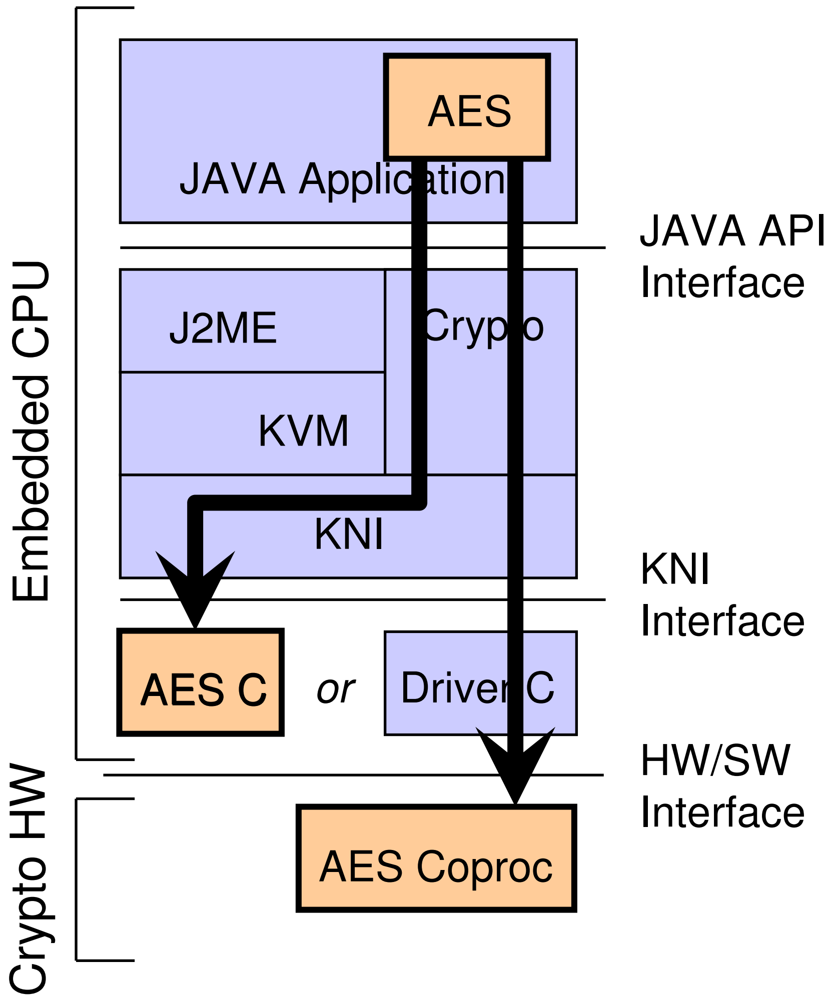
fingerprint sensor

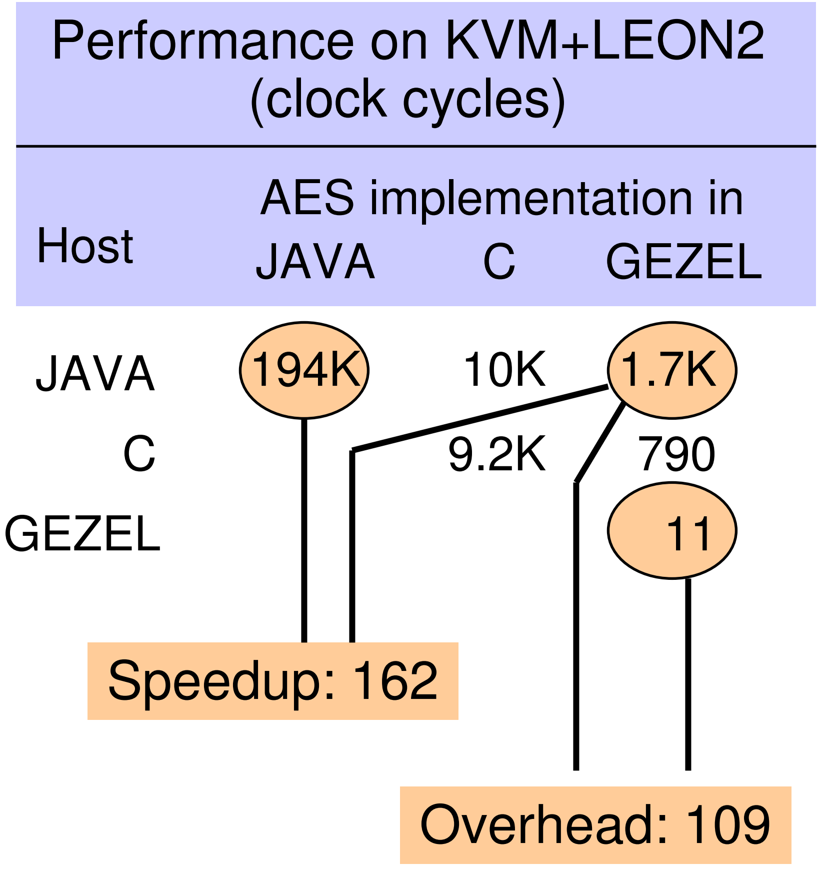
ThumbPod-1 Prototype





Crypto coprocessor for Embedded JAVA





Performance on KVM+LEON2 (clock cycles)

