Efficient architectures for streaming applications

Gerard Smit

University of Twente
Faculty EEMCS / CTIT
the Netherlands
e-mail: G.J.M.Smit@ewi.utwente.nl

University of Twente
The Netherlands

Contents

- Introduction energy-efficient systems
- Streaming DSP applications
- Tiled architectures
- Dynamic reconfiguration
- Run-time mapping of streams to architecture
- Conclusion

2

Motivations for energy-efficient systems

- Portable devices that rely on batteries
  - Batteries are heavy and large
  - There is no Moore's law for batteries
  - Exponential increase in demand for (streaming) communication and computation
    - Multimedia, wireless communication, etc.
- High-performance computing
  - Cost for cooling and packaging high
  - Reliability: 10°C increase doubles component failure rate
  - Power dissipation 100W to 2000W
  - Supply current 100A (per chip: Itanium) to 2000A (per board)!
- Environmental concerns
  - Pollution, EMC radiation
  - e.g. there are 7.000 GSM (x 5 providers) antennas in Netherlands
    - energy supply is a large portion of the exploitation budget
    - Energy bill of Google 50 M$ per year

3

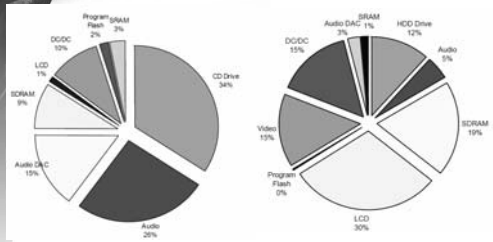
Sources of energy drain in a system

- Communication
  - energy spent by wireless interface
  - internal traffic between various parts of the system
- Computation
  - applications
  - operating system
  - wireless protocol processing
- Storage / memory
  - disk
- Display

→ there is no primary source of energy reduction

4

Energy profile of two mobile systems



Mobile audio player Mobile audio/video player

[source Philips Research]

5

Basic rules of energy reduction

- Do not do more than necessary
  - avoid overhead
  - do not optimize for 'worst case' but for the 'current case'
  - React on the environment: adaptability (QoS)
- Use Locality of reference
  - avoid communication over long distances
  - avoid off-chip communications (1000 times more expensive)
  - can we wait until the connection is better/cheaper?
  - Can we prefetch information when connection is cheap?
- Take a holistic view
  - Be energy aware at all levels of your system (QoS)
  - technological, system architecture, operating system, applications
- Do the tasks at the most energy-efficient platform/way
  - Heterogeneous architectures
  - Match algorithm with architecture
  - Migrate functions from mobile to wired system(?)

6

Myths and facts

- Myths in energy reduction
  - energy consumption is only a hardware problem
  - time will solve the energy problem
  - new battery technology will solve the problem
- Facts
  - functionality of a device is often limited by required energy consumption
  - batteries are the largest single source of weight in a portable
  - help from IC technology will slow down
  - energy is a 'vertical' parameter and involves all layers
  - solution might be in the higher levels: system architecture, communication protocols, operating system, applications
  - gap between battery energy and required energy grows
  - communication will require relative more energy than processing

7

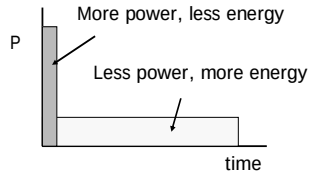
Battery gap

- Battery energy contents improves approx. 10% per year
- In most portables batteries contribute to 1/3 of the weight
- 2 AAA batteries have energy contents of ~3.3 Wh
- Required energy grows with far more than 10%

8

Metrics

- Power
  - Energy dissipated in a certain period of time
  - E = P. t [Ws]



9

Energy efficiency

- Energy efficiency =

$$\frac{\text{Essential energy dissipation for a certain function}}{\text{Actually used total energy dissipation}}$$

10

Design for energy efficiency

abstraction level	*examples*
system	dynamic power management compression method scheduling communication error control medium access protocols hierarchical memory systems application specific modules
logic	logic encoding data guarding clock management reversible logic asynchronous design
technological	reducing voltage chip layout packaging

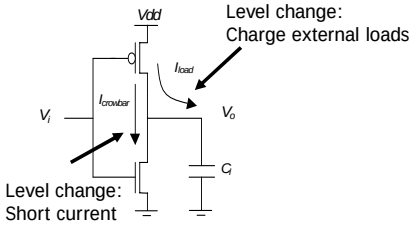
11

Technology and logic design level

12

CMOS inverter

- CMOS
  - Inherently low power
  - Cost effective



13

Where does energy go in CMOS digital logic?

