ESE5320:
System-on-a-Chip Architecture

Day 8: September 25, 2024

Spatial Computations

Preclass 3 pages

Today

- Accelerator Pipelines (Part 1)
- FPGAs (Part 2)
- Computational Capacity (Part 3)
  - Zynq

Message

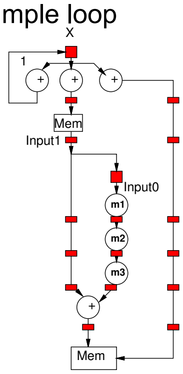
- Custom accelerators efficient for large computations
  - Exploit Instruction-level parallelism
  - Run many low-level operations in parallel
- Field-Programmable Gate Arrays (FPGAs)
  - Allow post-fabrication configuration of custom accelerator pipelines
  - Can offer high computational capacity

Accelerator Datapaths

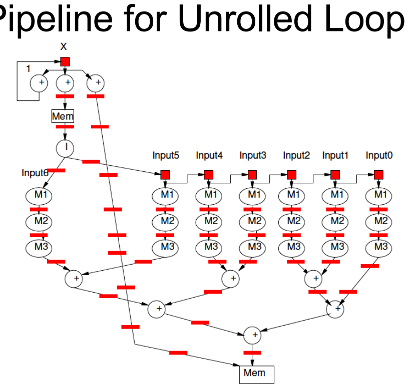
Pipeline Graph

- Last time: pipelined simple loop

```
unsigned char Input0, Input1;
Input1=Input[Y * INPUT_WIDTH + X + 0];
 for (int X = 0; X < OUTPUT_WIDTH; X++)
{
    unsigned int Sum;
    Input0=Input1;
    Input1=Input[Y * INPUT_WIDTH + X + 1];
    Sum = Coefficients_0 * Input0 + Input1;
    Output[Y * OUTPUT_WIDTH + X] = Sum;
}
```

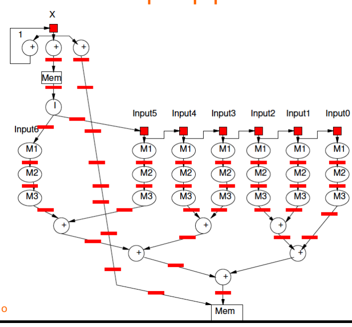


Pipeline for Unrolled Loop



Preclass 1

- For fully unrolled loop shown, how many instructions per pipeline cycle?
  - Add
  - Mpy
  - Load
  - Store



7

Spatial Pipeline

- Can compute equivalent of tens of "instructions" in a cycle
- Wire up primitive operators
  - No indirection through register file, memory
- Pipeline for operator latencies
- Any dataflow graph of computational operations

8

Operators

- Can assemble any custom operators
  - Ones may not have in generic processor
- Processor
  - Add, bitwise-xor/and/or
  - Maybe: floating-point add, multiply
- Less likely
  - Square-root, exponent, cosine, encryption (AES) step, polynomial evaluate, log-number-system

9

Accelerators

- Compression/decompression
- Encryption/decryption
- Encoding (ECC, Checksum)
- Discrete Cosine Transform (DCT)
- Sorter
- Taylor Series Approximation of function
- Transistor evaluator
- Tensor or Neural Network evaluator

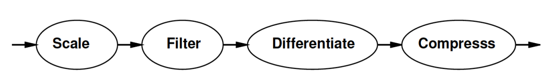
10

Streaming Dataflow

- Replace operator with custom accelerator
- Stream data to/from it

11

Streaming Dataflow Example



12

Application-Specific SoCs

- For dedicated applications may build custom hardware for accelerators
  - Layout VLSI, fab unique chips
  - ESE3700, 5700
- Video-encoder – include custom DCT, motion-estimation engines

13

Apple A16 Bionic

- ? 110+mm^2, 4nm
- 16 Billion Tr.
- iPhone 14
- 6 ARM cores
  - 2 fast (3.5GHz)
  - 4 low energy (2GHz)
- 5 custom GPUs (1.4GHz)
- 16 Neural Engines
  - 17 Trillion ops/s?

