# Energy efficient and reliable embedded computing systems

Bashir M. Al-Hashimi
bmah@ecs.soton.ac.uk

University
of Southampton

ICSA, School of Informatics, University of Edinburgh, Nov.15-2007

# Outline

- Overview of research activities focusing on
  - Automated low-power system-level design flows
  - Fault tolerant and low-power embedded systems
- Outline of other current projects
- Concluding remarks and future problems

# Power Minimization (background)

## Design Levels

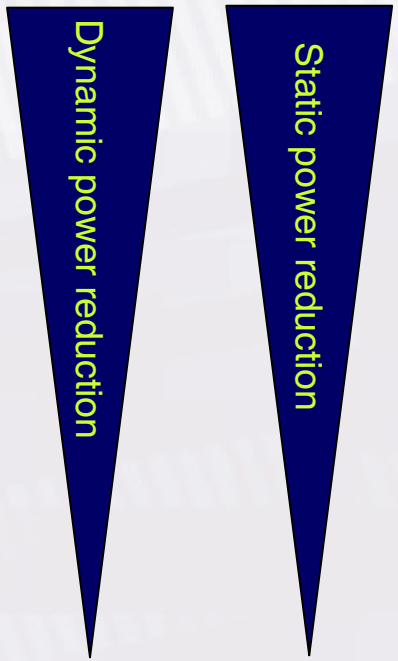

Behavioural (transformations, multiple Vt)

Architecture (clock gating, power gating)

Logic (pass-transistor)

Gate (min-size transistors, topology)

Technology (strained silicon, dual Vt process)



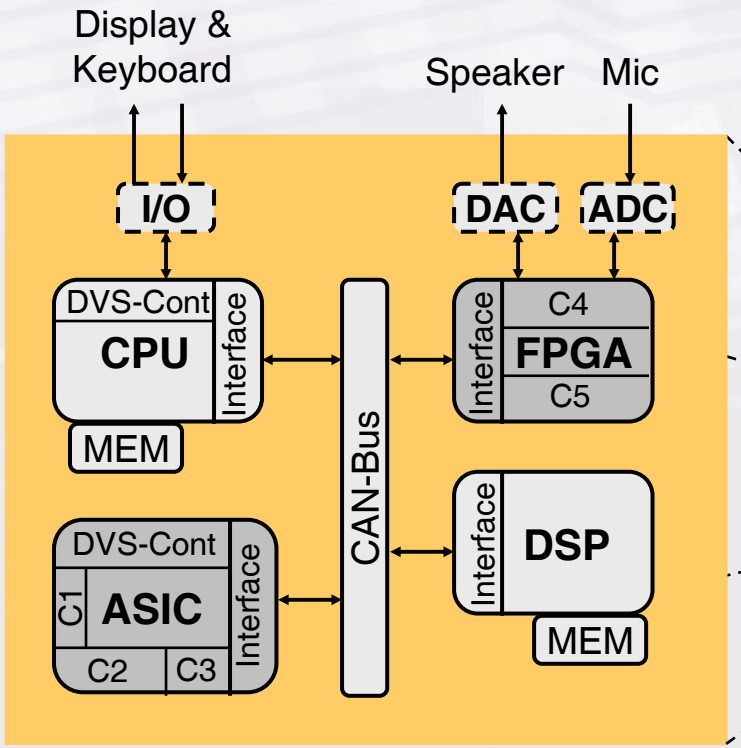

Dynamic power=0.5 $V^2_{dd}$ $F_{clk}$ $C_L$ $E_{sw}$

Leakage power (sub-threshold, gate oxide)

# Software Power Optimisation (background)

- Individual instructions: fundamental unit of SW, similar to gates in HW
- Program transformation techniques aims to reduce
  - use of instructions (MOV, JUMP,..) that consume large amount of supply current
  - memory access and unnecessary instruction fetching instruction involves memory consume energy > instructions involve registers
- Operating system techniques
  - Identify program regions where processor can be slowed down to reduce dynamic power through V/F scaling
  - Shut down unused parts of system to reduce leakage power

# Embedded Computing Systems, SoC, Multicore,..

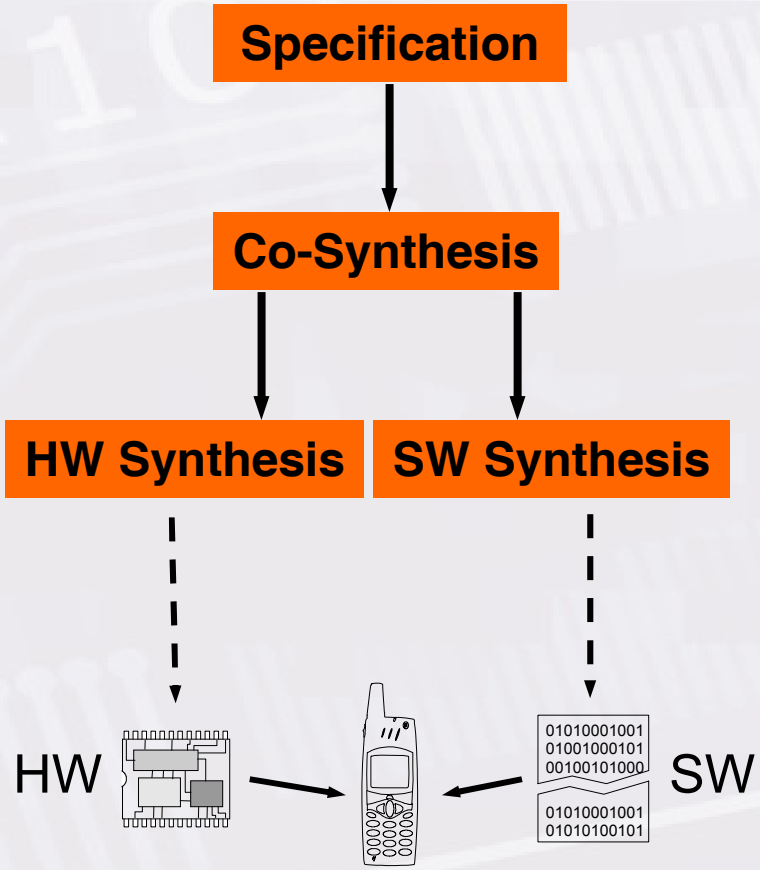



```
int main(int argc, char **argv) {
  Build_Interface();
  EventTrigger();
  while (!end_bs(&bs)) {
     sync = seek_sync(&bs, SYNC_WORD);
     ... ... ... ... ... ... ...
     decode_info(&bs, &fr_ps);
  }
  ... ... ... ...
}
```

```
void EventTrigger() {
   InitTriggerControl();
   while (Event) ReadEvent();
   switch (Event) {
      ... ... ... ... ...
      case 0: ... ... ...
      case 1: ... ... ...
   }
   ... ... ... ...
}
```

**System design requires optimisation in both *hardware (computation units, memory, communication)* and *software (application and system)***

# System-Level Design Flow*



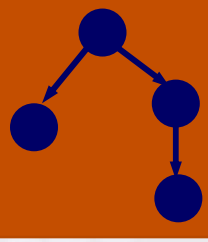

*M. Schmitz, B. Al-Hashimi, P. Eles, "System level design techniques for energy embedded systems", Kluwer Publishers, 2004

# Design Flow: Specification

Capturing system functionality using conceptual models

**Abstract Graph Representation:**

e.g.:-Finite State Machine
-Petri Net
-Task Graph

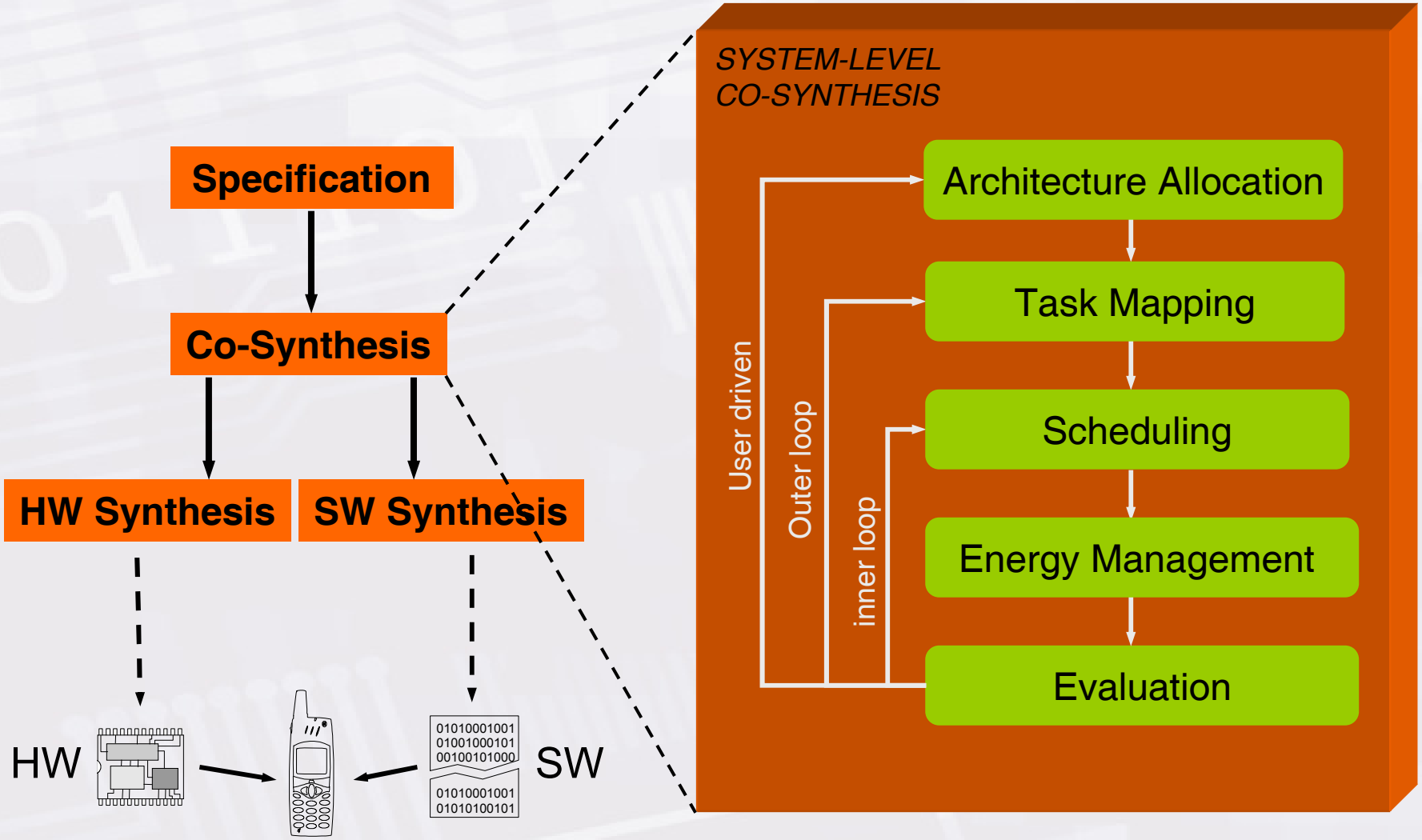


**High-Level Languages:**

e.g.: - SystemC
- VHDL / Verilog
- C / C++ / JAVA

```
for (i=0;i<x;i++) {
     a[i] += b[i];
}
```

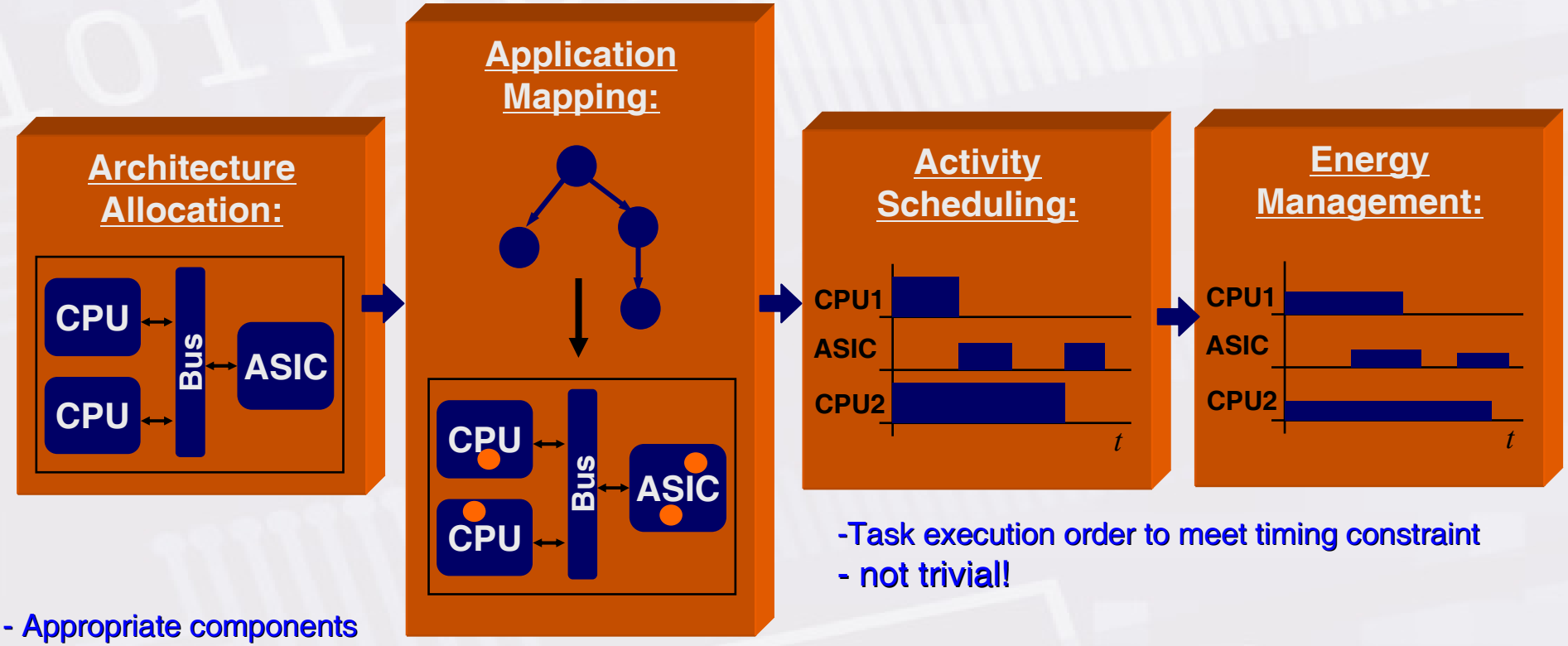
# Design Flow: Co-synthesis



# Design Flow: Co-Synthesis

## Identify the "most" suitable system architecture

sufficient performance to satisfy timing constraint, at the same time
cost and energy reduced to minimum




