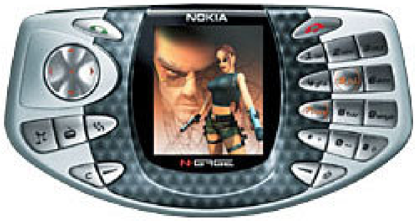
Balanced Energy Optimization

John Cornish
Director of Product Marketing
ARM Limited
john.cornish@arm.com

Energy Efficiency Matters to End Users

Users of consumer products care about:

- Convenient form factor
- Advanced Feature set
- Low Cost
- Long battery life
- High reliability

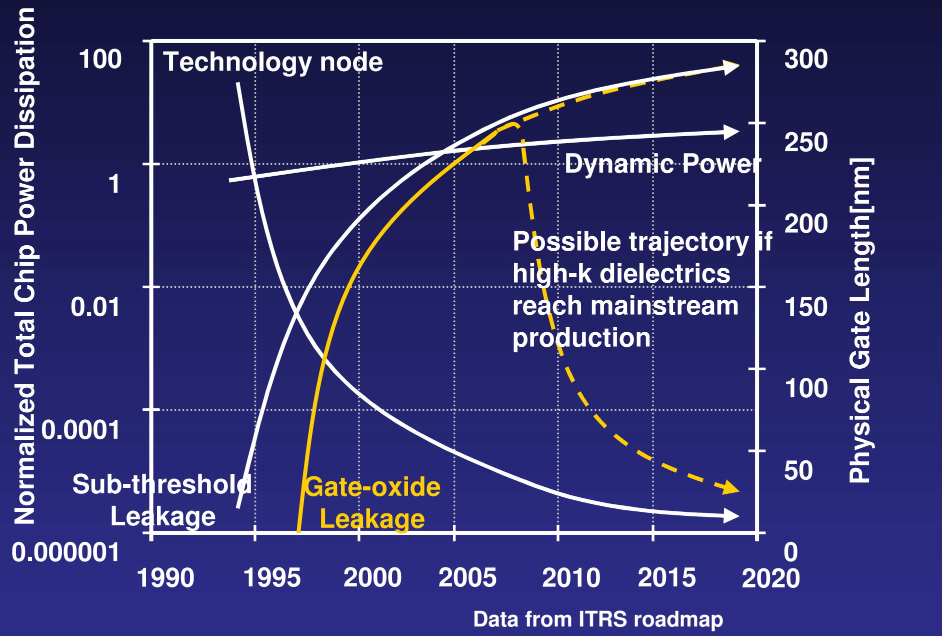


Higher Performance and Lower Cost

	1983 **Motorola DynaTAC 8000X**	1995 **Nokia 232**	2003 **Nokia 6600**
Battery life	0.5h - 1h talk, 8h standby	1h talk, 13h standby	4h talk, 240h (>1 week) standby
Battery	Lead Acid, 500g	NiMh, 100g	Li-Ion, 21g
Weight	800g	205g	125g
Features	Talk for brokers	Talk for the masses	Talk, play, web, photo, organize
Price	$3995	$500	$250

New Technology - New Rules

- Technology scaling trends are not in our favour
  - New processes are expensive
  - Diminishing performance gains from process scaling
  - Dynamic power remains high
  - Leakage power increasing exponentially



Data from ITRS roadmap

- Solutions need to cut across traditional boundaries
  - (SW / architecture / micro architecture / circuits)
  - Employ aggressive energy management techniques

Balancing Energy Efficiency

- Scalable processing

  Energy efficiency vs Responsiveness

- Customized processing

  Energy efficiency vs Generality

- Error tolerant processing

  Energy efficiency vs Determinism

Scalable Processing

The Need for Parallelism

- More bits to more places
  - ARMv5TE Signal processing instructions
  - ARMv6 SIMD media instructions
  - AMBA Multi-layer bus architectures
- More instructions per cycle
  - Deeper pipelines
  - Superscalar, VLIW
- More algorithms at once
  - Coprocessors, accelerators
- More applications
  - Multiple processors

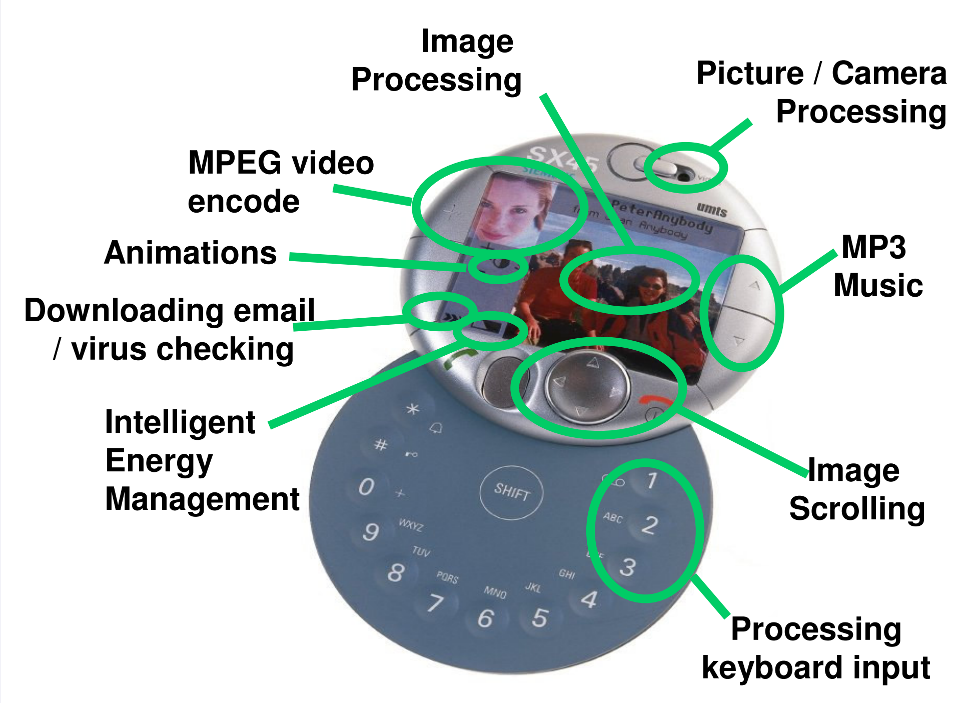
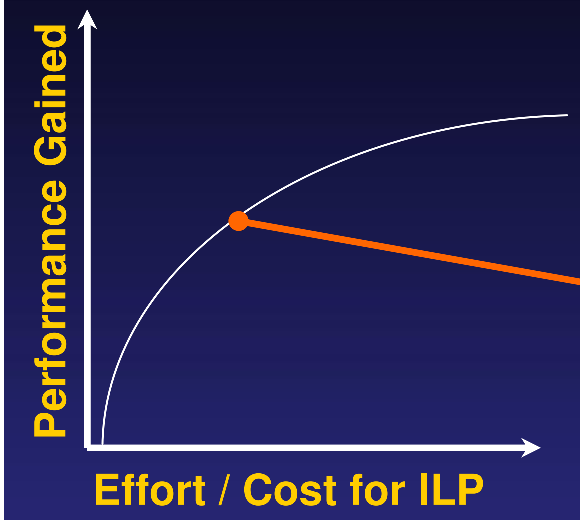
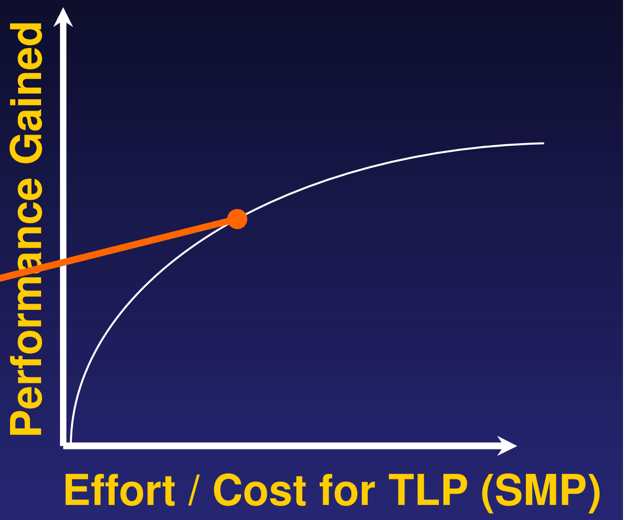


