Outline: technology = devices + design
1. overview: motivation and vision
2. field-programmable devices: today
   - Xilinx Virtex-4, Virtex-5; Stretch S5
3. field-programmable design: today
   - enhance optimality and re-use
4. field-programmable devices: tomorrow
   - hybrid FPGA, die stacking
5. field-programmable design: tomorrow
   - guided synthesis, representation, upgradability
6. summary

Thanks to colleagues, students and collaborators from Imperial College, University of British Columbia, Chinese University of Hong Kong, University of Massachusetts Amherst, UK Engineering and Physical Sciences Research Council, Stretch, Xilinx

1. Motivation: good - Moore’s Law
   - rising
     - capacity
     - speed
   - falling
     - power/MHz
     - price

2a. Devices today: Virtex-4 FPGA
   - fine-grain fabric + special function units
   - challenge: use resources effectively
2b. Virtex-5 FPGA

- capacity – rises
- logic speed – rises
- I/O speed – rises
- all good news?

Source: Xilinx

Growing gap: amount of gates vs I/O

- I/O off-chip
  - serial
- I/O on-chip
  - scalable
  - flexible
  - easy to use
- interconnect
  - heterogeneous
  - customisable

Source: ITRS

2c. Software configurable engine: S5

- RISC Processor
  - Tensilica – Xtensa V
  - 32 KB I & D Cache
  - On-Chip Memory, MMU
  - 24 Channels of DMA, FPU

- Wide Register File (WRF)
  - 32 Wide Registers (WR)
  - 128-bit Wide
  - Load/Store Unit
  - 128-bit Load/Store
  - Auto Increment/Decrement
  - Immediate, Indirect, Circular
  - Variable-byte Load/Store
  - Variable-bit Load/Store

- ISEF
  - Instruction Specialization Fabric
  - Compute Intensive
  - Arbitrary Bit-width Operations
  - 3 Inputs and 2 Outputs
  - Pipelined, Bypassed, Interlocked
  - Random Logic Support
  - Internal State Registers

Source: Stretch

3. Design today: overview

- structural or register-transfer level (RTL)
  - e.g. VHDL, Verilog
  - low-level, little automation, small designs
- behavioural, system-level descriptions
  - e.g. SystemC (public-domain: systemc.org)
  - MARTES: +UML for real-time embedded systems
- general-purpose software languages
  - e.g. C, Java; with hardware support: Handel-C
  - high-level, large automation, large designs
- special-purpose descriptions
  - e.g. System Generator (signal processing)
  - high-level, domain-specific optimisations

3a. Enhance optimality and re-use

- design optimality: quality
  - select algorithm and devices: meet requirements
  - mapping: regular to systolic, rest to processor
  - I/O: dictates on-chip parallelism, buffering schemes
  - control speed/area/power: pipelining, layout plan
  - partitioning: coarse vs fine grain logic and memory
- design re-use: productivity
  - separate aspects specific to application/technology
  - library of customisable components with trade-offs
  - compose and customise to meet requirements
  - uniform interface to memory and I/O: hide details
  - pre-verified parts: ease system verification

Example: systolic summation tree

- n-input adder
  - tree of (n-1) 2-input adders
- each adder
  - has k stages
  - each stage has s-bit adder
- figure shows
  - $k = 3$
  - $s = 3$
- high $k$, low $s$
  - less cycle time
  - more area
  - less power
Finance application: value-at-risk

- sampling from multivariate Gaussian distribution
- DSP units: matrix multiplication
- systolic tree: accumulate result
- RC2000 board
  - Virtex-4 xc4vsx55
  - 400MHz
- 33 times faster than 2.2GHz quad Opteron
  (including all IO overheads, PC-FPGA communications, and using AMD optimised BLAS for software)

3b. Stretch program development

- profile code
- identify hotspots
- special instruction
  - implement "C" functions in single instructions
  - bit-width optim.
- software compiler
  - generate instruction
  - schedule instruction
- multiple data (WR)
  - perform operations in parallel
- efficient data movement
  - intrinsic load store operations
  - 20+ DMA channels