- Appropriate components
- HW > performance < energy SW

-Which tasks done in HW or SW
- size of task (coarse, fine)

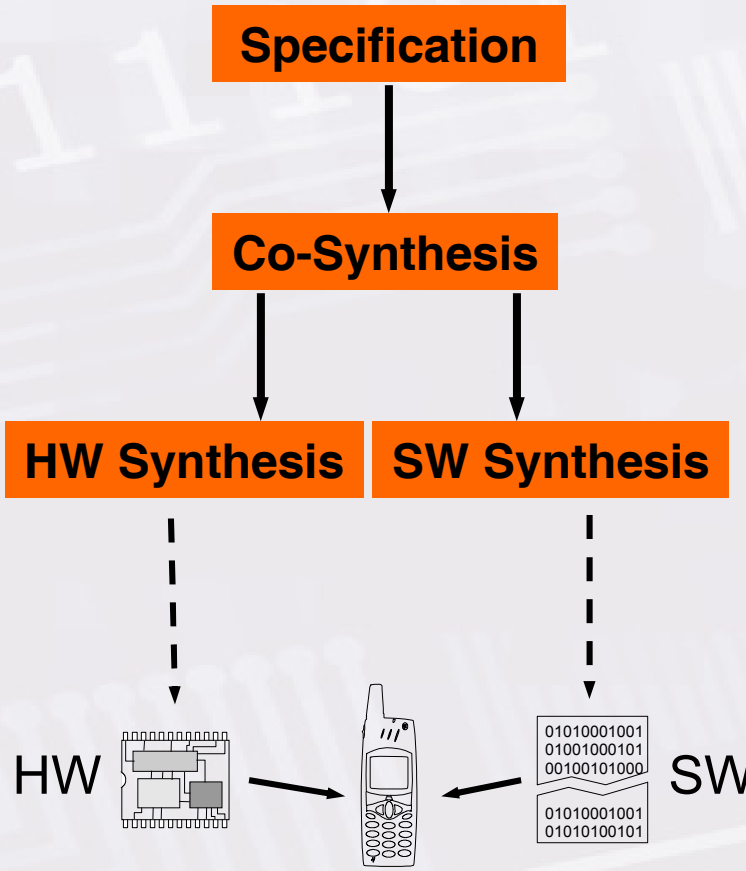
-Task execution order to meet timing constraint
- not trivial!

# Tasks Profiling

- Partitioning and Scheduling require tasks execution time and energy cost
  - In HW (HDL model of task)
    - Execution time through simulation
    - Energy estimation through power analysis on synthesised designs
  - In SW (coding of task)
    - Execution time through instruction-set simulator
    - Energy estimation= average power (pre characterised)*no. clock cycles*frequency

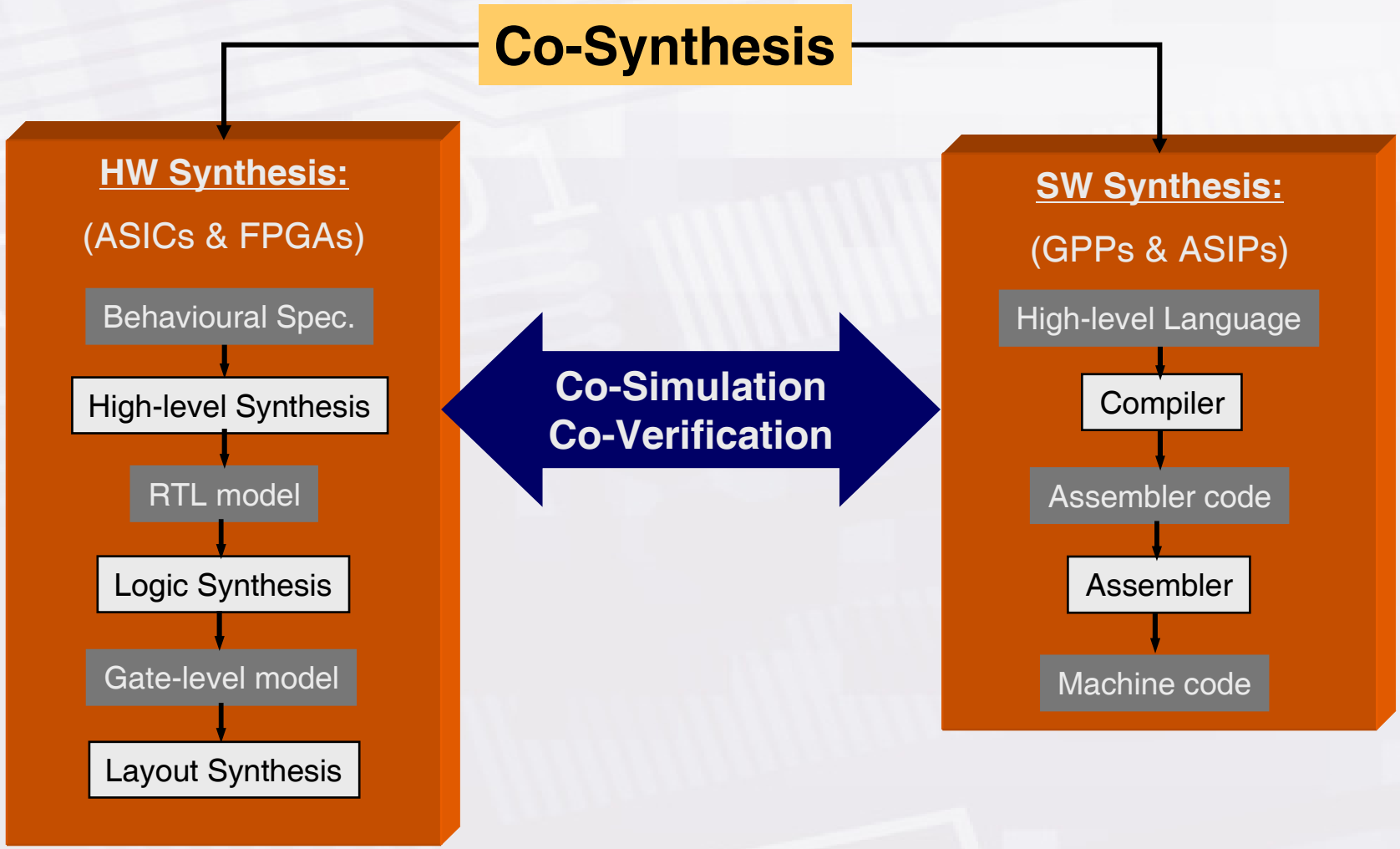
Commercial and academics timing and power estimation tools exist

# Design Flow: HW/SW Synthesis



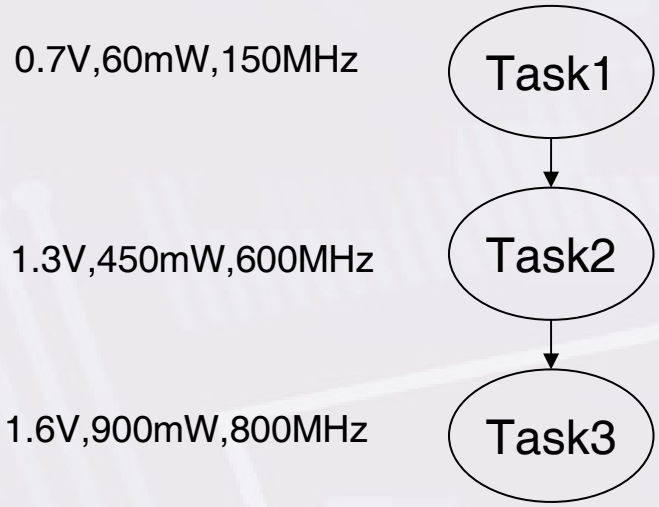

Transformation of system architecture into physical implementation

# Design Flow: HW/SW Synthesis

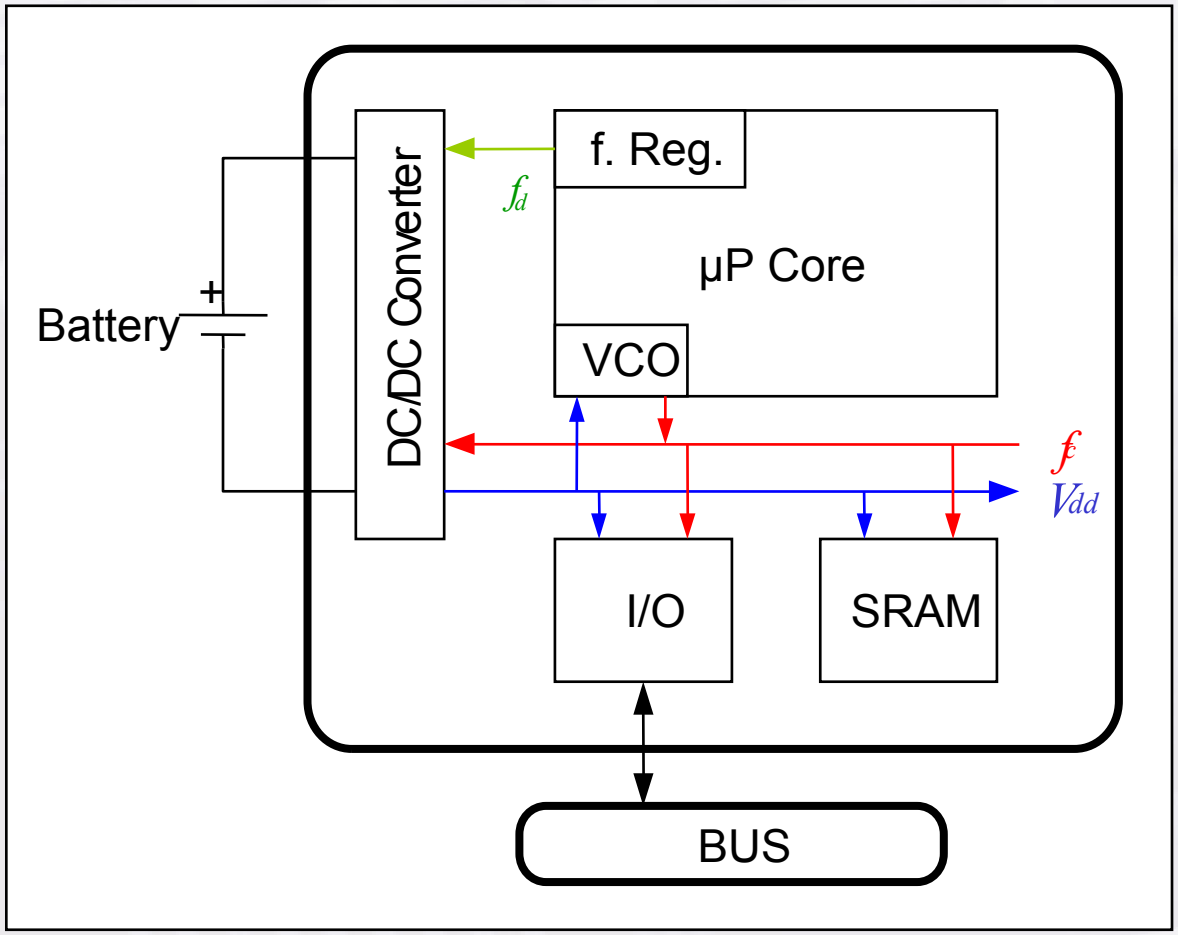


# Energy Management

- Dynamic Power Management
  - Idle components shutdown
- Dynamic Voltage and Frequency Scaling
  - Non-uniform workload, MPEG order of magnitude >MP3
  - Introduce slack times (deadline-finish time) and used to reduce processor performance to save energy
  - Adapt processor performance through V/F scaling



# Energy Management: DVS-Processor

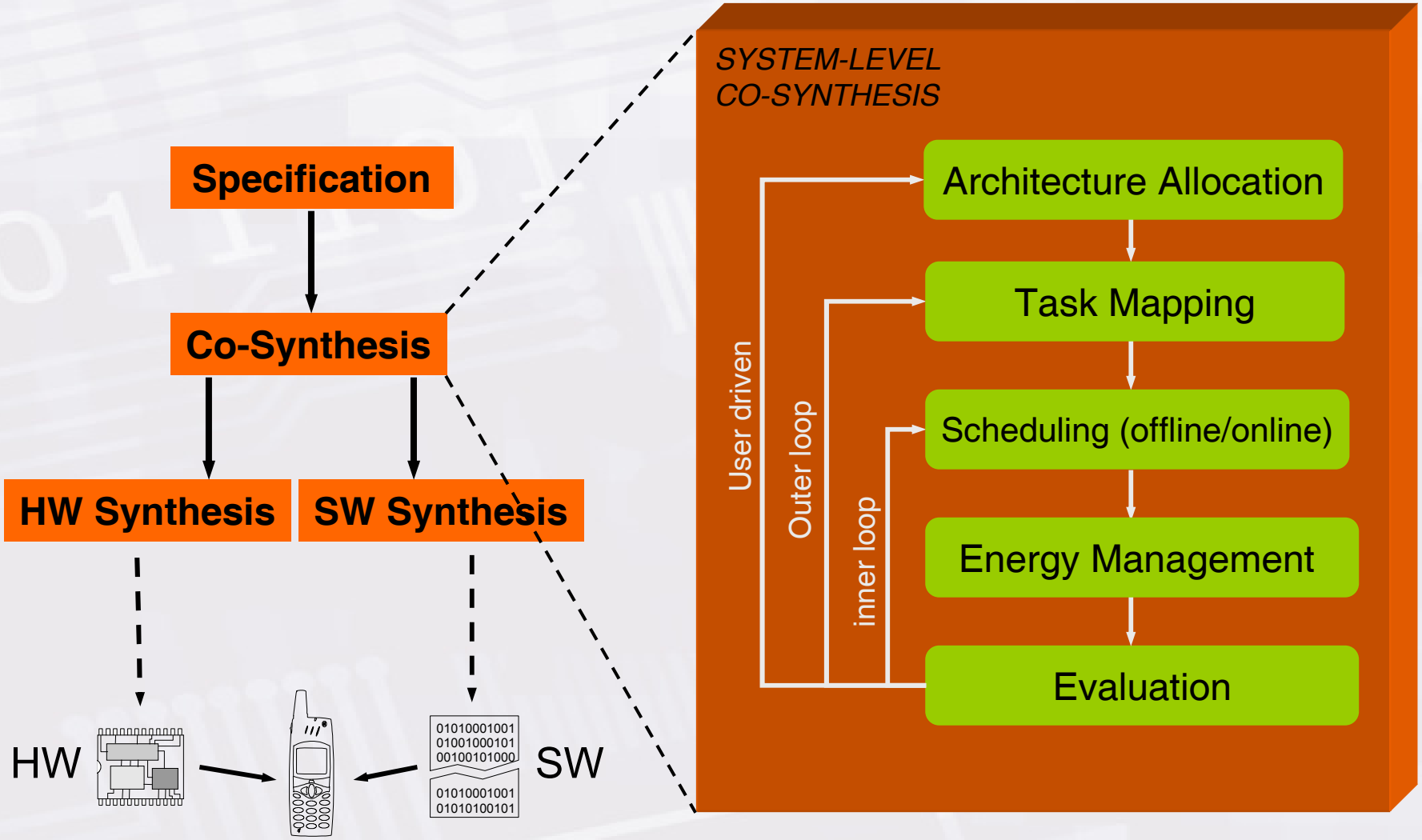



Examples:

- ARM (IEM)
- AMD (PowerNow)
- INTEL (Xscale)
- TRANSMETA

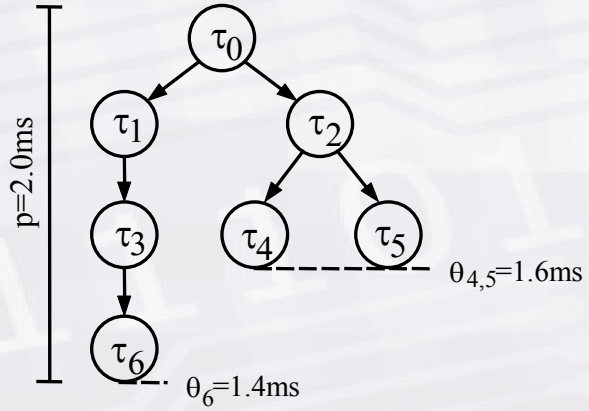
Intel Xscale
1.1V/150MHz
1.6V/600MHz
2.0V/800MHz

# Design Flow: Co-synthesis

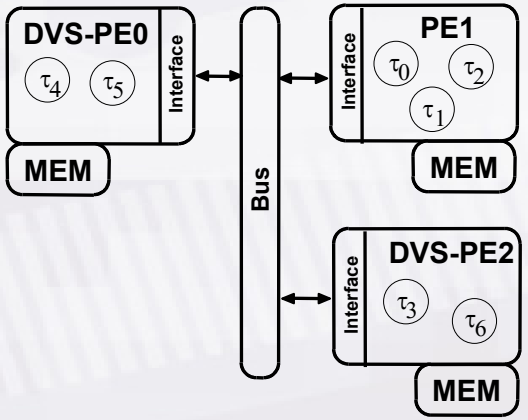



-Mapping and Scheduling NP-hard problems
- Heuristic methods employed

# DVS Example (Scheduling)




Task graph

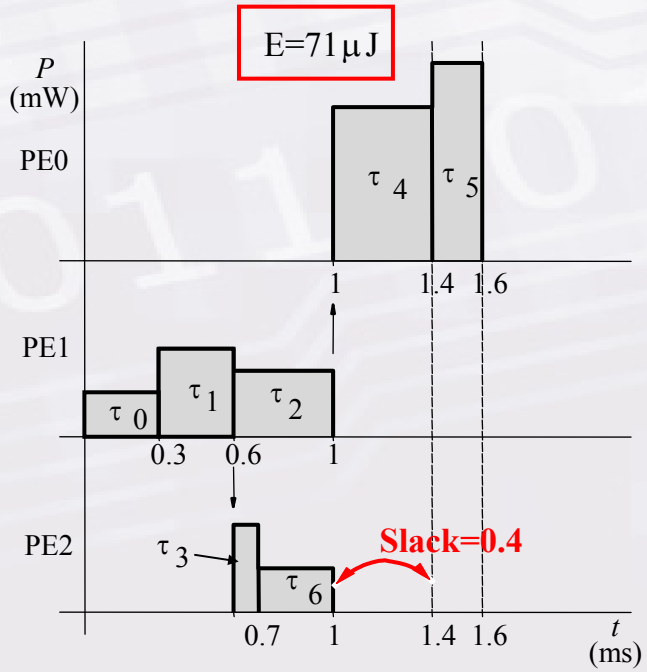



Target architecture and mapping

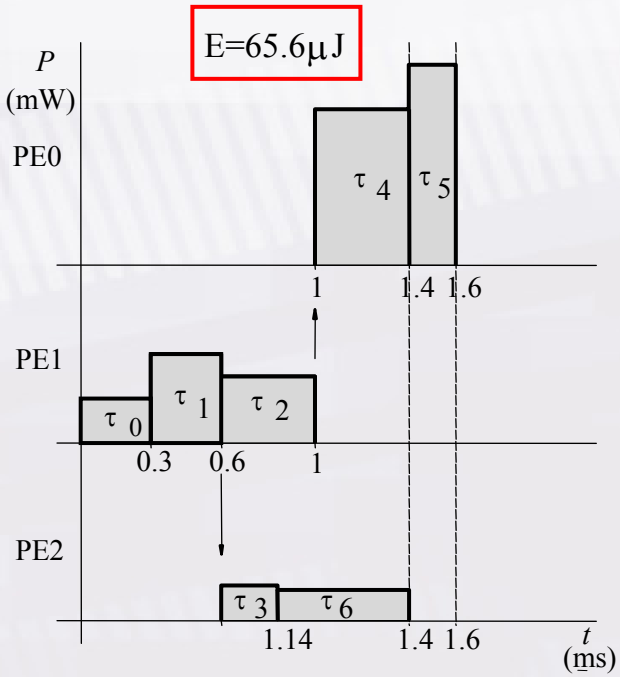
| Task | Exe. Time (ms) | Power (mW) | Mapped to |
|---|---|---|---|
| τ0 | 0.3 | 10 | PE1 |
| τ1 | 0.3 | 20 | PE1 |
| τ2 | 0.4 | 15 | PE1 |
| τ3 | 0.1 | 40 | PE2 |
| τ4 | 0.4 | 70 | PE0 |
| τ5 | 0.2 | 90 | PE0 |
| τ6 | 0.3 | 20 | PE2 |

Task properties (communication cost=0)

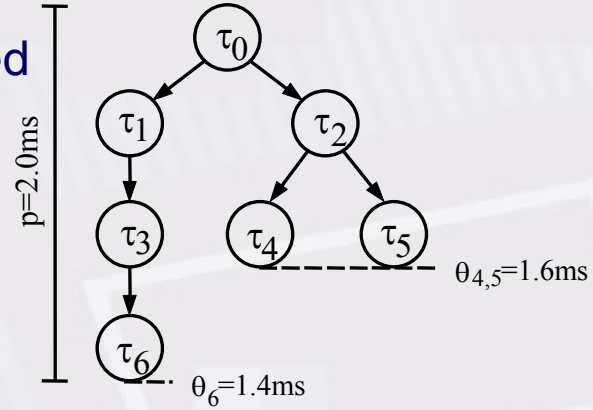
# DVS Example (cont..)




Apply DVS




Off-line schedule executed at nominal voltage (3.3V)/frequency

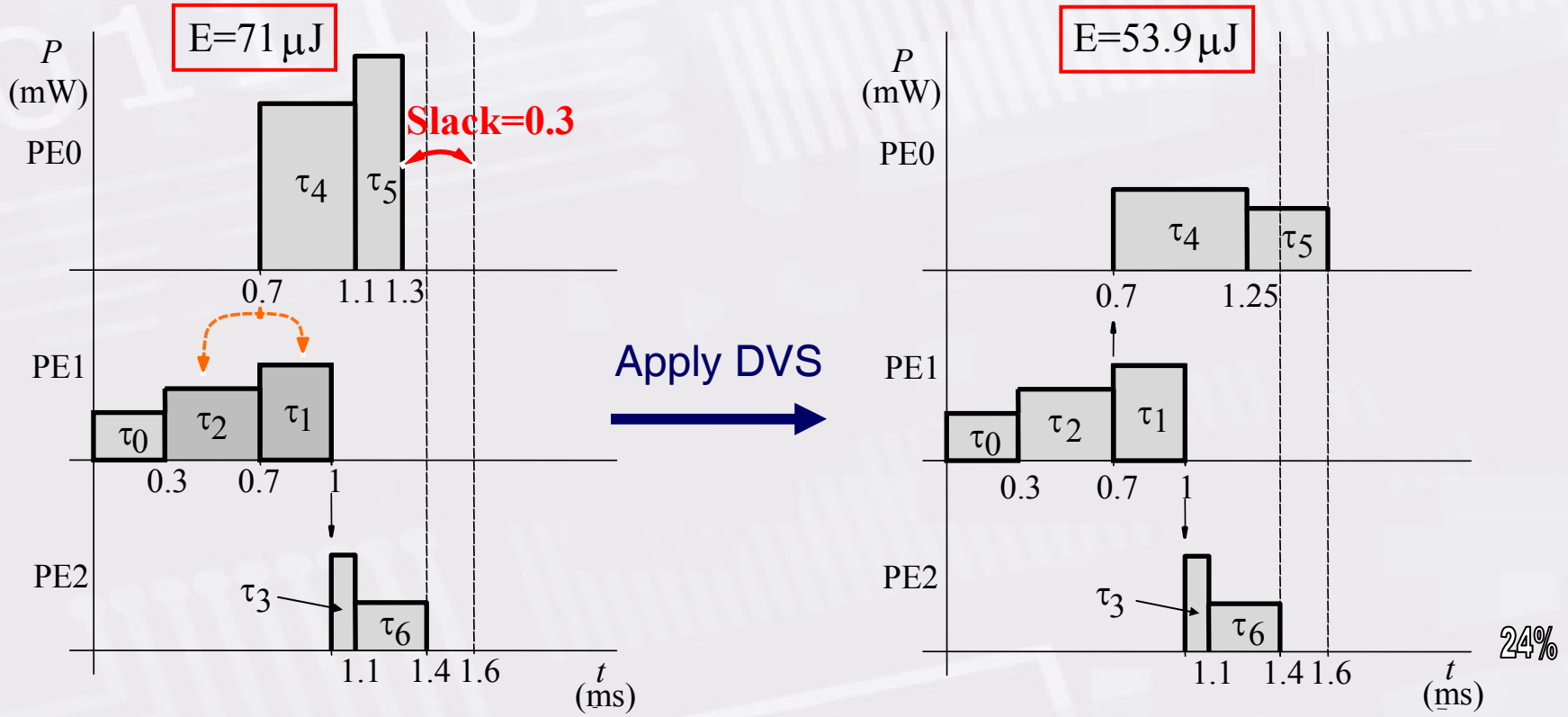



Scaled execution (task 3 and 6)

V3=2.1V, V6=2.34V, 8%

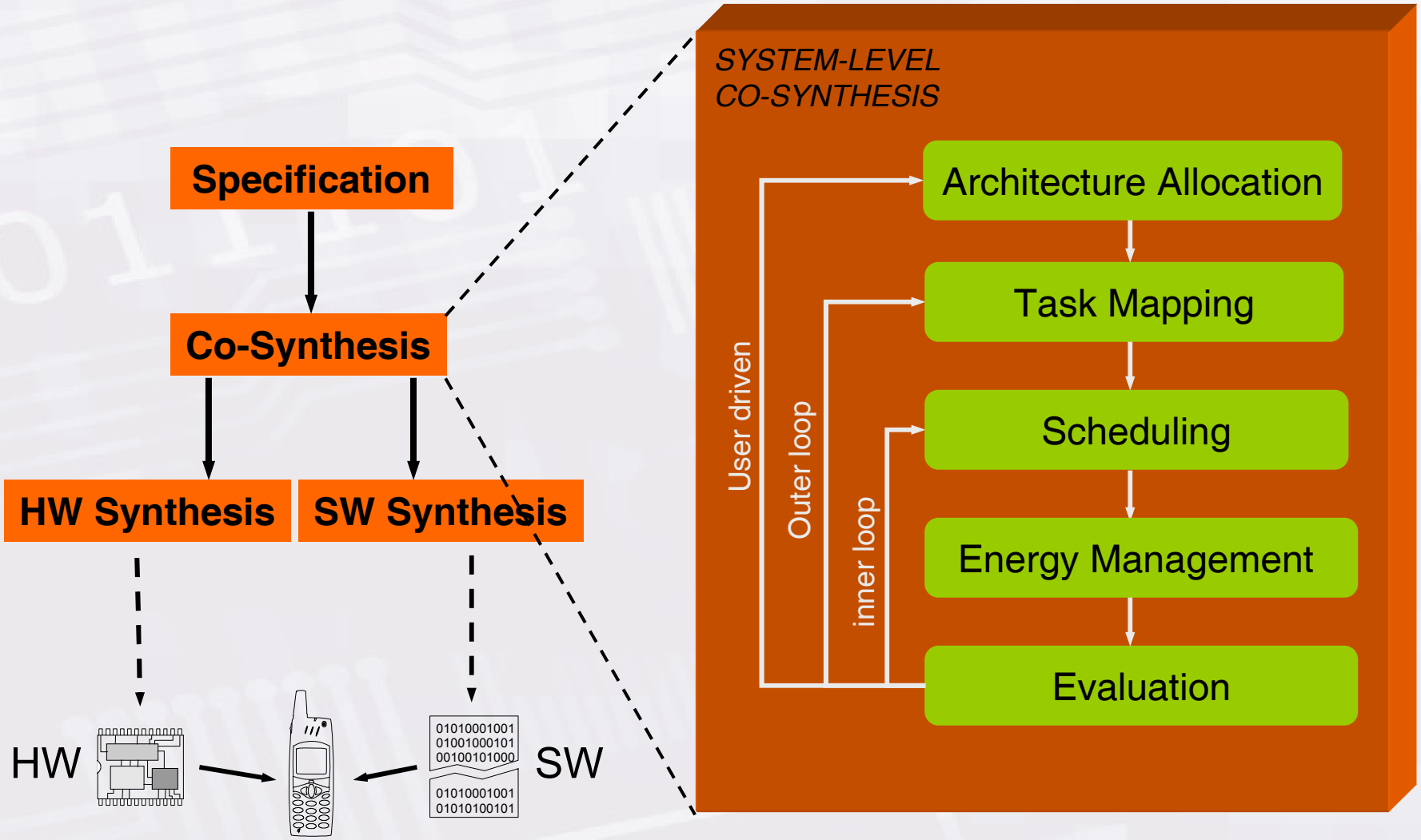
# DVS-Schedule Optimisation

- To increase the possible energy savings we can further optimise the execution order