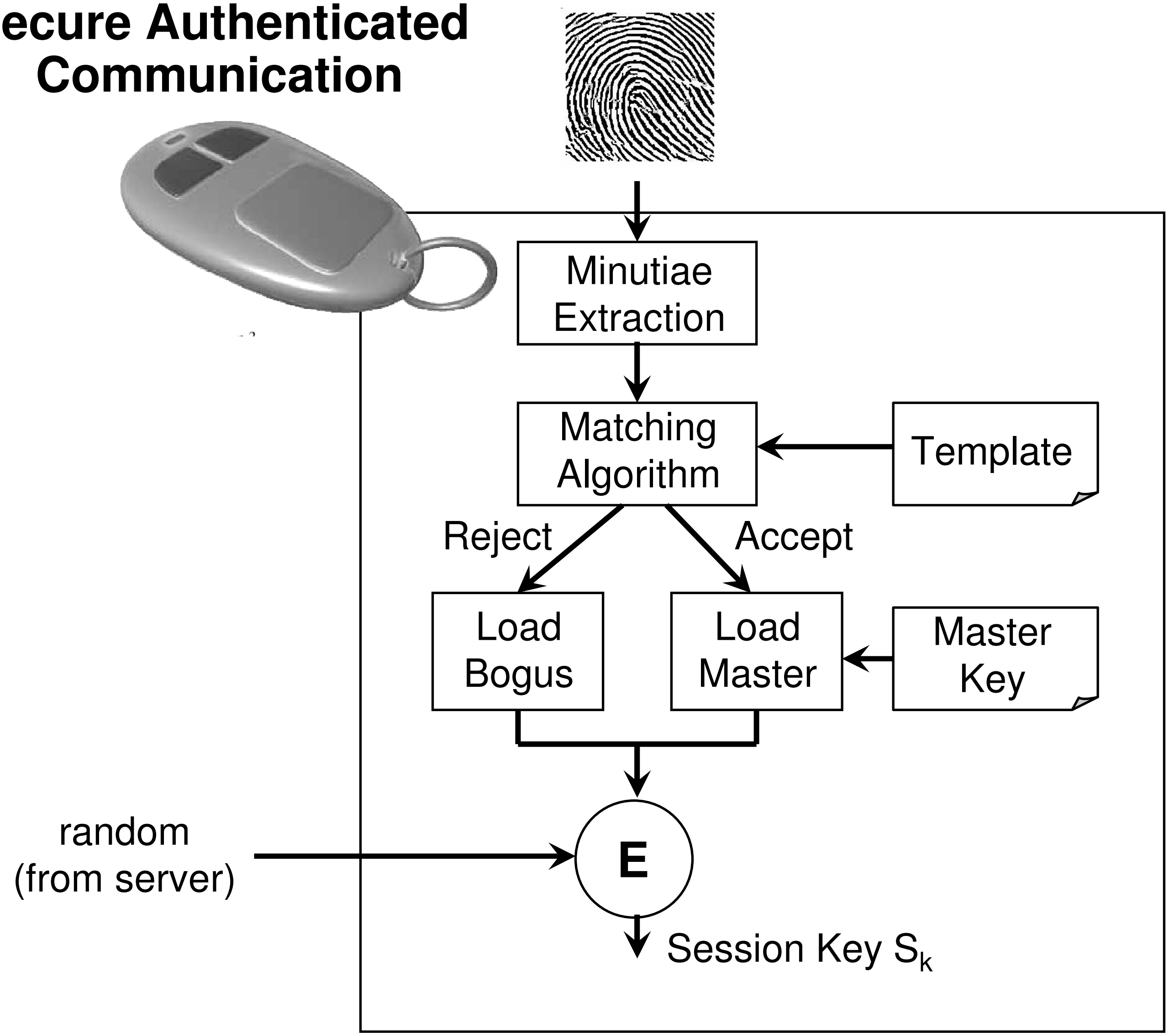
Host	AES implementation in JAVA	C	GEZEL
JAVA	194K	10K	1.7K
C		9.2K	790
GEZEL			11