Image Processing
Picture / Camera Processing
MPEG video encode
Animations
MP3 Music
Downloading email / virus checking
Intelligent Energy Management
Image Scrolling
Processing keyboard input

Where to Best Apply Parallelism



‘Optimal’ support for both ILP and TLP brings the most performance for the least cost / effort



Expensive on hardware to continue to attempt to extract instruction level parallelism (ILP)

Expensive on hardware and software to continue to attempt to share work between more instances of thread level parallelism (TLP)

ARM and NEC Collaboration

ARM NEC

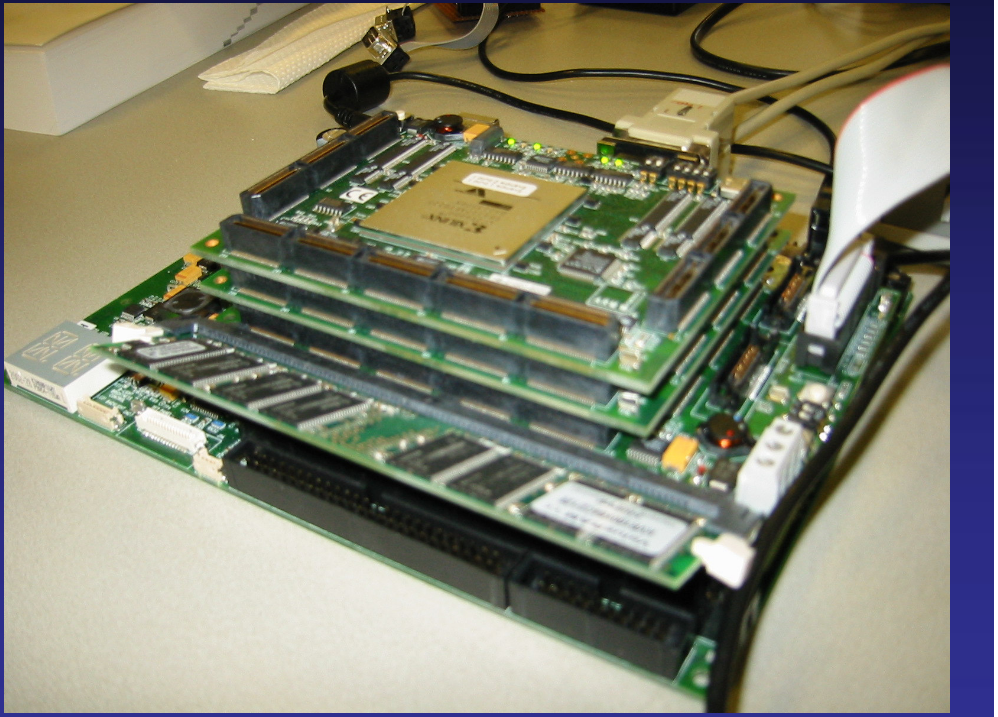
- NEC ELECTRONICS AND ARM ANNOUNCE LONG-TERM STRATEGIC COLLABORATION — Oct. 20, 2003
- NEC Electronics and ARM to co-develop and co-market next-generation multiprocessor-based CPU cores for home and automotive multimedia applications, and mobile handset markets
- NEC Electronics have licensed the current ARM11 family
- In addition there is a Multiprocessing collaboration effort
  - Multiprocessing can be higher performance and lower power
- ARM announced MPCore multiprocessor at EPF'04

Multi-Processing Models

- Asymmetric operation (AMP)
  - ‘Static’ task allocation
  - Distributed or common view of memory
    - Synchronization and communication via explicit message passing mechanism
  - Either homogeneous or heterogeneous CPUs
    - Manual allocation of work-items within definition of a SoC design (partitioned)
- Symmetric operation (SMP)
  - ‘Dynamic’ task allocation
  - Shared view of memory
    - Synchronization and communication via shared state in memory
  - Normally homogeneous CPU arrangement
    - Automatic allocation of work-items within an abstract definition of the SoC design

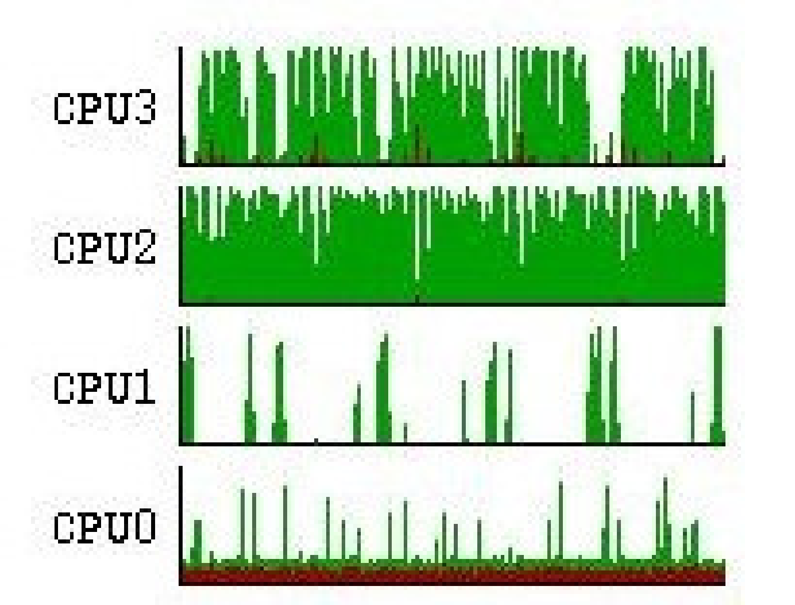
ARM SMP Evaluation Platform

- FPGA MPCore Implementation
  - 4 way ARM9-class CPU
- Software Environment
  - Pre-ported Linux SMP
  - GNU toolset
  - POSIX Thread library
- Availability
  - Today
  - Partners interested in MP application development

