
Reconfigurable Computing

Tony Givargis and Nikil Dutt

Introduction

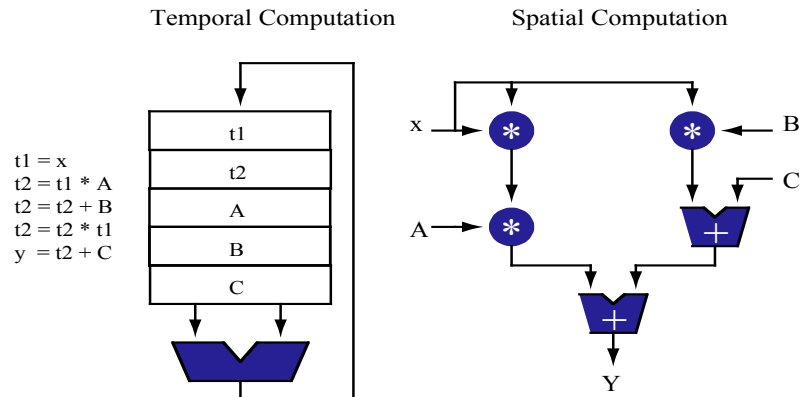
- ◆ Reconfigurable computing, a new paradigm for system design
 - Post fabrication **software personalization** for hardware computation
 - Traditionally based on FPGA technology
- ◆ Underlying technology enables swapping digital circuits to create **custom logic**:
 - In real-time, i.e., matter of seconds or milliseconds
 - In circuit
 - During execution time, i.e., on-the-fly
- ◆ Digital hardware (**custom logic**) used for speedup
 - Data parallel systems (*roots in array processing*)
 - DSP, multimedia
 - Long integer arithmetic
- ◆ Rapid prototyping of high performance systems (application specific)
- ◆ Computing structure granularity varies
 - Fine grained (FPGA) to coarse grained (array processors)

Why is Custom Logic Faster Than Software?

◆ Spatial vs. Temporal Computation

- Processors divide computation across time, dedicated logic divides across space

$$y = Ax^2 + Bx + C$$



3

Why is Custom Logic Faster Than Software?

◆ Specialization

- Instruction set may not provide the operations your program needs
- Processors provide hardware that may not be useful in every program or in every cycle of a given program
 - Multipliers
 - Dividers

◆ Instruction Memory

- Processors need lots of memory to hold the instructions that make up a program and to hold intermediate results.

◆ Bit Width Mismatches

- In general, processors have a fixed bit width, and all computations are performed on that many bits
 - Multimedia vector instructions (MMX) a response to this

4

Good Applications for Reconfigurable Computing

- ◆ Relatively small application graph
 - FPGAs have limited capacity
 - Simple control flow helps a lot
- ◆ Data Parallelism
 - Execute same computations on many independent data elements
 - Pipeline computations through the hardware
- ◆ Small and/or varying bit widths
 - Take advantage of the ability to customize the size of operators

5

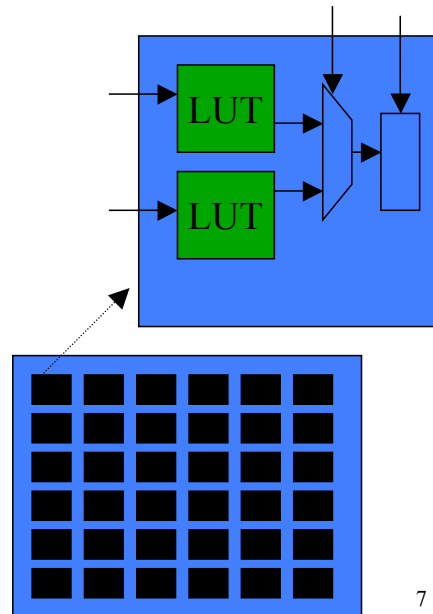
Field Programmable Gate Array (FPGA)

- ◆ Programmable logic blocks (PLB)
- ◆ Programmable connections between logic blocks (PIC)
- ◆ Programmable I/O (PIO)
- ◆ Digital only
- ◆ Low/medium/high performance (50/100/300 MHz)
- ◆ Low/medium/high density (20K/200K/2M gates)
- ◆ Bit stream programmable: SRAM, EEROM, Flash, Anti-fuse, etc.
- ◆ Execute/programming mode

6

FPGA Internals

- ◆ Traditionally 3 components:
 - PI/O, PIC, PLB
- ◆ Look up table (LUT) form the basic PLBs for logic generation
 - 3 to 4 inputs wide
 - Multiple LUTs / cell
- ◆ Programmable communication grid (PIC)
- ◆ Recent trends:
 - Incorporate memory and processor on chip



7

Why Compute With FPGAs?