4. Devices tomorrow: more diversified

- PowerPC™ Processors with Auxiliary Processor Unit
- Flexible Soft Logic Architecture
- 1Gbps Differential I/O
- 450MHz PowerPC™ Processors

4a. Hybrid FPGA: architecture

- most digital circuits
  - datapath: regular, word-based logic
  - control: irregular, bit-based logic
- hybrid FPGA
  - customised coarse-grained block
    - domain-specific requirements
  - fine-grained blocks:
    - existing FPGA architecture
  - good match to computing applications for given domain

4g. Coarse-grained fabric library

- 6 benchmark circuits
  - digital signal processing kernels: e.g. bfly (for FFT)
  - linear algebra: e.g. matrix multiplication
  - complete application: e.g. bgm (financial model)
- circuits: partitioned to control + datapath
  - control: vendor tools to fine-grained units
  - datapath: manually map to coarse-grained units
- comparison
  - directly synthesized to Xilinx Virtex-II devices

Evaluation
Results

<table>
<thead>
<tr>
<th>Floating Point hybrid FPGA</th>
<th>XC2V3000-6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area (slices)</td>
<td>Delay (ns)</td>
</tr>
<tr>
<td>bfly</td>
<td>565</td>
</tr>
<tr>
<td>dscg</td>
<td>661</td>
</tr>
<tr>
<td>fir4</td>
<td>371</td>
</tr>
<tr>
<td>mm3</td>
<td>642</td>
</tr>
<tr>
<td>ode</td>
<td>545</td>
</tr>
<tr>
<td>bgm</td>
<td>1810</td>
</tr>
<tr>
<td>Geometric Mean</td>
<td></td>
</tr>
</tbody>
</table>

4b. On-chip memory bandwidth

- storage hierarchy: registers, LUT RAM, block RAM
- processor cache: address lack of I/O bandwidth

High density die stacking

Programmable circuit board

5a. Design tomorrow: guided synthesis

- guided transformation of design descriptions
  - automate tedious and error-prone steps
  - applicable to various levels of abstraction
- focus: two timing models
  - strict timing model: cycle-accurate - efficiency
  - flexible timing model: behavioural - productivity
- combine cycle-accurate and behavioural models
  - rapid development with high quality
  - design maintainability and portability
- based on high-level language
  - library developer: provide optimised building blocks
  - application developer: customise building blocks

Timing models: strict vs flexible

- cycle accurate model
  - total-ordering
  - resource-bound
- behavioural model
  - partial-ordering
  - abstract operations

\[
\text{delay} = b \cdot b - ((a \cdot c) << 2);
\]

\[
\text{if} (\text{delta} > 0) 
\]

\[
\text{num_sol} = 2;
\]

\[
\text{else if} (\text{delta} == 0) 
\]

\[
\text{num_sol} = 1;
\]

\[
\text{else} 
\]

\[
\text{num_sol} = 0;
\]
 Rapid design: automated scheduling

Implementation

Automatically generated results

5b. Data representation optimisation

Maintainability: retarget design

User Specified design constraints

5b. Data representation optimisation

- In1..In5
  - known width
- Out1..Out2
  - width determines accuracy
  - defined by user
- find representation
  - minimise width of nodes, e.g. X, Y
- trade-off in speed, area, power, error

- ffd: free-form deformation; dct: discrete cosine transform

Rapid design: automated scheduling

Maintainability: retarget design

Implementation

Automatically generated results

5b. Data representation optimisation

User Specified design constraints

5b. Data representation optimisation

- In1..In5
  - known width
- Out1..Out2
  - width determines accuracy
  - defined by user
- find representation
  - minimise width of nodes, e.g. X, Y
- trade-off in speed, area, power, error

- ffd: free-form deformation; dct: discrete cosine transform

Par { // ================== [stage 1]
  pipe_mult[0].in(b,b);
  pipe_mult[1].in(a,c);
}