Speedup: 162

Overhead: 109

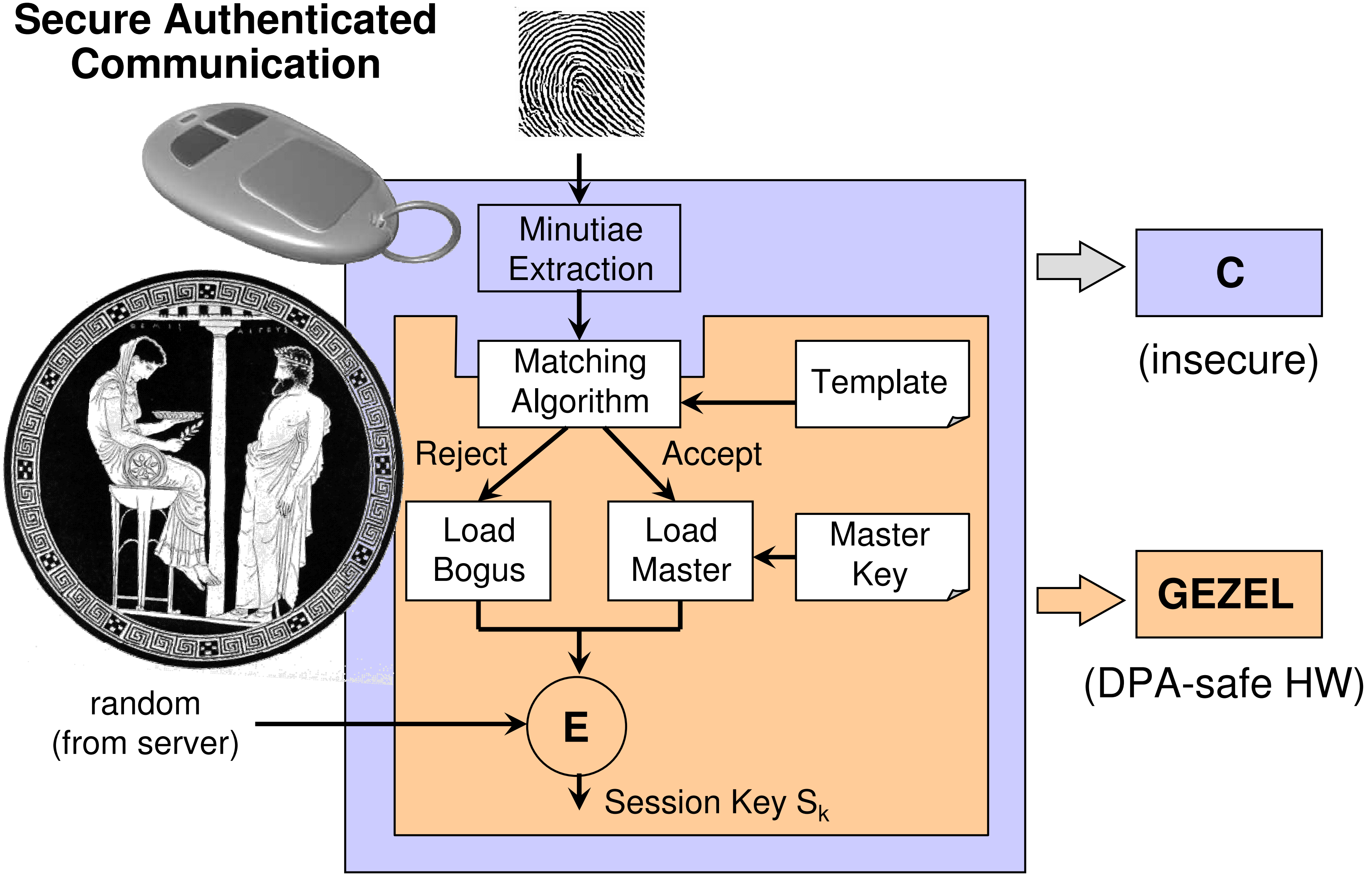
Security Partitioning in ThumbPod-2

Secure Authenticated Communication



Security Partitioning in ThumbPod-2

Secure Authenticated Communication



Minutiae Extraction

Matching Algorithm

Template

Reject

Accept

Load Bogus

Load Master

Master Key

random (from server)

E

Session Key S_k

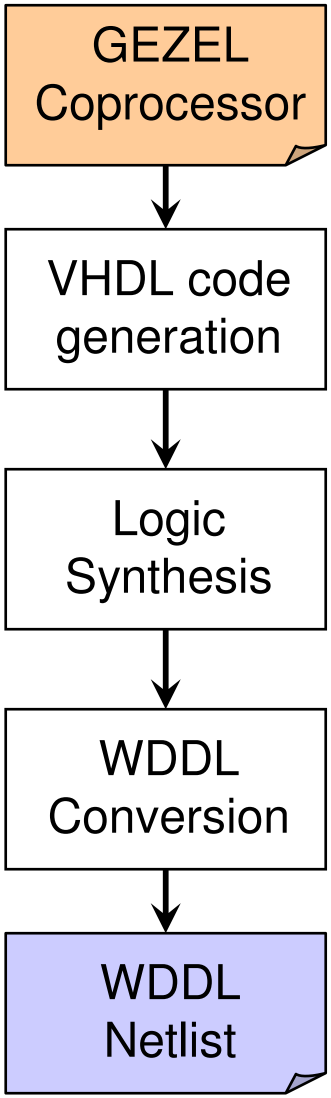
C

(insecure)

GEZEL

(DPA-safe HW)