- Dynamic power consumption
  - Charging and discharging capacitors
  - Most dominant (80-95%) in 130 nm technology
- Short circuit current
  - Short circuit path between supply rails when logic level changes
  - 10% – 15%
- Leakage
  - Leaking diodes and transistors
  - Problem even in standby
  - Effect is increasing with smaller feature size!
  - Will soon become a significant/dominant portion of total

14

Power consumption approximation

- Dynamic power consumption

$$P = \sum \alpha C V^2$$

with: α = switching activity
C = total capacity
V = voltage swing

- Semiconductor trend
  - V drops 5V → 1.8 V → 1.2 V → 0.8 V
  - smaller technologies
    - C decreases (on-chip C, *not* off-chip C)

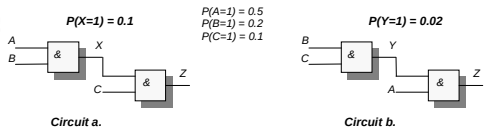
15

Minimise capacitance

- On-chip 10-50 femto Farads
  - Internal C reduced by technology scaling
  - E.g. MIPS 25% reduction in power due to migration from 0.8 um to 0.64 um
- Off-chip 14 pF
  - Energy required for 32 x 32 multiplication 5.7 nJ
  - 32 bits data to memory (24 address lines) 20 nJ
  - With smaller feature size gap will increase!!

16

Reordering logic inputs



17

Technology and logic Summary

- Numerous techniques are available
  - Packaging
  - Technology scaling
  - Circuits
  - Clock gating
  - Data guarding
  - Architecture
- Technological level gain is limited to x2
- Reduce switching activities is the most effective technique

18

System architectural level

19

System level

- Potentially high gain
- Three major mechanisms
  - Avoid unnecessary activity
  - Exploit locality of reference
  - Use most efficient platform

20

System architecture tricks

- Gated-clocks and power shutdown
  - Dynamic power management
- More efficient algorithms and architectures
- Proper I/O interconnect design and packaging
  - Single package
  - Coding of data
- Interconnection network
  - CPU centric / connection centric / NoC
- Memory access
  - Recompute rather than refetch from memory
  - Local memories/cache (locality of reference)

21

Memories & busses

- A significant fraction of total energy budget is consumed in busses and memories
  - Minimise bus access
    - Minimise memory access
    - caching
    - Clustering
    - compression
  - Reorder access
  - Bus encoding techniques
- Break memory in smaller sub-arrays/banks
  - Each bank can be individually powered down
  - Memory allocation and garbage collector

22

Encoding

- Large amount of energy goes into off-chip IO
- Encoding bus data and address can reduce power significantly
- Examples
  - Gray code: addresses usually increment sequentially by one
  - Compression
  - Bus invert coding
    - Transmit original or inverted data whichever results in fewer transitions from previous
    - Extra signal indicates polarity
  - 2's complement versus signed magnitude

23

Dynamic power management

- Natural focus of designers
  - Worst-case conditions
  - Peak performance
  - Peak utilisation
- Consequence is that system is not fully utilised
- Dynamic power management exploits periods of idleness caused by system under-utilisation

24

Problems with power management

- Cost of restarting
  - Latency (e.g. time to spin-up)
  - Extra energy, e.g. higher start-up current disk
    - Disk 2W in active, 1 W in idle, 3W in spin-up
- Two main questions
  - When to shut-down
  - When to wake-up

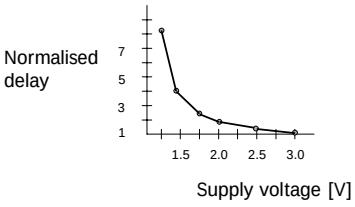
25

Barriers to voltage scaling

- Voltage scaling requires threshold voltage to be scaled as well
  - Decrease in noise margin
  - Leakage power will increase
- Requires special circuits
- soft error rate will increase
  - Caused by alpha particles and cosmic rays
  - Reduced capacitance implies lower energy to flip a bit
- Delay increases..

