CprE / ComS 583 Reconfigurable Computing

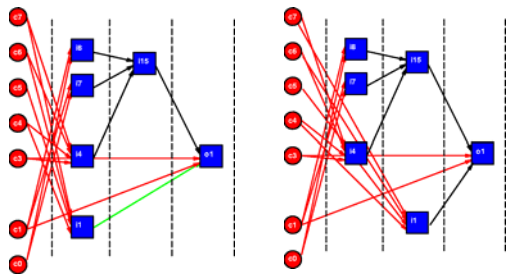
Prof. Joseph Zambreno
Department of Electrical and Computer Engineering
Iowa State University

Lecture #23 – Function Unit Architectures

Quick Points

- HW #3, #4 graded and returned
- Next week Thursday, project status updates
  - 10 minute presentations per group + questions
  - Upload to WebCT by the previous evening
  - Expected that you've made some progress!

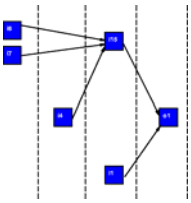
Allowable Schedules



Active LUTs (N_A) = 3

Sequentialization

- Adding time slots
  - More sequential (more latency)
  - Adds slack
    - Allows better balance



L=4 $\rightarrow N_A$=2 (4 or 3 contexts)

Multicontext Scheduling

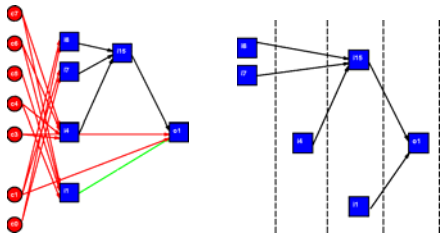
- "Retiming" for multicontext
  - **goal**: minimize peak resource requirements
- NP-complete
- List schedule, anneal
  - How do we accommodate intermediate data?
  - Effects?

Signal Retiming

- Non-pipelined
  - hold value on LUT Output (wire)
    - from production through consumption
  - Wastes wire and switches by occupying
    - For entire critical path delay L
    - Not just for 1/L'th of cycle takes to cross wire segment
  - How will it show up in multicontext?

Signal Retiming

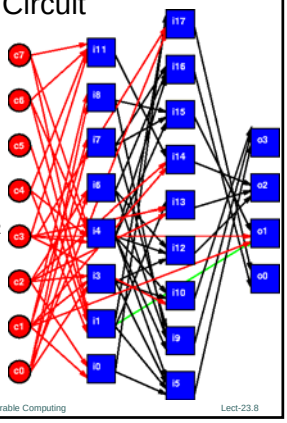
- Multicontext equivalent
  - Need LUT to hold value for each intermediate context



Full ASCII → Hex Circuit

- Logically three levels of dependence
- Single Context:

21 LUTs @ 880Kλ^2=18.5Mλ^2

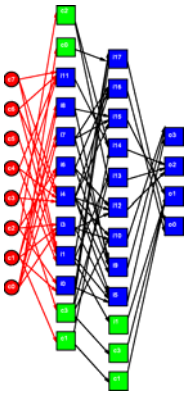


Multicontext Version

- Three contexts:

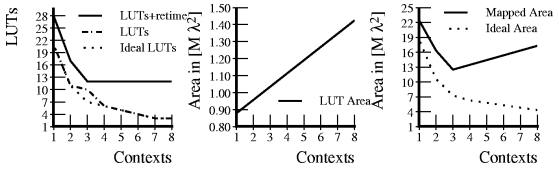
12 LUTs @ 1040Kλ^2=12.5Mλ^2

- Pipelining needed for dependent paths



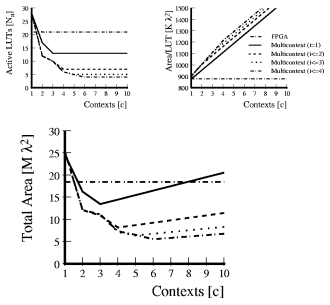
ASCII→Hex Example

- All retiming on wires (active outputs)
  - Saturation based on inputs to largest stage
  - With enough contexts only one LUT needed
  - Increased LUT area due to additional stored configuration information
  - Eventually additional interconnect savings taken up by LUT configuration overhead



Ideal≡Perfect scheduling spread + no retime overhead

ASCII→Hex Example (cont.)