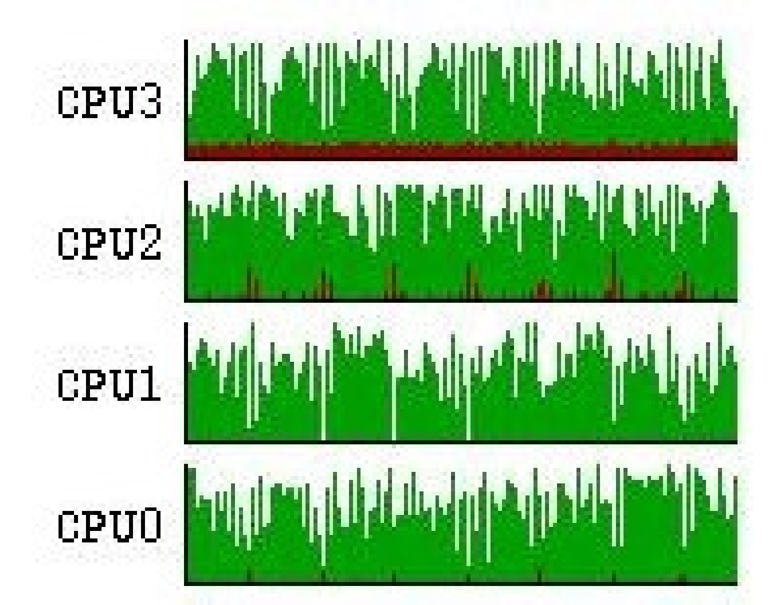


Thread Level Parallelism

- Measurements made on the ARM SMP Evaluation Platform
  - MPEG2 with color conversion deactivated, 20 frames



2 Threads



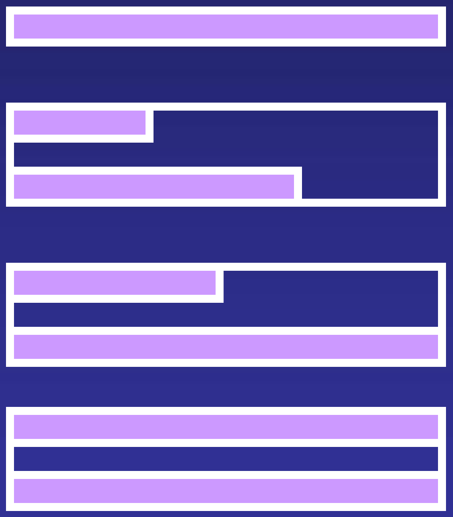
4 Threads

ARM Intelligent Energy Management

- Software components
    - Adaptively shutdown CPUs that are unused
        - Continual measurement of current level of TLP
    - Dynamic voltage and frequency scaling of active cores
        - Symmetric speed scaling – all active CPUs at same speed
    - Predict future software workloads by interacting with instrumented Operating Systems and application software
        - To determine the software deadlines
        - To balance workload and deadlines with performance
- Hardware component
    - To accurately measure the actual system performance
    - To independently manage the transitions of hardware scaling blocks. e.g. clock generators and power controllers
- Together these components determine and manage the lowest performance level that gets the work done

MP Energy efficiency

- Depending on TLP, each core can run slower.
- If TLP = 1.5 (50% of non-idle time 2 threads running)
  - **Symmetric-speed MP**: each core runs at 2/3 of uni-processor speed for equivalent throughput.
  - **Asymmetric-speed MP**: depends on synchronization between threads, slower thread could run at 1/3 of uni-processor speed for equivalent throughput.



Uni-processor (UP) workload

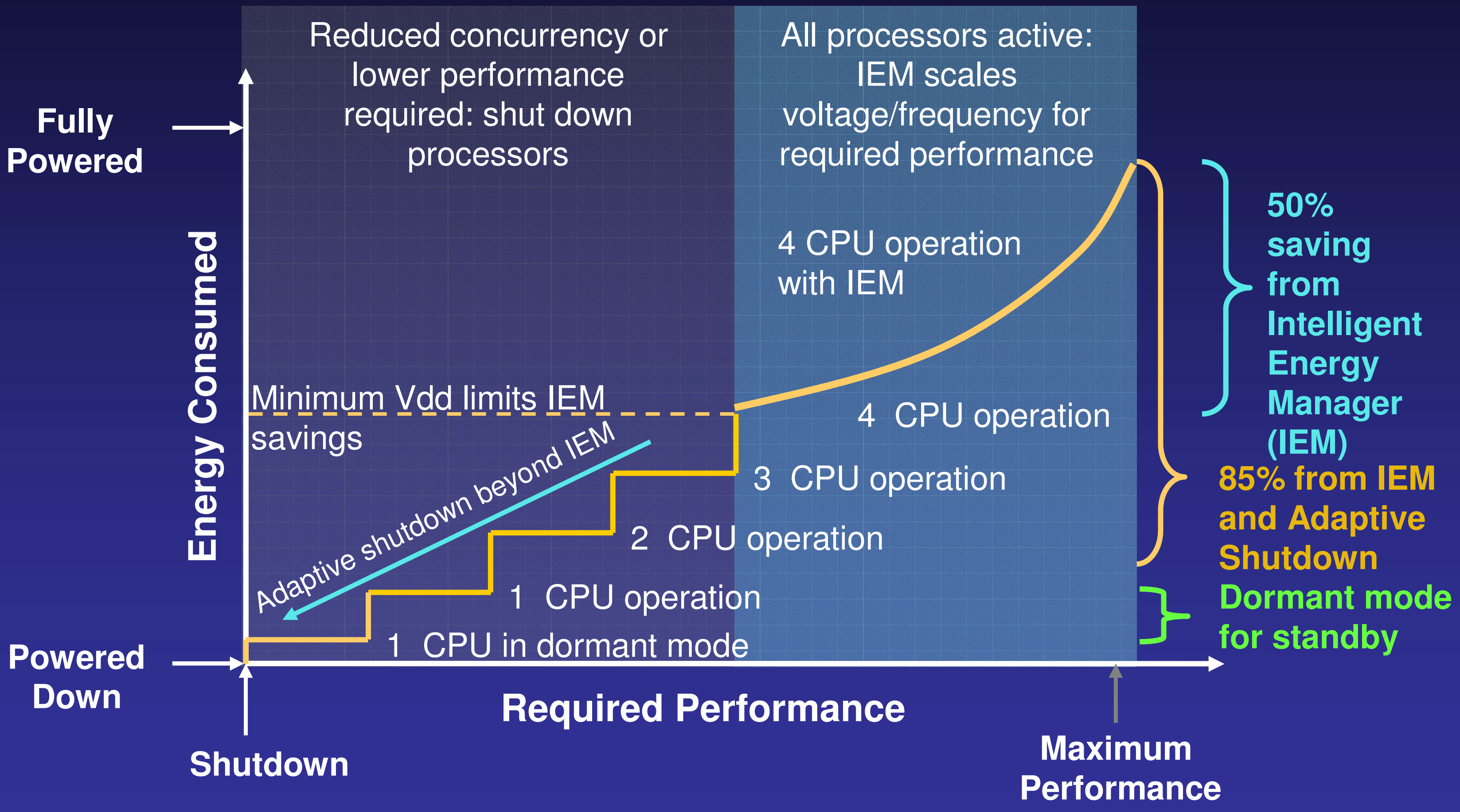
Workload on dual-processor (DP) at full speed. — **Increased performance**