26

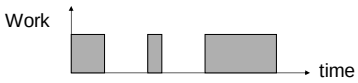
Speed vs. voltage



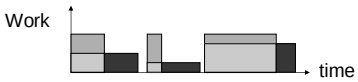
27

Why Dynamic Voltage Scaling?

- Execute only as fast as necessary to meet deadlines
- Workload in devices are typically bursty:

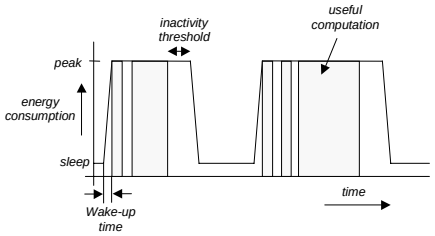


- We can save energy by slowing down and thus utilize the idle time.



28

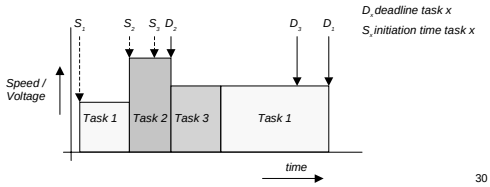
Traditional



29

Voltage scaling under deadline constraints

- Example:
  - task 100 ms deadline, needs 50 ms CPU full speed
    - Traditional: 50 ms computation, 50 ms idle
    - Half speed/voltage scaled: 100 ms comp., 0 idle
    - Ideal situation: ¼ energy reduction



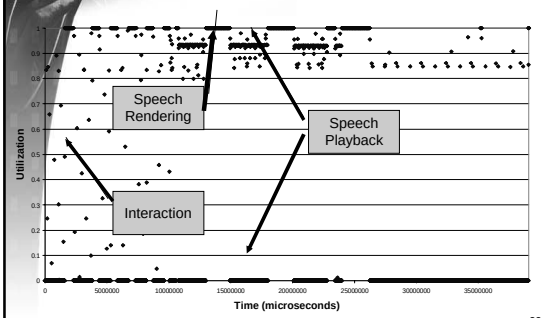
30

Why power-management in mobile systems

- Wireless systems have time varying computational loads
- Wireless systems often have a Quality / Power trade-off that can be exploited to suit application needs
- Wireless systems need to be resilient to variations in the environment
- Wireless systems have to be energy-efficient because they are battery powered

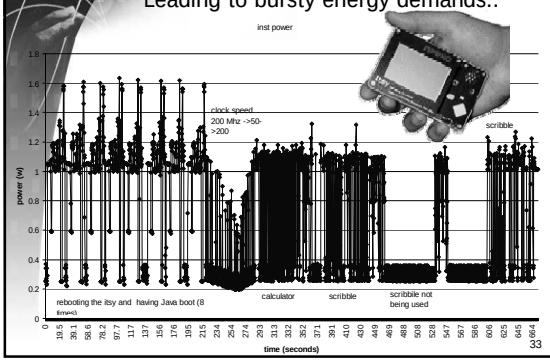
31

Interactive applications are usually bursty



32

Leading to bursty energy demands..



33

Efficient Architectures for Embedded Systems

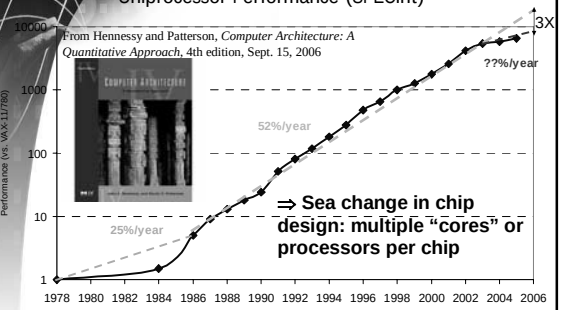
34

Conventional Wisdom (CW) in Computer Architecture [Patterson Hotchips2006]

- Old CW: Power is free, Transistors expensive
- New CW: "Power wall" Power expensive, Xtors free (Can put more on chip than can afford to turn on)
- Old: Multiplies are slow, Memory access is fast
- New: "Memory wall" Memory slow, multiplies fast (200 clocks to DRAM memory, 4 clocks for FP multiply)
- Old : Increasing Instruction Level Parallelism via compilers, innovation (Out-of-order, speculation, VLIW, ...)
- New: "ILP wall" diminishing returns on more ILP HW
- New: Power Wall + Memory Wall + ILP Wall = Brick Wall
  - Old CW: Uniprocessor performance 2X / 1.5 yrs
  - New CW: Uniprocessor performance only 2X / 5 yrs?

