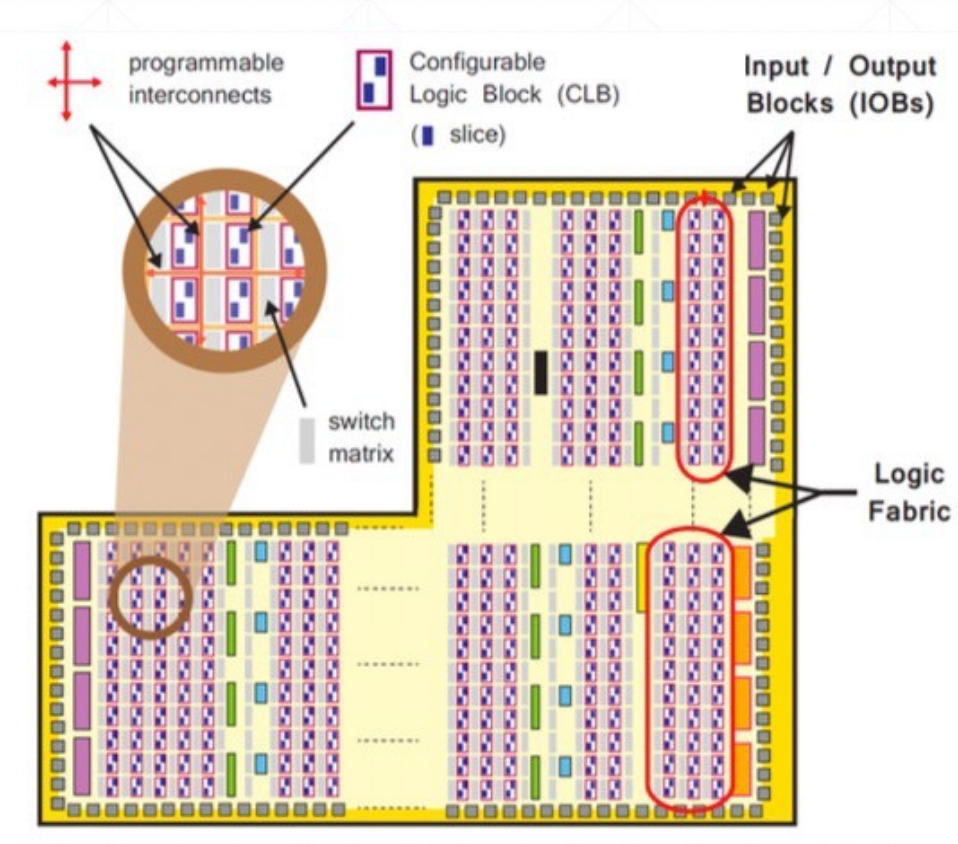
FPGA Resources

Lecture 7

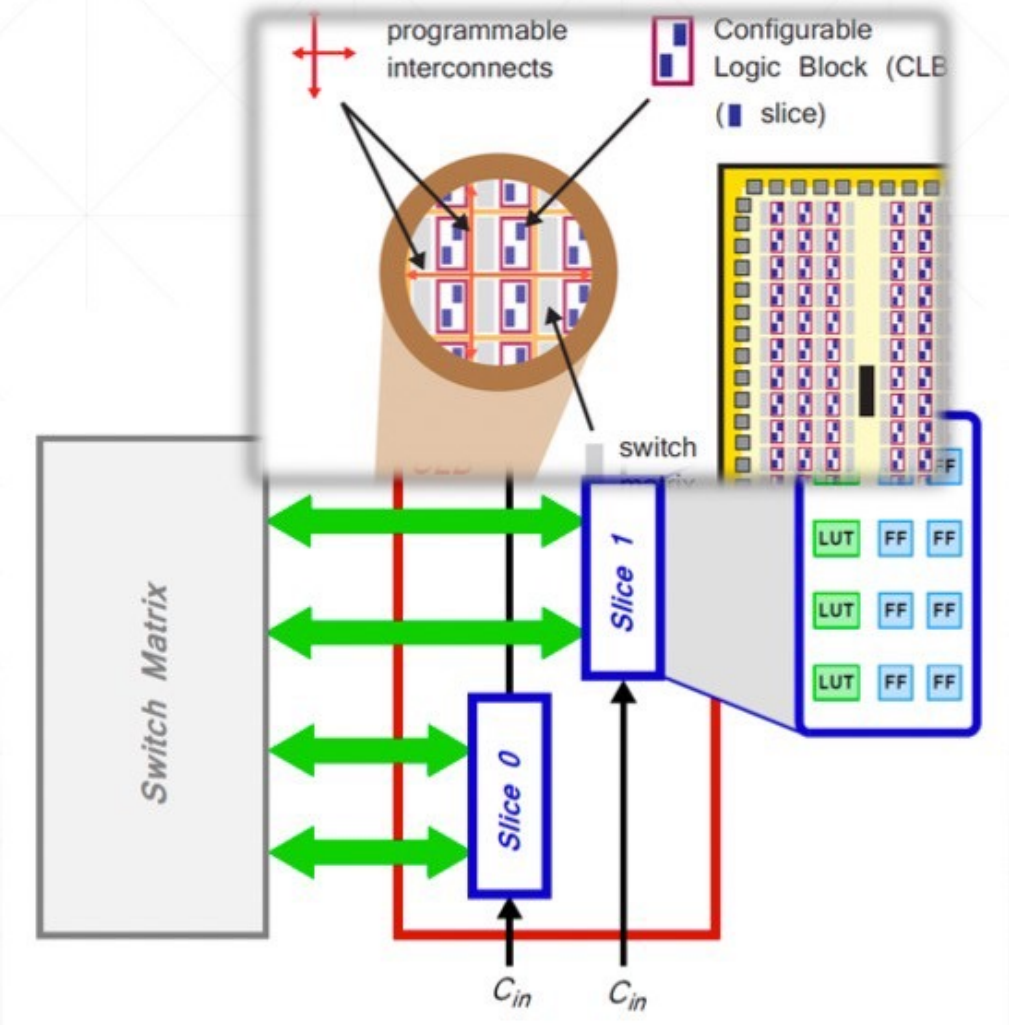
The Logic Fabric: Overview



- Programmable logic units
  - Configurable Logic Blocks (CLBs)
  - Can be programmed to realise different digital functions
- Programmable interconnect
  - Allow different blocks to be connected together
- Programmable I/O pins
  - Send/receive data external to chip

