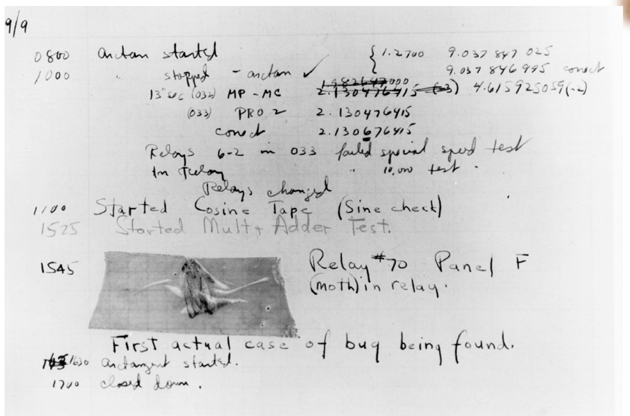
EECS151 : Introduction to Digital Design and ICs

Lecture 10 – Pipelining, FPGAs

Bora Nikolić

Stalking the Elusive Computer Bug

From at least the time of Thomas Edison, U.S. engineers have used the word “bug” to refer to flaws in the systems they developed. This short word conveniently covered a multitude of possible problems. It also suggested that difficulties were small and could be easily corrected. IBM engineers who installed the ASSC Mark I at Harvard University in 1944 taught the phrase to the staff there. Grace Murray Hopper used the word with particular enthusiasm in documents relating to her work. In 1947, when technicians building the Mark II computer at Harvard discovered a moth in one of the relays, they saved it as the first actual case of a bug being found. In the early 1950s, the terms “bug” and “debug,” as applied to computers and computer programs, began to appear not only in computer documentation but even in the popular press.



Peggy Aldrich Kidwell,
IEEE Annals of the History of Computing , 1998.

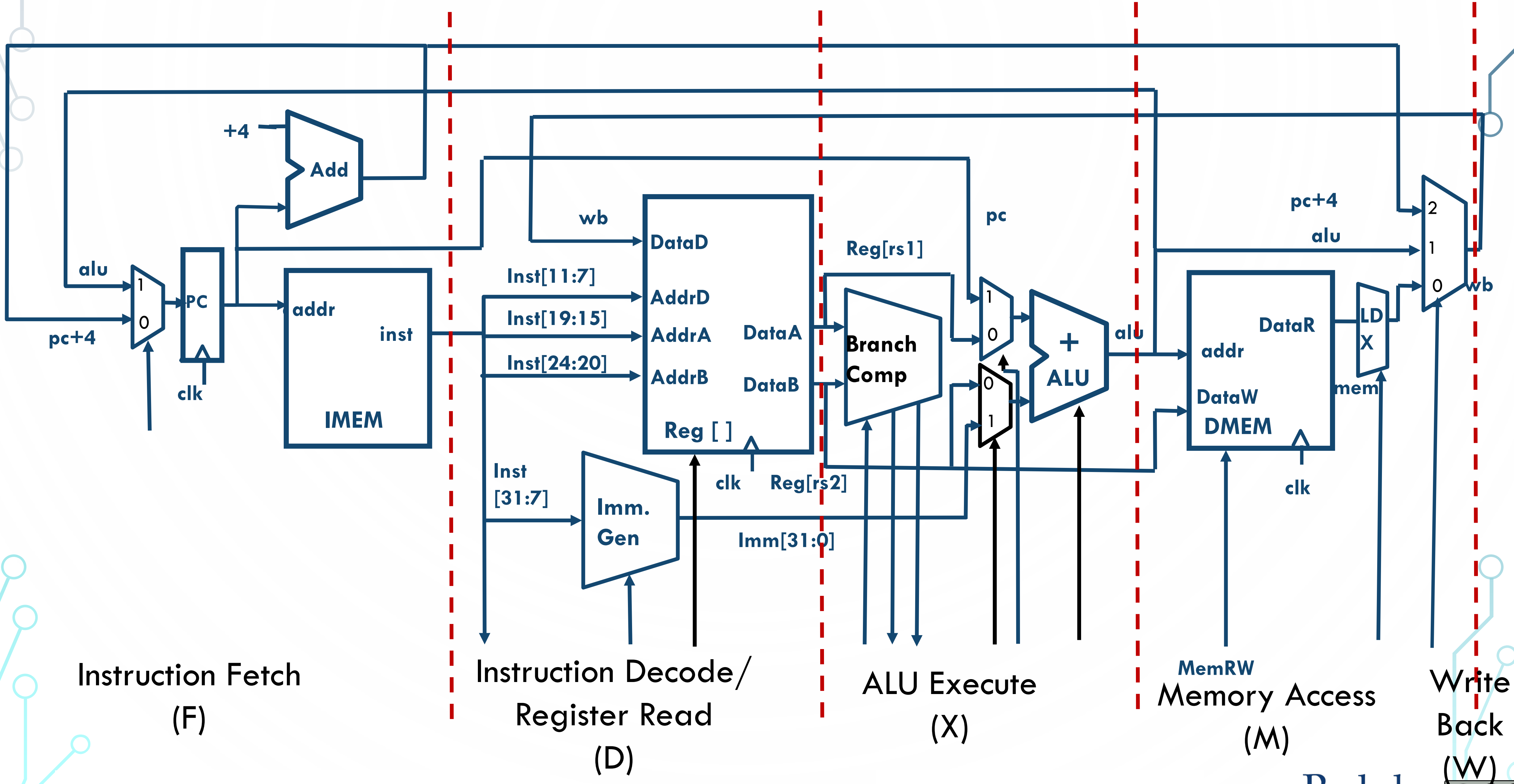
Grace Murray Hopper
Logbook of the Mark II for 9/9/1947

Review

- RISC-V ISA
  - Completed the datapath with B-, J-, U-instructions
- Control
  - Can be implemented as a ROM while prototyping
  - Synthesized as custom logic
- Pipelining to increase throughput
  - 5-stage pipeline example

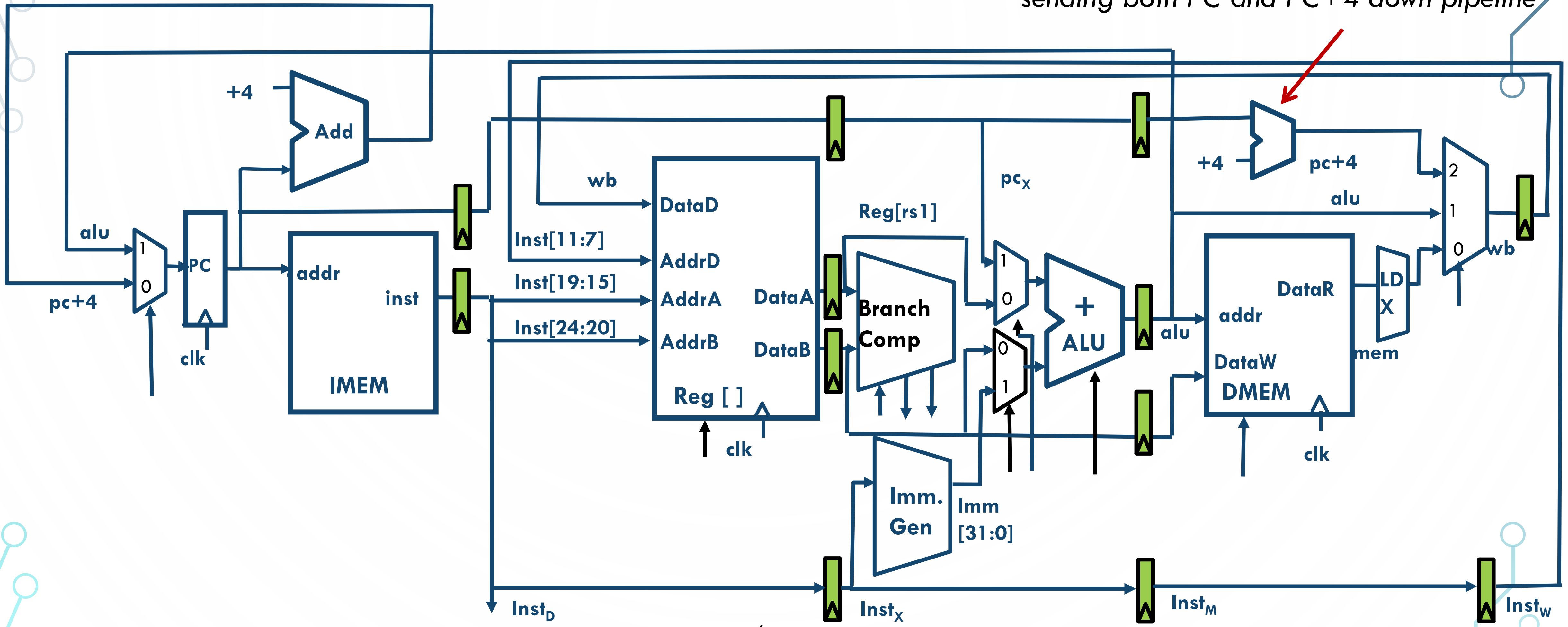
Pipelining

Complete RV32I Datapath with Control



Pipelining RV32I Datapath

Recalculate PC+4 in M stage to avoid sending both PC and PC+4 down pipeline



Instruction Fetch (F)

Instruction Decode/ Register Read (D)

ALU Execute (X)

Memory Access (M)

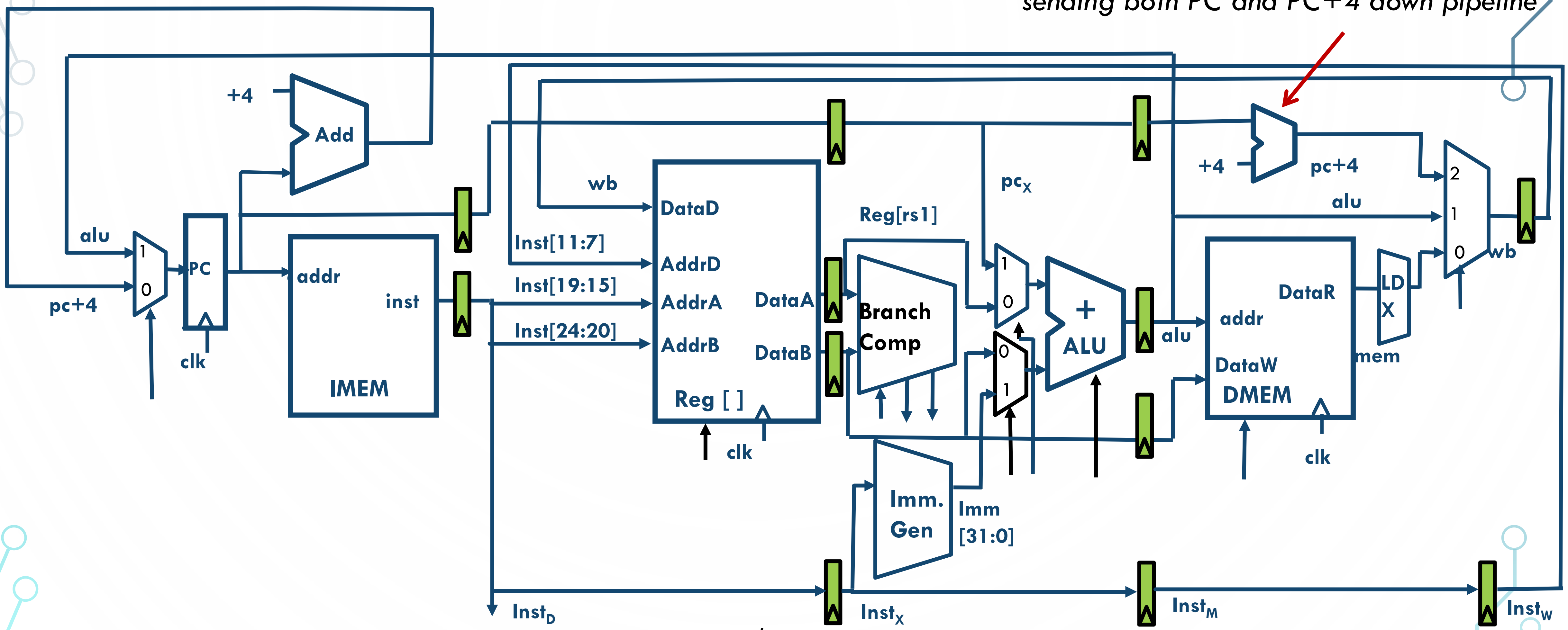
Write Back (W)