Symmetric-speed DP: speed scaled to match UP throughput. — **E savings, simple implementation**

Asymmetric-speed DP: speed of each processor scaled separately. — **Largest possible E savings.**

MPCore - Control Over Energy Usage

MPCore multiprocessor extends control over power usage by providing both voltage and frequency scaling and turning off unused processors



Scalable Processing - Summary

- Exploit workload variance to minimise energy consumption (e.g Toyota Prius)
- Attempt to match processing resources to the computational workload in real time
- Need to use multiple processors to have fine granularity and wide dynamic range
  - This constrains maximum single thread performance so software must be multi-threaded
- Employ adaptive algorithms to predict what level of performance will be required
- System responsiveness constrained by time constant of power management devices

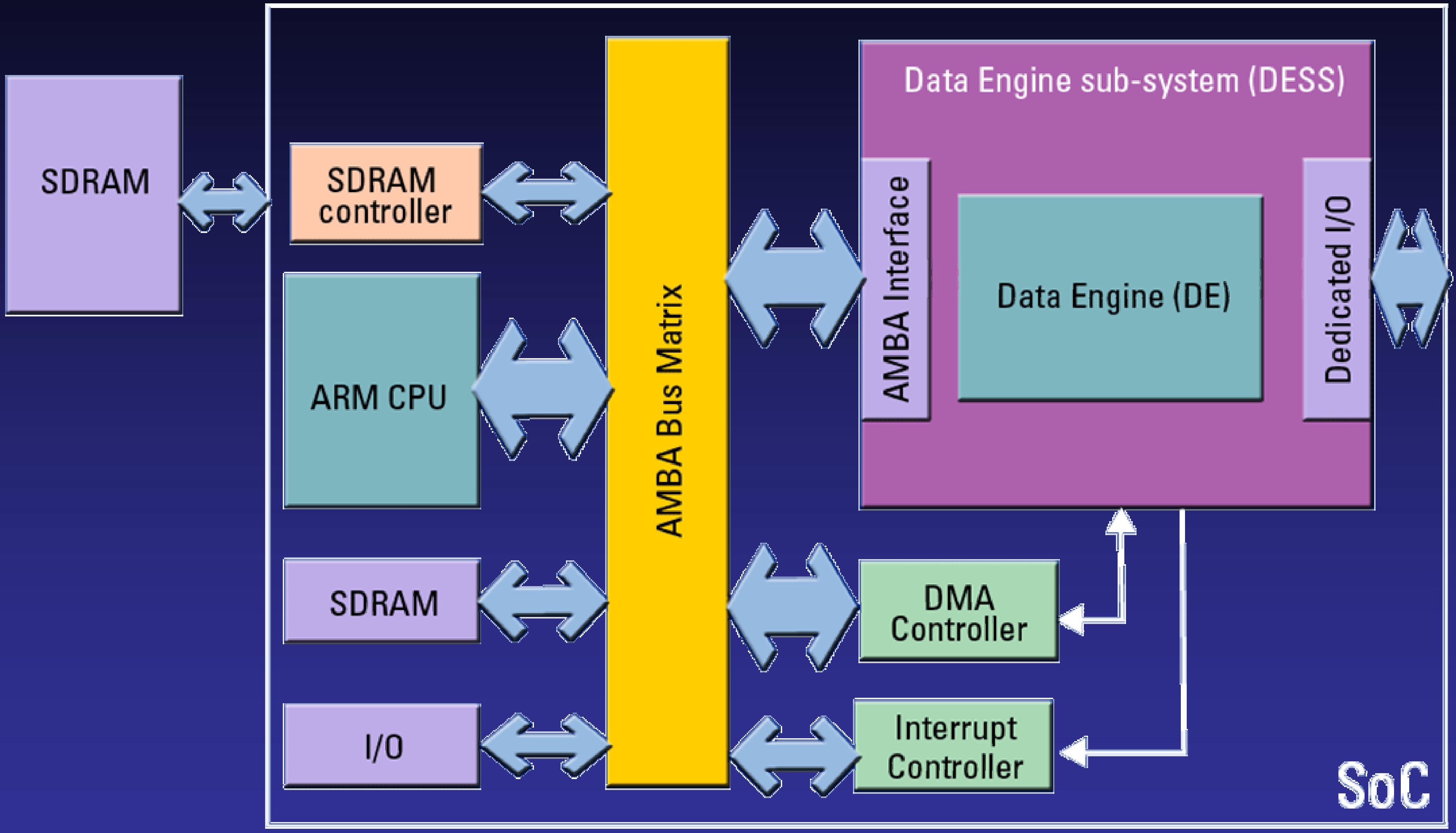
Customized Processing

ARM OptimoDE

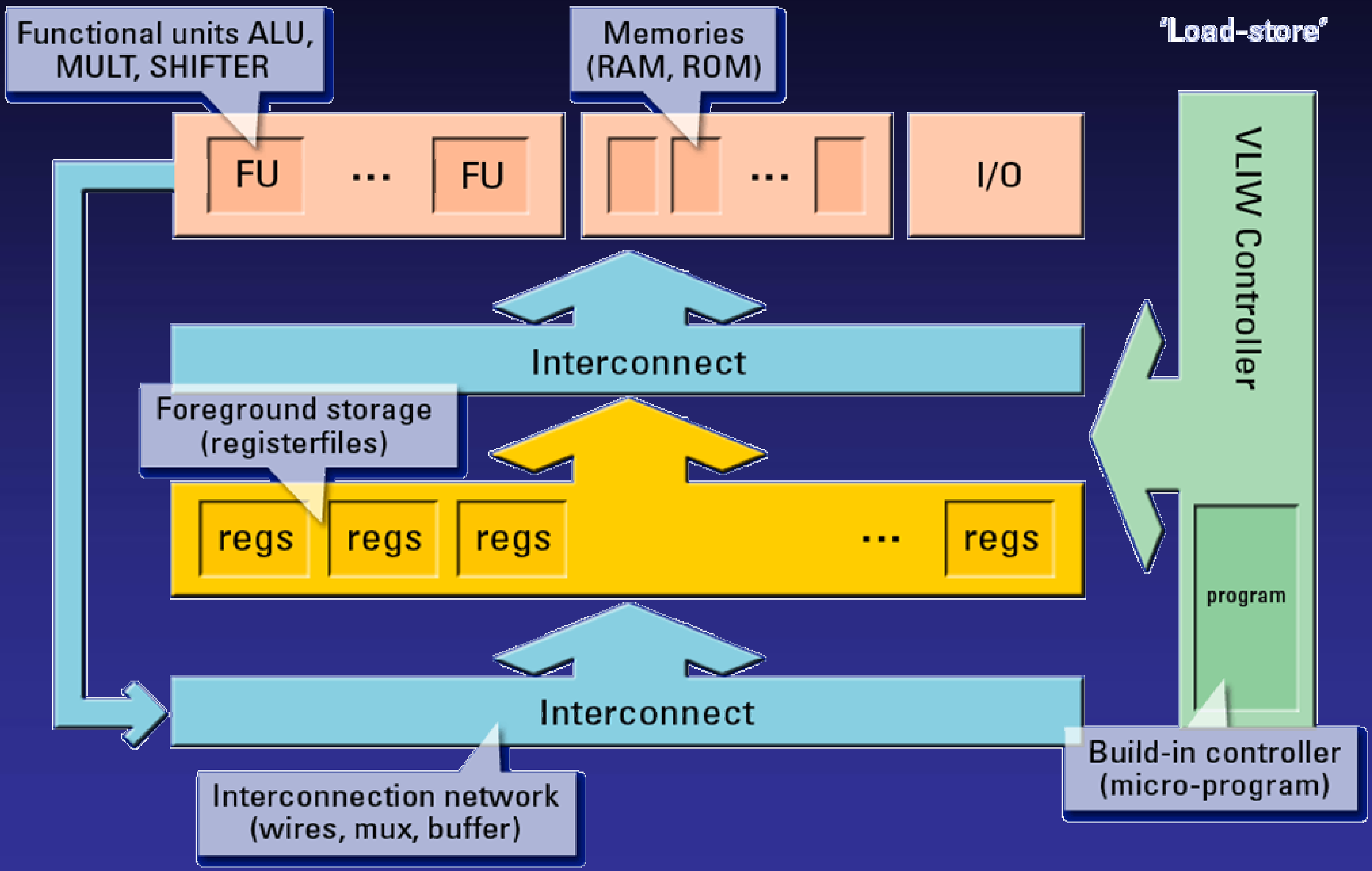
- Configurable VLIW style architecture
  - High datapath parallelism → high performance (> 3GOPS)*
  - Low clock rate and low power implementation (0.0215mW/MHz)*
  - Allows small core (from 9,500 gates)*
  - Efficient code density using code compaction scheme
  - Fully customizable data path including addition of application specific units
- Dynamic parallelizing (patented) HLL compiler
  - C and C++ supported
  - Compiler automatically targets application specific units