ThumbPod-2: DPA-resistant matching



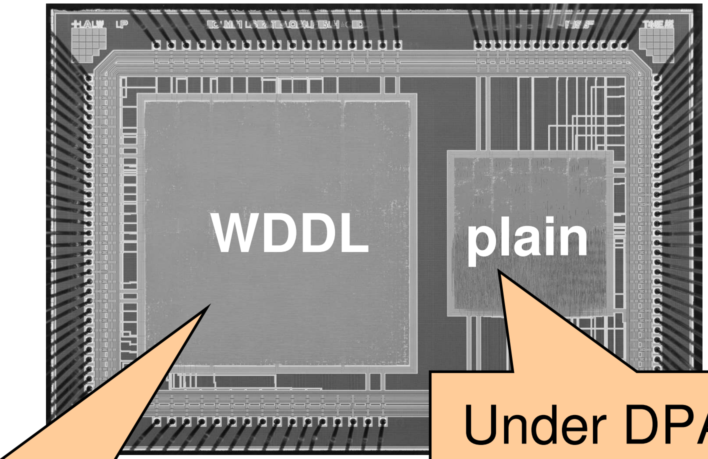
GEZEL Coprocessor

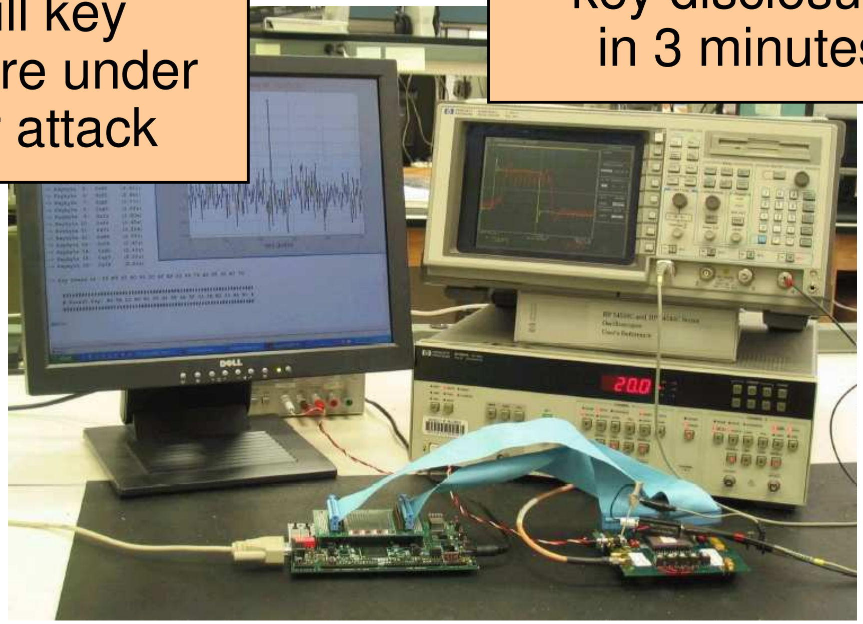
VHDL code generation

Logic Synthesis

WDDL Conversion

WDDL Netlist

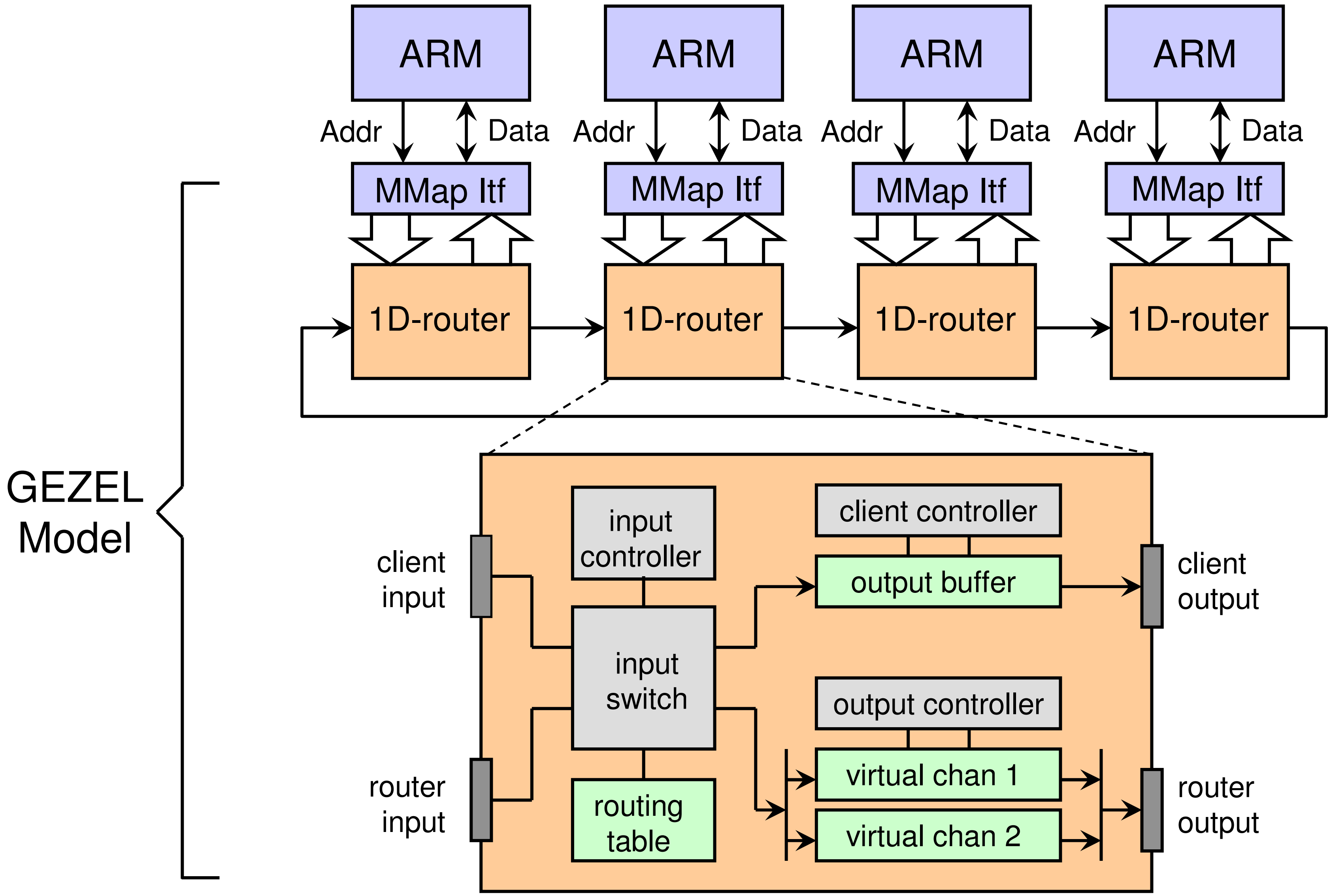




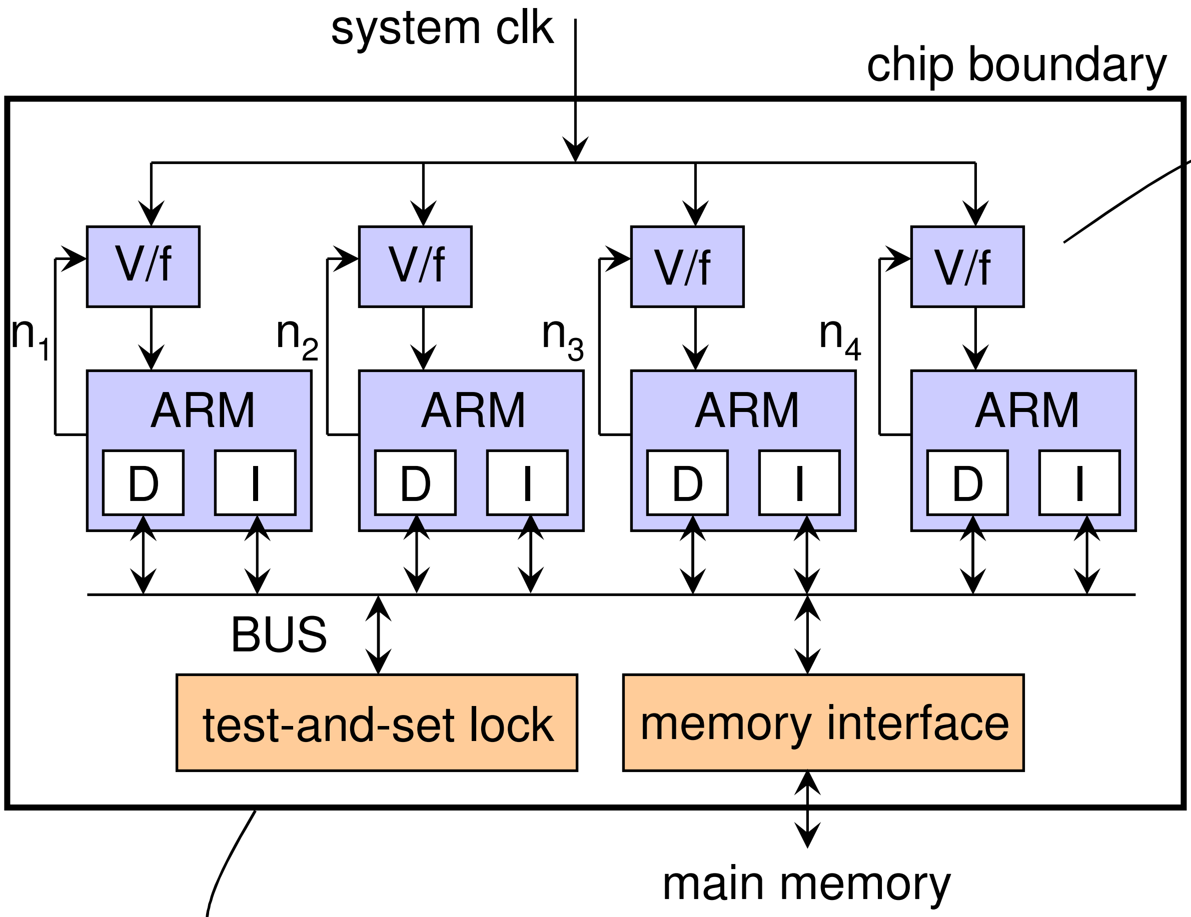
No full key disclosure under similar attack