- ◆ Huge performance gap between software and hand-designed hardware systems
 - Often 100-to-1 ratio of performance or performance/area
- ◆ Hardware systems not so good for general computing
 - Big design, cost barriers to implementation
 - Not practical to buy a new machine every time you want to run a different program
- ◆ Reconfigurable systems offer best-of-both-worlds
 - Run-time programmability
 - Hardware-level performance

8

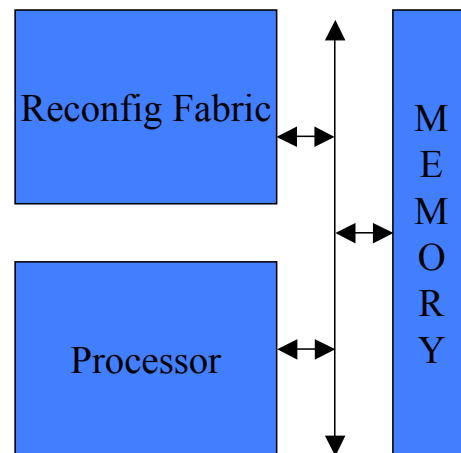
Why Haven't FPGA's Defeated CPUs?

- ◆ Capacity: Instructions are very dense representation, logic blocks aren't
- ◆ Tools: Compilers for reconfigurable logic aren't very good
 - Some operations are hard to implement on FPGAs
- ◆ One approach to capacity is to exploit the 90-10 rule of software
 - Run the 90% of code that takes 10% of execution time on a conventional processor
 - Run the 10% of code that takes 90% of execution time on reconfigurable logic
- ◆ Programmable-reconfigurable processors

9

Reconfigurable Computing Architecture

- ◆ Processor implements most of the function
- ◆ Reconfigurable fabric (e.g., FPGA) implements compute/data intensive
- ◆ Shared memory (dual port or DMA driven)
- ◆ Shared bus
 - bottleneck

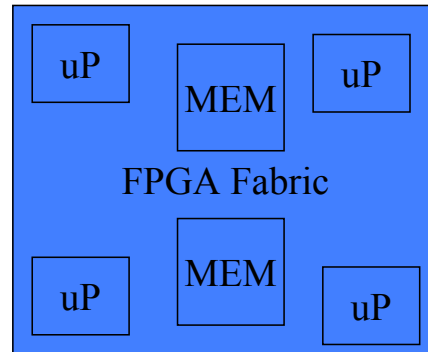


10

Reconfigurable Computing

Alternate Architecture

- ◆ One or more processors on FPGA chip
- ◆ One or more memory blocks on FPGA chip
- ◆ FPGA fabric used to design the communication and storage infrastructure



11

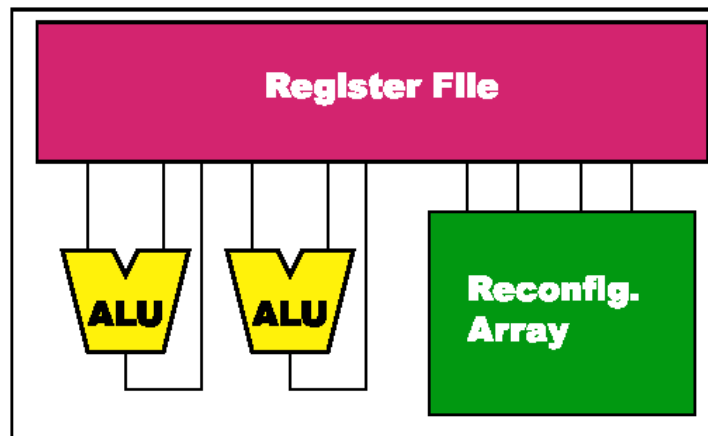
Introduction

- ◆ Reconfigurable computing, a new paradigm for system design
 - Post fabrication **software personalization** for hardware computation
 - Traditionally based on FPGA technology
- ◆ Underlying technology enables swapping digital circuits to create custom logic:
 - In real-time, i.e., matter of seconds or milliseconds
 - In circuit
 - During execution time, i.e., on-the-fly
- ◆ Digital hardware (custom logic) used for speedup
 - Data parallel systems (*roots in array processing*)
 - DSP, multimedia
 - Long integer arithmetic
- ◆ Rapid prototyping of high performance systems (application specific)
- ◆ Computing structure granularity varies
 - **Fine grained (FPGA) to coarse grained (array processors)**
 - Let's look at some examples of how they're used

12

Fine-Grained System: CHIMERA E

- Treat reconfigurable array as ALU within superscalar
 - Array implements some number of custom instructions for each program
 - Register file is interface between programmable and reconfigurable