- Supported by
  - Data engine customization and profiling tools
  - AMBA™ design integration kit (data engine sub-system design)

** Taken from actual implementations*

Data Engine System in Context



OptimoDE - Architecture Model

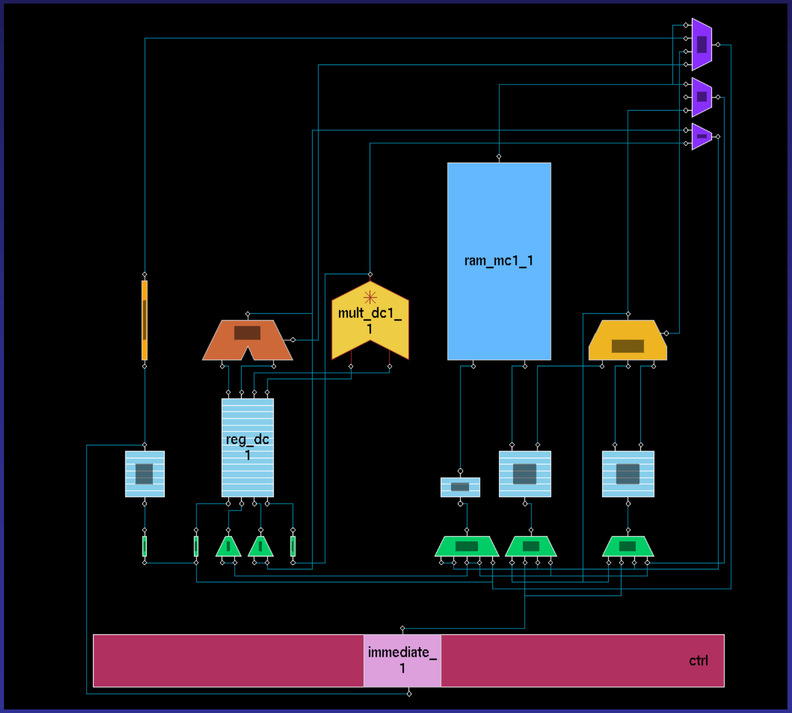


Functional units ALU, MULT, SHIFTER
Memories (RAM, ROM)
'Load-store'
FU
...
FU
...
I/O
VLIW Controller
Interconnect
Foreground storage (registerfiles)
regs
regs
regs
...
regs
program
Interconnect
Build-in controller (micro-program)
Interconnection network (wires, mux, buffer)

Example Configurations

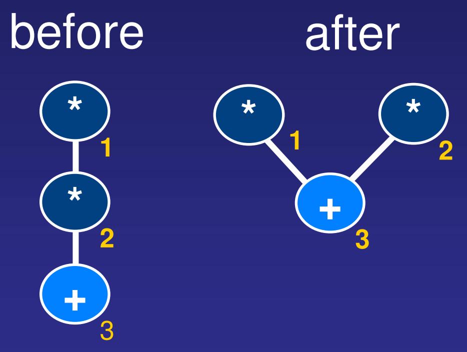
- Wide range of configurations and customizations possible
- To reduce development time ARM supplies
  - Variety (19) of partially and fully defined example configurations
  - Library of optional engine resources for any data engine implementation
- Applications programs (C or C++) can be immediately compiled and profiled to these implementations

- Example configuration OMA111FSc
  - 1 ALU, 1 multiplier, 1 RAM
  - Central register file
  - Support for subroutine spilling



OptimoDE Compiler

- Converts C-source into a Data and Control flow graph
- Performs patented Dataflow Analysis
  - Analyzes data-dependencies in C code
  - Reconstructs parallelism from sequential C code
  - Works across hierarchy and arrays
  - Improves results in several ways
    - Parallelism
    - Software pipelining
    - Address computation



```
p1 = a * b;
p2 = c * d;
s = p1 + p2;
```