Under DPA attack, key disclosure in 3 minutes

Network-on-chip design in GEZEL



Energy-scaled embedded multiprocessor



V/f scaling under control of the application

e.g. StrongARM (LART)

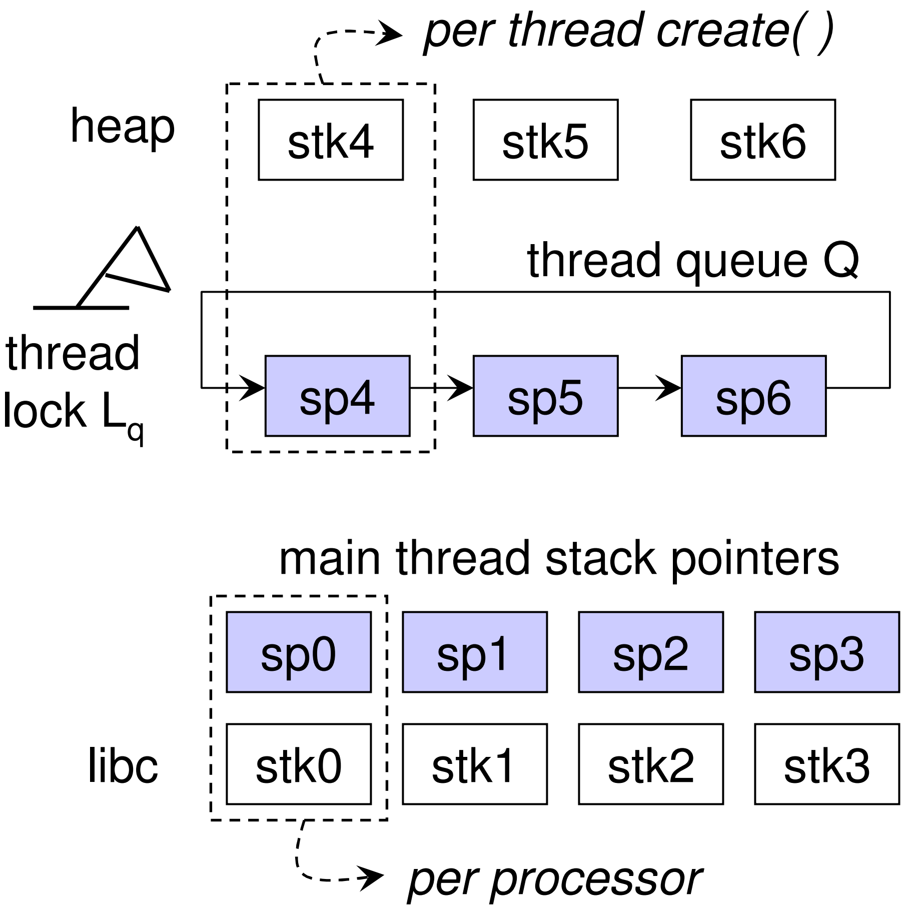
High V/f	1.65/251
Low V/f	0.79/59
V^2f ratio	18.5
f ratio	4.25
switching	140 us