@ depth=4, c=6: 5.5Mλ^2 (compare 18.5Mλ^2)

General Throughput Mapping

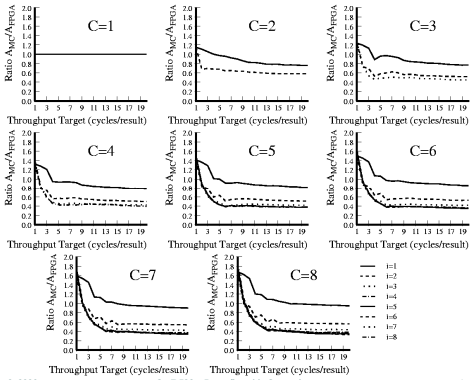
- If only want to achieve limited throughput
- Target produce new result every t cycles
- Spatially pipeline every t stages
  - cycle = t
- Retime to minimize register requirements
- Multicontext evaluation w/in a spatial stage
  - Retime (list schedule) to minimize resource usage
- Map for depth (i) and contexts (c)

Benchmark Set

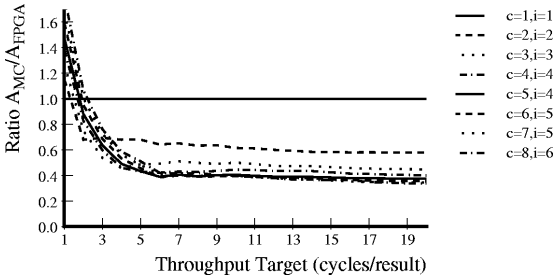
- 23 MCNC circuits
  - Area mapped with SIS and Chortle

Circuit	Mapped LUTs	Path Length	Circuit	Mapped LUTs	Path Length
5xp1	46	10	des	1267	13
9sym	123	7	e64	230	9
9symml	108	8	f51m	45	17
C499	85	10	misex1	20	6
C880	176	21	misex2	38	8
alu2	169	19	rd73	105	10
apex6	248	9	rd84	150	9
apex7	77	7	rot	293	16
b9	46	7	sao2	73	9
clip	121	9	vg2	60	9
cordic	367	13	z4ml	8	7
count	46	16			

Area v. Throughput



Area v. Throughput (cont.)



Reconfiguration for Fault Tolerance

- Embedded systems require high reliability in the presence of transient or permanent faults
- FPGAs contain substantial redundancy
- Possible to dynamically “configure around” problem areas
- Numerous on-line and off-line solutions

Column Based Reconfiguration

- Huang and McCluskey
- Assume that each FPGA column is equivalent in terms of logic and routing
  - Preserve empty columns for future use
  - Somewhat wasteful
- Precompile and compress differences in bitstreams

Figure 2: Configuration frame architecture in Xilinx Virtex-series FPGAs.

Column Based Reconfiguration

- Create multiple copies of the same design with different unused columns
- Only requires different inter-block connections
- Can lead to unreasonable configuration count

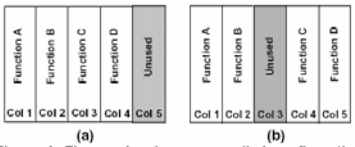


Figure 4: The overlapping precompiled configuration. (a) Base configuration. (b) Alternative configuration when column 3 is intentionally unused.

Column Based Reconfiguration

- Determining differences and compressing the results leads to “reasonable” overhead
- Scalability and fault diagnosis are issues

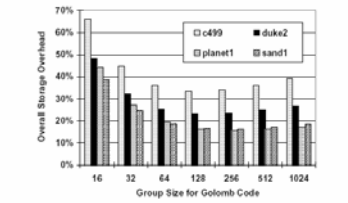


Figure 7: Storage overhead for the overlapping precompiled configuration scheme.

Summary

- In many cases cannot profitably reuse logic at device cycle rate
  - Cycles, no data parallelism
  - Low throughput, unstructured
  - Dissimilar data dependent computations
- These cases benefit from having more than one instructions/operations per active element
- Economical retiming becomes important here to achieve active LUT reduction
- For c=[4,8], I=[4,6] automatically mapped designs are 1/2 to 1/3 single context size

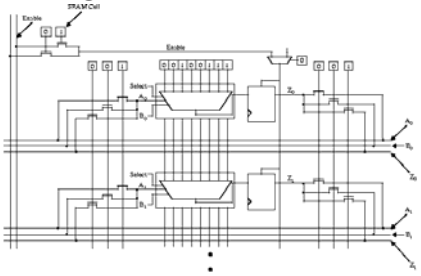
Outline

- Continuation
- Function Unit Architectures
  - Motivation
  - Various architectures
  - Device trends