35

Uniprocessor Performance (SPECint)



- **VAX : 25%/year 1978 to 1986**
- **RISC + x86: 52%/year 1986 to 2002**
- **RISC + x86: ??%/year 2002 to present**

36

Déjà vu all over again?

"... today's processors ... are nearing an impasse as technologies approach the speed of light.."

David Mitchell, *The Transputer: The Time Is Now* (1989)

- Transputer had bad timing (Uniprocessor performance↑)

"We are dedicating all of our future product development to multicore designs. ... This is a sea change in computing"

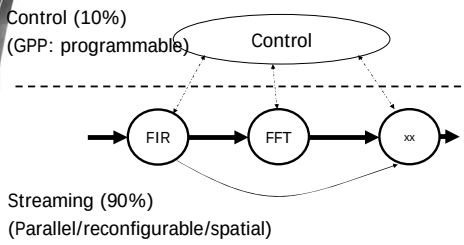
Paul Otellini, President, Intel (2005)

- All microprocessor companies switch to MP (2X CPUs / 2 yrs)

Manufacturer/Year	AMD/'05	Intel/'06	IBM/'04	Sun/'05
Processors/chip	2	2	2	8
Threads/Processor	1	2	2	4
Threads/chip	2	4	4	32

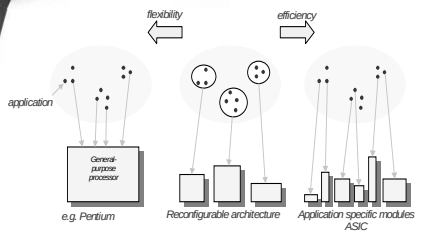
37

Typical Embedded Systems applications combination



38

Reconfigurable architectures

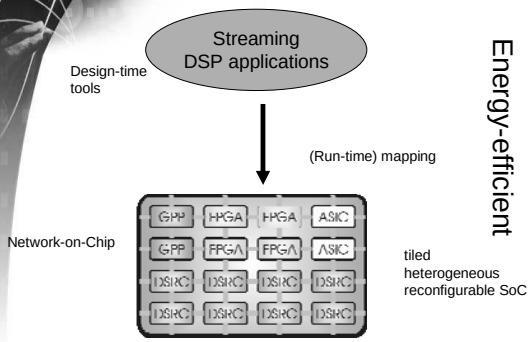


heterogeneous

Flexibility versus energy efficiency

39

Focus of Chameleon group



40

Efficient implementation of streaming applications

- Holistic approach: everything should fit together like a jigsaw puzzle

- Efficient processing platform
  - heterogeneous tiled SoC platform
  - efficient tile processors (e.g. Montium like)
- Efficient and predictable reconfigurable NoC
  - e.g. virtual channel network + network interface
- Efficient design-time tools for 'compiling' processes to tiles
- Run-time mapping of process graphs to SoC/NoC
  - determine at run-time near-optimal mapping
- Fast (partly) reconfiguration of tiles&communication
  - dynamic: while the system is in operation

41

Streaming Applications (1)

- Process graph with node (=processes/tasks) and edges (=communication/synchronization)
  - process: e.g. FFT, FIR, DCT, ...
  - communication: e.g. sample, OFDM symbol, video frame, ...
- Constant stream of data flows through the network
  - modeled as dataflow graph
  - like a lemming network
- For our domain streaming applications typically takes 80 to 100% of the processing / communication resources in a system
  - streams remain relatively fixed for a longer period
- Characteristics
  - predictable temporal behavior
  - predictable spatial behavior
  - relative simple local processing but huge amount of data
  - trend: communication will dominate energy costs rather than processing
  - adaptive: dynamic (partial) reconfiguration