GEZEL is used for system integration of ARM ISS

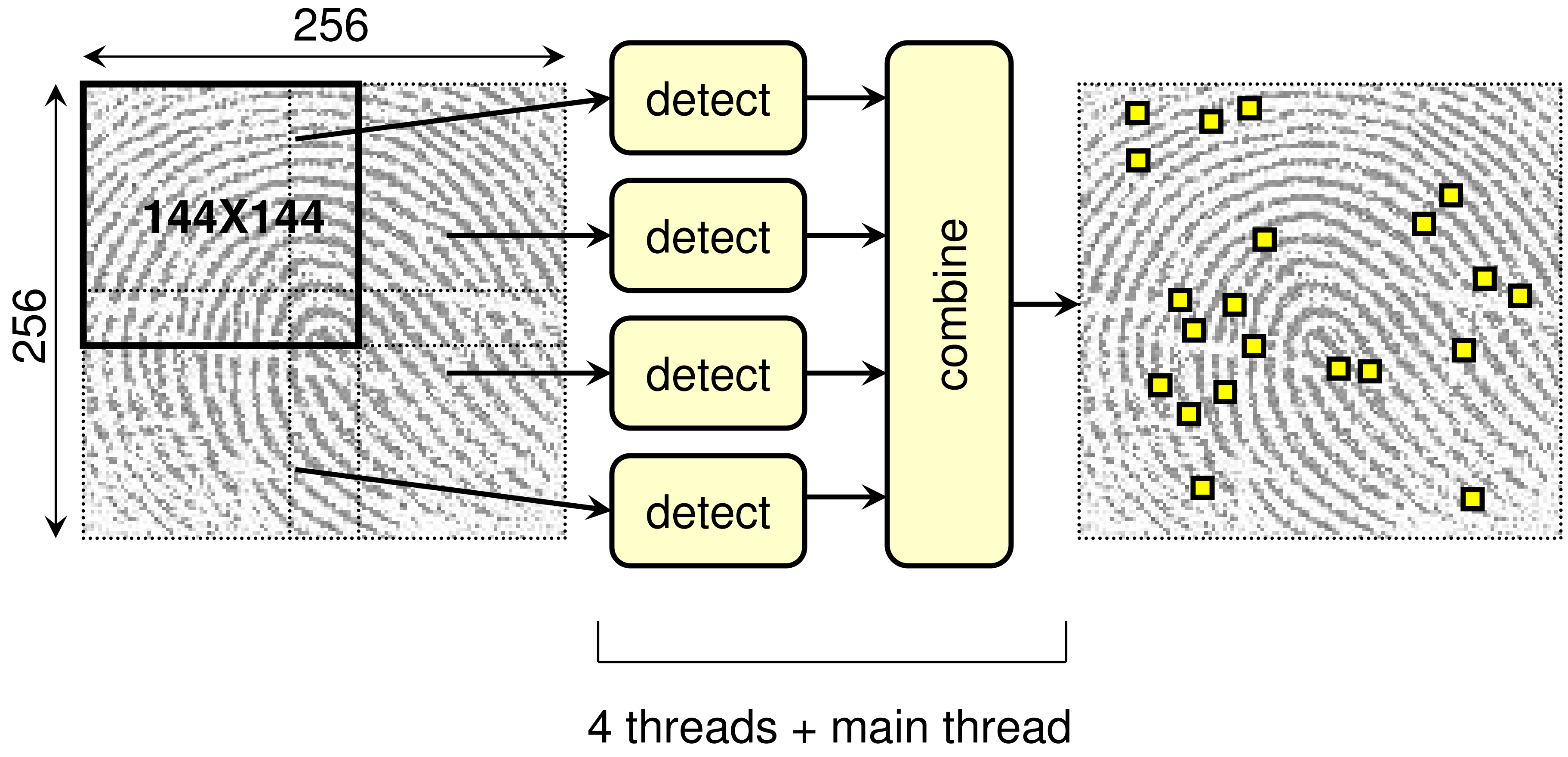
Cooperative Threading Model

```
int main( ) {
  create(my_thread);
  start();
}

int slave_main {
  .. getprocid();
}

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

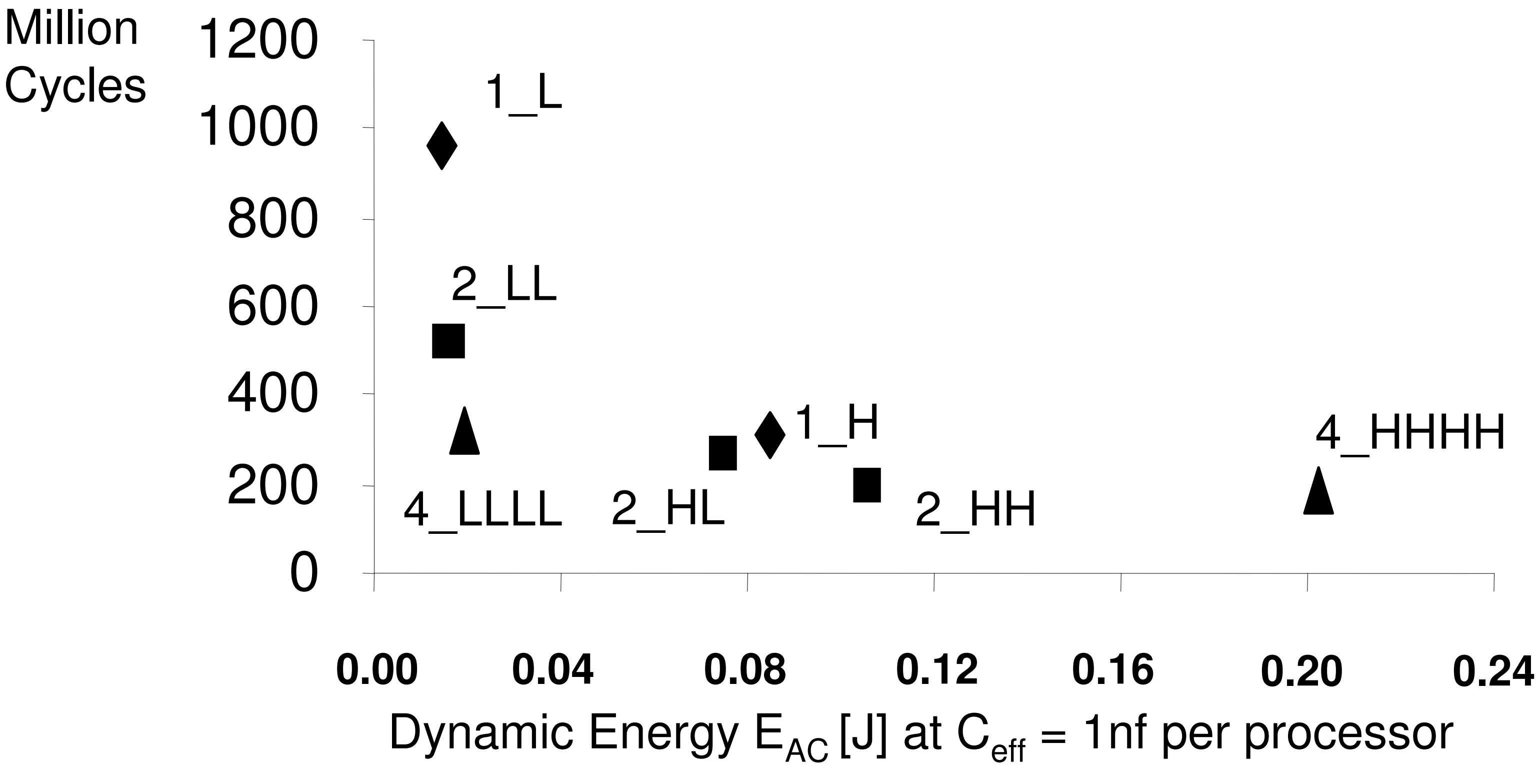
Quickthreads-based library
(350 lines C + 25 lines asm, 1600 bytes obj)



Thread-parallel Minutiae Detection



Energy Scaling Results

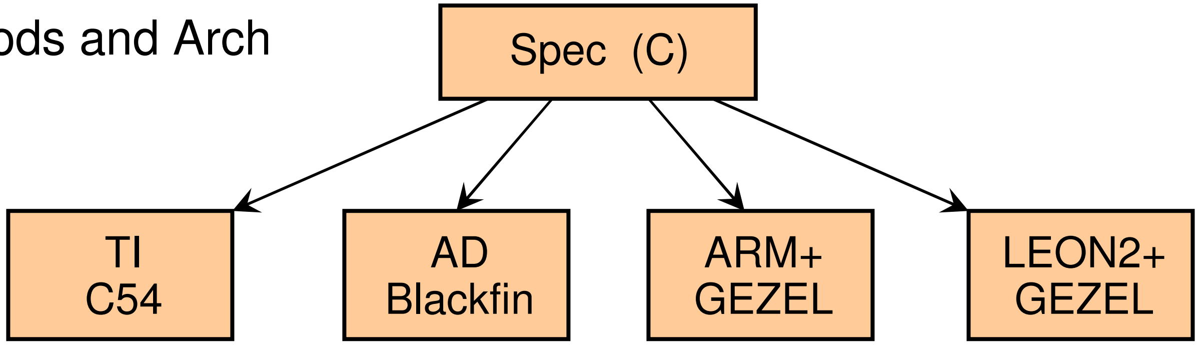


