

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

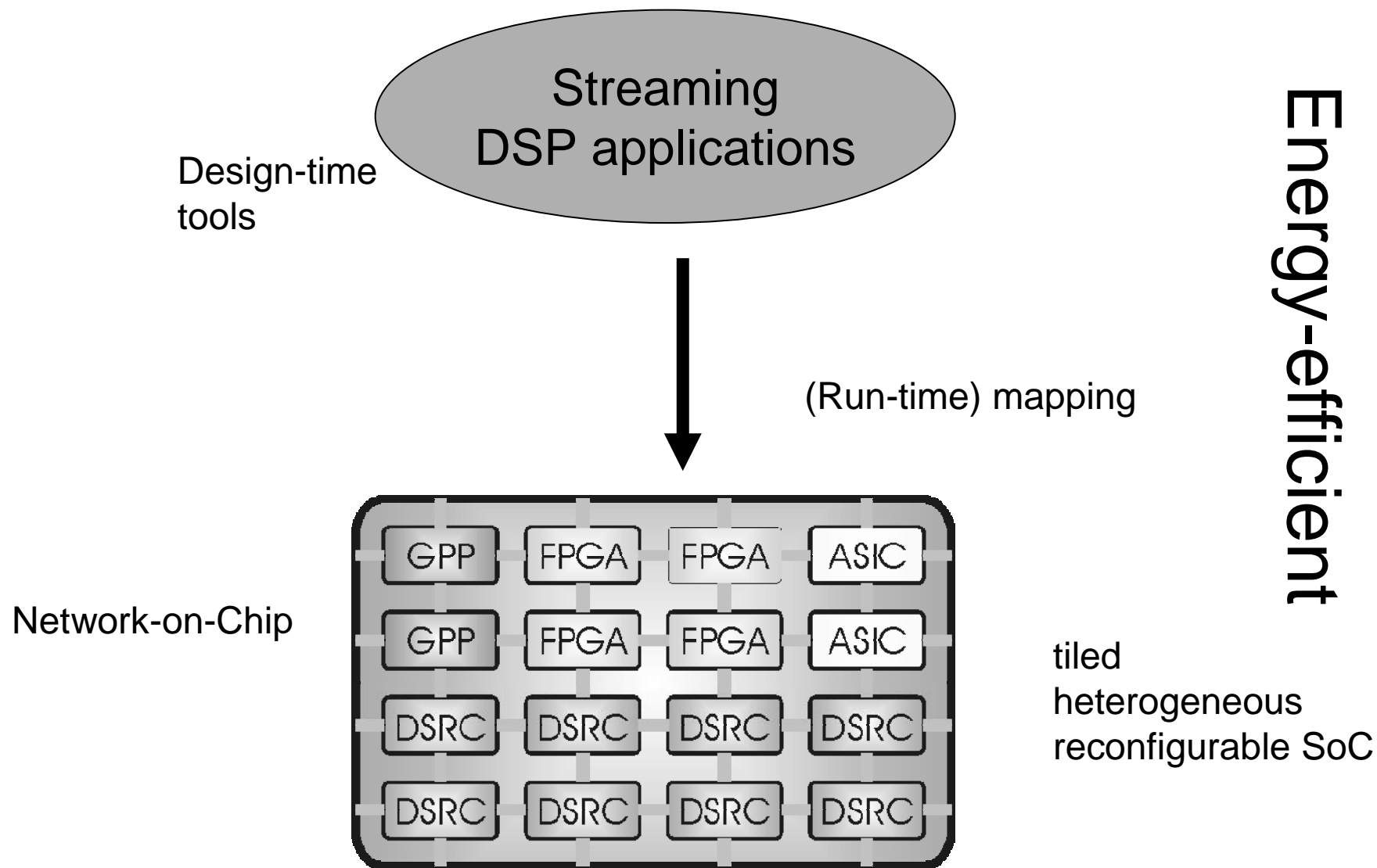
The Netherlands

Contents

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



Focus of Chameleon group



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

Motivations for 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

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

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

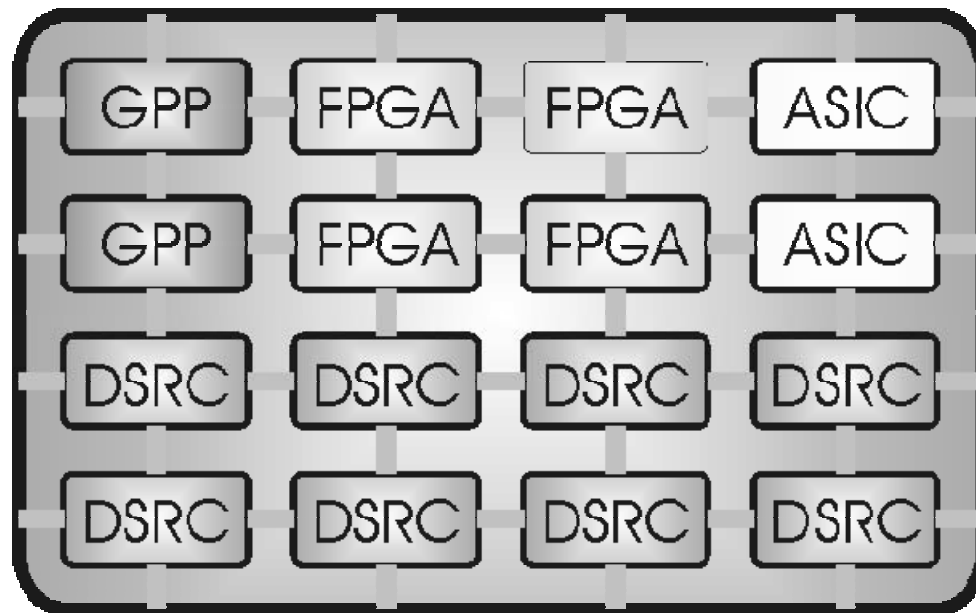
What is not our focus?

- 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