=

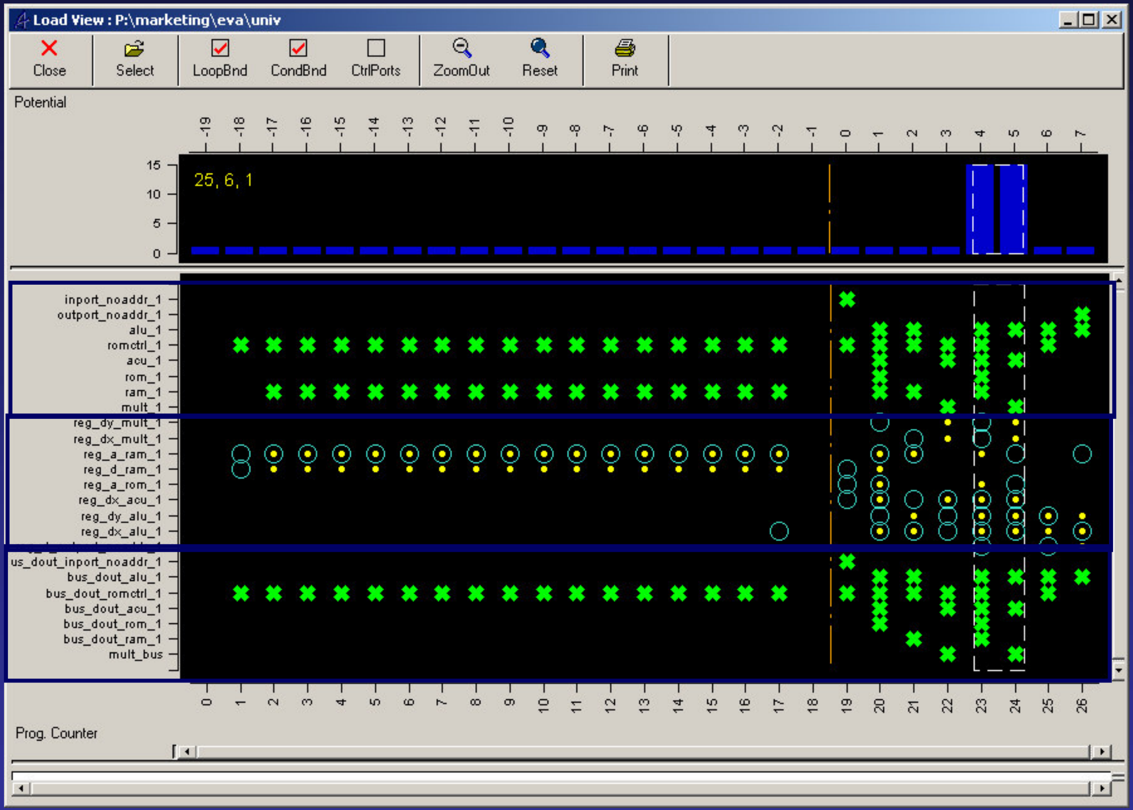
```
p2 = c * d;
p1 = a * b;
s = p1 + p2;
```