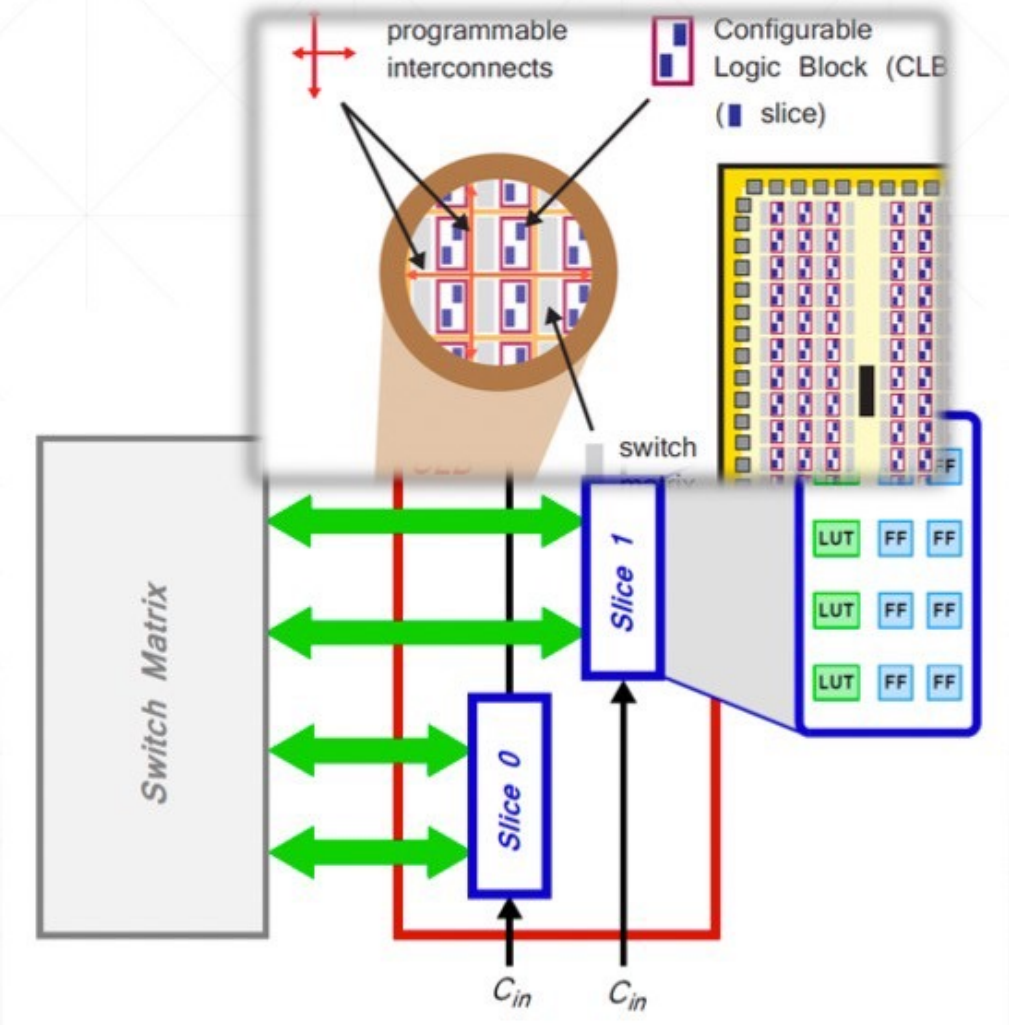
The Logic Fabric: CLB

- 2D arrays of small blocks
  - Made up of ‘slices’ containing basic logic elements
- FFs allow for synchronous logic
- Switch matrix provides flexible routing facility between CLBs



The Logic Fabric: CLB

- LUTs used to implement combinatorial logic
  - Versatile, used in many ways
  - RAM-based function generator
  - Stores truth table, which can implement logic function or small memory block
  - Shift register



LUTs: Computations and Logic Functions

LUT Logic Function:
A xor B = C

A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

LUT Computation:
Full Adder Truth Table

A	B	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

LUTs: Computations and Logic Functions

- Distributed RAM is also referred to as LUT RAM
  - Uses CLBs to form RAM
  - Fast and localised. Ideal for small data buffers, FIFOs etc.
- Truth table of LUT stores data instead of function
  - The LUT inputs become address lines
  - LUTs can be combined to provide larger address space
- Writing to Distributed RAM is synchronous, reading is asynchronous (so old data is read at start of write)
- Can access small amounts of memory from any part of the logic fabric with no long routing paths