13

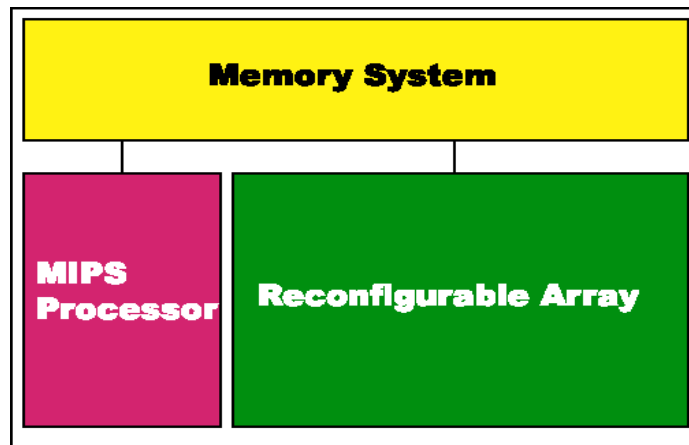
CHIMERA E

- Programmed in C
 - Instruction combining
 - Control localization
 - SIMD Within a Register
- Simulation Studies
 - Example applications only require 8 RFUOPs in the reconfigurable array
 - Equivalent to 32 rows in RFU
- Performance Results
 - Vary strongly from application to application
 - Also dependent on model used for RFU delay
 - Average speedup of 20-30%, one application sees >2x improvement

14

Coarse-Grained System: Garp

- ◆ Small programmable processor with large reconfigurable array
 - Interface through memory system



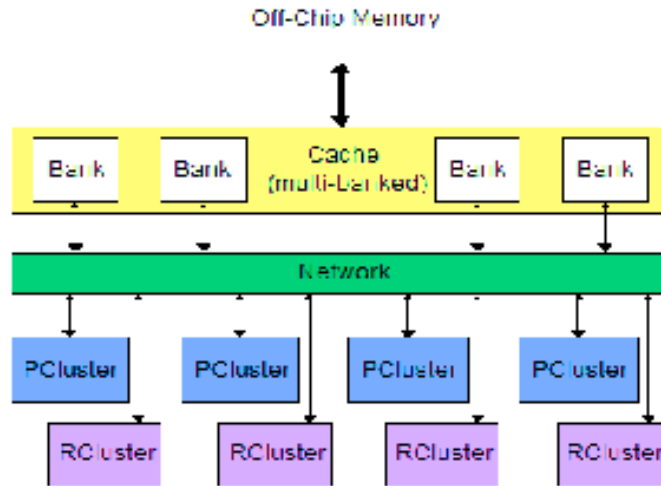
15

Garp

- Again, Programmed in C
 - Compiler attempts to map loop nests onto the reconfigurable array
- Data Encryption Standard
 - Estimate 24x speedup over UltraSPARC
- Image Dithering
 - 9x Speedup
- Sorting
 - 2x Speedup

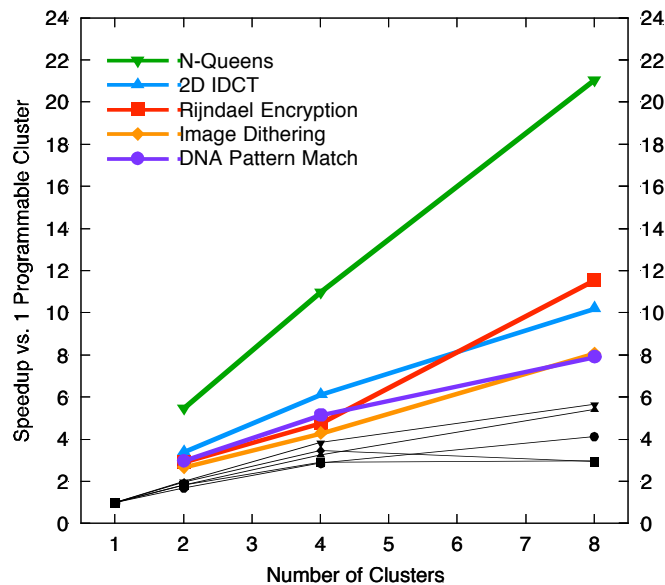
16

Mid-Grain: Amalgam



17

Amalgam Performance



18

Computing Abstraction

- ◆ One or more FPGA contexts (bit streams) pre-synthesized $B_0, B_1, B_2 \dots B_n$
- ◆ Cost for reconfiguration $C_{reconfig}$
- ◆ FPGA holds one context at a time
 - Entire computation/data can not be configured onto a single FPGA
- ◆ Notion of scheduling
 - Processing stops during reconfiguration
 - Next context being configured while current context is executing (double buffering)
 - NP complete problem (similar to partitioning/scheduling in codesign)

19

State of the Art in Reconfigurable Computing

- ◆ Some see RC as what will replace ASIC or even system design
- ◆ But:
 - Conventional FPGAs aren't really designed for reconfigurable computing
 - Design methodology not standard or automated
 - Architecture
 - Computation model
 - Compiler challenges
- ◆ A hot research topic

20

Some Reconfigurable Computing Successes

- ◆ **RSA Decryption**
 - Programmable-Active-Memory machine set record for decryption of RSA-encrypted data
- ◆ **DNA Sequence Matching**
 - Reconfigurable hardware has achieved 100x better performance than contemporary supercomputers
- ◆ **Signal Processing**
 - FPGA-based filters often get 10x better performance than DSP chips
 - Benefit from customization of hardware to the application
- ◆ **Emulation**
 - Use reconfigurable logic to simulate new processors at high speeds
- ◆ **Cryptographic Attacks**
 - High-performance low-cost implementations for breaking encryption algorithms