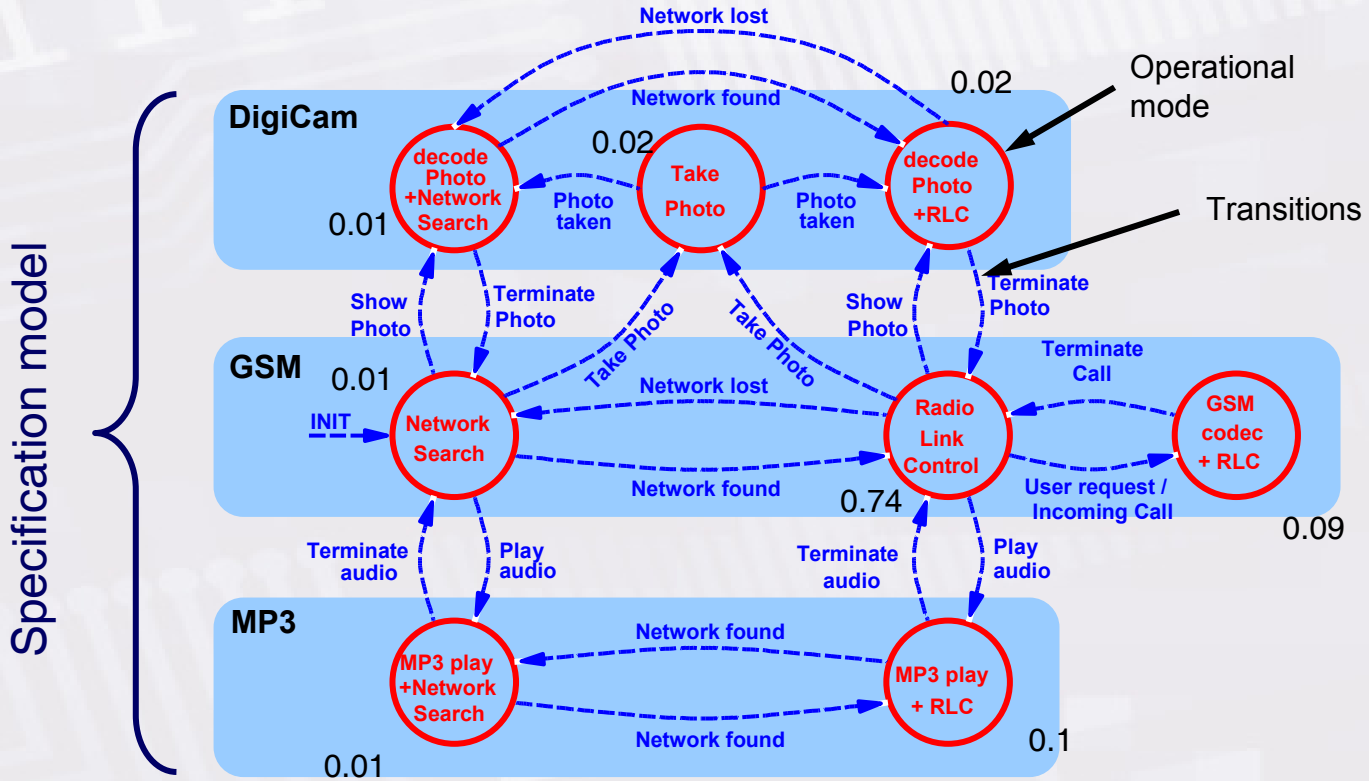

*Schmitz, M. T., Al-Hashimi, B. M. Eles, P. Iterative schedule optimisation for voltage scalable distributed embedded systems. *ACM Transactions on Embedded Computing Systems* 3(1):pp. 1-36, Feb.04.

# Design Flow: Co-synthesis



# Multi-Mode Embedded Systems

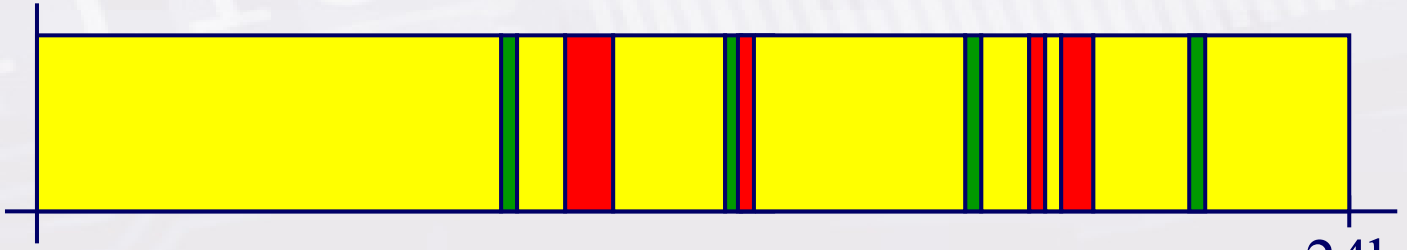
- Emerging embedded systems work across a set of different interacting applications
- **Smart Phone** consisting of three applications:
  - GSM Phone, MP3 Player, Digital Camera (JPEG compression/decompression)
- Specification model captures both mode interaction and functionality




*Schmitz, M. T., Al-Hashimi, B. M. , Eles, P. Co-Synthesis of Energy-Efficient Multi-Mode Embedded Systems with Consideration of Mode Execution Probabilities. *IEEE TCAD, 24(2),pp.153-170, Feb.05*

# Mode Execution Probabilities

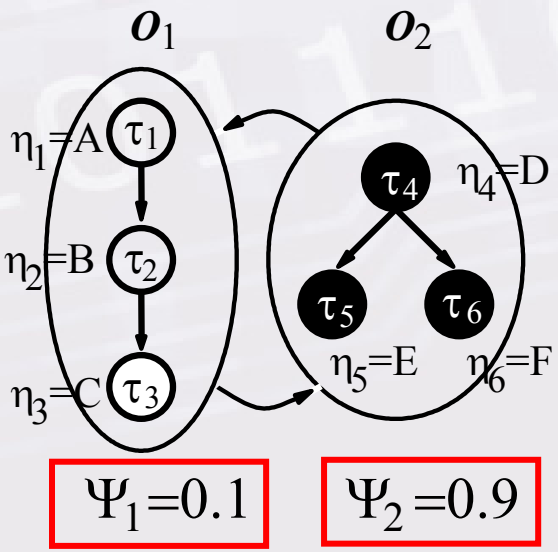
**Typical mode activation profile of a mobile phone:**




| | Search | RLC | Calling |
|---|---|---|---|
| **Execution Prob.:** | **2%** | **88%** | **10%** |

**Depending on the application, the time spend in a certain operational mode is user-typical!**

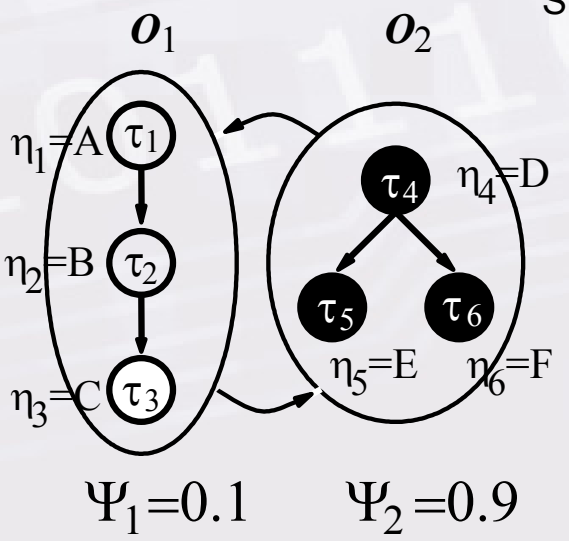
# Multi-Mode Example



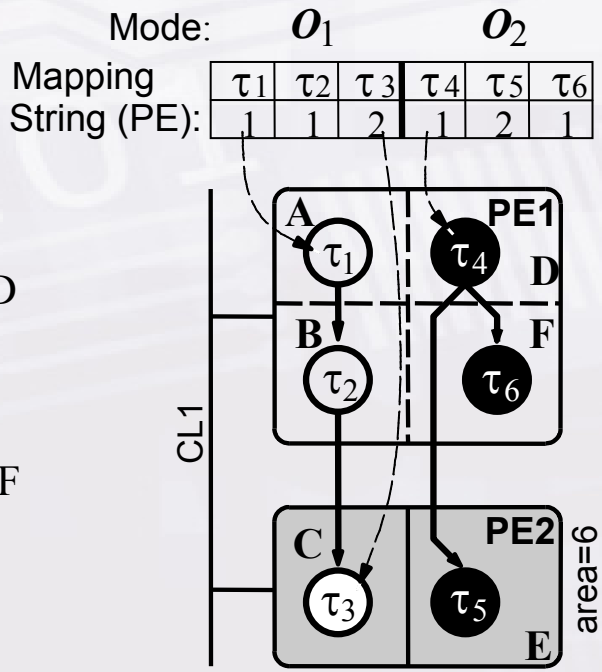

Application specified
by two interacting modes

| Task type | PE1 (SW) | | PE2 (HW) | | |
|---|---|---|---|---|---|
| | exe. time (ms) | dyn. energy (mJ) | exe. time (ms) | dyn. energy (mJ) | area ($mm^2$) |
| A | 20 | 10 | 2 | 0.010 | 2.40 |
| B | 28 | 14 | 2.2 | 0.012 | 3.00 |
| C | 32 | 16 | 1.6 | 0.023 | 2.75 |
| D | 26 | 13 | 3.1 | 0.047 | 2.45 |
| E | 30 | 15 | 1.8 | 0.015 | 2.10 |
| F | 24 | 14 | 2.2 | 0.032 | 2.80 |

# Multi-Mode Example (Cont..)

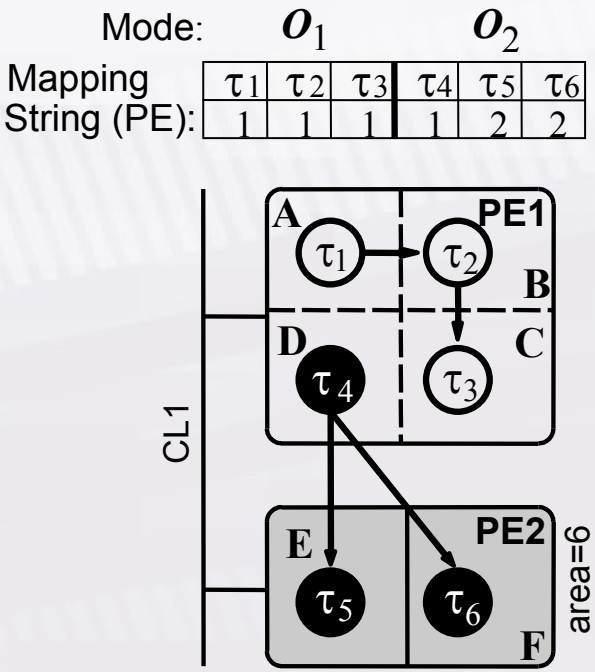



Application specified
by two interacting modes




Optimised without
mode consideration

26.72mJ




Optimised with
mode consideration

15.74mJ, 41%

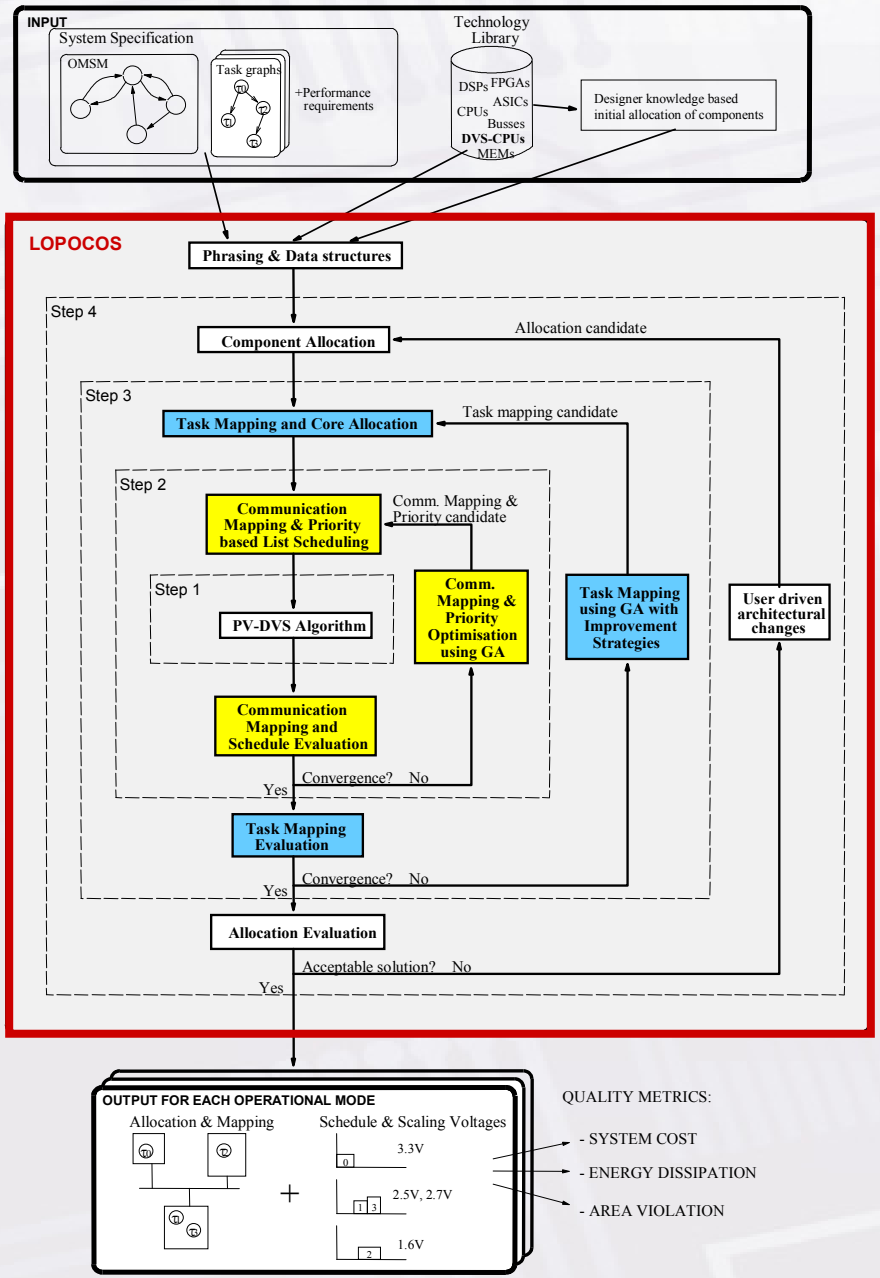
# Leakage power

- Reduced using adaptive body biasing (ABB) by increasing the processor threshold voltage and decreasing its frequency
- Simultaneous reduction of dynamic and leakage power in MPSoC*
- Considering the overheads (energy, time) imposed by changing voltage levels