LUTs: Distributed RAM

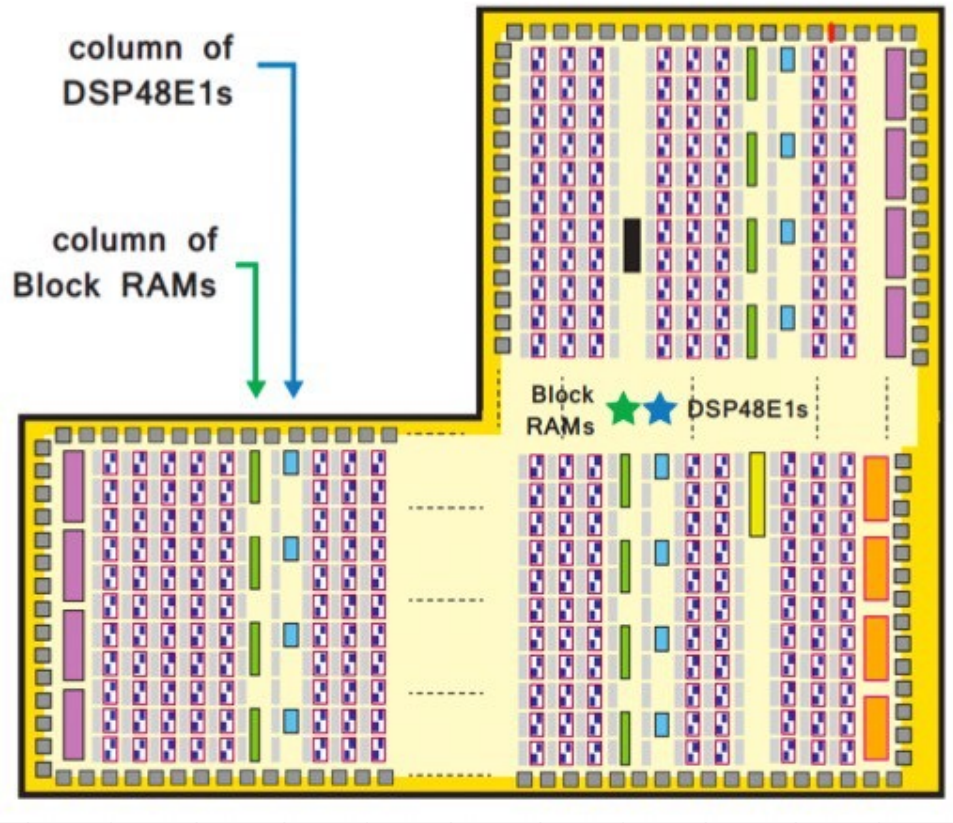
LUT Distributed RAM:

- When writing the contents of the LUT, may write any values in output columns
- A, B need not have specific logic function producing C,D and E
- A and B become our address pins, allowing us to store 4 separate 3-bit words
- Note actual size of LUTs (number of inputs and outputs) depends on chip.

Input signals acts as Address:

A	B	C	D	E
0	0	0	0	1
0	1	0	1	0
1	0	0	1	1
1	1	?	?	?

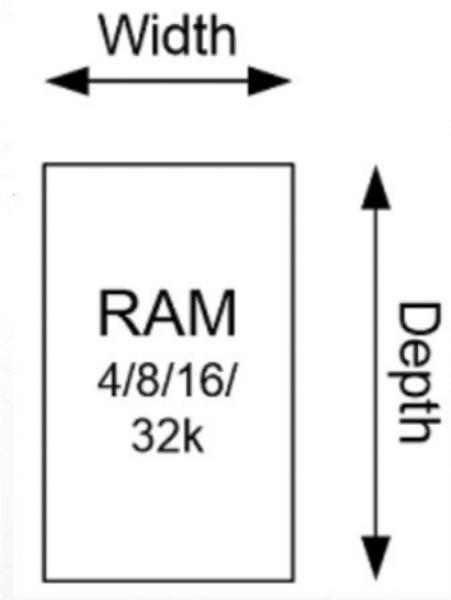
The Logic Fabric: Fixed Resources



- As well as generic CLBs for logic functions, have specialised blocks for specific functions
- Integrated into logic array with close proximity to logic blocks
- Block RAMs for high-speed memory requirements
- DSP48E1s or just ‘DSP Slices’ for high-speed computations
- Many others ...