instruction along with data

ctly in each stage

Pipelining RV32I Datapath

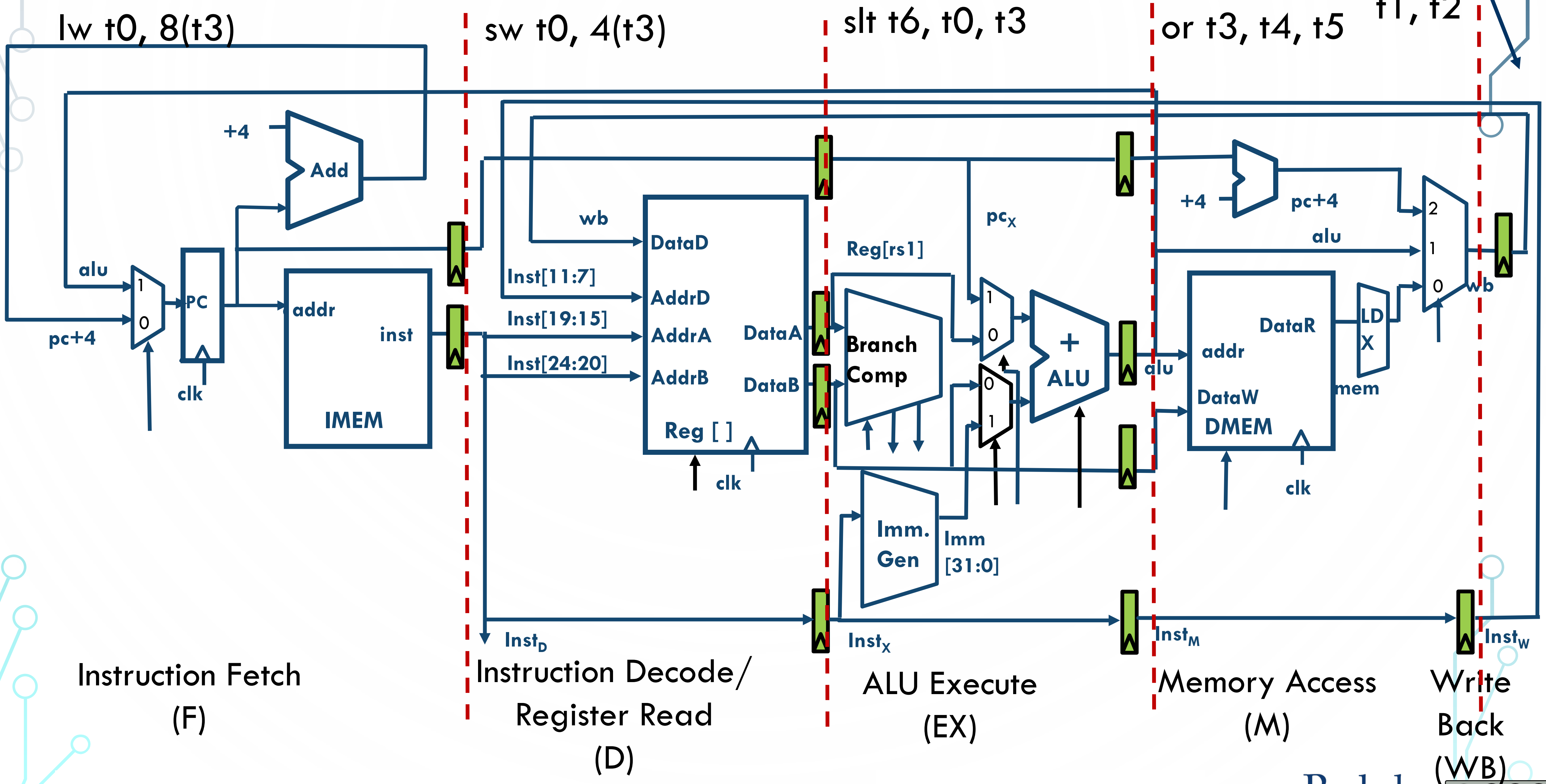
Recalculate PC+4 in M stage to avoid sending both PC and PC+4 down pipeline



Instruction Fetch (F) | Instruction Decode/ Register Read (D) | ALU Execute (X) | Memory Access (M) | Write Back (W)

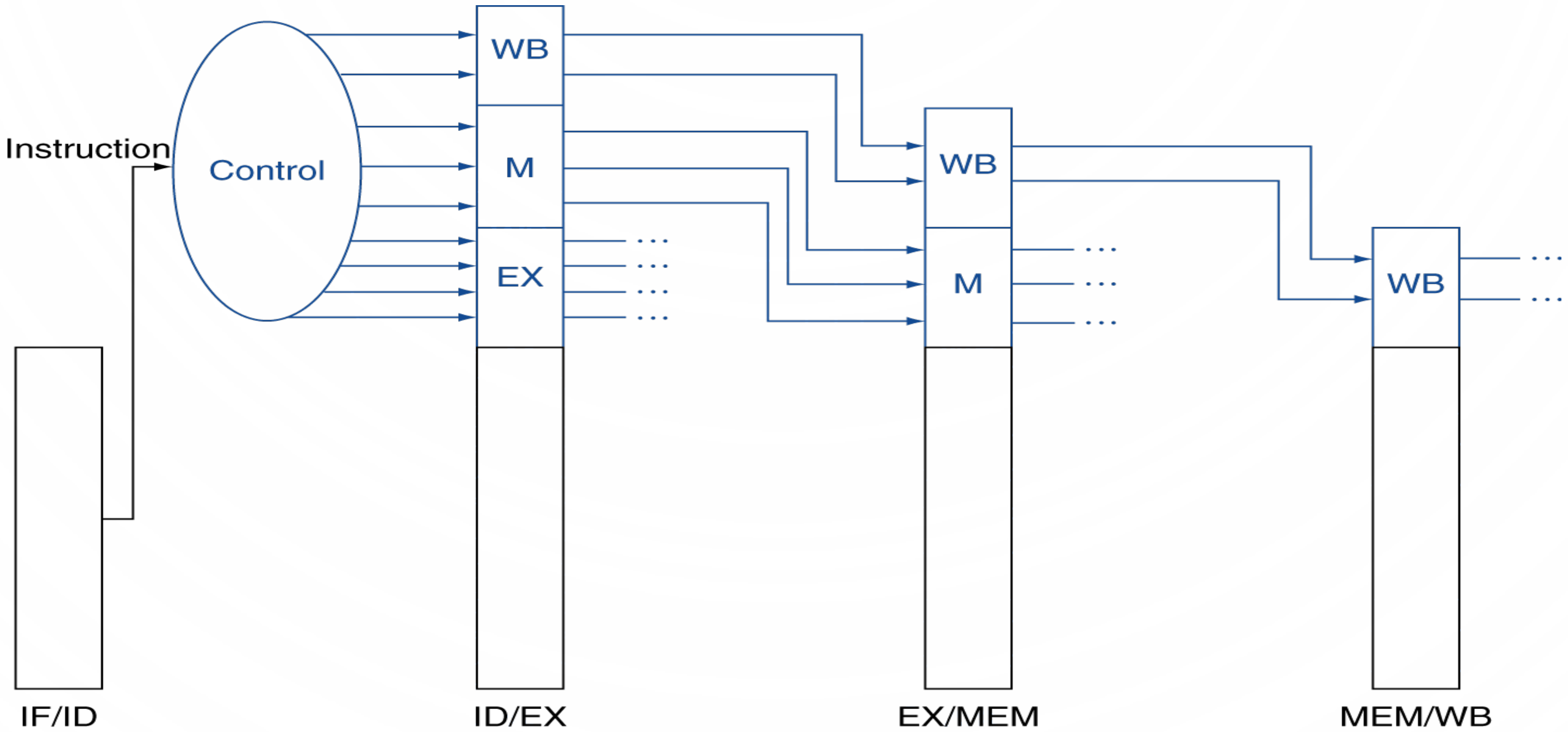
instruction along with data *…ctly in each stage*

Different Instructions in Flight



Pipelined Control

- Control signals derived from instruction
  - As in single-cycle implementation
  - Information is stored in pipeline registers for use by later stages



Pipeline Hazards

Pipelining Hazards

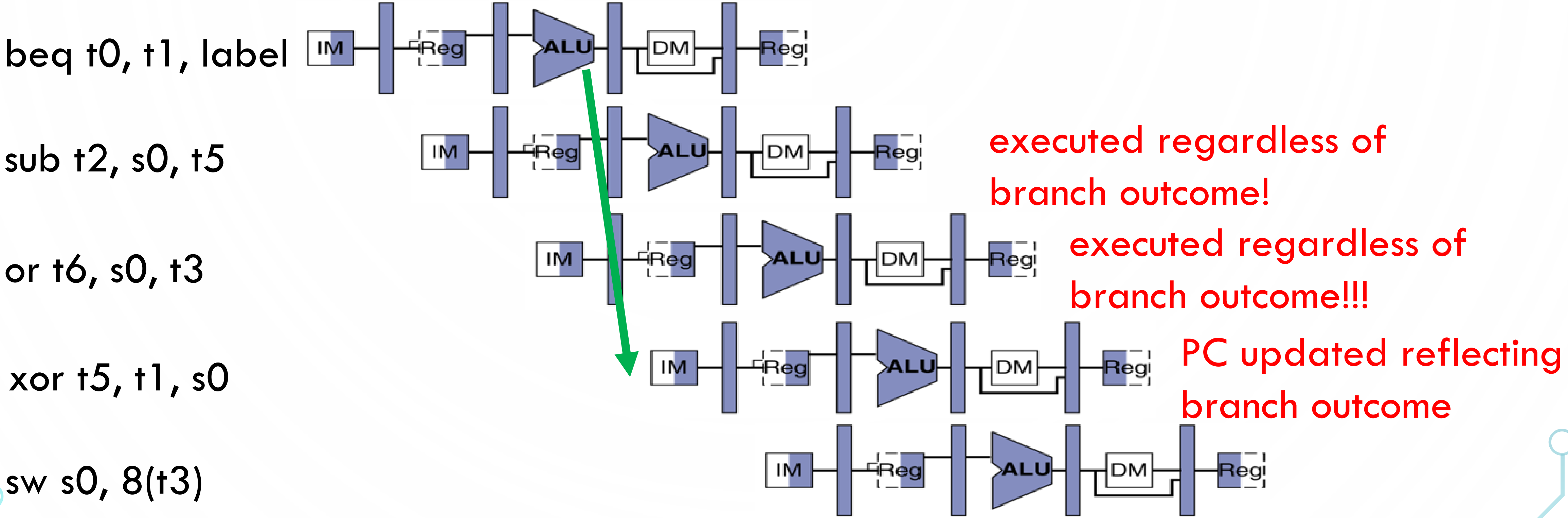
A *hazard* is a situation that prevents starting the next instruction in the next clock cycle

1) *Structural hazard*
   - A required resource is busy
     (e.g. needed in multiple stages)

2) *Data hazard*
   - Data dependency between instructions
   - Need to wait for previous instruction to complete its data read/write

3) *Control hazard*
   - Flow of execution depends on previous instruction

Control Hazards



Observation

- If branch not taken, then instructions fetched sequentially after branch are correct
- If branch or jump taken, then need to flush incorrect instructions from pipeline by converting to NOPs