*Andrei, Eles, Peng, Schmitz, Al-Hashimi, "Energy optimization of multiprocessor systems on chip by voltage selection", IEEE TVLSI, March 2007

# LOPOCOS

(Low-Power Co-Synthesis)

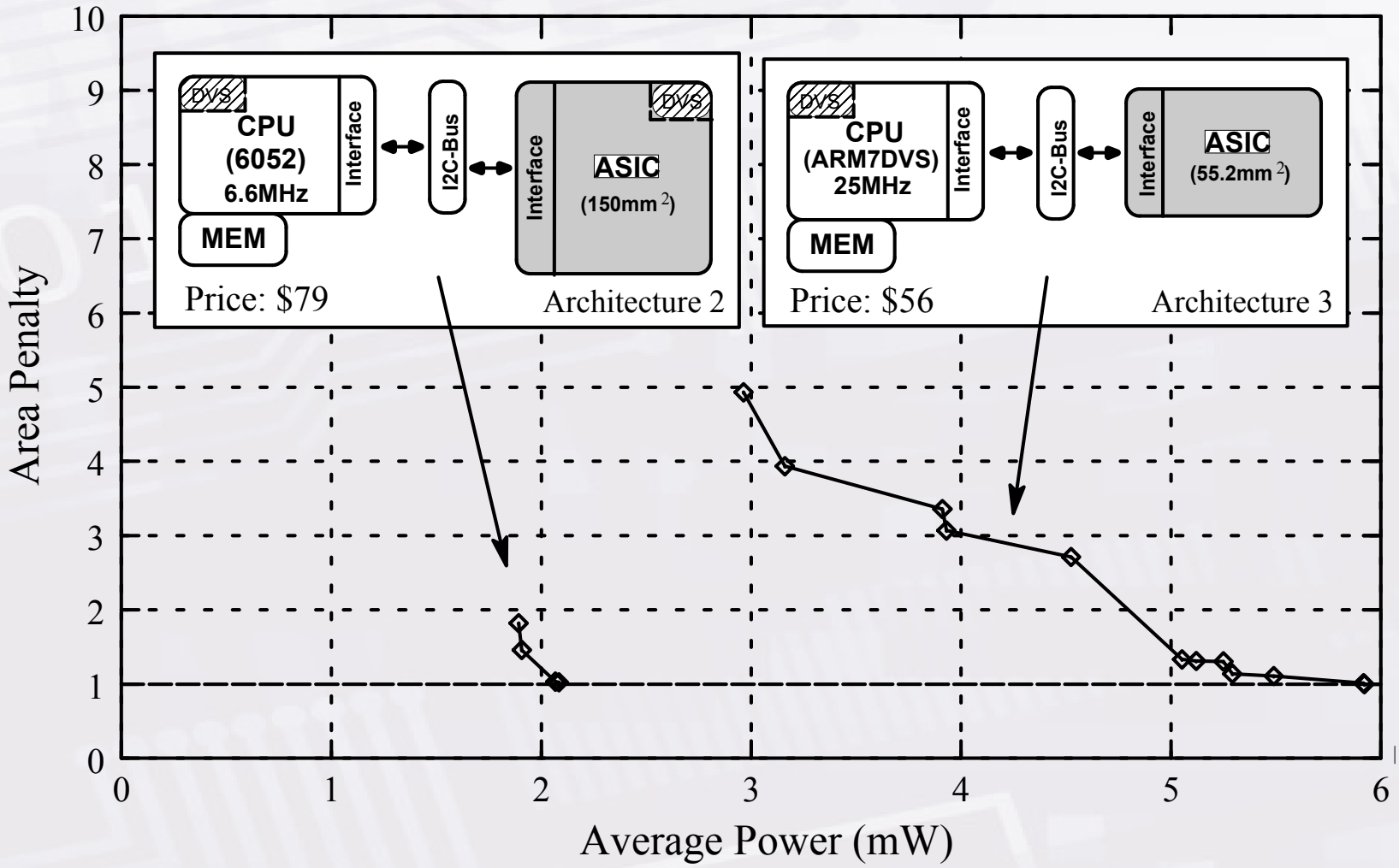



- Explore different system architectures
  - Optimisation targets
    - Energy-efficiency
    - Cost

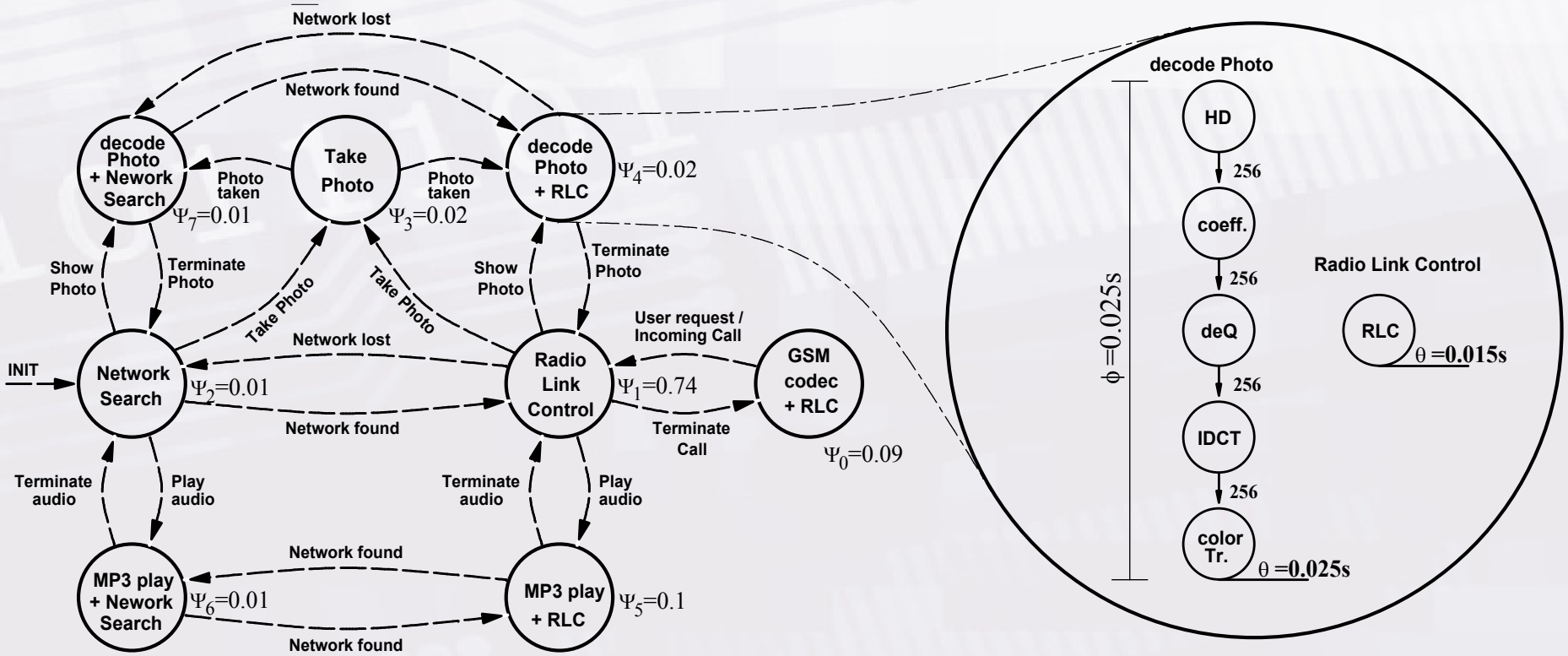
# Smart Phone: Case Study

- Smart Phone Specification
  - GSM Phone
  - MP3 player
  - Digital Camera
- System price: < $120
- Power consumption: < 1.6mW

# Smart Phone: Case Study (cont..)

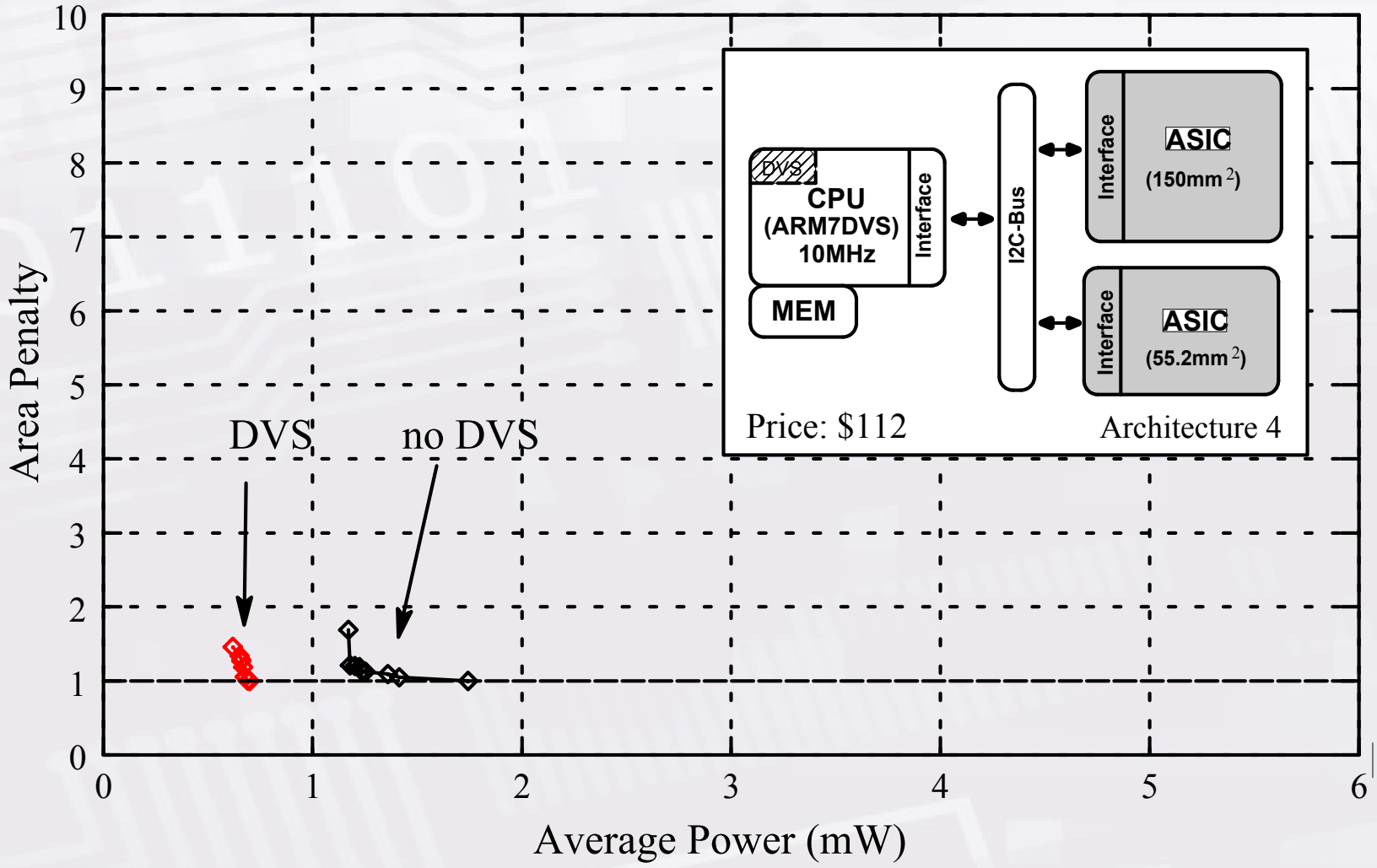


# Multi-Mode Embedded Systems (cont..)



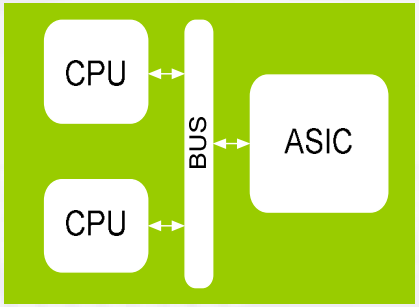

Schmitz, M. T., Al-Hashimi, B. M. , Eles, P. Co-Synthesis of Energy-Efficient Multi-Mode Embedded Systems with Consideration of Mode Execution Probabilities. *IEEE TCAD, 24(2),pp.153-170, Feb.05*

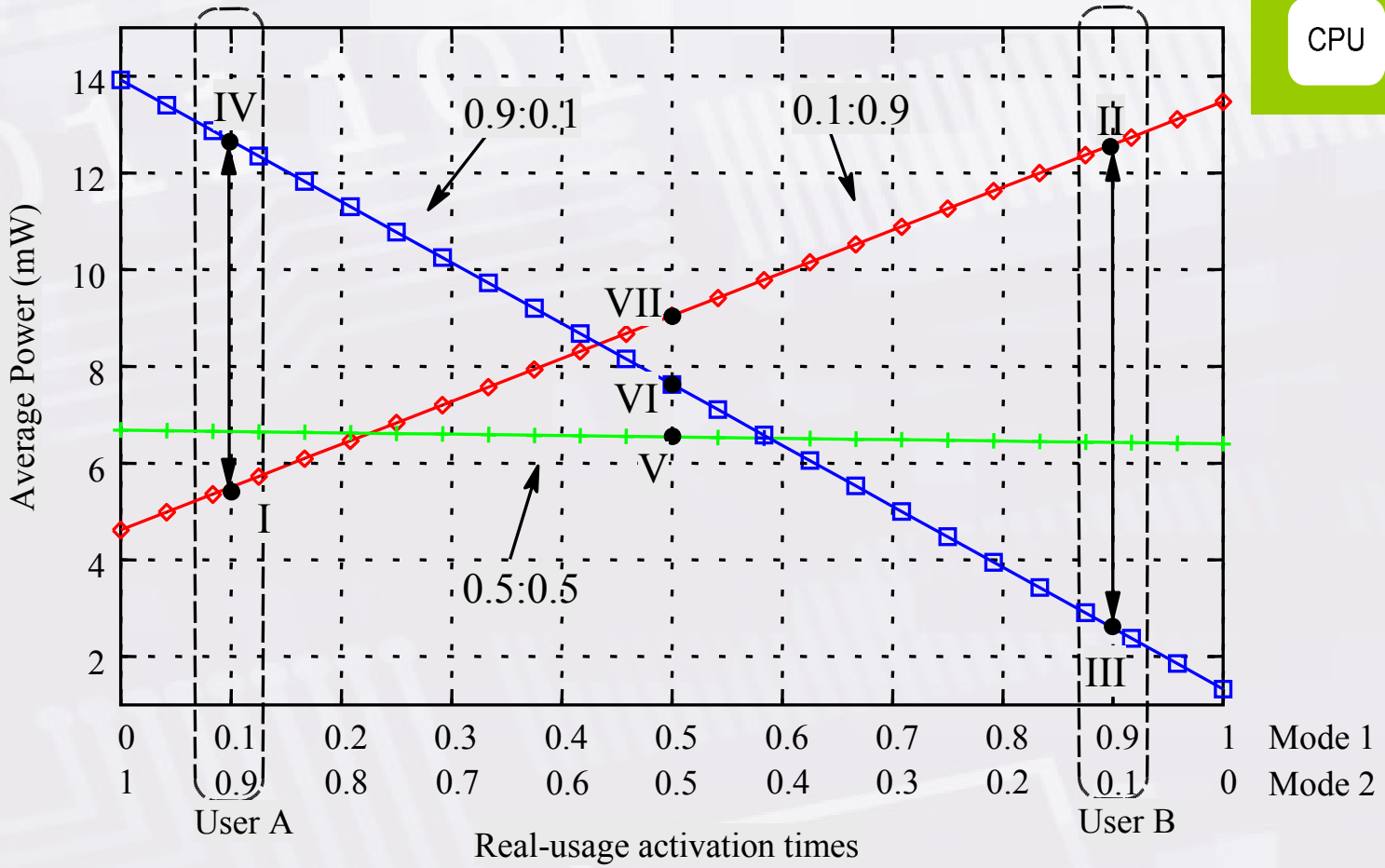
# Smart Phone: Case Study (cont..)