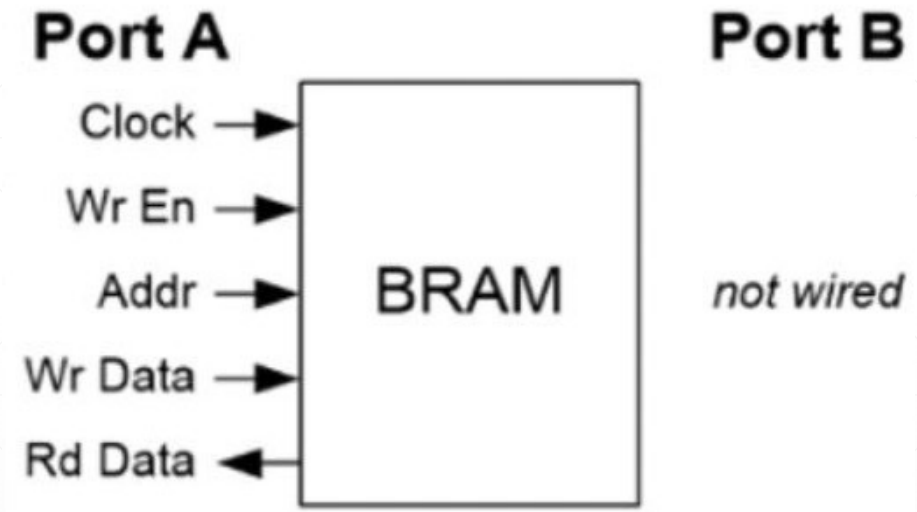
Fixed Resources: Block RAM

- Dedicated memory resource, separate column
  - Used only as memory resource
  - Limited number of BRAM blocks
- Different configurations available
  - Multi port, multi clock options
  - Single port, Dual Port, FIFO
- Built in error detection and correction
- Synchronous write and read
- Can be written by bit-stream



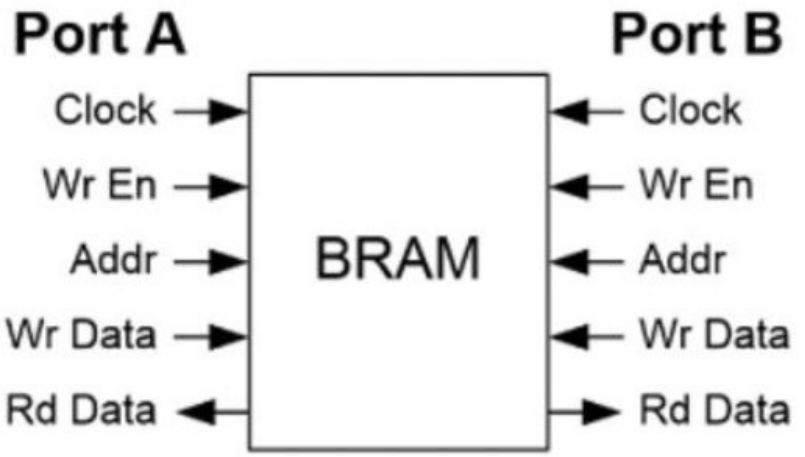
Fixed Resources: Block RAM Single Port

- Useful if only one port needs to access data
- All signal changes aligned with clock edge (synchronous)
- Write is performed if `wr_en` is high
  - `wr_data` is stored at `addr`
- Read is performed while `wr_en` is low
  - `rd_data` loaded with value stored at `addr`
  - How much energy used on subsequent clock cycles?



Fixed Resources: Block RAM Dual Port

- If need to write and read data at same time...
- Two ports, signals on each behave like single port
- Take care with dual ports
  - What if read and write attempted at same address?
  - What if two writes attempted at same address?



Fixed Resources: Block RAM Inference and Instantiation

- RTL with large memory may infer block RAM
  - Must use synchronous reset
- Can generate a core and then instantiate it
  - Can be laborious if creating many FIFOs of different sizes in the design

Block RAM vs Distributed RAM

- When to choose distributed RAM or block RAM?
- Resource usage:
  - Block RAM resources are already available and are quite large. Using them leaves more CLBs available, but there is a fixed number of Block RAMs.
- Size of data:
  - Distributed RAM is ideal for small memories, don’t occupy full Block RAM resource with small data and can implement in same region as logic that uses the memory
- Read operation:
  - Synchronous for Block RAM and asynchronous for Distributed RAM