https://en.wikipedia.org/wiki/Apple_A16

14

Customizable Accelerators

- With post-fabrication configurability can exploit without unique fabrication
- Need programmable substrate that allows us to wire-up computations

15

Field-Programmable Gate Arrays

FPGAs

Part 2

16

FPGA

- Idea: Can wire up programmable gates in the "field"
  - After fabrication
  - At your desk
  - When part "boots"
- Like a "Gate Array"
  - But not hardwired

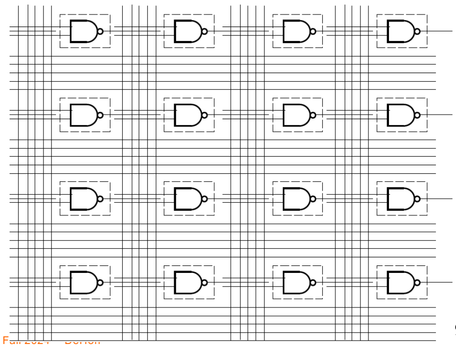
17

Gate Array

- Idea: Provide a collection of uncommitted gates
- Create your "custom" logic by wiring together the gates
- Less layout, fewer masks than full custom IC Chip
  - Since only wiring together pre-fab gates
  - → lower cost (fewer masks)
  - → lower manufacturing delay

18

Gate Array



19

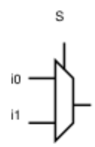
GA→FPGA

- Remove the need to even fabricate the wiring mask
- Make "customization" soft
- Key trick:
  - Use reprogrammable configuration bits
  - Typically: static-RAM bits
    - Like SRAM cells or latches in memory
    - Hold a configuration value
  - Configuration bits define gates, wiring

20

Multiplexer Gate

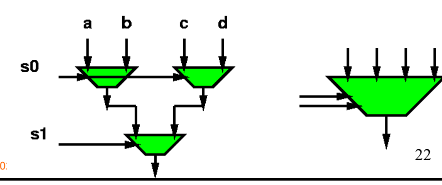
- MUX
  - When S=0, output=i0
  - When S=1, output=i1



Out = /s*i0 + s*i1

21

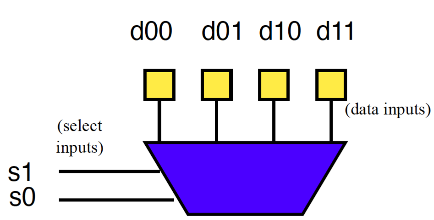
Mux with configuration bits = programmable gate



22

Preclass 4a

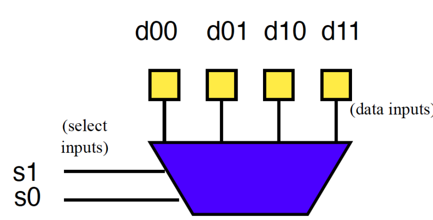
- How do we program to behave as and2?



23

Preclass 4b

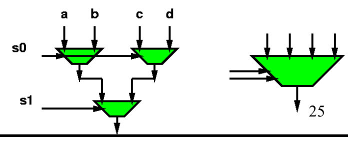
- How do we program to behave as xor2?



24

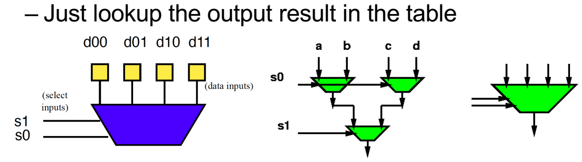
Mux as Logic

- Just by “configuring” data into this mux4,
  - Can select **any** two input function



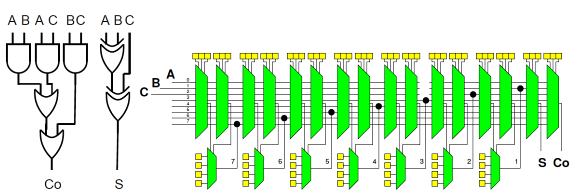
LUT – LookUp Table

- When use a mux as programmable gate
  - Call it a LookUp Table (LUT)
  - Implementing the Truth Table for small # of inputs
    - k-LUT: # of inputs =k (need mux-2^k)
  - Just lookup the output result in the table



Preclass 6

- How do we program full adder?