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

A tiled architecture

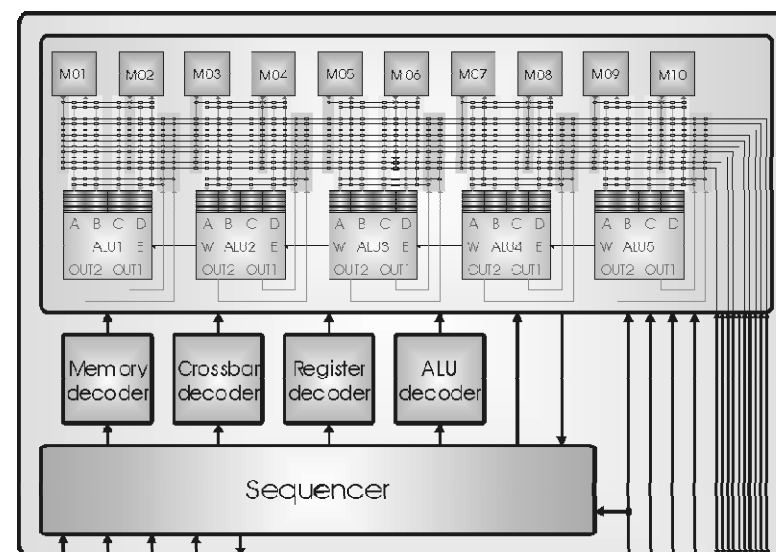
- Inherently exploits locality of reference
 - Energy and delay costs of transporting a signal over a wire will become much higher compared to the costs of computation
- Tiles do not grow in complexity with technology
 - Technology scales → more tiles on chip
- On-chip network
 - Higher bandwidth, lower 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



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 mm ²
Clock speed	45-150 MHz