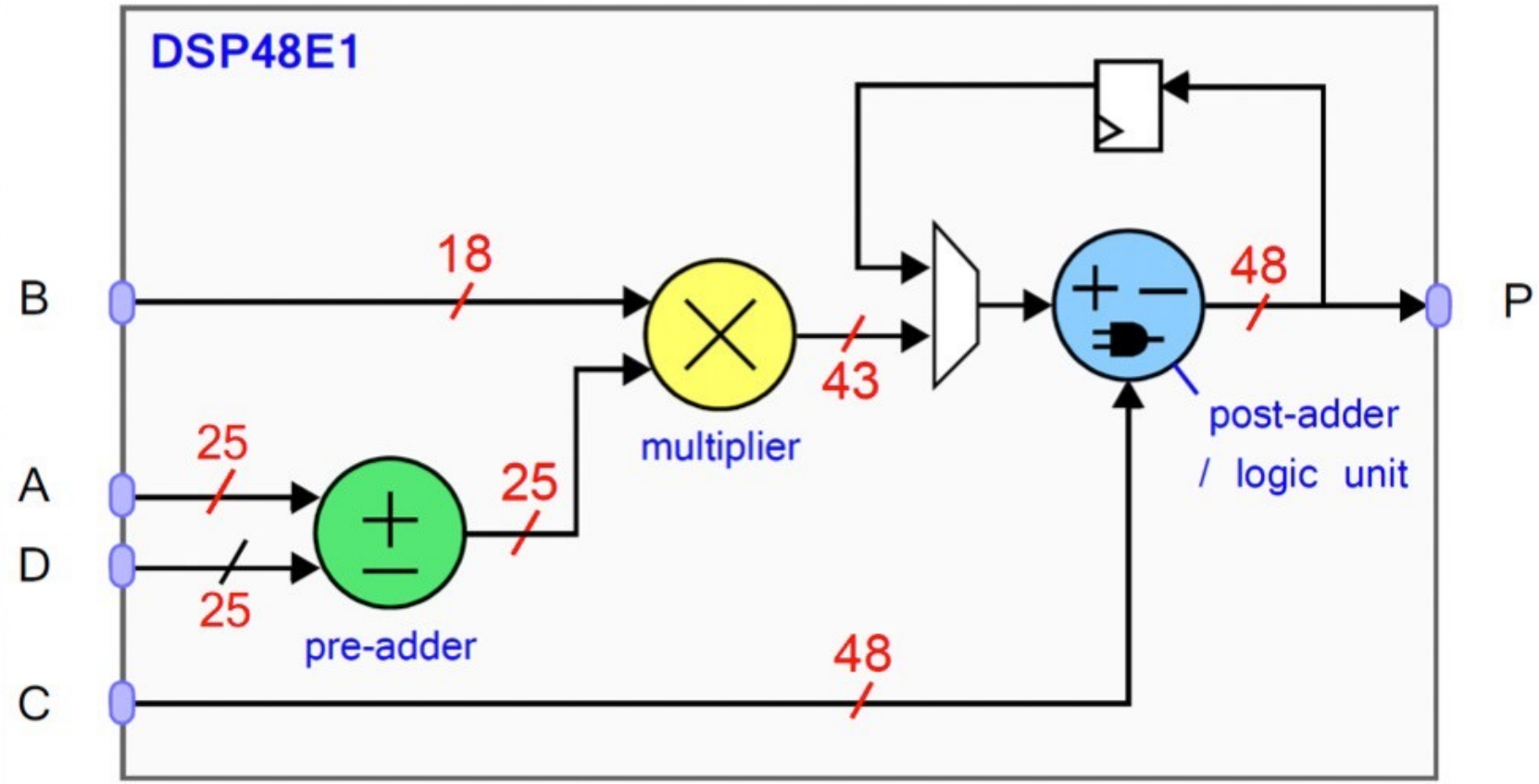
Memory Terminology

- RAM
  - Random Access Memory
    - SRAM, DRAM
    - Single Port vs Dual Port
- ROM
  - Read Only Memory
    - PROM, EPROM, EEPROM
- Volatile
  - Memory that loses the data it has stored when power cycled