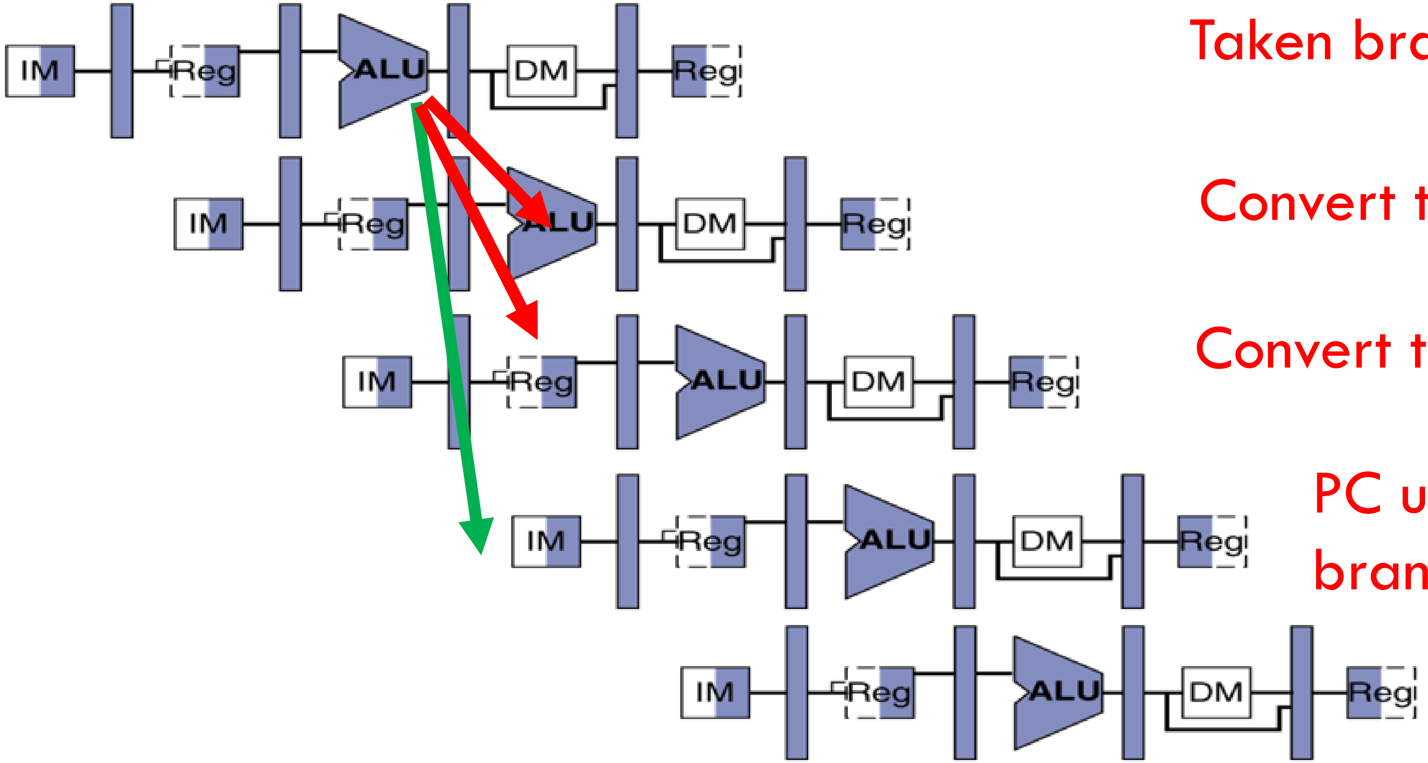
Kill Instructions after Branch if Taken

beq t0, t1, label

sub t2, s0, t5

or t6, s0, t3

label: xxxxxx



Taken branch

Convert to NOP

Convert to NOP

PC updated reflecting branch outcome

Reducing Branch Penalties

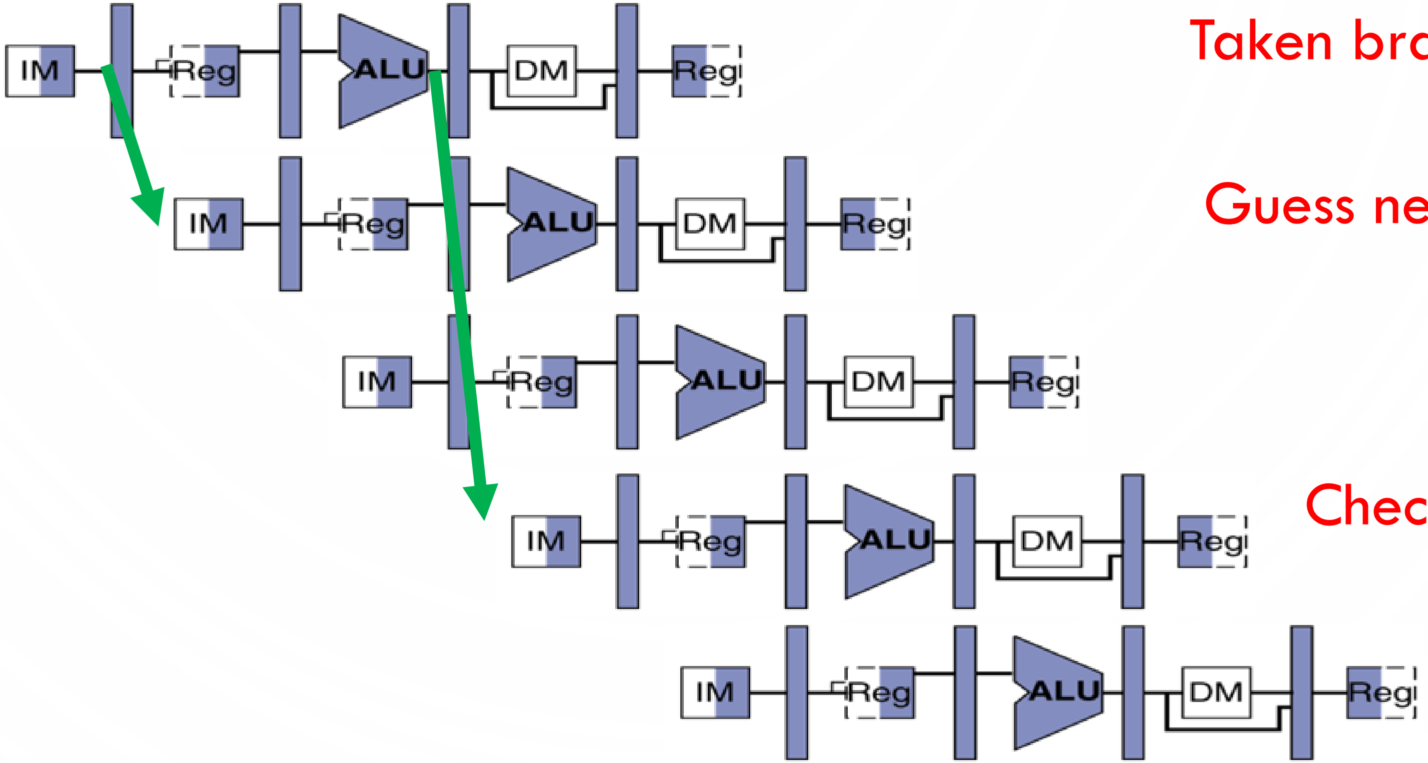
- Every taken branch in simple pipeline costs 2 ‘dead’ cycles
- To improve performance, use “branch prediction” to guess which way branch will go earlier in pipeline
- Only flush pipeline if branch prediction was incorrect

Branch Prediction

beq t0, t1, label

label: …..

…..



Taken branch

Guess next PC!

Check guess correct

Quiz: Hazards

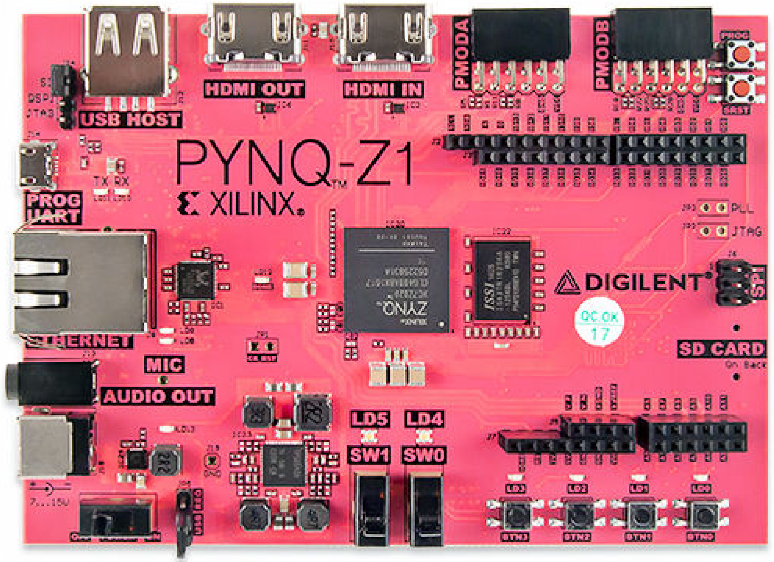
- How many data hazards exist in the following sequence (assuming a 5-stage pipeline)?

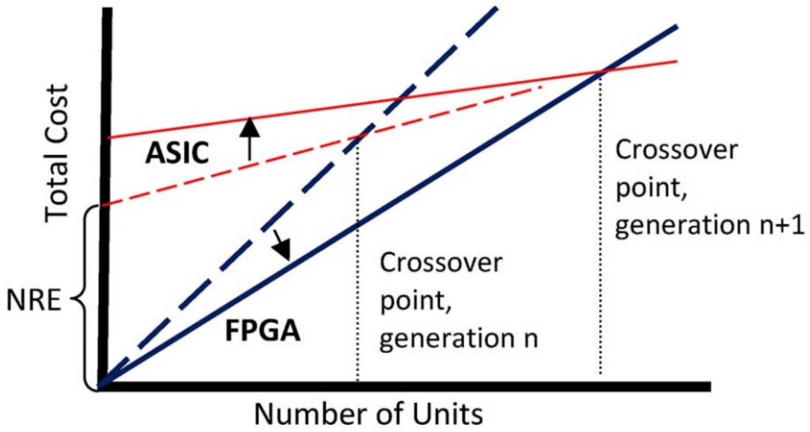
```
add x3, x1, x2
or x5, x3, x4
add x2, x5, x3
lw x6, x2, 12
sw x1, x6, 36
```

FPGAs: Overview

Field Programmable Gate Arrays (FPGAs)

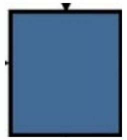
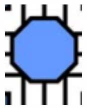
- An integrated circuit designed to be configured by a customer or a designer after manufacturing, i.e., field programmable.
- The FPGA configuration is generally specified using a hardware description language, similar to that used for ASICs.
- Two dominant FPGA makers:
  - Xilinx
  - Altera (now Intel)

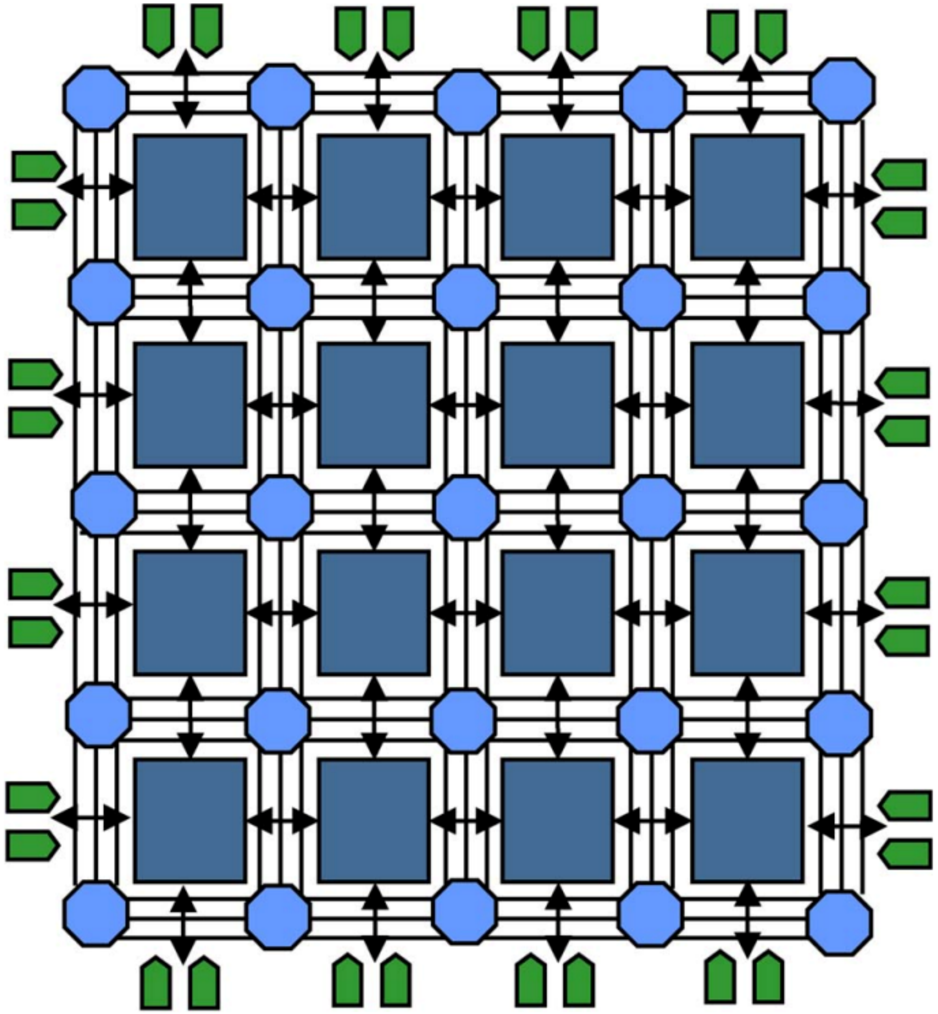




Trimberger, IEEE Micro'2015

FPGA Overview

- Basic idea:
  - Two dimensional array of logic blocks and flip-flops with means for the user to configure:
    - The function of each block
    - The interconnection between blocks
- Configurable Logic Blocks (CLBs)
  - FPGA's Functional Units 
- Reconfigurable Interconnect
  - Connecting CLBs together 



State-of-the-art Xilinx FPGAs

45nm	28nm	20nm	16nm
SPARTAN 6	VIRTEX 7, KINTEX 7, ARTIX 7, SPARTAN 7	VIRTEX UltraSCALE, KINTEX UltraSCALE	VIRTEX UltraSCALE+, KINTEX UltraSCALE+