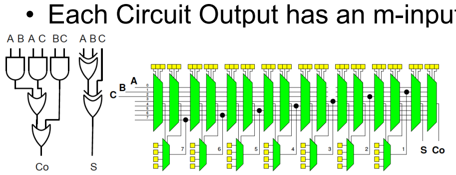
[Switch Google Doc]

FPGA

- Programmable gates + wiring
  - (both built from muxes w/ config. bits)
- Can wire up any collection of gates
  - Like a gate array

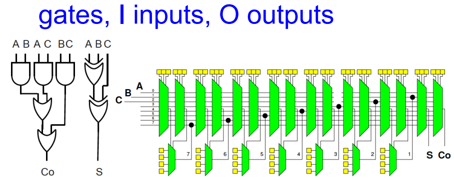
Simplistic FPGA (illustrate possibility)

- Every LUT input has a mux
- Every such mux has m=(N+I) inputs
  - An input for each LUT output (N 2-LUTs)
  - An input for each Circuit Input (I Circuit inputs)
- Each Circuit Output has an m-input mux



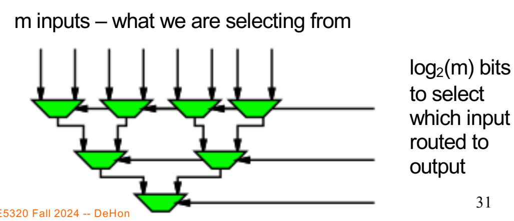
Simplistic FPGA (illustrate possibility)

- N 2-LUTs, I Circuit Inputs, O Circuit Outputs
- 2N+O muxes to connect
- Can build **any** combinational logic circuit that doesn’t need more than N 2-input gates, I inputs, O outputs



Preclass 3
How big is an m-input mux?

- In terms of 2-input muxes?
  - Warmup: how many for 4-input (Preclass 2)
  - Warmup: how many for 8-input (below)



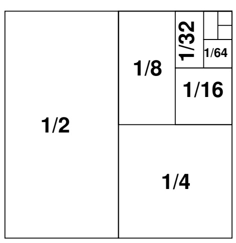
31

Math: Series Sums

- $A_0(1+r+r^2+r^3+r^4+\ldots)$
- $A_0(1+r+r^2+r^3+r^4+\ldots)*(1-r)$
  $=A_0+ A_0 r+ A_0 r^2+ A_0 r^3+ A_0 r^4+\ldots$
  $- A_0 r - A_0 r^2 - A_0 r^3 - A_0 r^4-\ldots$
  $= A_0$ (when $r<1$)
- $A_0(1+r+r^2+r^3+r^4+\ldots)*(1-r)=A_0$
- $A_0(1+r+r^2+r^3+r^4+\ldots)=A_0/(1-r)$

32

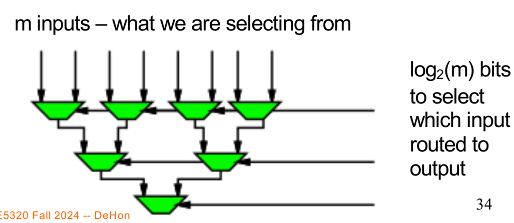
Receding Sum



33

Preclass 3
How big is an m-input mux?

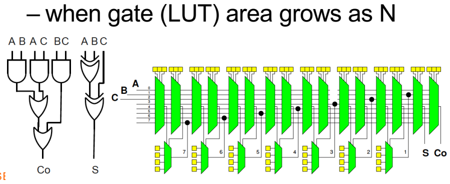
- m-input mux will require m-1 2-input muxes



34

Simplistic FPGA
(illustrate possibility…and expense)

- 2N+O m-input muxs; m=N+I
- Each m-input mux is m-1 2-input muxes
- Requires: (2N+O)*(N+I-1) 2-input muxes
- Mux area grows as $\sim N^2$
  - when gate (LUT) area grows as N