=

```
p1 = a * b; p2 = c * d;
s = p1 + p2;
```

Static Profiling

Performance profiling reports automatically generated



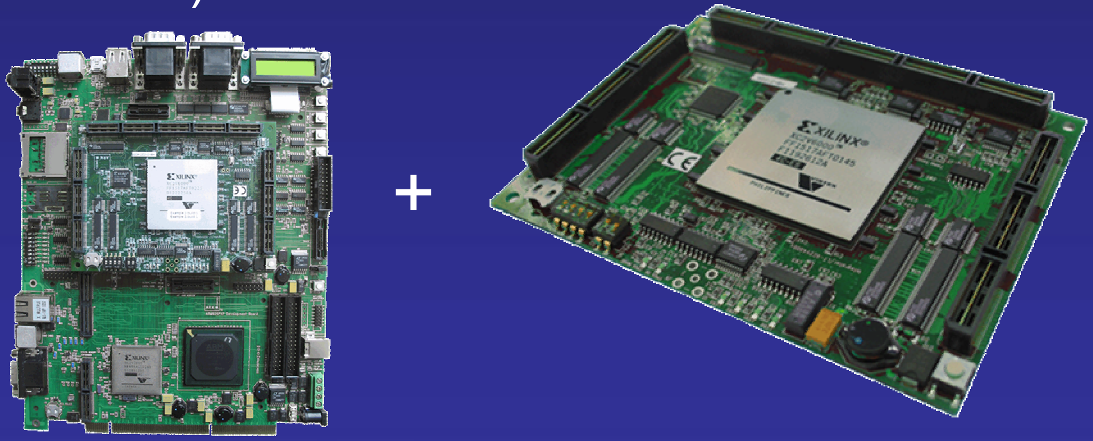
core_activity

register_activity

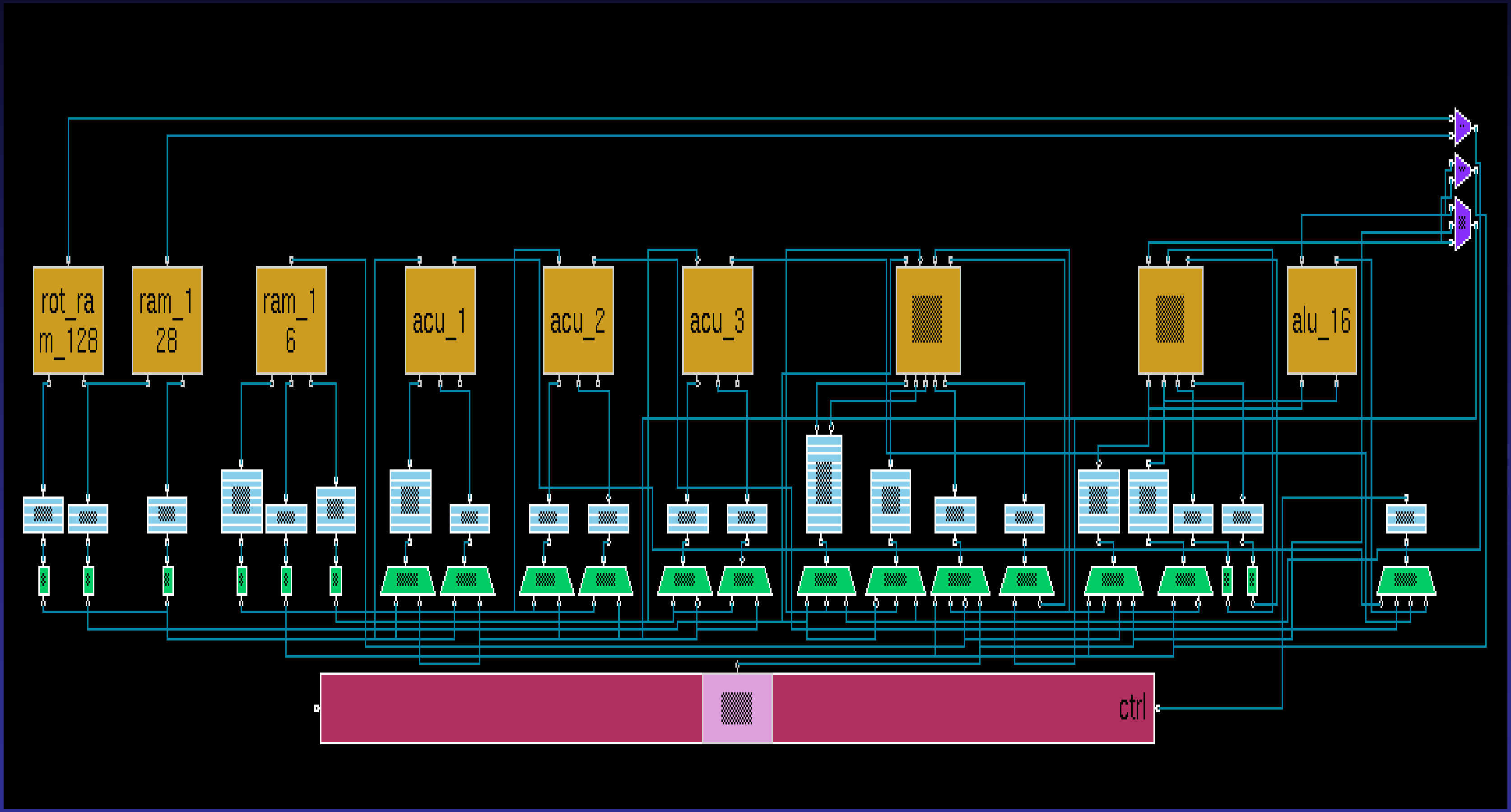
bus_activity

Video Example - Target Platform

- MPEG4 Decoder implementation
- Prototype using ARM Versatile platform with FPGA card
  - OptimoDE data engine targeting an FPGA implementation
    for prototype development
    - Fast (near real-time) verification



VideoDE™ – Schematic View

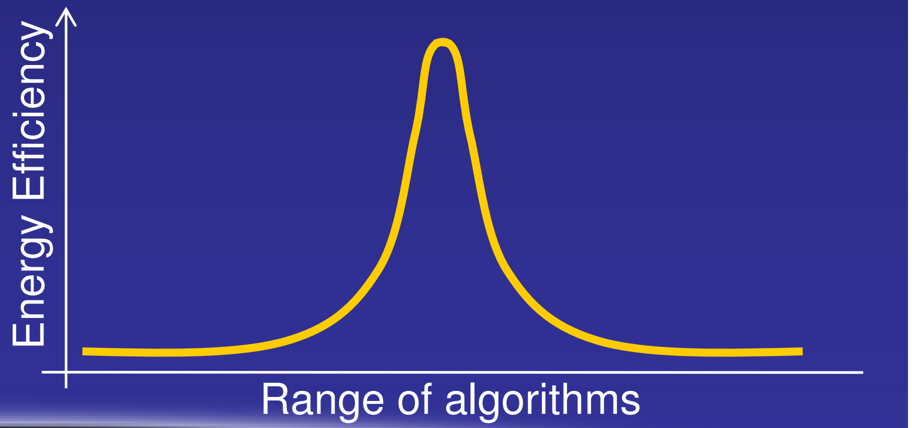


Example VideoDE Result

- Single Instruction Multiple Data (SIMD) datapath
  - 128-bit configured as 8 * 16 bit or 16 * 8 bit
  - Parallel multiplier and adder
  - Sum of Absolute Difference (SAD)
- VLIW controller - 160-bit instruction word
- 16-bit ALU
- Three address generators
- Two RAM (128-bit and 16-bit wide)
- Support for efficient block rotation
- 150K gates
- 150-200MHz clock rate

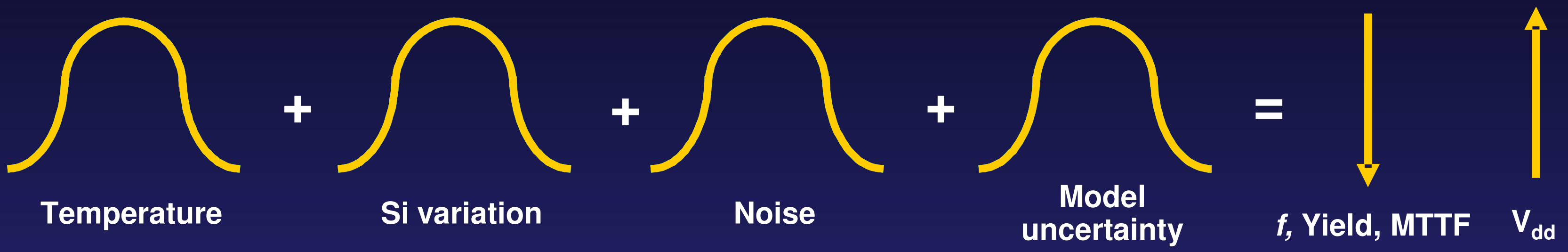
Customised Processing - Summary

- Optimise the processing resources to the intended workload at design time
- Trade-off generality for energy efficiency
  - Processor will be efficient for a class of algorithms
  - For many embedded applications this is acceptable
- Need tools to analyze, compile, and profile target algorithms against the design



Error Tolerant Processing

Uncertainty in Design Parameters



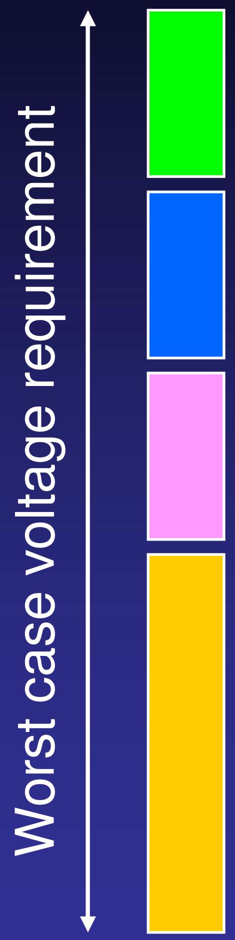
- Uncertainly leads to performance and power overheads
  - Increasing uncertainty with design scaling
  - Intra-die process/temperature variations, inductive noise
- Key Observation: worst-case conditions are highly improbable
  - Significant gain for circuits optimized for common case
  - Efficiency mechanisms needed to tolerate infrequent worst-case scenarios

“Better Than Worst-Case” Design

- ‘Making Typical Silicon matter with Razor’
  - Austin, Blaauw, Mudge, Flautner
- Low-power pipelines based on circuit timing error detection & correction
- Uses voltage scaling to eliminate worst case design margins