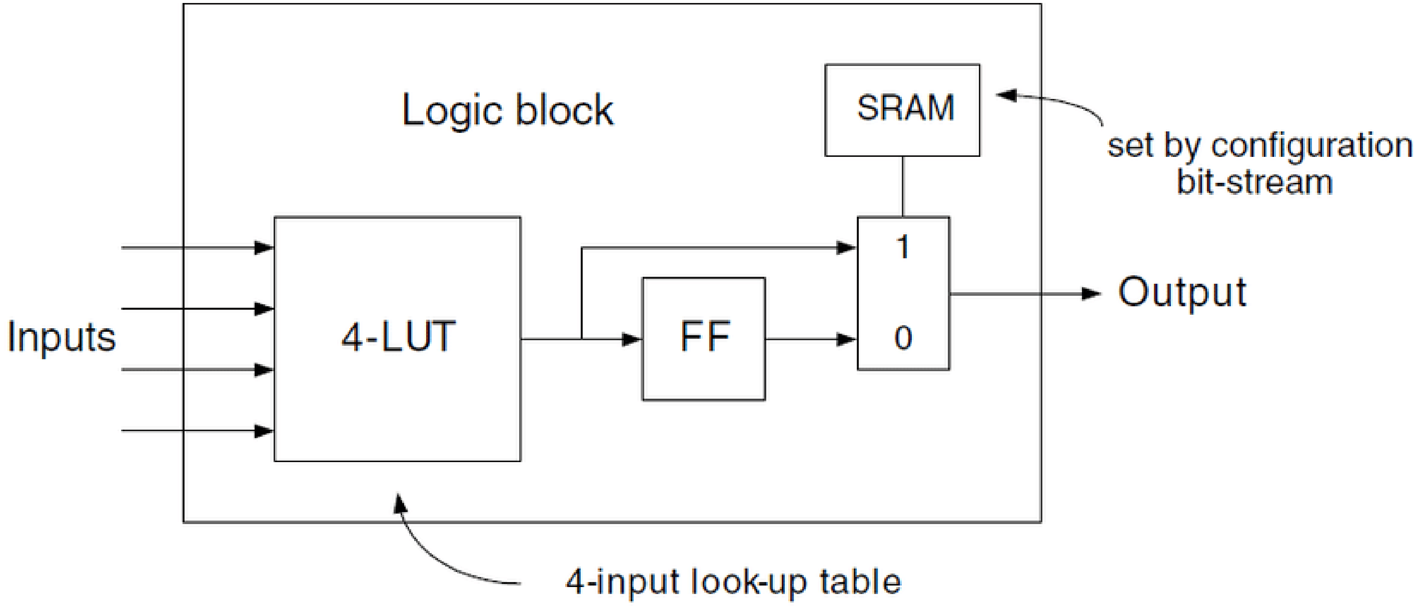
Virtex Ultra-scale

Device Name	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P	VU27P	VU29P	VU31P	VU33P	VU35P	VU37P
System Logic Cells (K)	862	1,314	1,724	2,586	2,835	3,780	2,835	3,780	962	962	1,907	2,852
CLB Flip-Flops (K)	788	1,201	1,576	2,364	2,592	3,456	2,592	3,456	879	879	1,743	2,607
CLB LUTs (K)	394	601	788	1,182	1,296	1,728	1,296	1,728	440	440	872	1,304
Max. Dist. RAM (Mb)	12.0	18.3	24.1	36.1	36.2	48.3	36.2	48.3	12.5	12.5	24.6	36.7
Total Block RAM (Mb)	25.3	36.0	50.6	75.9	70.9	94.5	70.9	94.5	23.6	23.6	47.3	70.9
UltraRAM (Mb)	90.0	132.2	180.0	270.0	270.0	360.0	270.0	360.0	90.0	90.0	180.0	270.0
HBM DRAM (GB)	–	–	–	–	–	–	–	–	4	8	8	8
HBM AXI Interfaces	–	–	–	–	–	–	–	–	32	32	32	32
Clock Mgmt Tiles (CMTs)	10	20	20	30	12	16	16	16	4	4	8	12
DSP Slices	2,280	3,474	4,560	6,840	9,216	12,288	9,216	12,288	2,880	2,880	5,952	9,024
Peak INT8 DSP (TOP/s)	7.1	10.8	14.2	21.3	28.7	38.3	28.7	38.3	8.9	8.9	18.6	28.1
PCIe® Gen3 x16	2	4	4	6	3	4	1	1	0	0	1	2
PCIe Gen3 x16/Gen4 x8 / CCIX(1)	–	–	–	–	–	–	–	–	4	4	4	4
150G Interlaken	3	4	6	9	6	8	6	8	0	0	2	4
100G Ethernet w/ KR4 RS-FEC	3	4	6	9	9	12	11	15	2	2	5	8
Max. Single-Ended HP I/Os	520	832	832	832	624	832	520	676	208	208	416	624
GTY 32.75Gb/s Transceivers	40	80	80	120	96	128	32	32	32	32	64	96
GTM 58Gb/s PAM4 Transceivers							32	48				
100G / 50G KP4 FEC							16 / 32	24 / 48				
Extended(2)	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3	-1 -2 -2L -3
Industrial	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	–	–	–	–

FPGA: CLBs

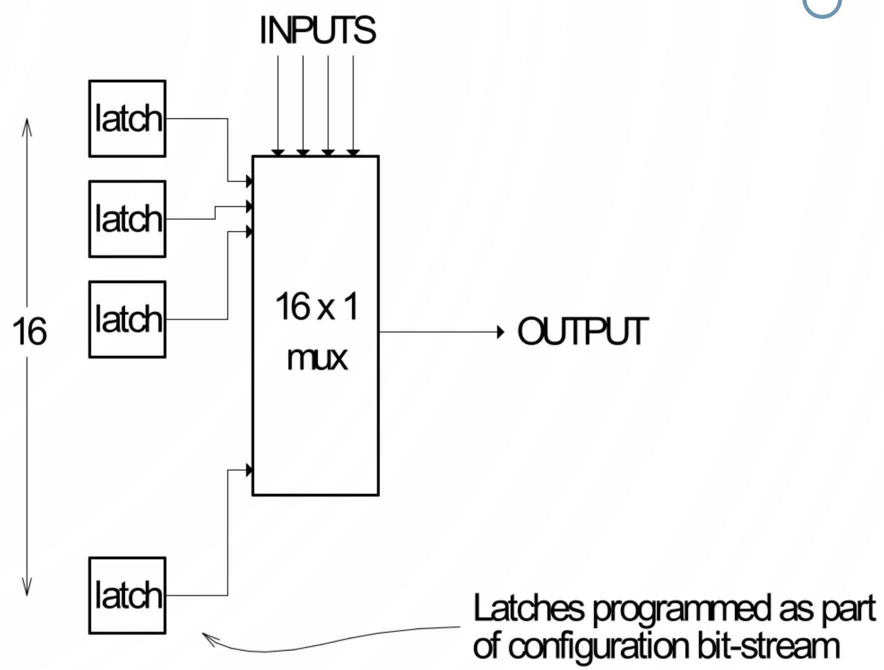
Configurable Logic Blocks (CLBs)

- Basic FPGA functional unit
- Implements both combinational and sequential logic
- Includes:
  - Look-up table
  - Register (Flip-Flop)
  - Multiplexers



Look-Up Table Implementation

- Implement truth table in small memories
  - SRAM/Latch arrays
  - “Latch” is actually a flip-flop
- n-bit LUT is implemented as a $2^n * 1$ memory:
  - inputs choose one of 2^n memory locations.
  - memory locations (latches) are normally loaded with values from user’s configuration bit stream.
  - Inputs to mux control are the CLB inputs.
- Result is a general purpose “logic gate”.
  - n-LUT can implement any function of n inputs!



Look-Up Table Implementation

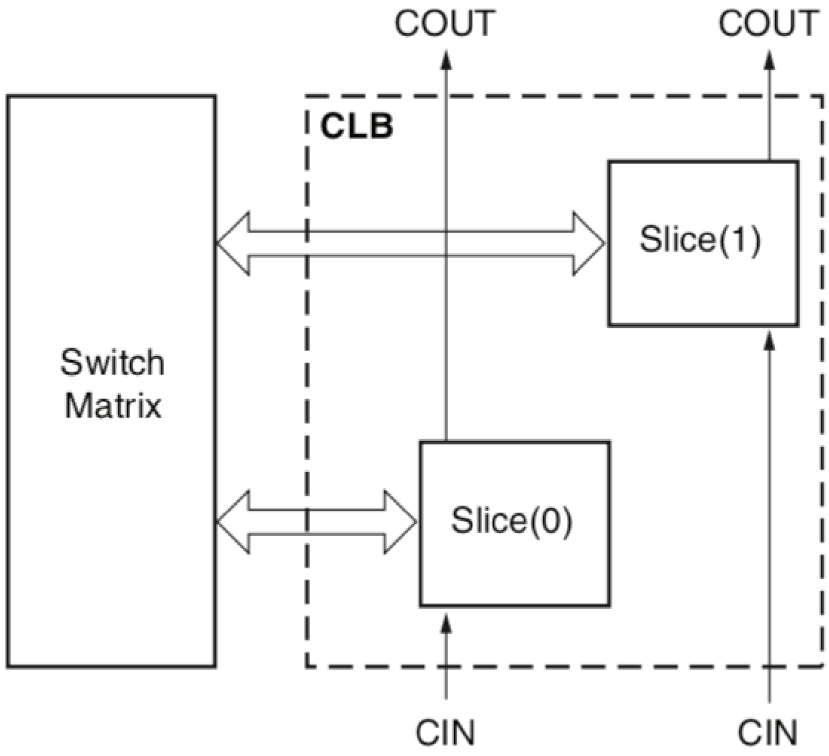
- An n-LUT is a direct implementation of a function truth-table.
- Each location holds the value of the function corresponding to one input combination.
- LUT size grows exponentially with # of inputs.
  - 64 input LUT requires $2^{64} = 1.84 * 10^{19}$ bits storage.
  - 4-input ~ 8-input LUT

Example: 4-LUT

INPUTS		
0000	F(0,0,0,0)	store in 1st latch
0001	F(0,0,0,1)	store in 2nd latch
0010	F(0,0,1,0)	
0011	F(0,0,1,1)	
0011		
0100	•	
0101	•	
0110	•	
0111		
1000		
1001		
1010		
1011		
1100		
1101		
1110		
1111		

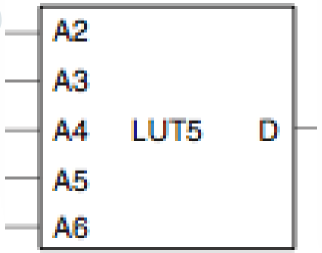
Slices

- Each CLB contains two slices.
- LUTs and registers are split across slices.
  - 4 LUTS and 8 FFs in 7-series.
- Two types of slices:
  - SLICEM: Full slice
    - LUT can be used for logic *and* memory/shift registers.
    - Has wide multiplexers and carry chain
  - SLICEL: logic and arithmetic only
    - LUT can only be used for logic (no memory)
    - Has wide multiplexers and carry chain

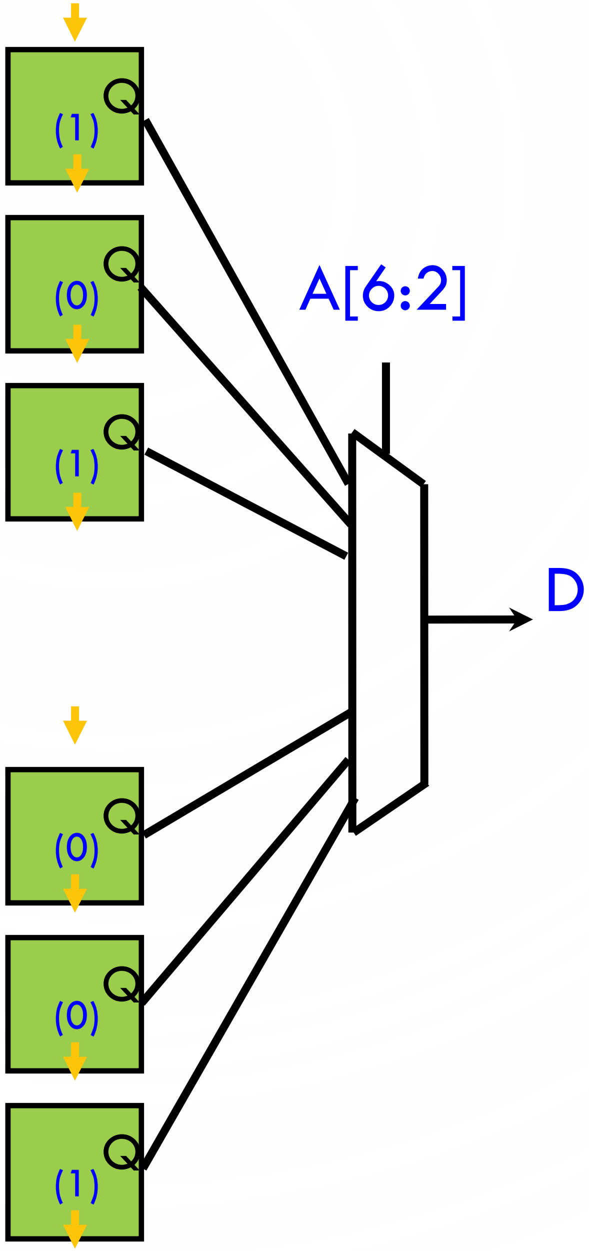


Constructing a SLICE

- 5-Input Look-Up Table



A[6:2]	D
00000	1
00001	0
00010	1
11101	0
11110	0
11111	1



Computes any 5-input logic function.