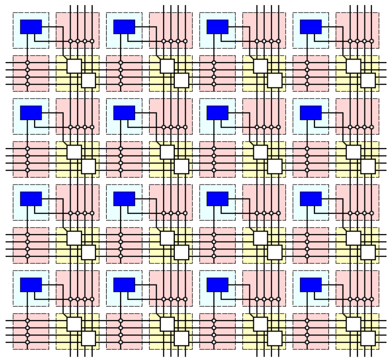
35

Interconnect

- Fully connected mux input is too expensive, growing as N^2
  - …and not necessary
- Want
  - To be able to wire up gates
  - Economical with wires and muxes
    - …and configuration bits
  - Exploit locality (keep wires short)

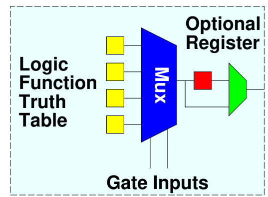
36

Simple FPGA

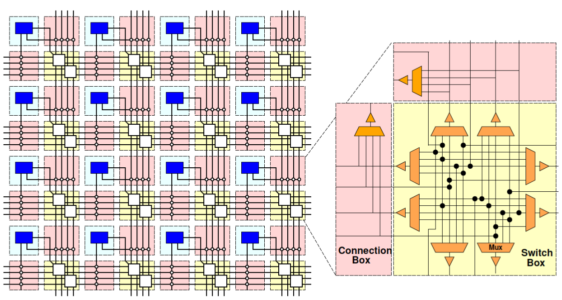


Register

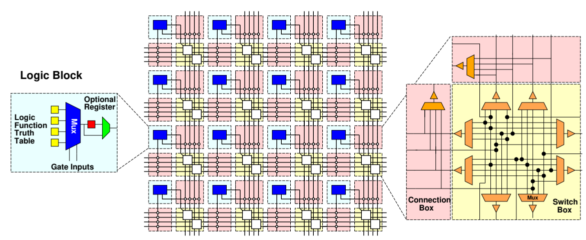
- Want to be able to pipeline logic
- …and generally hold state
  - E.g. implement hold Input-N in preclass 1
- Add optional register on each gate



Simple FPGA



Simple FPGA



FPGA Design

- Raises many architectural design questions
  - How big (many inputs) should the gates have? (what k should use for k-LUTs?)
    - Are LUTs really the right thing…
  - How rich is the interconnect?
    - Wires/channel
    - Wire length
    - Switching options

Modern FPGAs

- Logic Blocks
  - hardwired fast-carry logic
    - Can implement adder bit in single "LUT"
  - Speed optimized: 6-LUTs
  - Energy, Cost optimization: 4-LUTs
  - Clusters many LUTs into a tile
- Interconnect
  - Mesh, segments of length 4 and longer

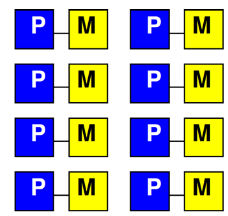
More than LUTs

- Should there be more than LUTs in the "array" fabric?
- What else might we want?

43

Embedded Memory

- One flip-flop per LUT doesn't store state densely
- Want memory close to logic



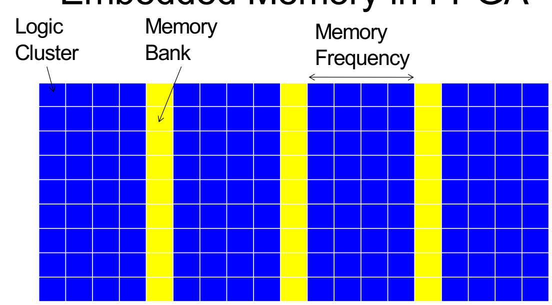
44

Embed Memory in Array

- Replace logic clusters
- Convenient to replace columns
  - Since area of memory may not match area of logic cluster

45

Embedded Memory in FPGA



Memory banks on AMD/Xilinx called BRAMs (Block RAMs)