(*) 2_HL = two-processors: one high-power, one low-power

(**) 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

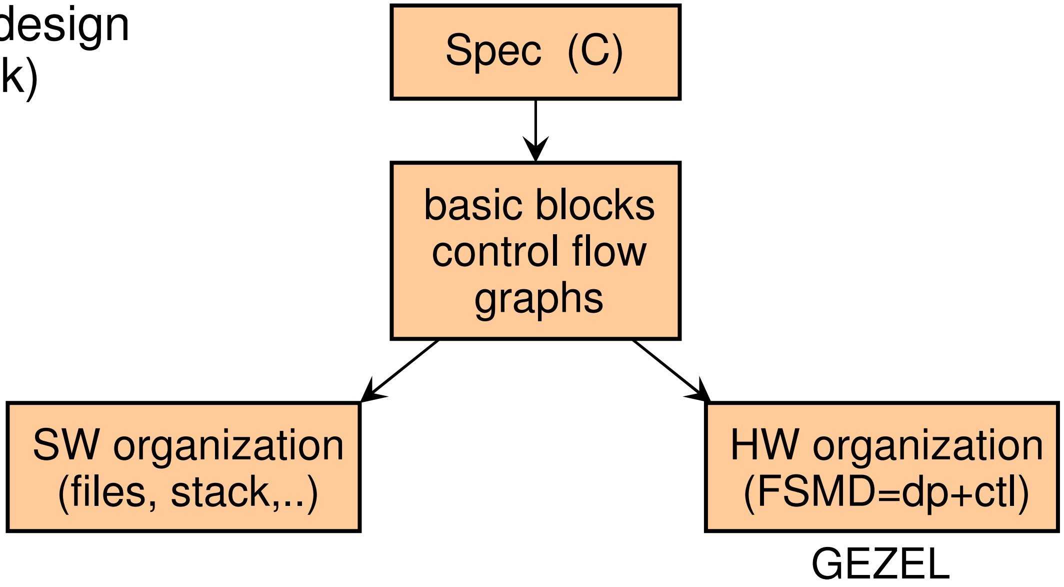
AD
Blackfin

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)

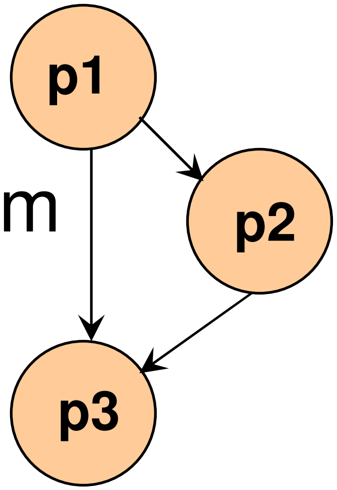
GEZEL

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

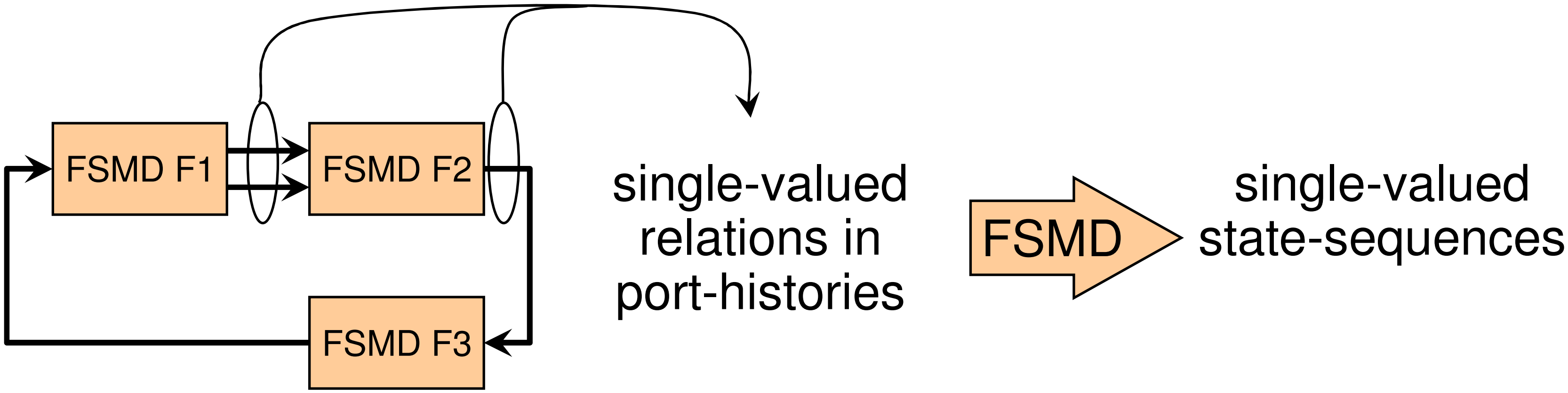
GEZEL yields deterministic HW



The Kahn Principle: systematic determinism