# LOPOCOS Synthesis Results

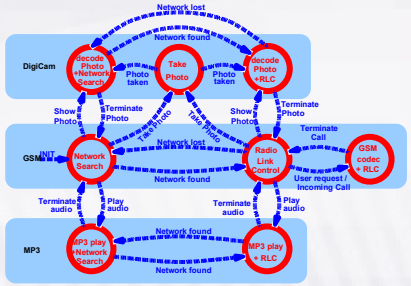
- Influence of the user behaviour

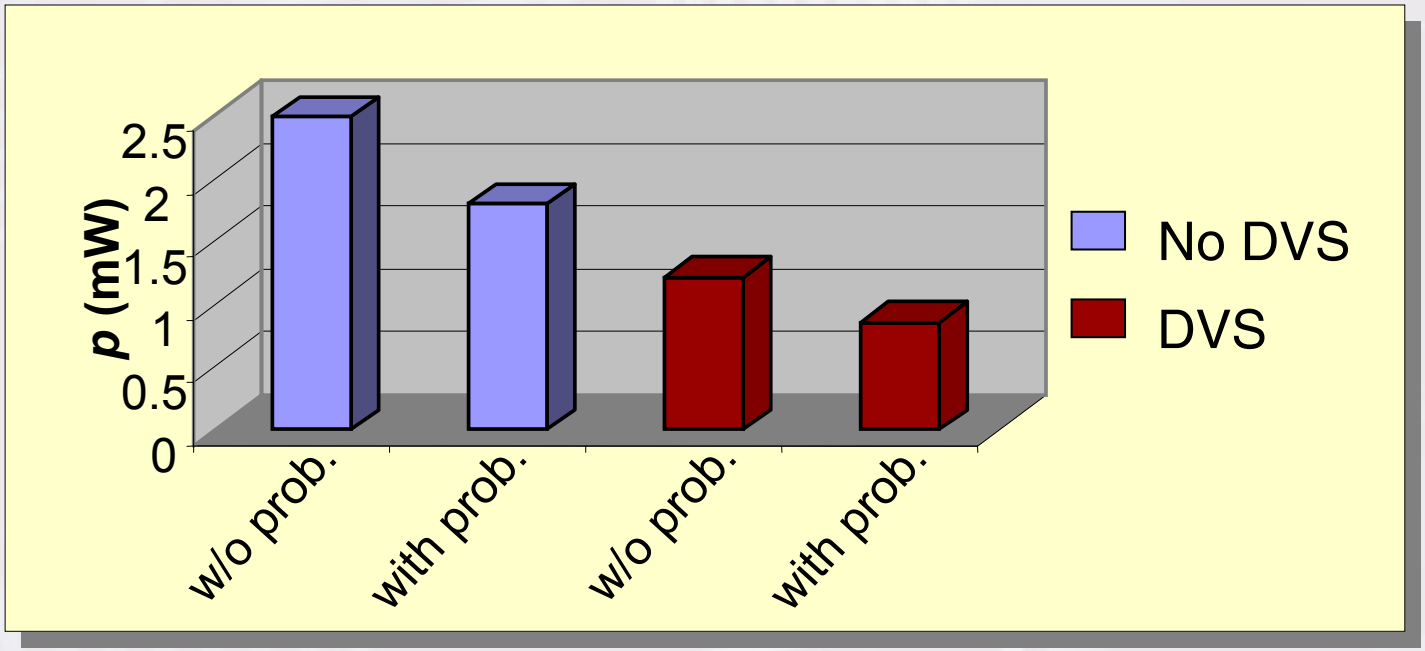


# Smart Phone: Case Study (cont..)

- Specification consists of
  - 3 applications 8 operational modes, w/o, 100/8% prob.
  - 5—88 tasks and 0—137 communications
- Finding the "best" solution took less than 8 hours



# Reliable and Energy Efficient Embedded Computing Systems

Ejlali, Schmitz, Al-Hashimi, Miremadi, Rosinger, " Combined Time and Information Redundancy for SEU-Tolerance in Energy-Efficient Real-Time Systems", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(4), pp.323-335, April 2006

University of Southampton

ICSA, School of Informatics, University of Edinburgh, Nov.15-2007

# Background

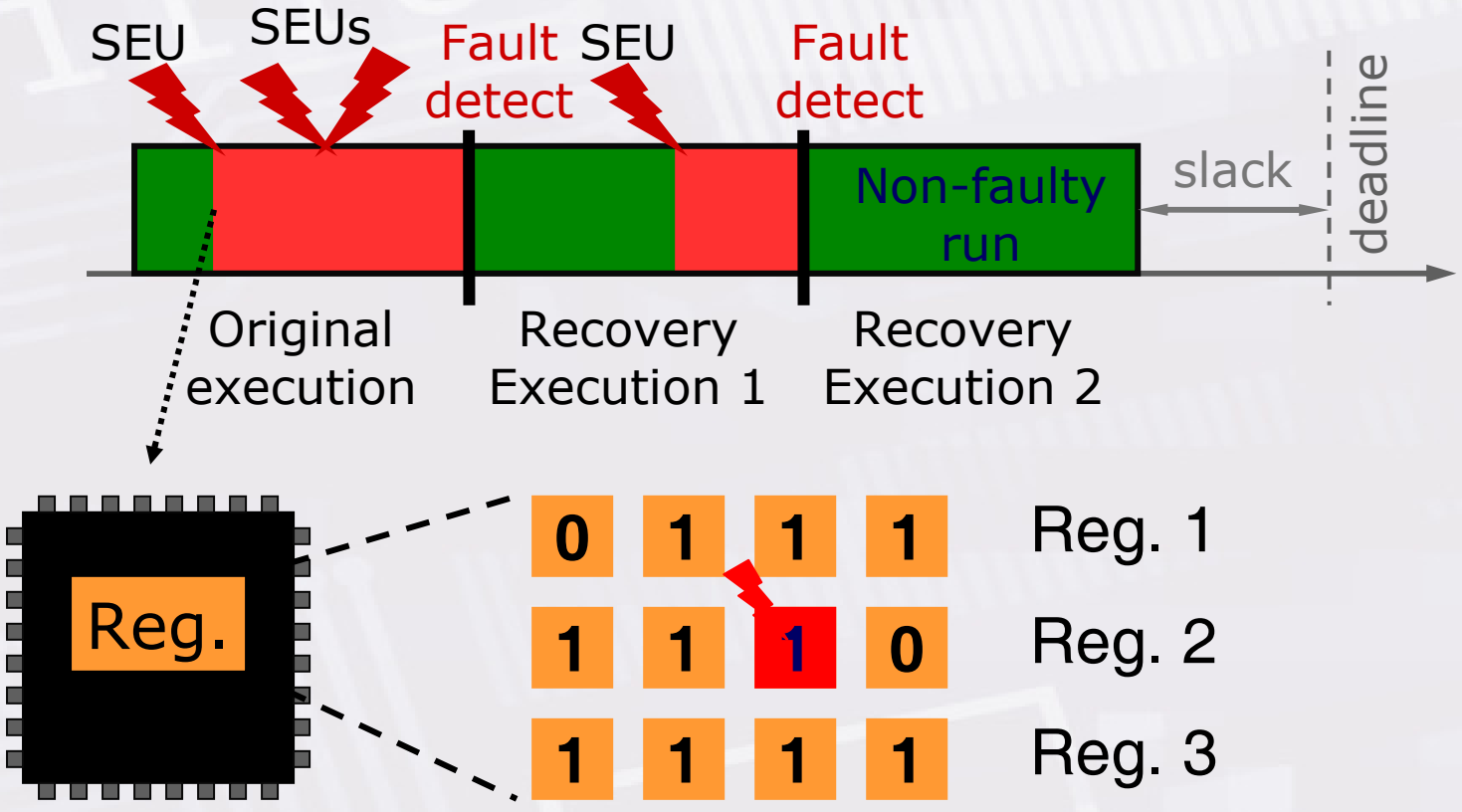
- How to design embedded computing systems that can tolerate faults (transient, soft, SEU, bit-flip) is becoming important
  - Demand from industry (high reliability even in commodity microprocessors)
  - Technology scaling and power management are making designs more sensitive to faults (SEU sensitivity increases by 1-2 order of magnitude as Vdd reduces by 1V)
- Additional dimension to an already complex design problem, need to be done carefully?

# Fault Tolerance

- Numerous techniques exists in Fault Tolerant Computing
  - Can they be applied directly to embedded computing systems?
  - Which FT technique or combination of techniques best match to specific requirements ?
- Same principles but different constraints
  - consume as little power as possible
  - cost (silicon area, execution time) as little as possible
  - Little or no performance degradation

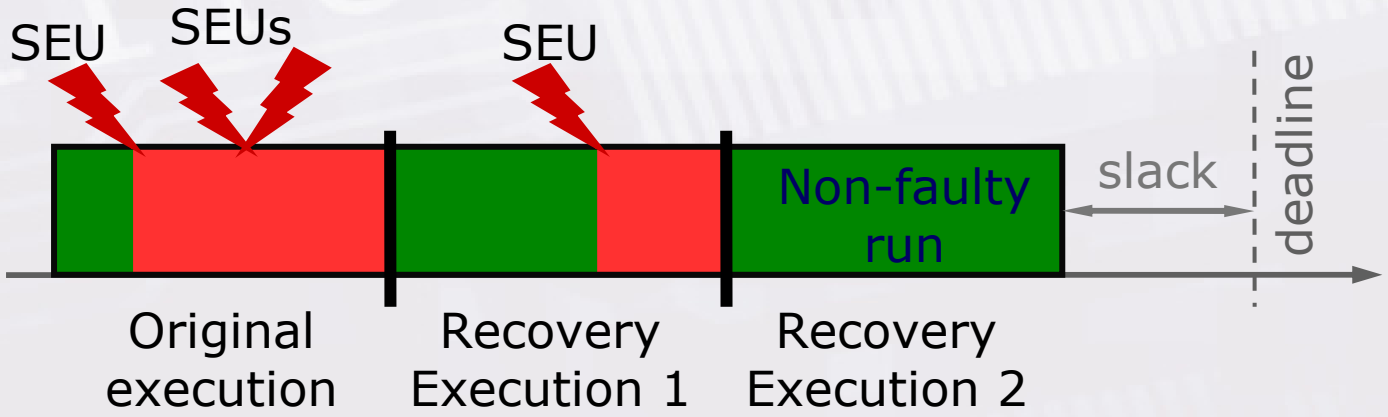
# Fault-Tolerance: Time Redundancy

- Rollback recovery (re-execution of faulty tasks)



# Fault-Tolerance: Time Redundancy

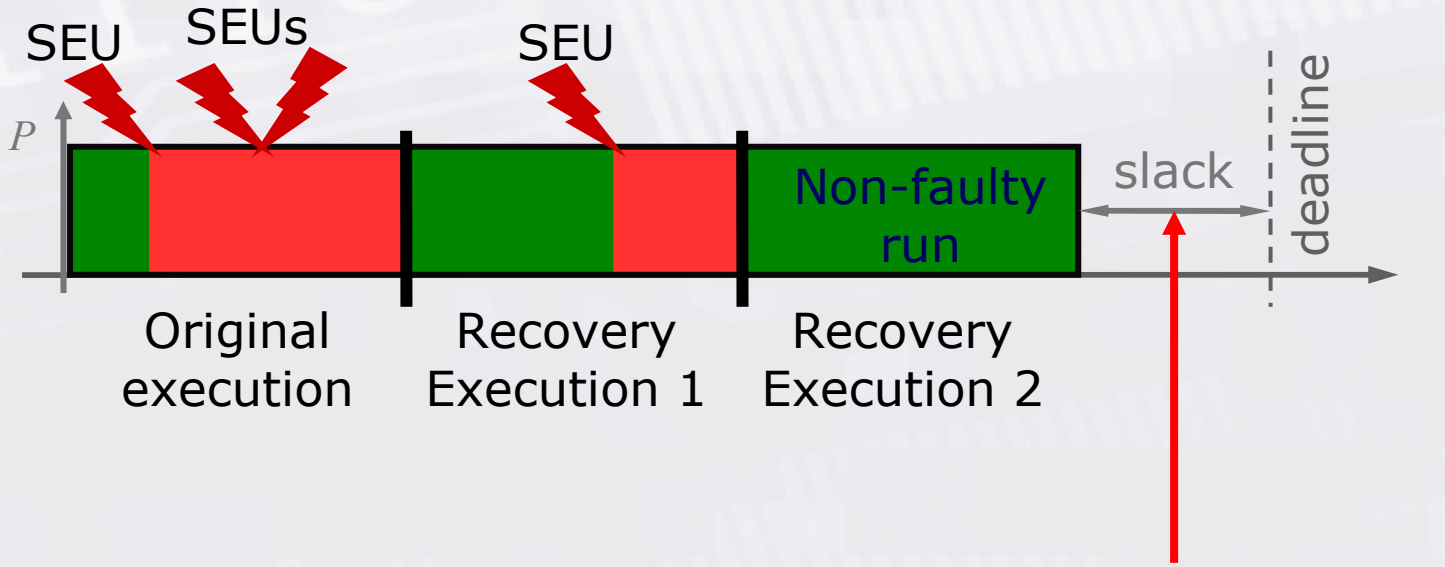
- Rollback recovery (re-execution of faulty tasks)