Timing is independent of function.

Latches set during configuration.

Constructing a SLICE

- 6-input LUT

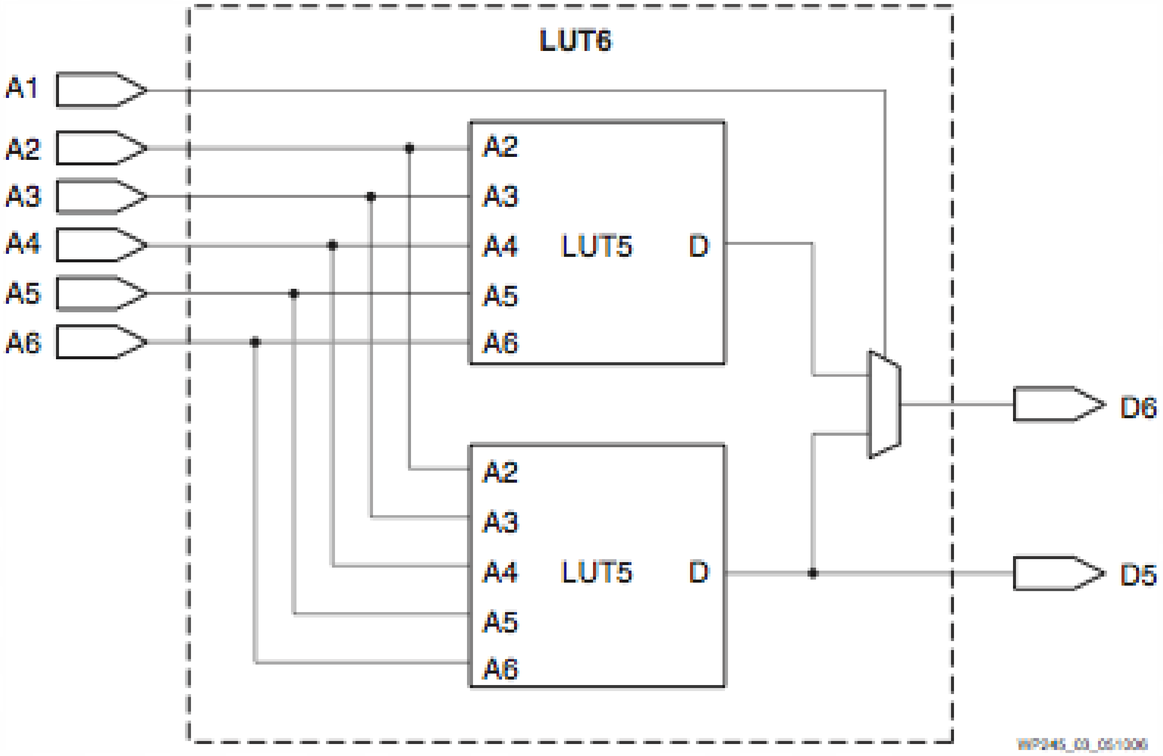


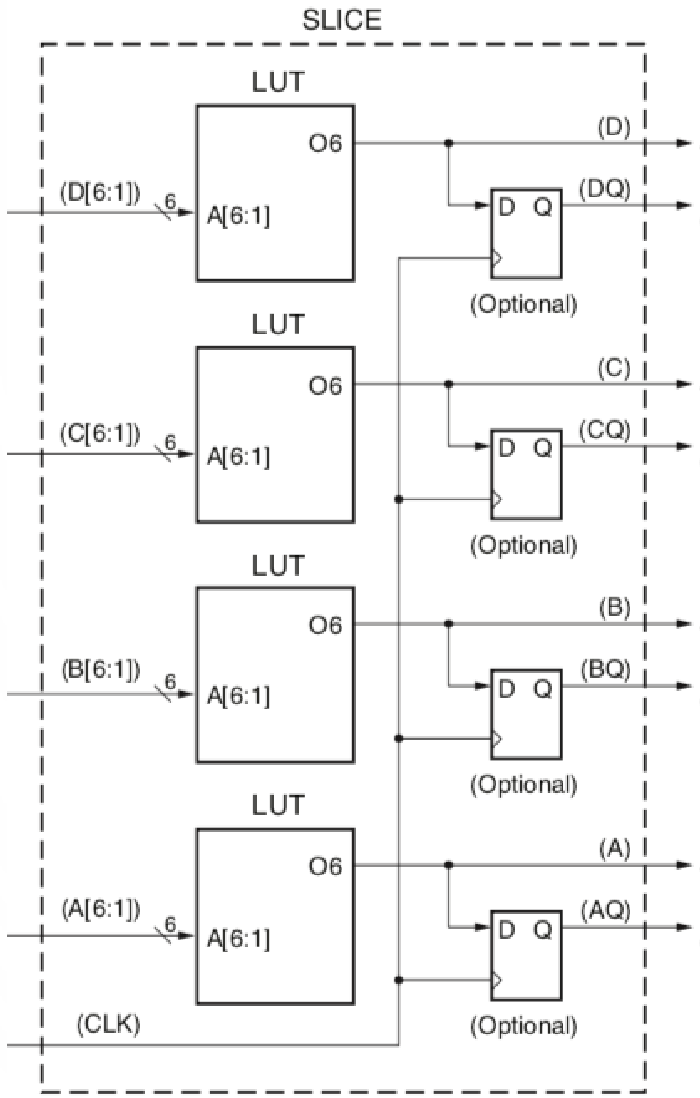
Figure 3: **Block Diagram of a Virtex-5 6-Input LUT**

May be used
as one 6-input LUT
(D6 out)
...

... or as two
5-input LUTS
(D6 and D5)

Combinational
logic
(post configuration)

The Simplest View of A Slice



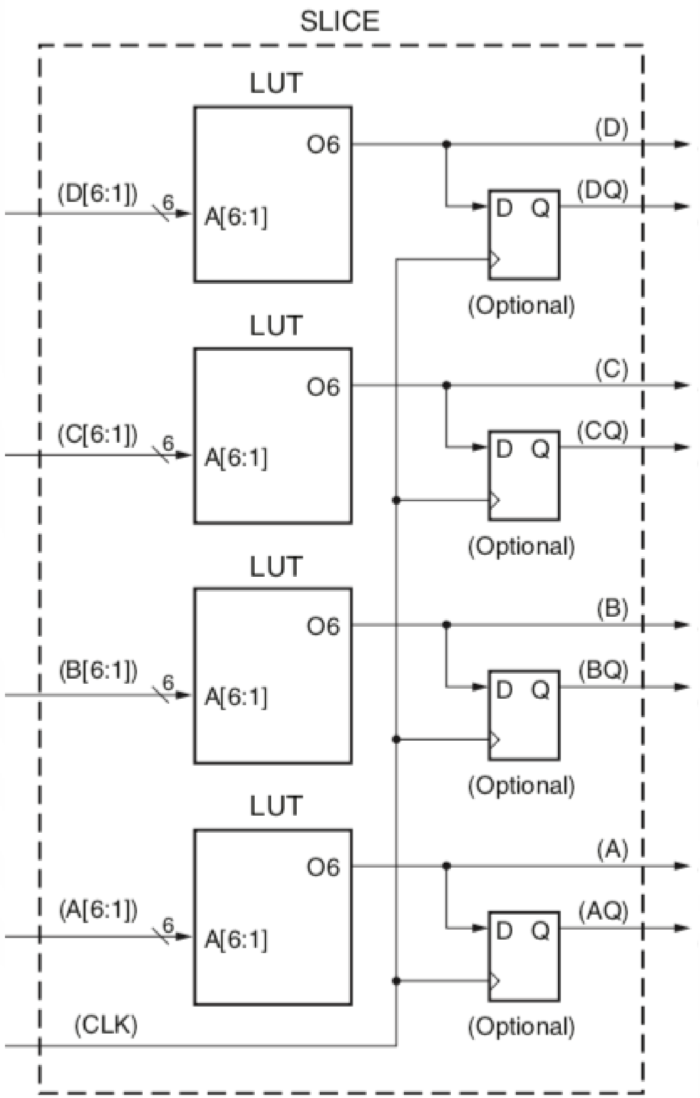
Four 6-LUTs

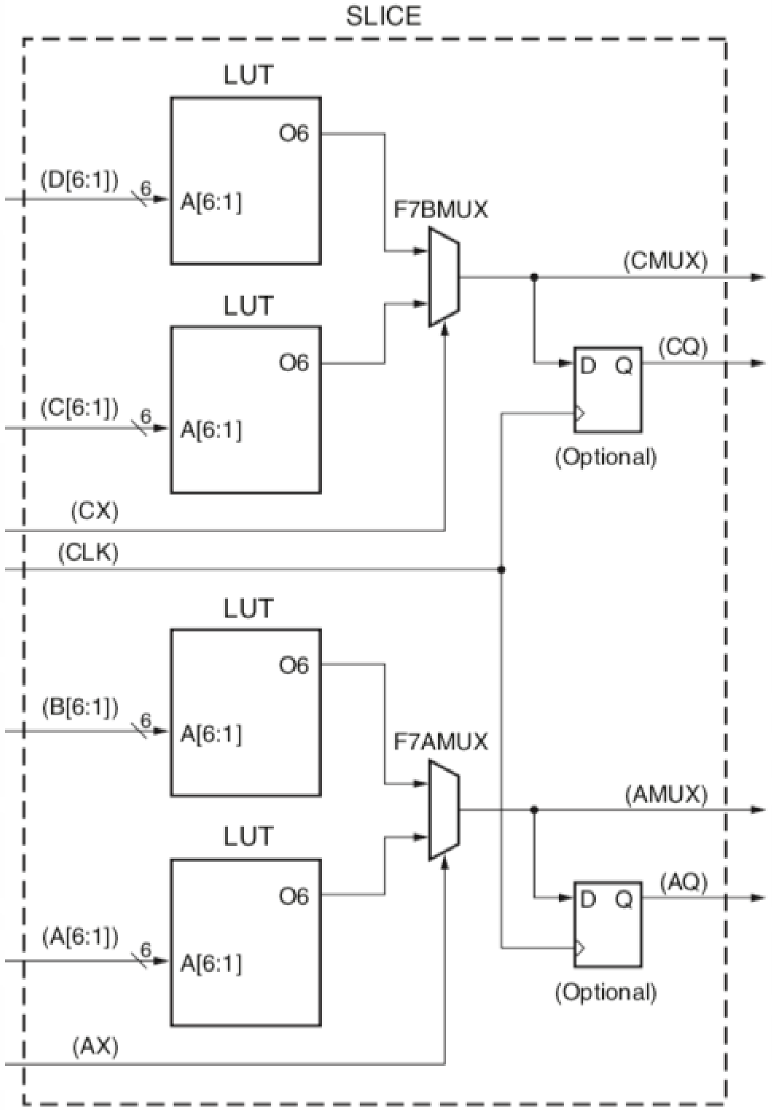
Four Flip-Flops

Switching fabric may see combinational and registered outputs.

An actual Virtex slice adds many small features to this simplified diagram.
We show them one by one ...

How about 7-input LUT in a slice?