- System = Σ (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



single-valued relations in port-histories

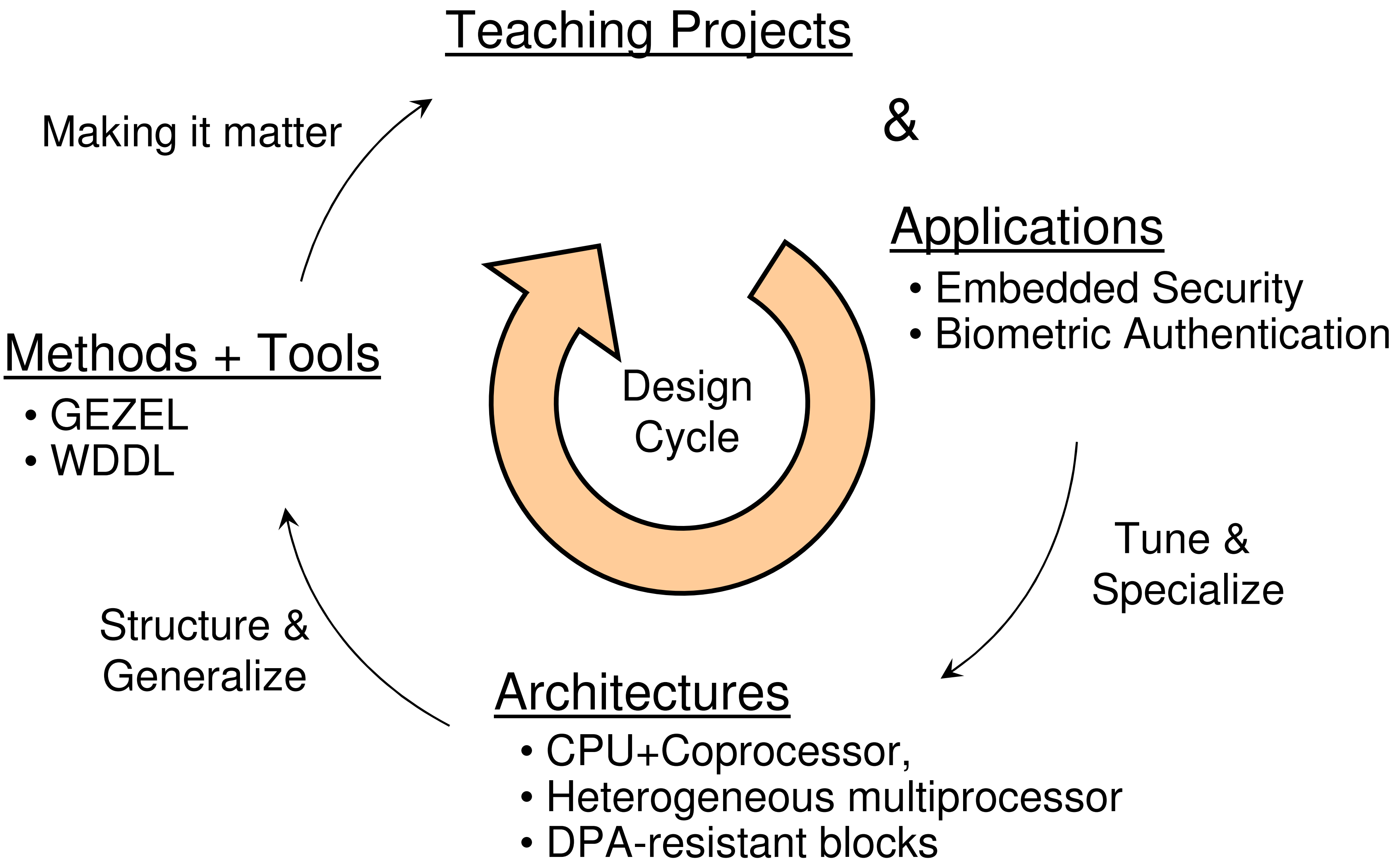
single-valued state-sequences

‘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



Teaching Projects

&

Applications

- Embedded Security
- Biometric Authentication

Tune & Specialize

Architectures

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

Structure & Generalize

Methods + Tools

- GEZEL
- WDDL

Making it matter

Design Cycle