- Time reserved for rollback-recoveries affects fault-tolerance and energy saving

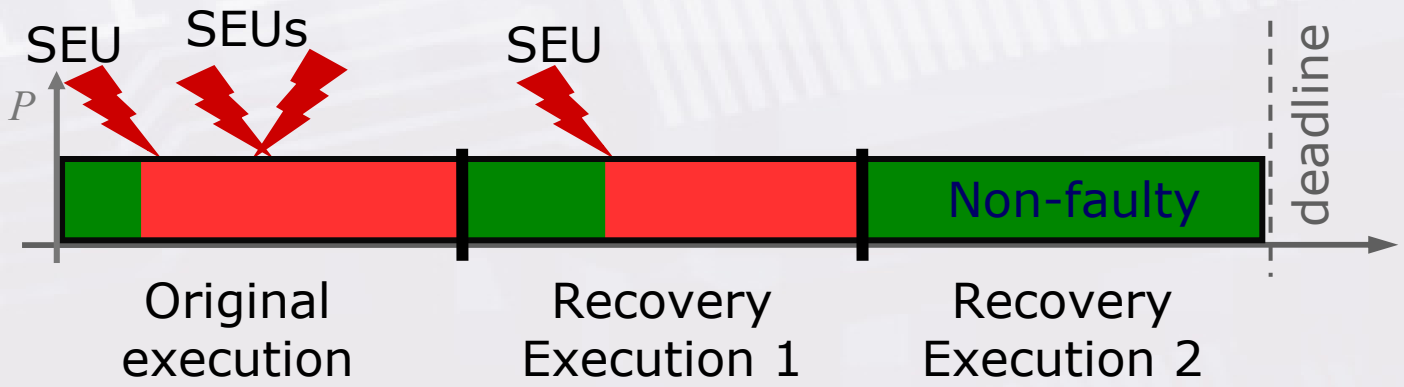
# Dynamic Voltage Scaling (DVS)

- Energy depends quadratic on frequency/voltage




Task executions can be extended

# Fault tolerance/Energy Trade-off



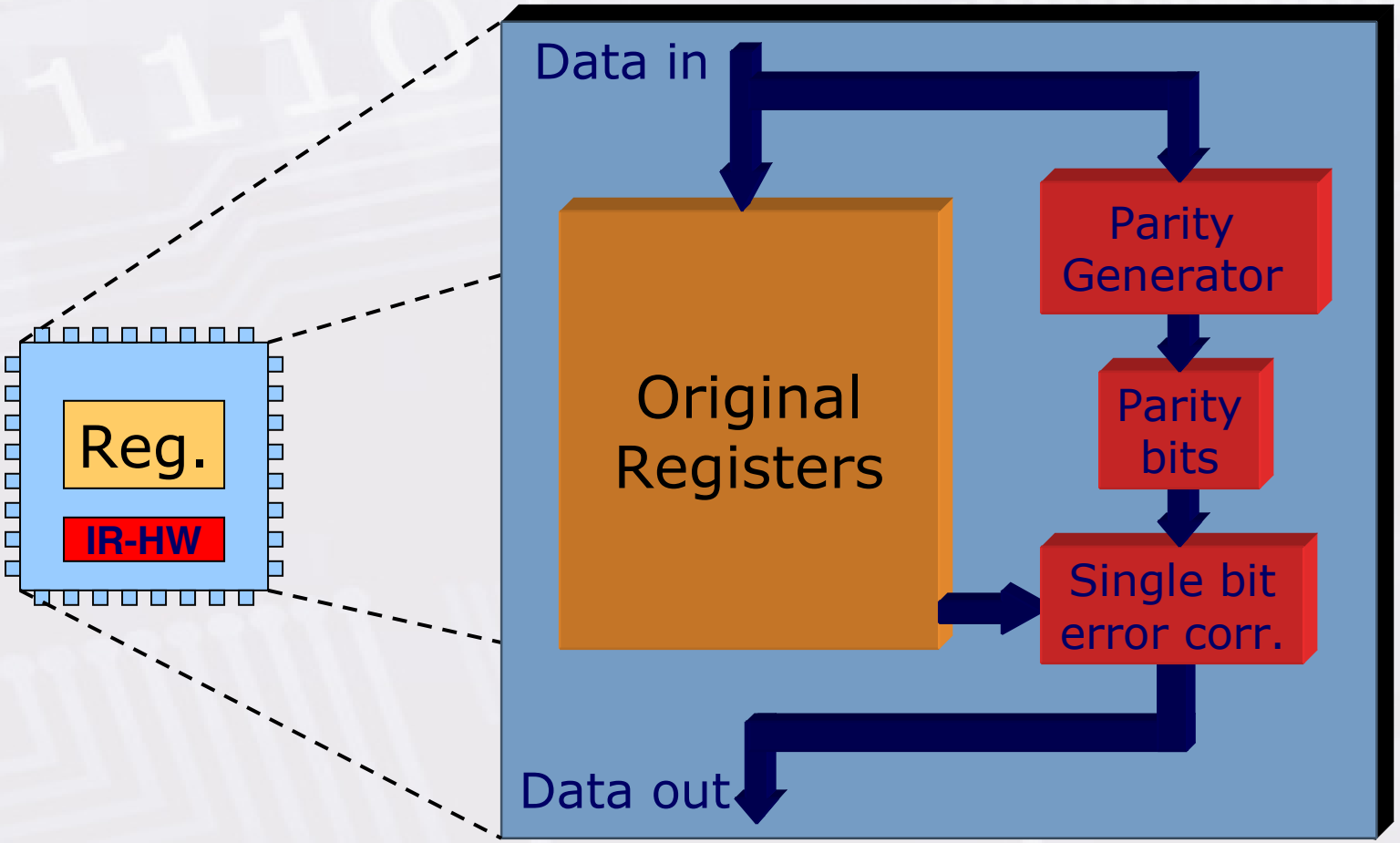
# Time Redundancy FT & Energy Management Conflict

- Competing for the same resource
  - Fault-tolerance requires slack time
  - Dynamic voltage scaling requires slack time

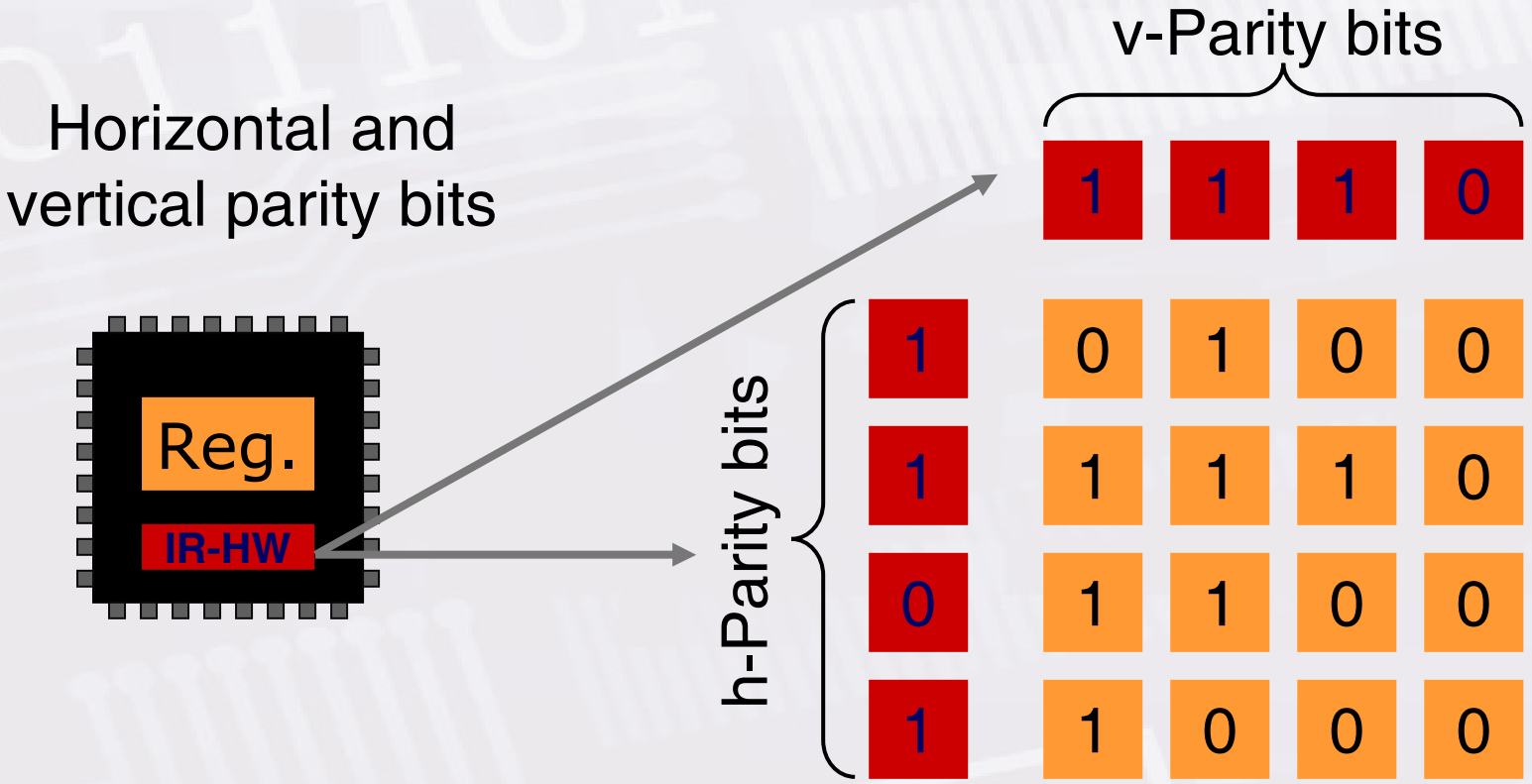
**Carefully trading off between fault-tolerance and energy management is necessary**

- Our approach for reliable & energy efficient systems
  - SEU (common), solve with information redundancy (error correction codes) and save slack time
  - Multiple SEU (infrequent), solve with time redundancy (error detection + retransmission)