Two 7-LUTs

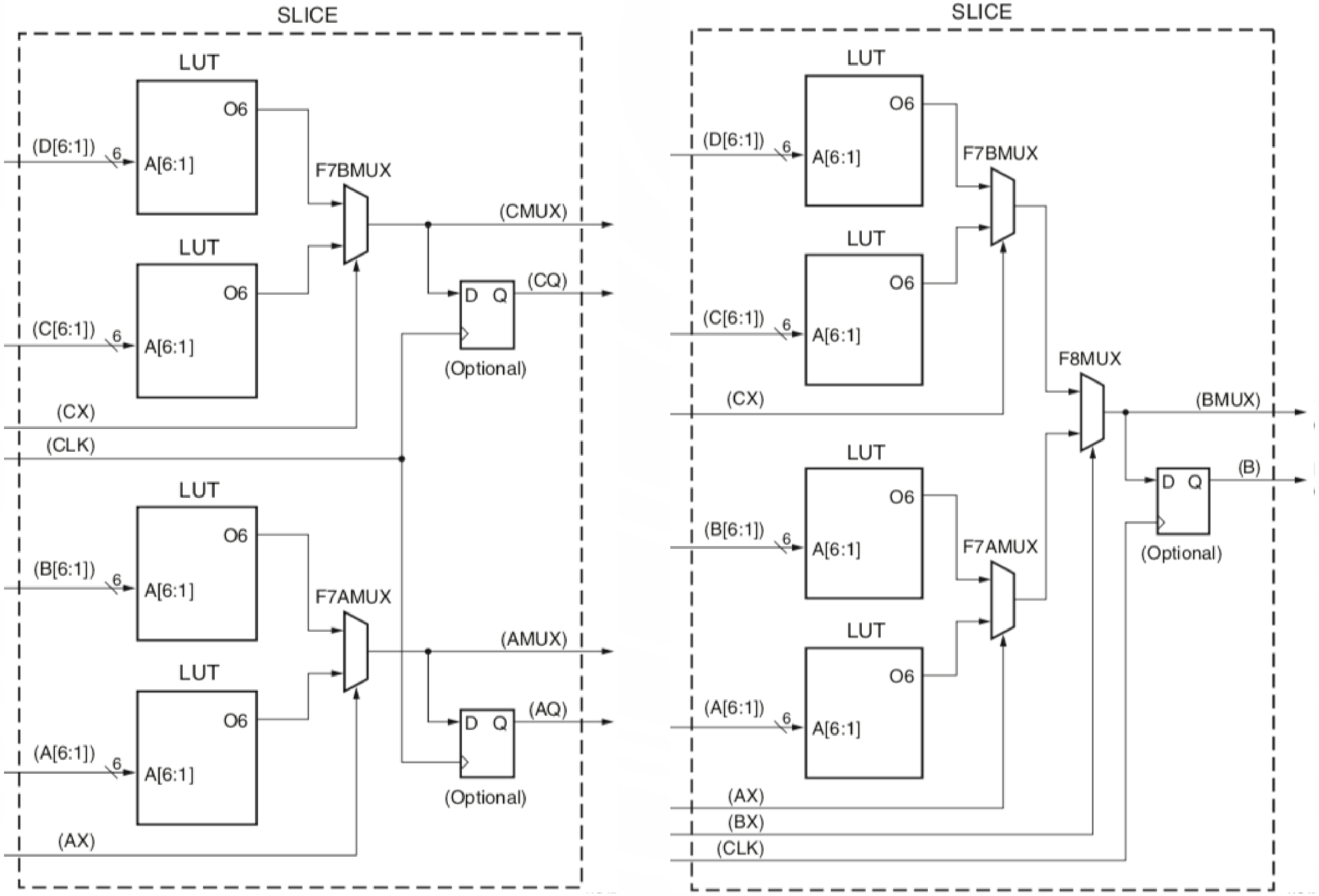
Extra MUX

(F7AMUX, F7BMUX)

Extra inputs

(AX and CX)

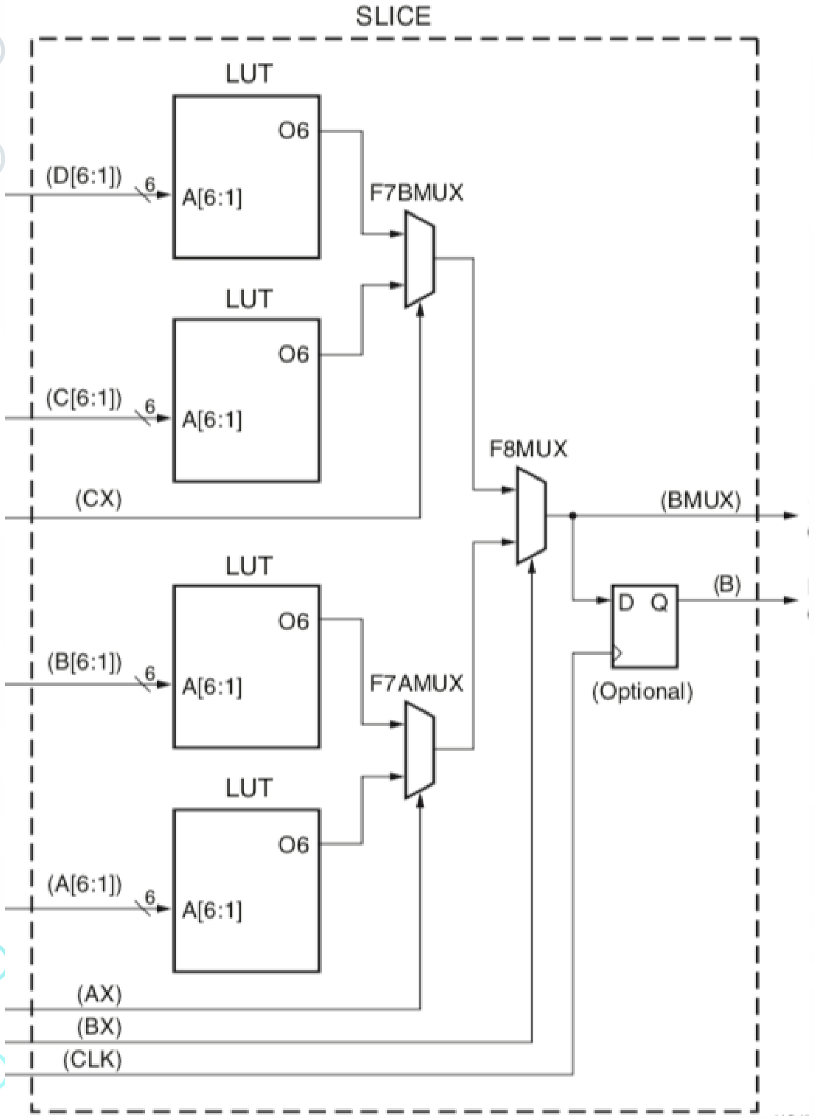
How about 8-input LUT in a slice?

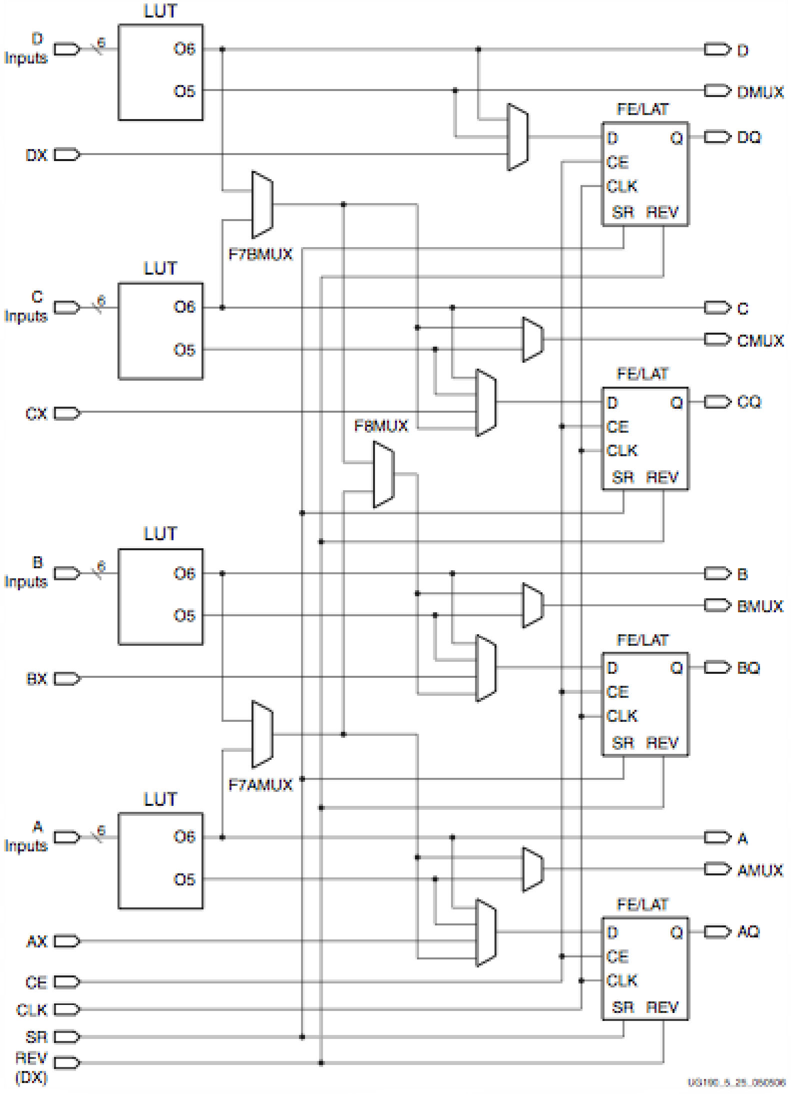


Third MUX (F8MUX)

Third input (BX)

Extra MUXes to choose LUT outputs





From eight 5-LUTs

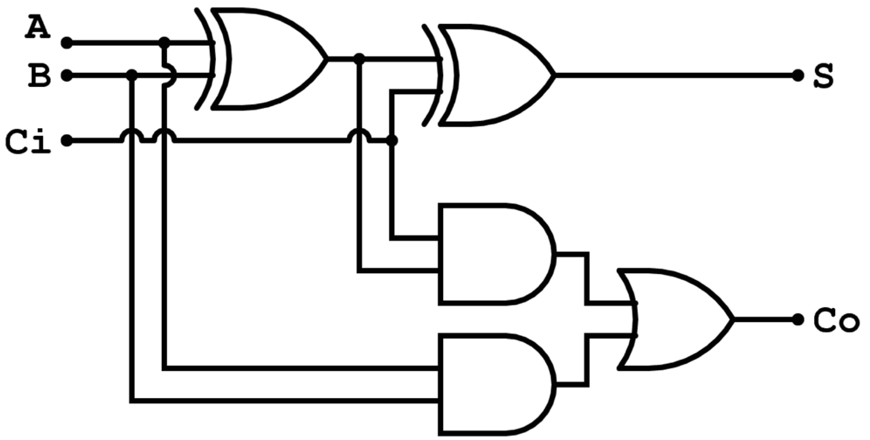
… to one 8-LUT.

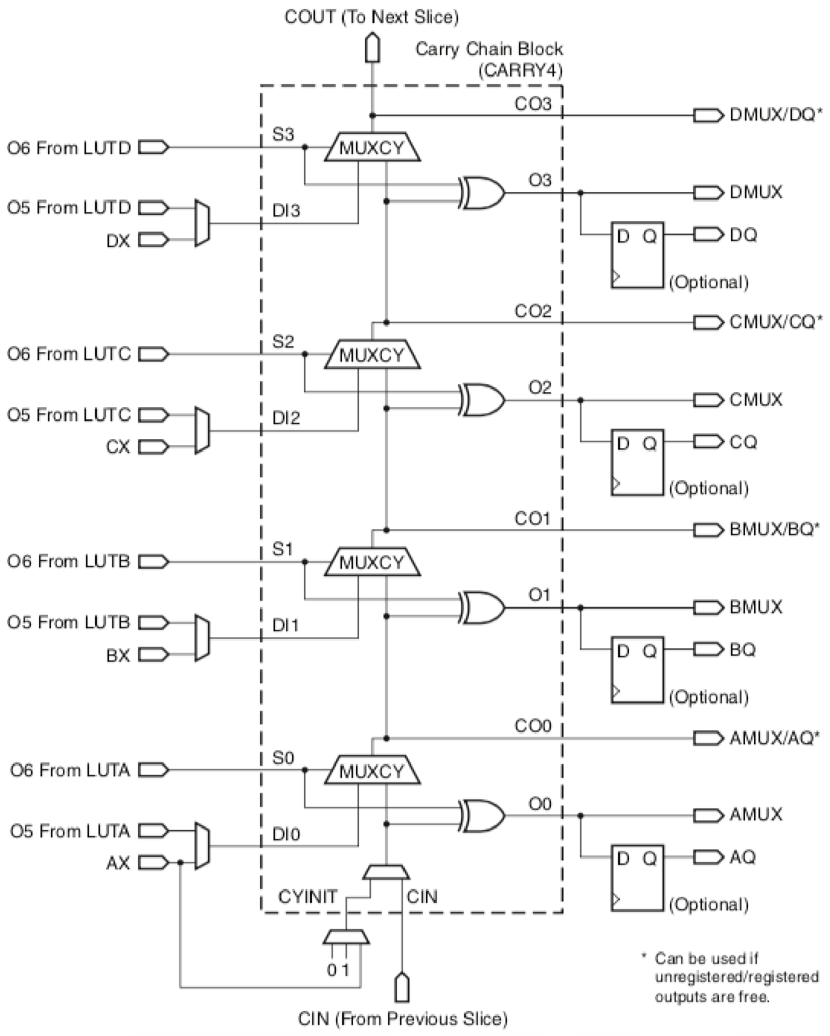
Combinational

or registered outs.