Coarse-grained Architectures

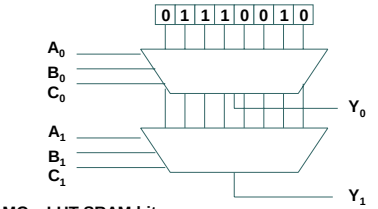
- DP-FPGA
  - LUT-based
  - LUTs share configuration bits
- Rapid
  - Specialized ALUs, mutlipliers
  - 1D pipeline
- Matrix
  - 2-D array of ALUs
- Chess
  - Augmented, pipelined matrix
- Raw
  - Full RISC core as basic block
  - Static scheduling used for communication

DP-FPGA



- Break FPGA into datapath and control sections
- Save storage for LUTs and connection transistors
- Key issue is grain size
- Cherepacha/Lewis – U. Toronto

Configuration Sharing



MC = LUT SRAM bits

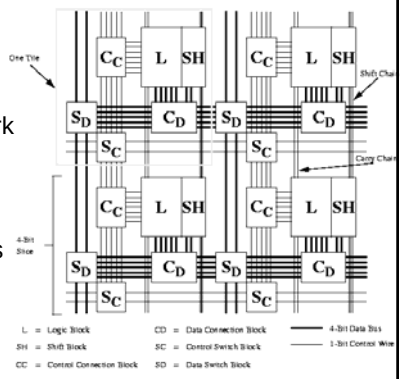
CE = connection block pass transistors

$$A(N) = \frac{MC + N * CE}{N} = \frac{MC}{N} + CE$$

Set MC = 2-3CE

Two-dimensional Layout

- Control network supports distributed signals
- Data routed as four-bit values



DP-FPGA Technology Mapping

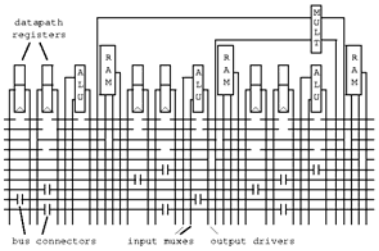
- Ideal case would be if all datapath divisible by 4, no "irregularities"
- Area improvement includes logic values only
- Shift logic included

RaPiD

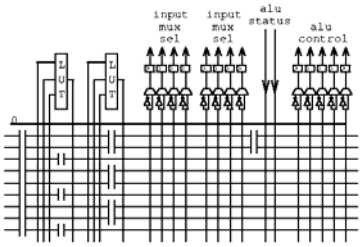
- Reconfigurable Pipeline Datapath
- Ebeling –University of Washington
- Uses hard-coded functional units (ALU, Memory, multiply)
- Good for signal processing
- Linear array of processing elements

RaPiD Datapath

- Segmented linear architecture
- All RAMs and ALUs are pipelined
- Bus connectors also contain registers



RaPiD Control Path

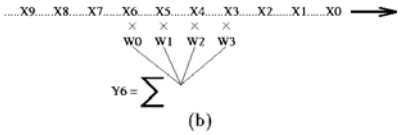


- In addition to static control, control pipeline allows dynamic control
- LUTs provide simple programmability
- Cells can be chained together to form continuous pipe

FIR Filter Example

```
for i := NumTaps-1 to NumX-1
  Y[i] := 0
  for j := 0 to NumTaps-1
    Y[i] := Y[i] + X[i-j]*W[j]
  end
end
```

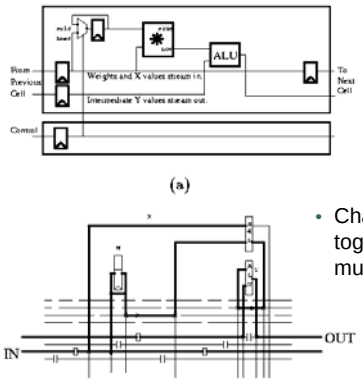
(a)



(b)

- Measure system response to input impulse
- Coefficients used to scale input
- Running sum determined total

FIR Filter Example (cont.)

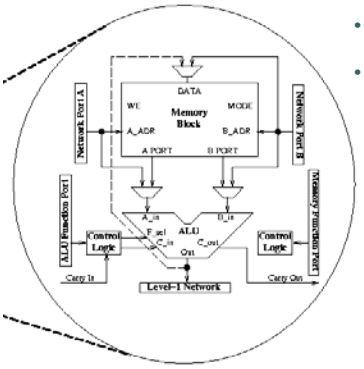


- Chain multiple taps together (one multiplier per tap)

MATRIX

- Dehon and Mirsky -> MIT
- 2-dimensional array of ALUs
- Each Basic Functional Unit contains "processor" (ALU + SRAM)
- Ideal for systolic and VLIW computation
- 8-bit computation
- Forerunner of SiliconSpice product