46

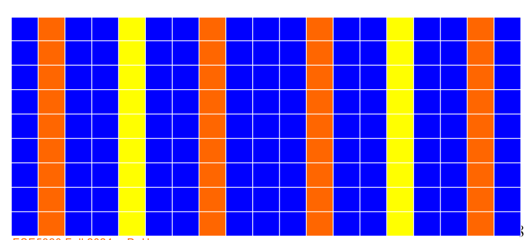
Hardwired Multipliers

- Can build multipliers out of LUTs
  - Just as can implement multiplies on processor out of adds
- But, custom multiplier is smaller than LUT-configured multiplier
  - …and multipliers common in signal processing, scientific/engineering compute

47

Multiplier Integration

- Integrate like memories
  - Replace columns



48

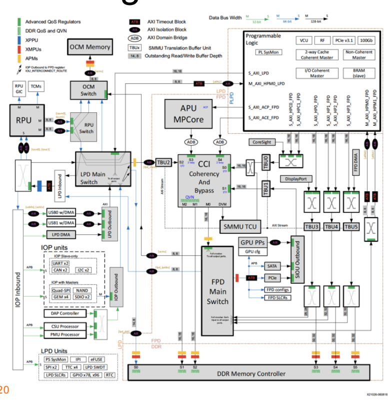
More FPGA Architecture Design Questions

- Size of Memories? Multipliers?
- Mix of LUTs, Memories, Multipliers?
- Add processors? Floating-point?
- Other hardwired blocks?
- How manage configuration?

Zynq MPSoC

Part 3

Programmable SoC



UG1085 Xilinx UltraScale Zynq TRM (p27)

ZU3EG (Ultra96)

- 6-LUTs: 70,560
- DSP Blocks: 360
  - 18x27 multiply, 48b accumulate
- Block RAMs (BRAMs): 216
  - 36Kb
  - Dual port
  - Up to 72b wide (512x72)

DSP48

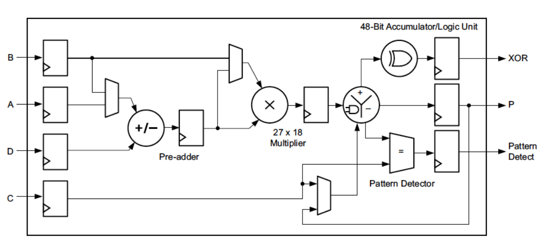


Figure 1-1: **Basic DSP48E2 Functionality**

Xilinx UG579 UltraScale DSP Slice User's Guide

Preclass 5

Approximating	Resources	Cycle	Per second
Zynq LUTs	70,000 adder bits	0.5 GHz	
4x ARM Scalar	4x2x64 adder bits	1.2 GHz	
4x ARM Neon	4x1x64 adder bits	1.2 GHz	
Zynq DSP	360 multiply-accumulates	0.5 GHz	
4x ARM Scalar	4x(1 mpy+1add)	1.2 GHz	
4x ARM Neon	4x1x4 multiply-accumulates	1.2 GHz	

- How compare between ARM scalar, ARM NEON and FPGA array?
  - Adder-bits/second?
  - Multiply-accumulators/second?

Capacity → Density

- Says Zynq has high computational capacity in FPGA
- More broadly
  - FPGA can have more compute/area than processor
    - E.g., more adder bits in some fixed area
  - SIMD can have more compute/area than processor (Day 6)
    - How wide SIMD can you exploit?

55

FPGA Potential

- FPGA Array has high raw capacity
- Exploitable when computation has high regularity
  - Uses the same computation over-and-over
  - High throughput on a computation
  - Build customized accelerator pipeline to match the computation
- Low-hanging fruit
  - Operator/function takes most of the compute time

56

90/10 Rule

- Observation that code is not used uniformly
- 90% of the time is spent in 10% of the code
- Knuth: 50% of the time in 2% of the code
- Opportunity
  - Build custom datapath in FPGA (hardware) for that 10% (or 2%) of the code

57

Big Ideas

- Custom accelerators efficient for large computations
  - Exploit Instruction-level parallelism
  - Run many low-level operations in parallel
- Field Programmable Gate Arrays (FPGAs)
  - Allow post-fabrication configuration of custom accelerator pipelines
  - Can offer high computational capacity

58

Admin

- Reading for Day 9 on canvas
- HW4 due on Friday
- HW5 out soon
  - Heavier – start early…
  - Vivado HLS synthesis slow (plan for it)

59