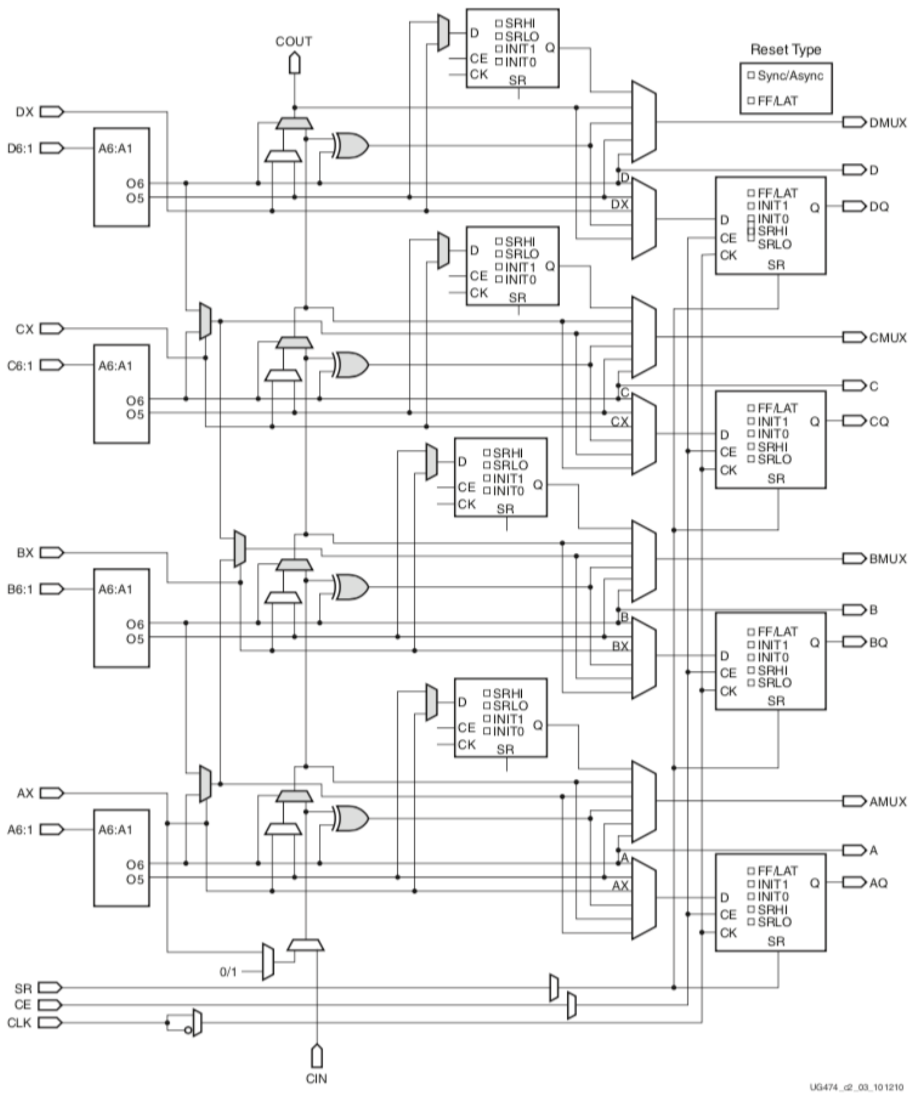
Extra Carry Chain

We can map ripple-carry addition onto carry-chain block.





Putting it all together … a SLICEL.



The previous slides explain all SLICEL features.

About 50% of the are SLICELs.

The other slices are SLICEMs, and have extra features.

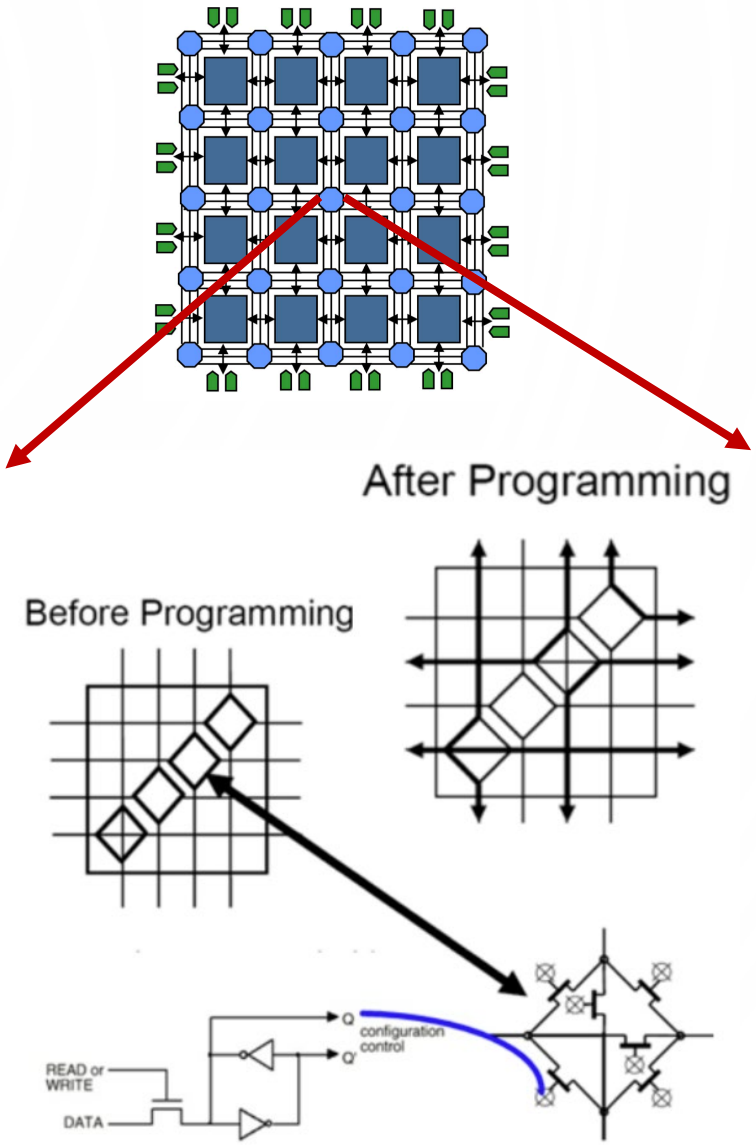
Administrivia

- Homework 4 is due next Monday
  - No new homework this week
  - Homework 5 will be posted next week, due after the midterm
- Lab 5 this week
  - No lab next week
  - Lab 6 (last) after the midterm
- Midterm 1 on October 7, 7-8:30pm
  - You will be assigned a classroom

FPGA Interconnect

Configurable Interconnect

- Between rows and columns of CLBs are wiring channels.
- These are programable. Each wire can be connected in many ways.
- Switch Box:
  - Each interconnection has a transistor switch.
  - Each switch is controlled by 1-bit configuration register.



FPGA Features: BRAMs, DSP, AI

Diverse Resources on FPGA

Colors represent different types of resources:

Logic

Block RAM

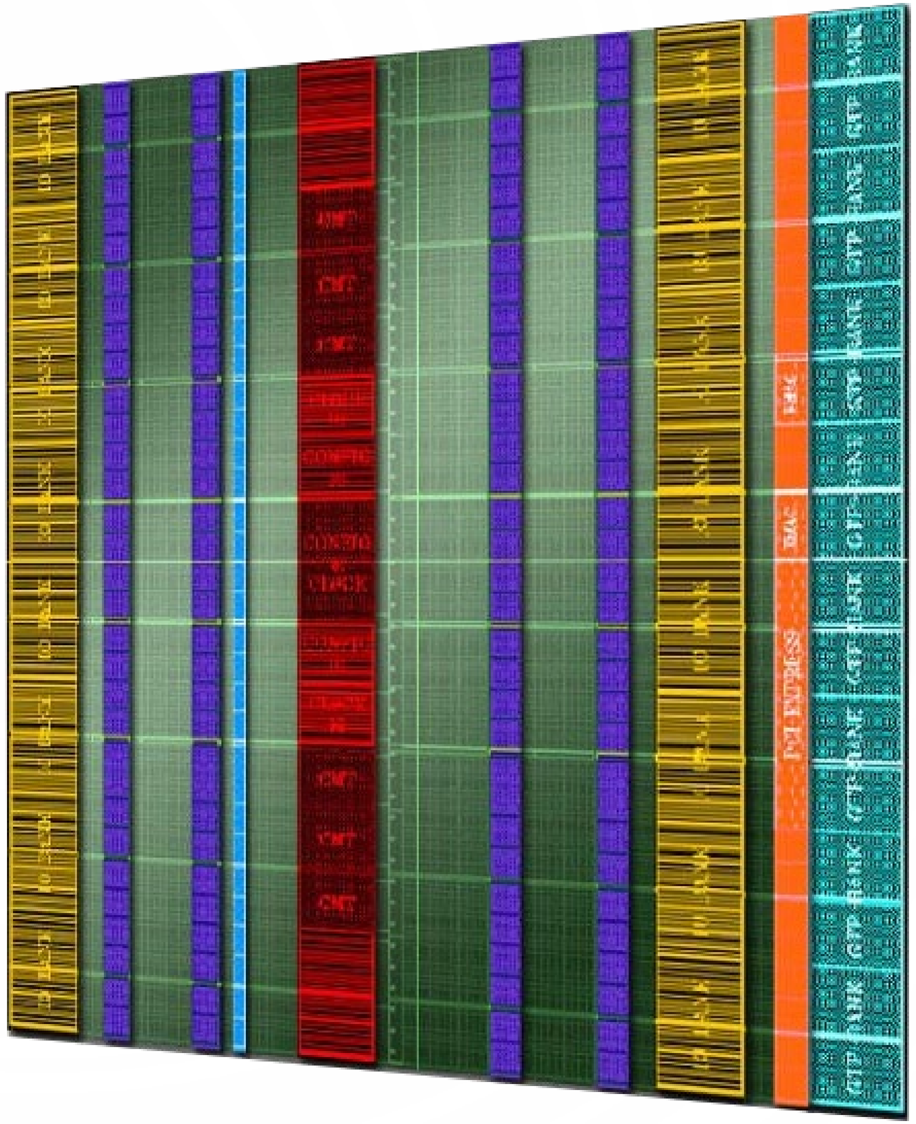
DSPs

Clocking

I/O

Serial I/O + PCI

A routing fabric runs throughout the chip to wire everything together.



Virtex-5 Die Photo [Xilinx]

Block RAM

- Block Random Access Memory
- Used for storing large amounts of data:
  - 18Kb or 36Kb
  - Configurable bitwidth
  - 2 read and write ports
- More recently
  - UltraRAM in UltraScale+ devices