March 2004

Voltage Margins



Noise Margin

Ambient Margin

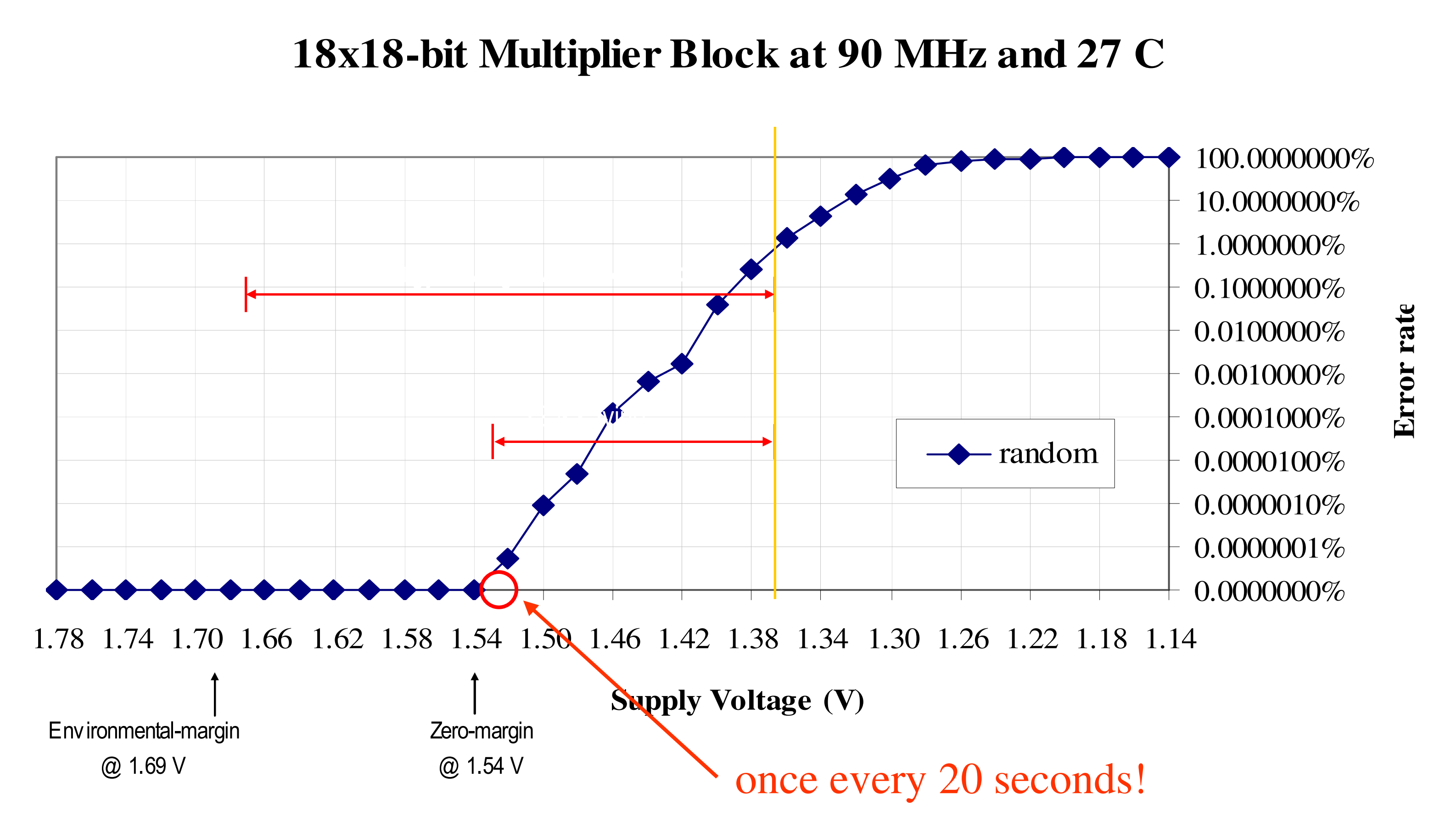
Process Margin

Critical voltage determined
by critical circuit path

Detecting & Correcting Timing Errors

- Razor flip-flop implements a shadow latch using a locally inverted clock signal
- Allowable operating voltage constrained to ensure shadow latch setup time is always met
- Error signal is XOR of latch and critical flip-flop
- Many non-critical flip-flops will not need Razor
- Two methods developed for recovering pipeline state after timing error detection: global clock gating, and counterflow pipelining
- Multiplyer experiments showed 35% energy saving at 1.3% error rate

Error Rate Studies – Empirical Results

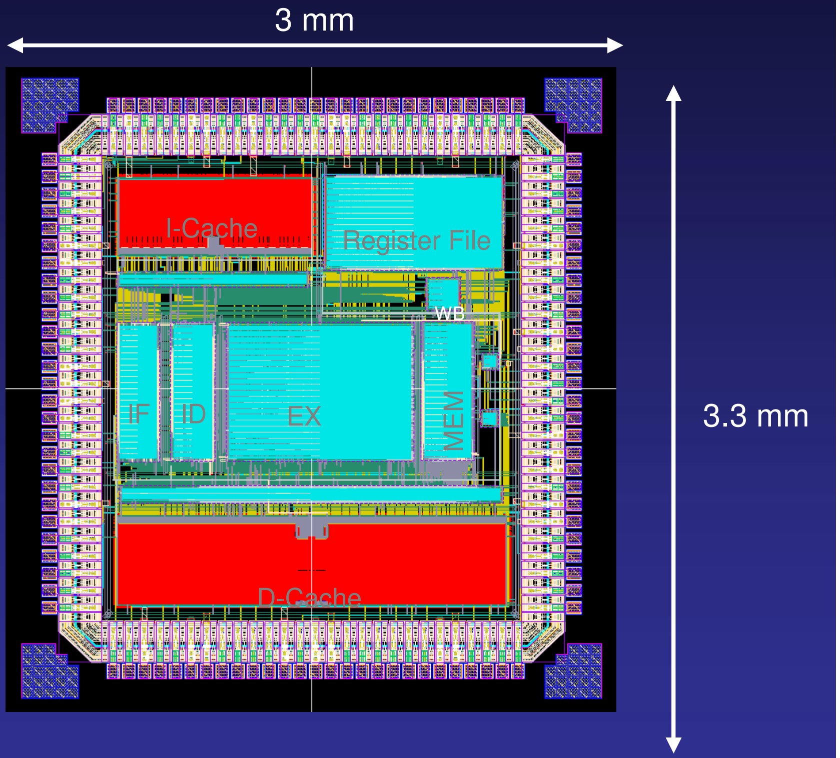


once every 20 seconds!

- Error rates rise even slower for real program data

Razor I - Razor Prototype Design

- Six stage 64-bit Alpha pipeline
  - 200MHz in 0.18mm @ 1.8V
  - Tunable via software from 200-50MHz, 1.8-1.1V
- 32-entry, 3-port register file, 8K I-Cache, 8K D-Cache
- Branch-not-taken branch predictor
- Implements substantial subset of 64-bit Alpha integer instructions
- Ability to service software interrupts
- Full scan capability
- Razor overhead:
  - Total of 192 Razor flip-flops out of 2408 total (9%)
  - Error-free power overhead 3.1%
  - Recovery power overhead at 10% error rate 1%



Conclusions

- Energy efficiency is the #1 issue for many applications
  - Mobile devices need more performance & longer battery life
- Scalable processing
  - Multi-processor designs with DVS and adaptive shutdown increase efficiency by scaling to widely varying workloads
- Customized processing
  - Trades-off generality for energy efficiency which is acceptable for many embedded applications
- Error tolerant processing
  - Offers very attractive efficiency gains for small overhead