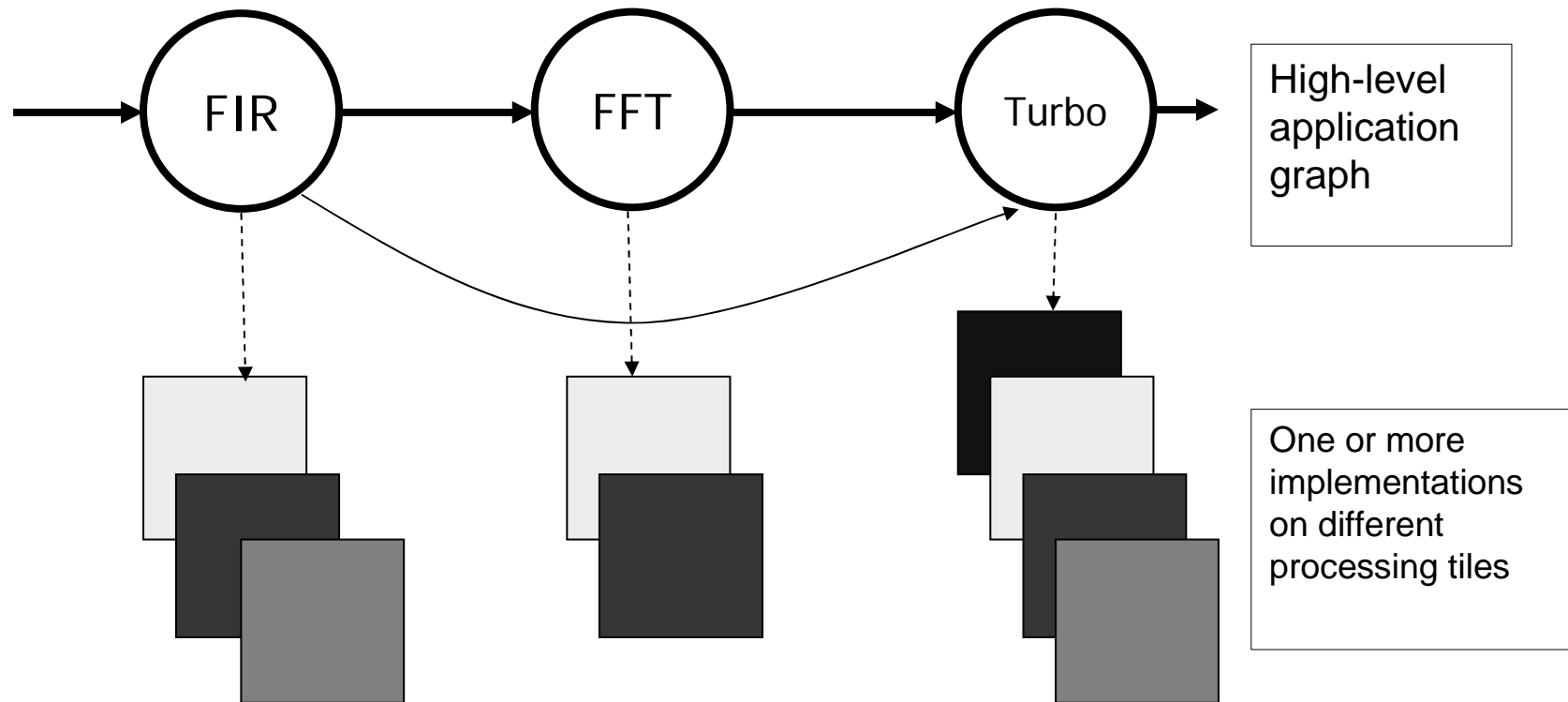
Dynamic Reconfiguration

- 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

Energy-efficiency

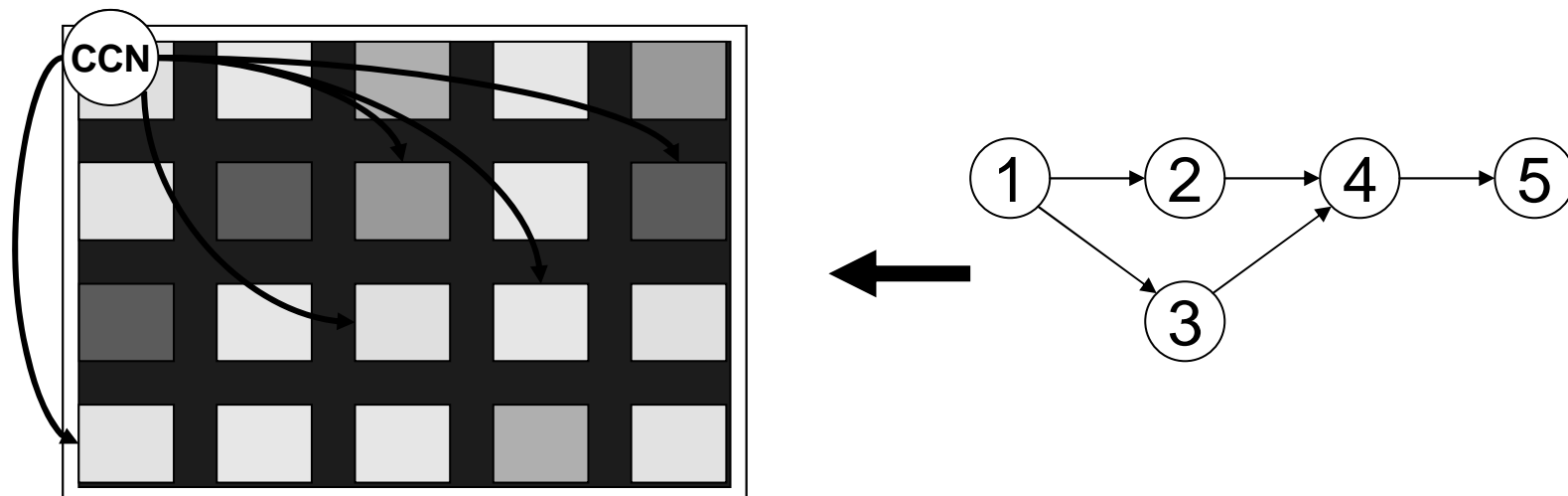
- Locality of reference
 - Data and storage in one tile
 - Communication as local as possible
- Reduce control overhead
 - e.g. running a FIR almost all control signals stable
- Match algorithms with hardware architecture
 - PN-code generation in bit-level reconfigurable
 - Control-oriented on GPP
 -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

Definition of streaming DSP applications



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.





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 / ...



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



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

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)

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 in 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

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
 - more info see chameleon.ctit.utwente.nl



RECORE

www.recoresystems.com