// ================== [stage 8]

tmp0 = pipe_mult[0].q;
tmp1 = pipe_mult[1].q << 2;

// ================== [stage 9]

tmp2 = tmp0 - tmp1;

// ================== [stage 10]

if (tmp2 > 0) num_sol = 2;
else if (tmp2 == 0) num_sol = 1;
else num_sol = 0;

delta = tmp2;

Par { // ================= [stage 1]
  pipe_mult[0].in(b,b);
  pipe_mult[1].in(a,c);
}

Implementation

Automatically generated results

5b. Data representation optimisation

User Specified design constraints

5b. Data representation optimisation

- In1..In5
  - known width
- Out1..Out2
  - width determines accuracy
  - defined by user
- find representation
  - minimise width of nodes, e.g. X, Y
- trade-off in speed, area, power, error

- ffd: free-form deformation; dct: discrete cosine transform

Par { // ================== [stage 1]
  pipe_mult[0].in(b,b);
  pipe_mult[1].in(a,c);
}

// ================== [stage 8]

tmp0 = pipe_mult[0].q;
tmp1 = pipe_mult[1].q << 2;

// ================== [stage 9]

tmp2 = tmp0 - tmp1;

// ================== [stage 10]

if (tmp2 > 0) num_sol = 2;
else if (tmp2 == 0) num_sol = 1;
else num_sol = 0;

delta = tmp2;
FIR filter and DFT: area vs error

FIR filter and DFT: speed vs error

5c. Upgradable design

- Number of new users
- Time (months)
- Initial release
- Upgrade 1
- Upgrade 2
- Upgradable design
- Non-upgradable design

- Upgradability: minimise time-to-market
- Maximise time-in-market
- Add new functions, fix bugs
- Very rapid upgrade?

Dynamic upgrade: turbo coder

- Error correction code: add redundancy
- Need: fast, low-power, adapt to noise level
- Recursive systematic convolutional (RSC) encoder, decoder, interleaver

Self-tuning: run-time reconfiguration

- Adapt: less channel noise, so lower power
- Larger N_max: better correction, more area/power
- Sample channel noise every 250K bits
- Find Signal to Noise Ratio (SNR), select N_max
- If N_max ≠ current N_max, configure new bitstream
- Configuration overhead: about 30ms, 54.8mW

Performance of dynamic upgrade

<table>
<thead>
<tr>
<th></th>
<th>code 165.57</th>
<th>code 31.27</th>
<th>code 15.13</th>
</tr>
</thead>
<tbody>
<tr>
<td>speed (Kbps)</td>
<td>173.4</td>
<td>301.2</td>
<td>487.9</td>
</tr>
<tr>
<td>power (mW)</td>
<td>447.7</td>
<td>205.8</td>
<td>134.3</td>
</tr>
</tbody>
</table>

Dynamic

- Required reconfig: 10000
- Speed (Kbps): 359.1, 429.4, 598.6
- Power (mW): 216.2, 131.7, 111.6
- Power saving: 52%, 36%, 18%

- Up to 0.5x power, 2x speed over static decoder
- 100 times faster than processor decoder

Source: Xilinx

Source: Liang, Tessier, Goeckel
Other directions

- domain-specific design automation
  - languages + tools: for particle physics systems?
- multi-core, sensor network co-design
  - multiple hardware/software: FPGA + CPU + sensors
- extending processor and compiler capabilities
  - static and dynamic optimizations, self-tuning
- power-aware, radiation-aware design
  - transforms e.g. pipelining, damage monitoring
- rapid and informative design validation
  - simulation + FPGA prototype + formal verification

6. Summary

- good: Moore's Law, bad: productivity gap
- vision: unified design synthesis and analysis
- devices and design today
  - growing gap: amount of I/O and amount of logic
  - enhance optimality and re-use: I/O driven
- devices tomorrow
  - hybrid FPGA: multi-granularity fabric
  - 3D FPGA: customisable system-in-package
- design tomorrow
  - guided synthesis: optimised and portable design
  - data representation optimisation
  - upgradable and self-tuned design