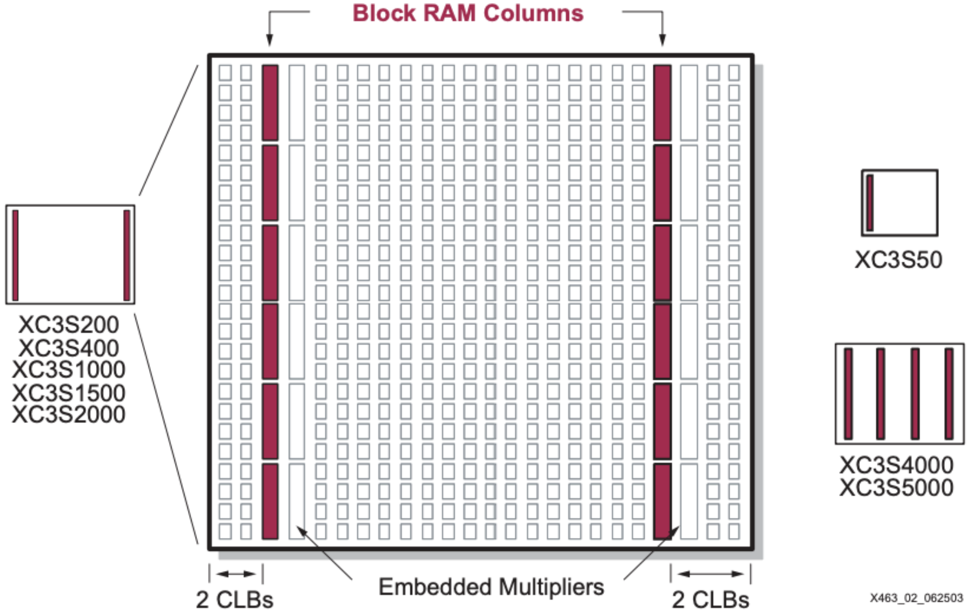
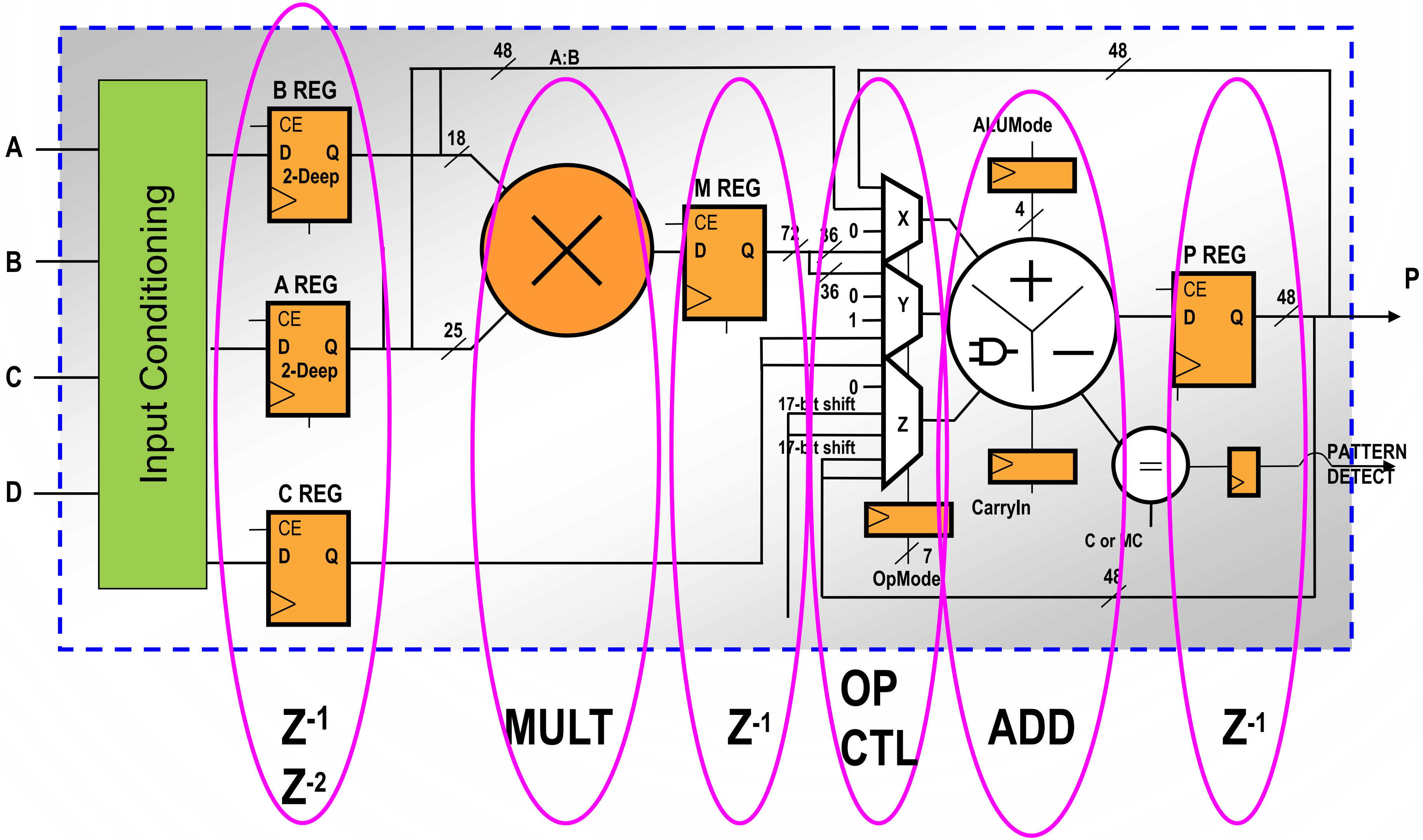


Figure 2: **Block RAMs Arranged in Columns with Detailed Floorplan of XC3S200**

Xilinx Datasheet

DSP Slice

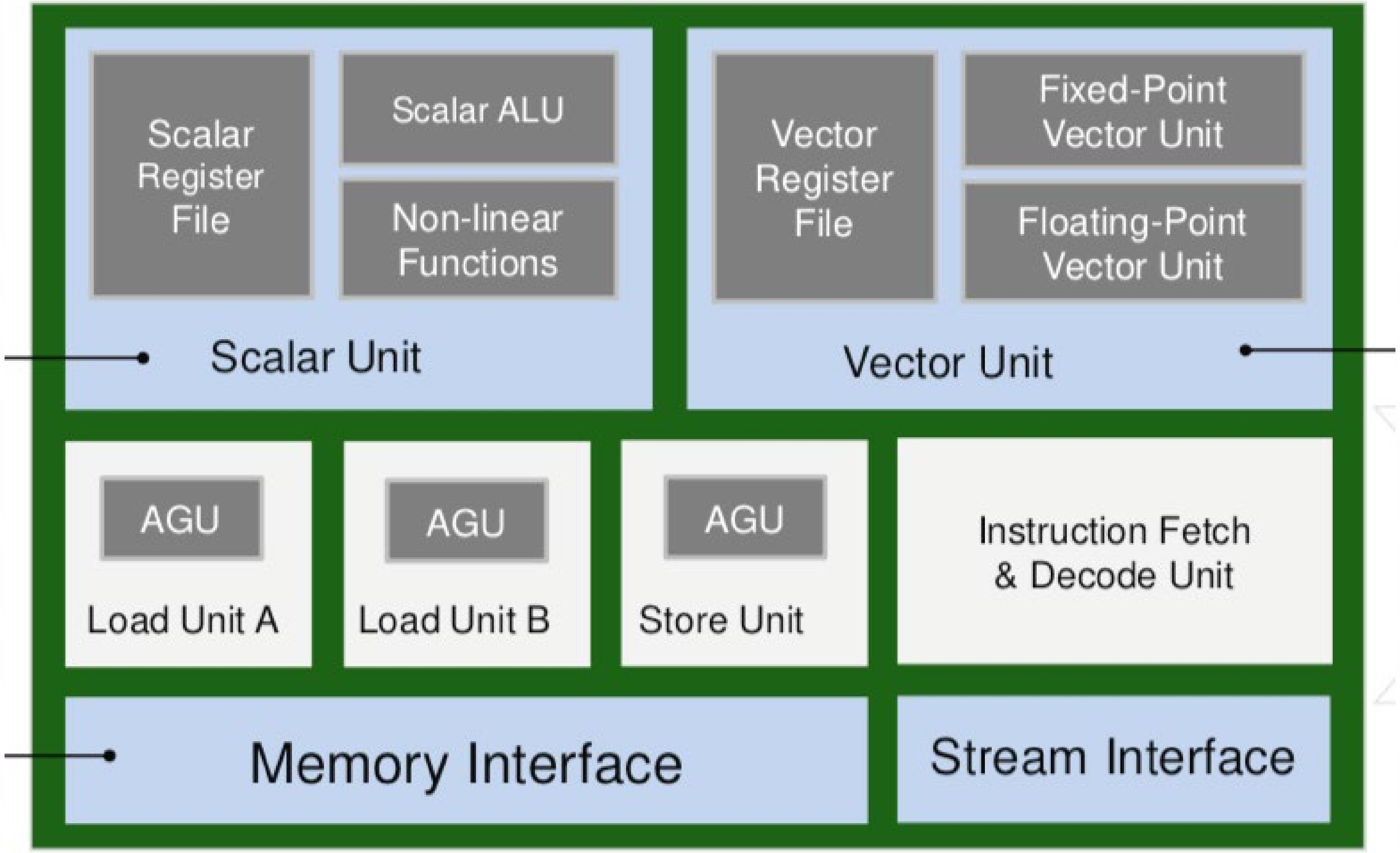


Efficient implementation of multiply, add, bit-wise logical.

Xilinx Resource

AI Engine

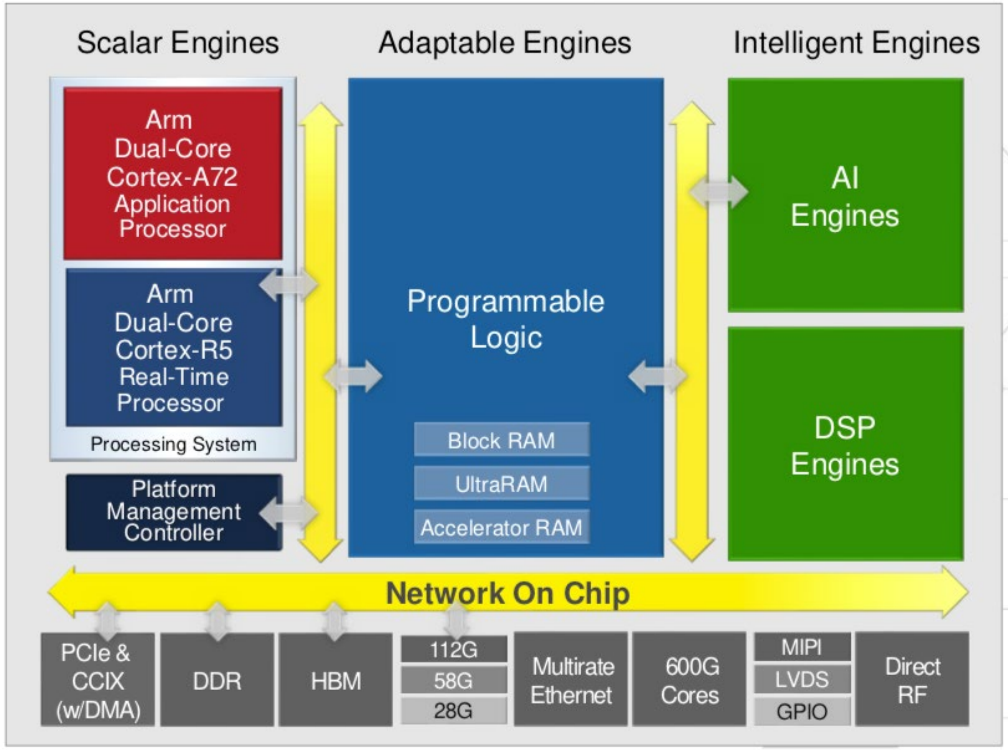
- Versal AI Core

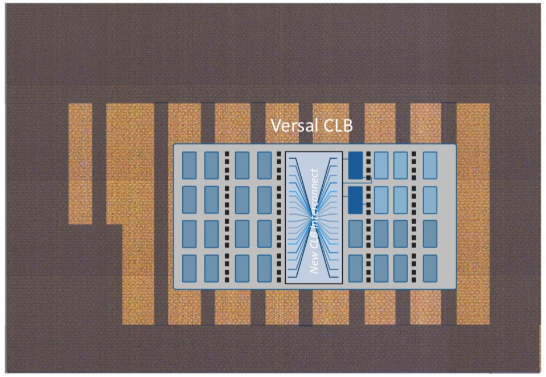


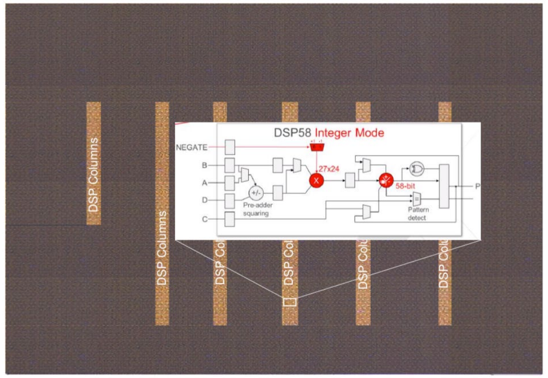
Xilinx HotChips'2019

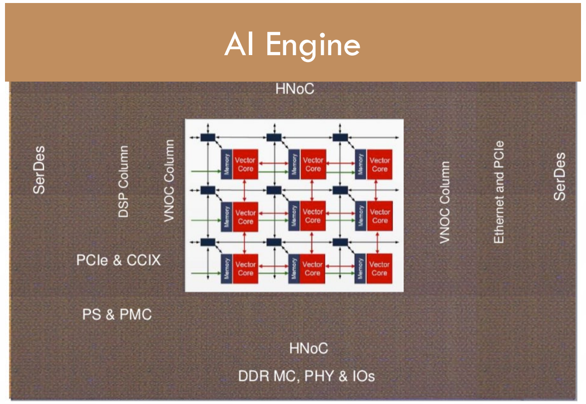
State-of-the-art Xilinx FPGA Platform

- Versal (ACAP: Adaptive Compute Acceleration Platform)









Xilinx HotChips'2019

Summary

- Pipelining increases throughput
  - Structural, control and data hazards exist
- FPGAs are widely used for hardware prototyping and accelerating key applications.
- Core FPGA building blocks:
  - Configurable Logic Blocks (CLBs)
    - Slices
      - Look-Up Tables
      - Flip-Flops
      - Carry chain
  - Configurable Interconnect
    - Switch boxes
- Modern FPGA Designs:
  - BRAMs, DSPs, and AI Engines