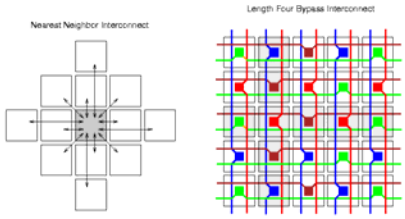
Basic Functional Unit



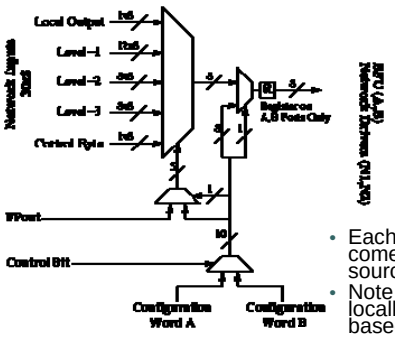
- Two inputs from adjacent blocks
- Local memory for instructions, data

MATRIX Interconnect

- Near-neighbor and quad connectivity
- Pipelined interconnect at ALU inputs
- Data transferred in 8-bit groups
- Interconnect not pipelined



Functional Unit Inputs



- Each ALU inputs come from several sources
- Note that source is locally configurable based on data values

FIR Filter Example

- For k-weight filter 4K cells needed
  - One result every 2 cycles
- K/2 8x8 multiplies per cycle

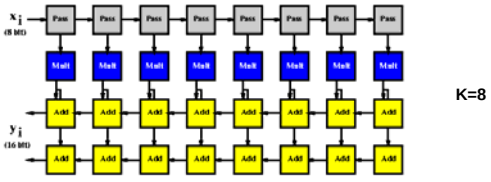
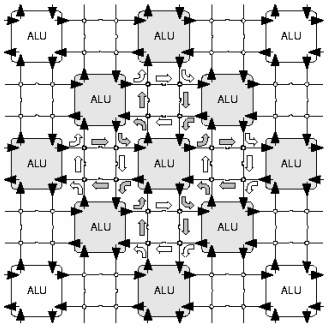


Figure 5: Systolic Convolution Implementation

Chess

- HP Labs – Bristol, England
- 2-D array – similar to Matrix
- Contains more “FPGA-like” routing resources
- No reported software or application results
- Doesn’t support incremental compilation

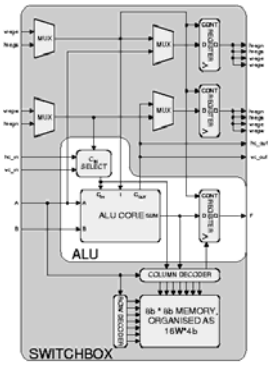
Chess Interconnect



- More like an FPGA
- Takes advantage of near-neighbor connectivity

Figure 1 CHESS layout and nearest neighbour wiring

Chess Basic Block



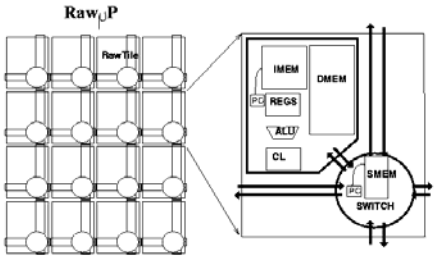
- Switchbox memory can be used as storage
- ALU core for computation

Reconfigurable Architecture Workstation

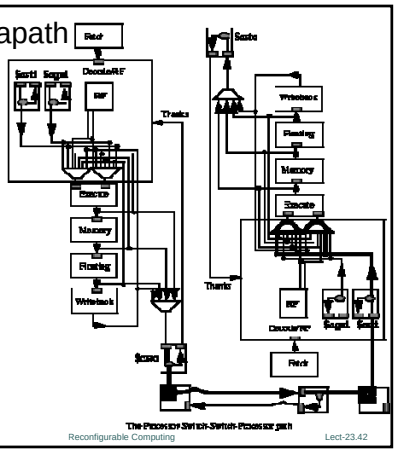
- MIT Computer Architecture Group
- Full RISC processor located as processing element
- Routing decoupled into switch mode
- Parallelizing compiler used to distribute work load
- Large amount of memory per tile

RAW Tile

- Full functionality in each tile
- Static router located for near-neighbor communication



RAW Datapath



Raw Compiler

- Parallelizes compilation across multiple tiles
- Orchestrates communication between tiles
- Some dynamic (data dependent) routing possible

Summary

- Architectures moving in the direction of coarse-grained blocks
- Latest trend is functional pipeline
- Communication determined at compile time
- Software support still a major issue