# Information Redundancy

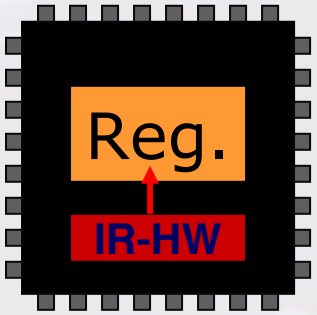


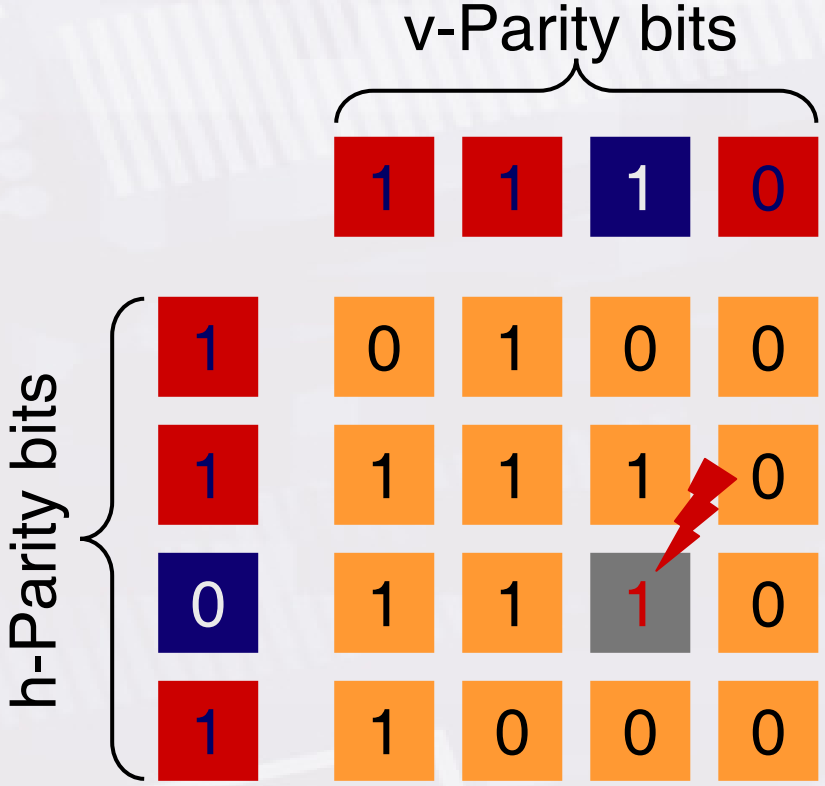
# Information Redundancy



# Information Redundancy

Horizontal and vertical parity bits

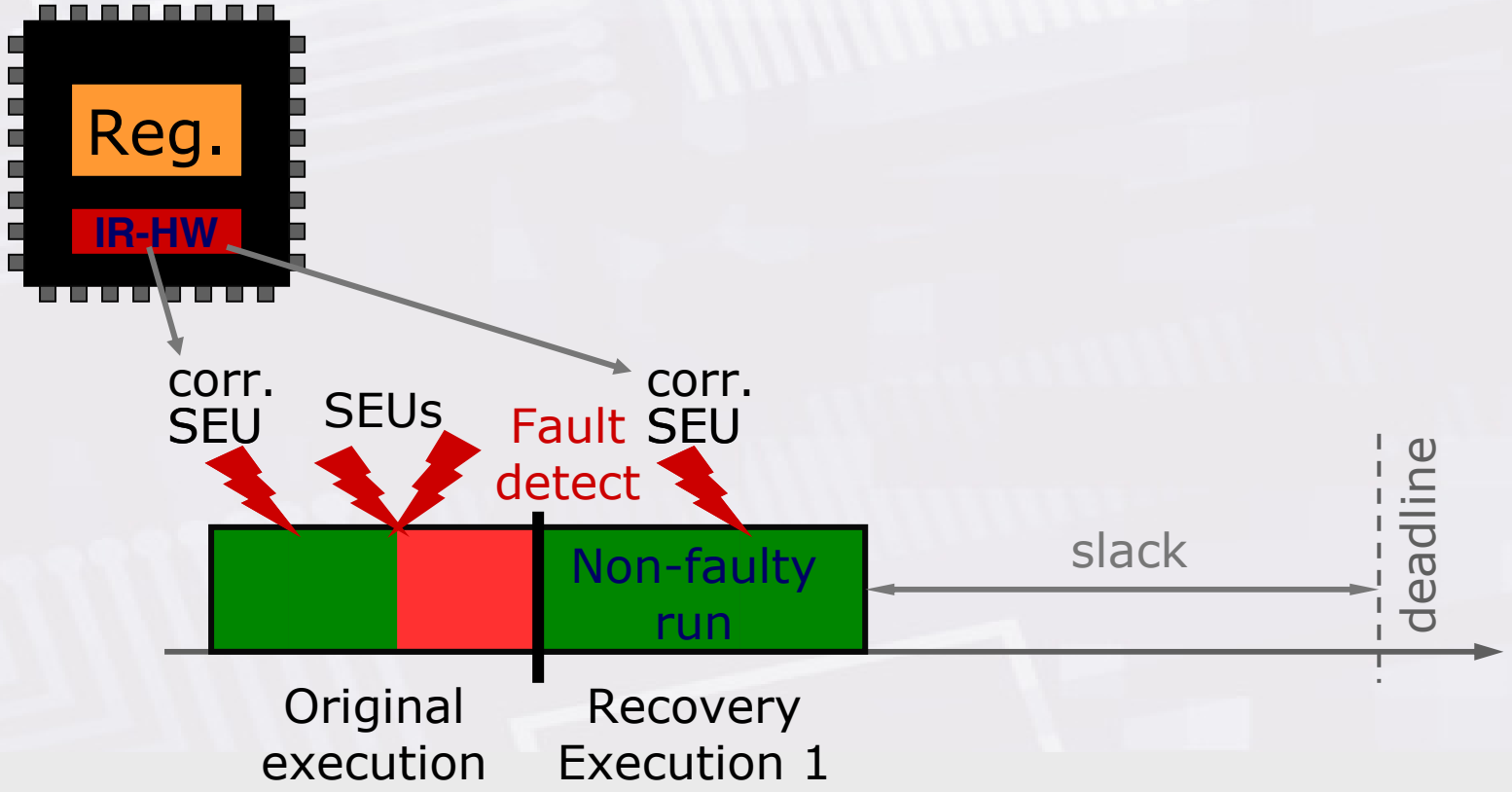



Identified fault can be corrected during execution

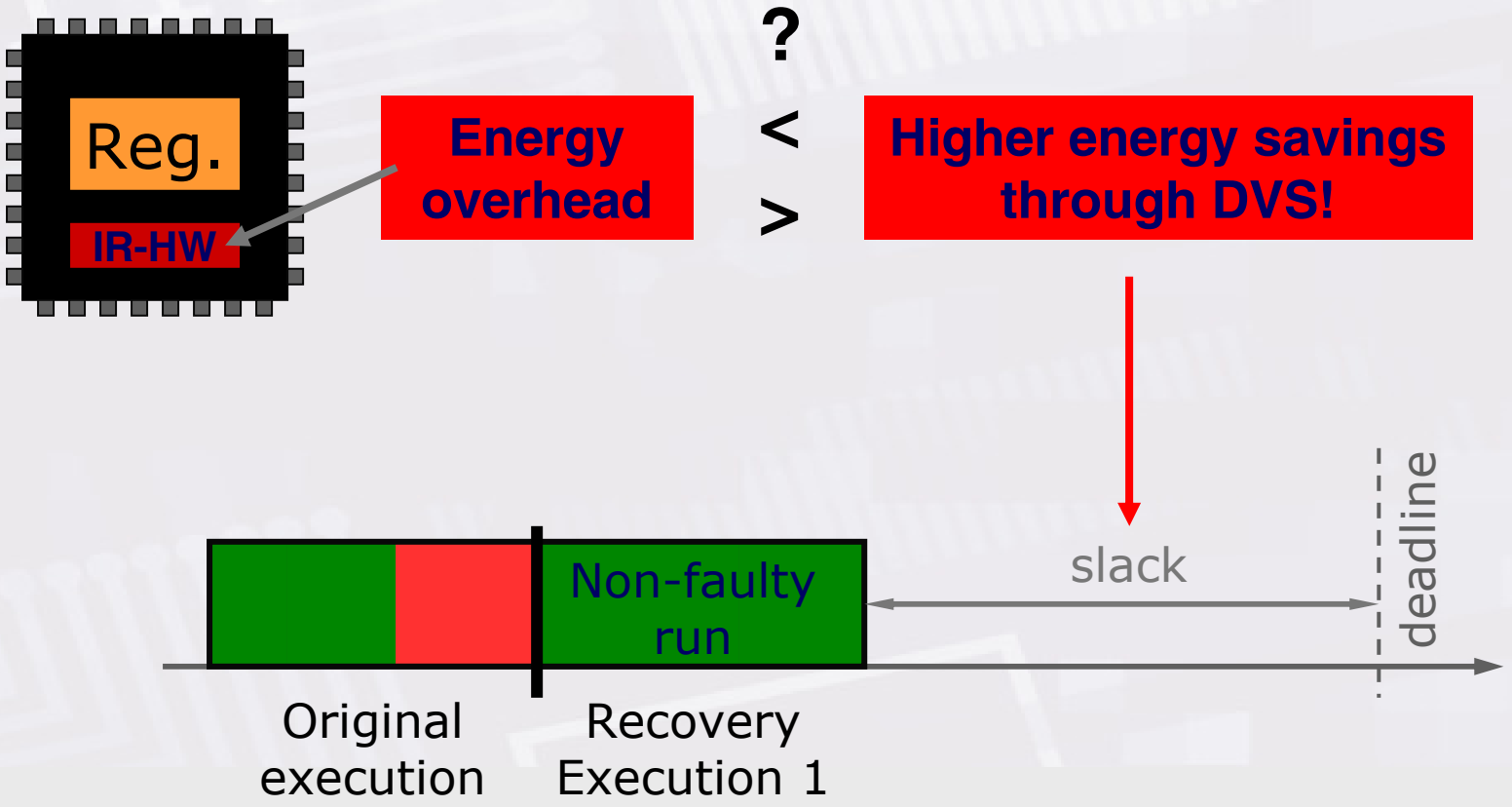
# Energy efficient FT

- FT through *rollback-recoveries* and *information redundancy (depending on fault occurrence)*



# Energy efficient FT

- FT through *rollback-recoveries* and *information redundancy*



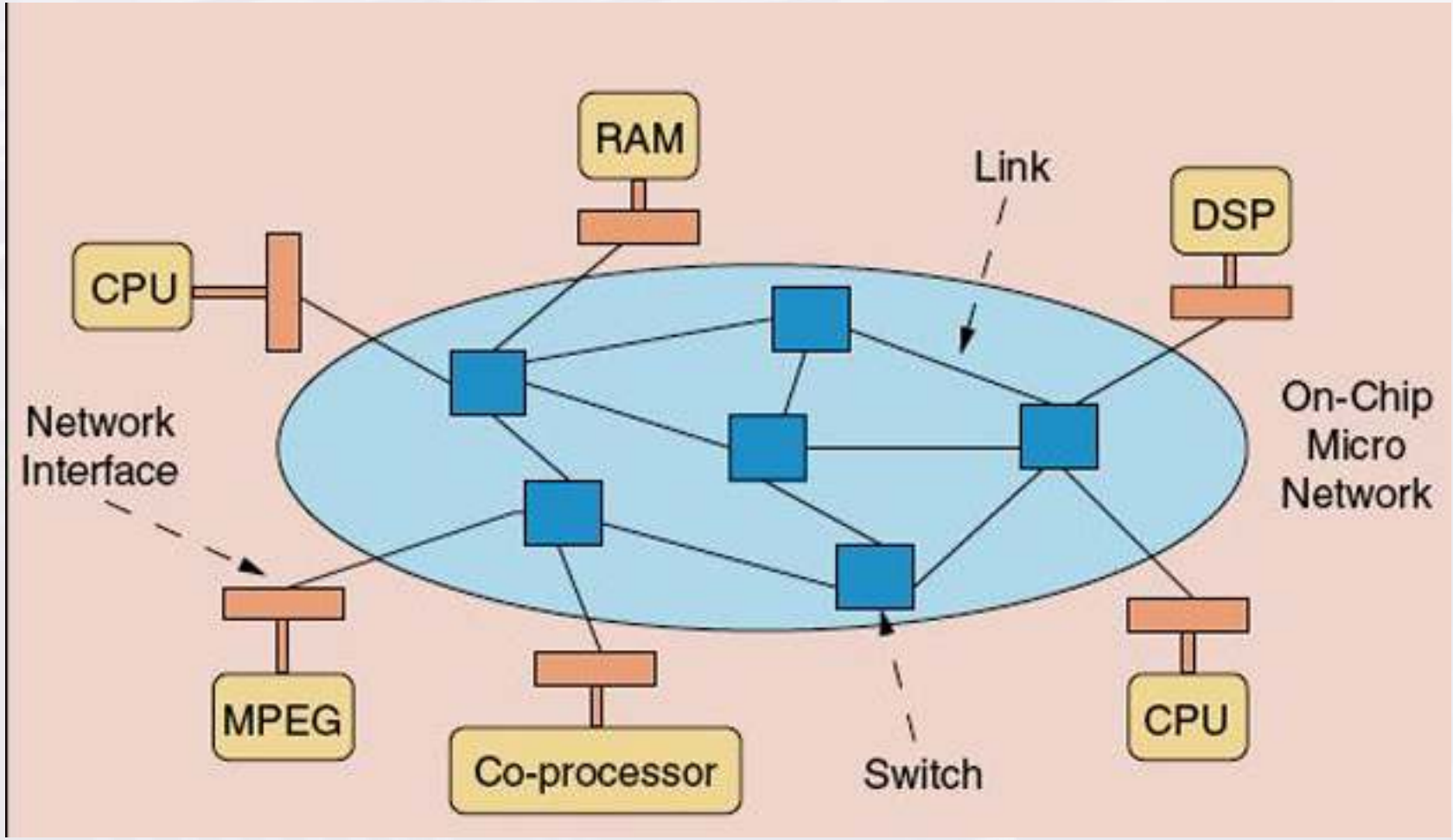
# Key observations

- Using various ITC benchmarks synthesized and analysed (hardware, energy) using EDA tools and different fault rates
  - It is possible to improve embedded computing systems reliability to transient faults without compromising energy saving through DVS
- Employ information redundancy for SEU and time redundancy for multiple SEU
- Adaptive Body Bias used to reduce leakage power also increases SEU rate by up to 36%

# Network-on-Chip

- Future embedded computing systems will contain hundred's of processors and memories blocks
- Bus communication may prove to be system bottleneck because
  - shared bandwidth and not compatible with the required Gbits/s bandwidth requirements
  - limited opportunities for parallelism and not compatible with the highly parallel system architectures
  - not scalable
- Network-on-chip attempts to solve the above issues

# Network-on-Chip



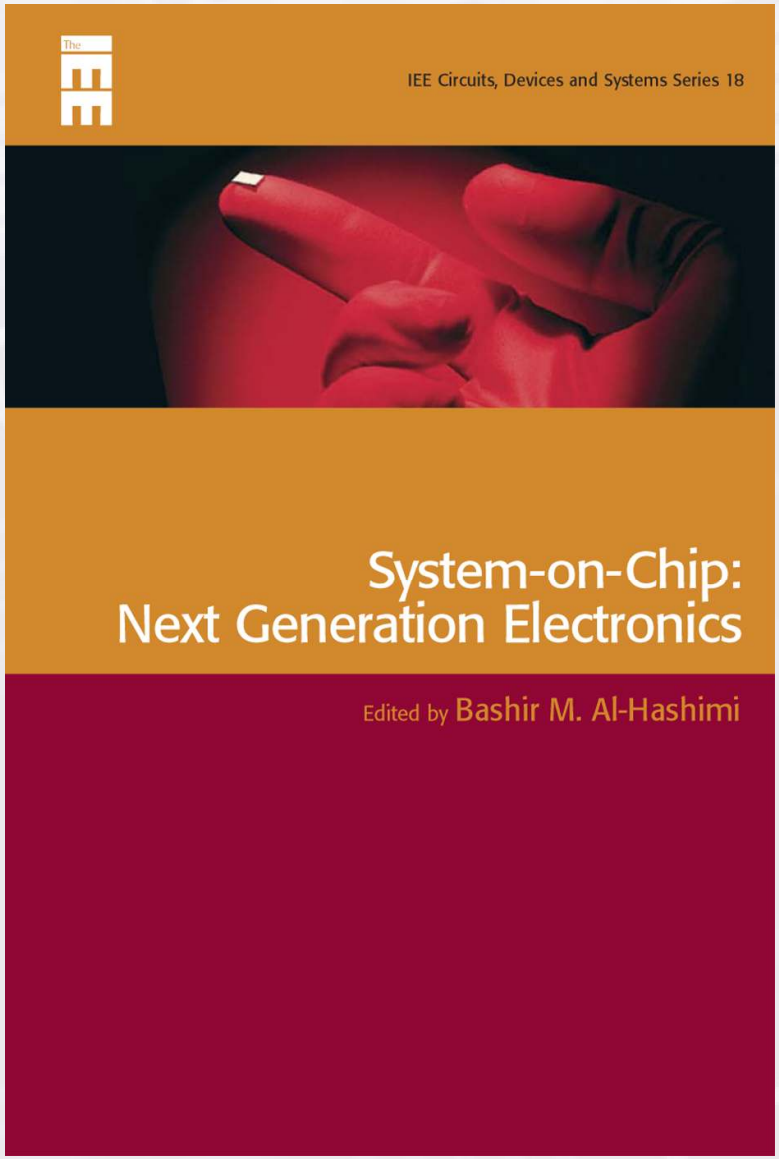
# HW/SW Co-Design with NoC Platform

- Scheduling and mapping optimisation to include not only computation cost but also variable communication cost (power, latency, reliability,..) as there are a number of possible routing options in NoC
- NIRGAM SystemC cycle accurate simulator, analyse NoC in terms of routing algorithms and applications on various topologies.

  http://nirgam.ecs.soton.ac.uk/

# Final Thoughts

- How to develop effective energy-efficient system-level automated design flows is reasonably well understood as demonstrated by the recent availability of SL-EDA tools
  - Extending SL flows to NoC platform is the next step?
- Low power and reliability are 2 key objectives when designing future embedded computing systems
  - Selecting the appropriate FT techniques fit for the application is important but not trivial
  - Developing application specific and *light weight* FT (employed only where needed) may be necessary to gain acceptance


The IEE
IEE Circuits, Devices and Systems Series 18
System-on-Chip:
Next Generation Electronics
Edited by Bashir M. Al-Hashimi