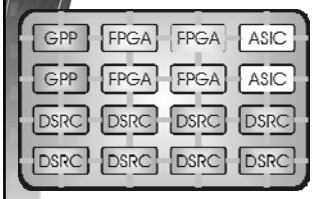
42

Streaming Applications (2)

- Application examples
  - signal processing for phased array antennas (6000)
    - radar + radio astronomy
  - wireless baseband processing (OFDM)
    - HiperLAN/2, WiMax, DAB, DRM, DVB, ...
  - multi-media processing (encoding/decoding)
    - e.g. MPEG / TV
  - medical image processing
    - one page of code but needs several hours of processing time on the fastest Pentium4 processor
  - sensor processing
    - e.g. remote surveillance cameras
    - automotive

43

Chameleon template heterogeneous tiled reconfigurable SoC



- General-purpose
- Fine-grained
- Coarse-grained
- Application specific

DSRC = domain specific reconfigurable core

- Heterogeneous reconfigurable processing tiles interconnected by a predictable on-chip interconnection network
- Match algorithm with architecture

44

Heterogeneous Match algorithm with architecture

- Bit-level architectures (FPGA)
  - PN-code generation
  - Turbo-encoding
- Word-level architectures (Montium, DSP)
  - FFT
  - FIR filters
  - Turbo-decoding
- General-purpose processor (GP)
  - Control oriented programs (frequent if/then/else constructs)
- Reconfigurable interconnect
  - Real-time multimedia streaming traffic
  - Bus vs. connection centric

45

What is <u>not</u> our focus?

- Control intensive applications
  - more suitable for standard GPPs
  - complex software operations on small amount of data
    - data caches help here (not in streaming case)
- We do assume
  - (part of the) data can be held locally in a tile
    - fast local data access
    - might mean tiling of datasets

46

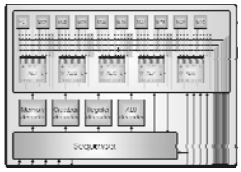
A tiled architecture

- Inherently exploits locality of reference
  - Energy and delay costs of transporting a signal over a wire will become much higher compared to the costs of computation
- Tiles do not grow in complexity with technology
  - Technology scales → more tiles on chip
- On-chip network
  - Higher bandwidth, lower power
- Small tile design
  - Can be highly optimized for low-power
  - Identical tiles have to be verified only once
- Partial and dynamic reconfiguration
- Tile can have its own (configurable) clock domain
- Our vision
  - FPGA like structure with cores instead of CLBs for streaming applications

47

Characteristics of the Montium Tile Processor

- Design goals
  - Energy-efficiency
  - Flexibility
  - Small control overhead
  - Avoid compiler and scheduler bottlenecks
- Algorithm domain
  - Digital Signal Processing
  - 2048p FFT, FIR, correlation, ...
- Features
  - 16-bit datapath
  - Signed integer and signed fixed-point arithmetic
  - Streaming I/O
  - Reconfigurable instruction set



Montium Tile Processor	
Process	0.12 µm
Voltage	1.10V
Energy	0.5 mW/MHz
Area (excluding wires)	1.8 mm²
Clock speed	45-150 MHz

www.recoresystems.com

48

Dynamic Reconfiguration

- Configuration per tile
- Fast re-configuration (micro-second scale)
  - all tiles can be configured in parallel
  - coarse-grain reconfiguration
    - word-level not on bit-level
    - e.g. Montium has 2.6kB configuration size
  - partial re-configuration
    - reconfiguration memory is RAM
    - changing # filter taps, coefficients, etc
    - e.g. change from FFT → iFFT takes 16 bytes

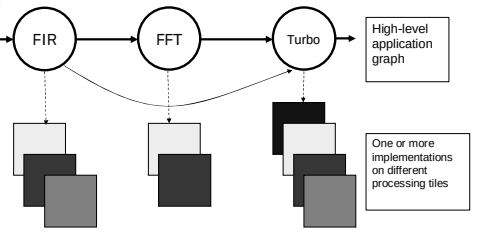
49

Energy-efficiency

- Locality of reference
  - Data and storage in one tile
  - Communication as local as possible
- Reduce control overhead
  - e.g. running a FIR almost all control signals stable
- Match algorithms with hardware architecture
  - PN-code generation in bit-level reconfigurable
  - Control-oriented on GPP
  - DSP algorithms on DSP or coarse-grain reconfigurable
- Dynamic Voltage (Frequency) Scaling per tile
  - even switch off unused tiles

50

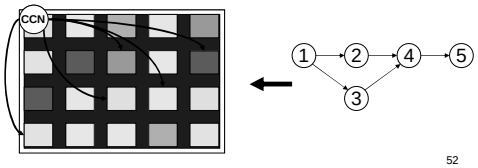
Definition of streaming DSP applications



51

Mapping applications to a SoC

- Mapping of the application is done at run-time
  - Processes → Processor Tiles
  - Communication streams → Network-on-Chip
- Central Coordination Node (CCN)
  - The node has a global overview of the system
  - SoC wide optimization to minimize the energy consumption
  - For NoC allocation of channels, routes, bandwidth etc.



52

Run-time mapping of applications to tiled architectures (dynamic reconfiguration?)

- Only at run-time the mix of applications is known
  - More applications might be running
  - new applications (after 'upgrade')
- Only at run-time the environment is known
  - Systems work in a dynamic environment
  - Adaptive: select algorithms/parameters at run-time
  - Applications can have QoS parameters
    - Video / sound quality requirements
- At run-time the available tiles are known
  - Other applications use your preferred tiles
  - Some tiles / communication links might be faulty
  - tiles may breakdown due to aging / ...

53

Seven reasons why coarse-grain reconfigurable failed as efficient architecture for streaming DSP domain

E.g. Chameleon Systems, Quick Silver, ...

- 1. Software trap
  - Parallel machines / CG reconfigurable hard to program
  - lack of high-level tool support
- 2. No holistic view
  - E.g. nice HW architecture but no software toolflow
  - E.g. a minimal NoC router but a complex Network Interface
- 3. Locality of Reference trap
  - hundreds of ALUs and memories and long wires doomed to fail
- 4. Communication / computation trap
  - HW accelerator designers tend to forget communication overhead (see also 2)
- 5. Lack of predictability
  - Compensate unpredictability (e.g. shared busses) with large buffers
- 6. Cost trap (configuration has overhead in area and energy)
- 7. fill in your own reason ....

54

Why are we still working on CG reconfigurable?

- It is our only option left
  - Pentiums are too inefficient
  - FPGAs have their problems
    - Long configuration times
    - Not energy efficient (locality of reference)
    - Difficult to program
    - Little support for dynamic reconfiguration
  - ASICs are too expensive / too inflexible
- But we have to learn from our past mistakes

55

Where can we find the clue?

- Tiled architectures
  - Have many tiles instead of one complex high-speed single processor
  - Distributed memory & distributed control
  - Dynamic (partial) reconfiguration
- Holistic view
  - Design teams with EE + CS + application developers
- Locality of Reference
- Streaming applications
  - Predictable
  - Guaranteed QoS NoC (throughput & latency)

56

Conclusion

- Tiled architectures are a good match for streaming DSP applications
- Holistic approach pays off
  - Efficient tiles
  - Efficient NoC
  - Dynamic partial reconfiguration
  - Good tooling
  - Run-time mapping
- Lots of open issues
  - how do we model applications
  - how do we find parallelism is sequential code?
    - implicit: use sequential C/Matlab code → let the compiler do the trick
    - explicit: use parallel programs e.g. Simulink, TTL, ... → let the user find the parallelism
  - what are the ideal tiles?
  - Efficient inter-tile communication with different clocks/voltages

57

Questions

- Acknowledgements
  - PhD students
    - Gerard Rauwerda
    - Marcel vd Burgwal
    - Yuanqing Guo
    - Nikolay Kavaldjiev
    - Pascal Wolkotte
    - Lodewijk Smit
    - Philip Holzenspies
    - Qiwei Zhang
    - Maarten Wiggers
    - Paul Heysters
    - Jan Jacobs
    - Mohammed Khatib
    - Tjerk Bijlsma

www.recoresystems.com

- more info see chameleon.ctit.utwente.nl

58