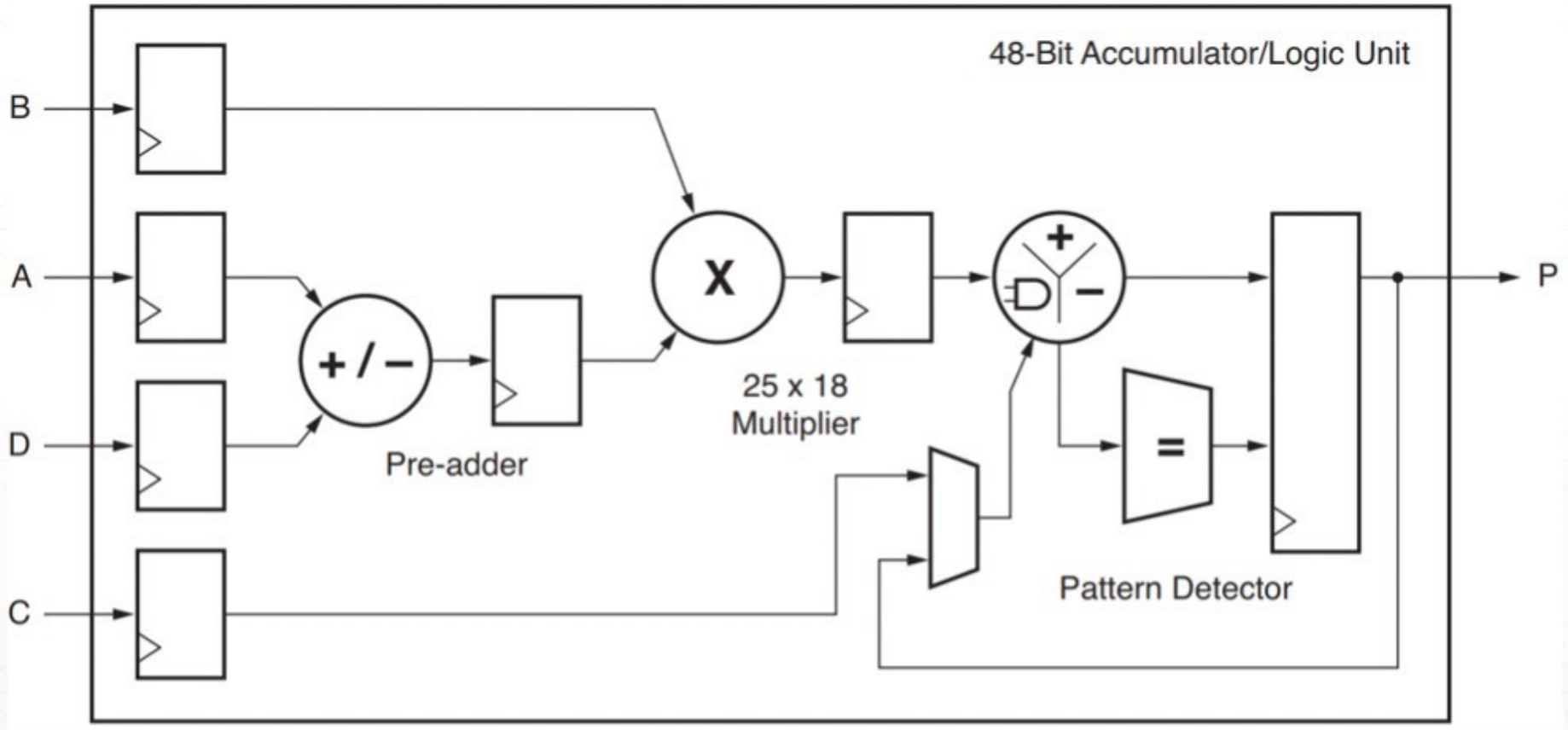
DSP Slice

- Dedicated arithmetic resource
  - Implements custom algorithms
  - High speed, low power, configurable
- Introduced on Virtex-4, we'll study 7 Series DSP48E1
- Particularly suitable for high-speed arithmetic on signals with medium to long word-lengths
- Major features include pre-adder/subtractor, multiplier and a post-adder/subtractor or logic function

DSP Slice: Simplified Structure



DSP Slice: (Less) Simplified Structure



A 25b, B 18b, C 48b, D 25b

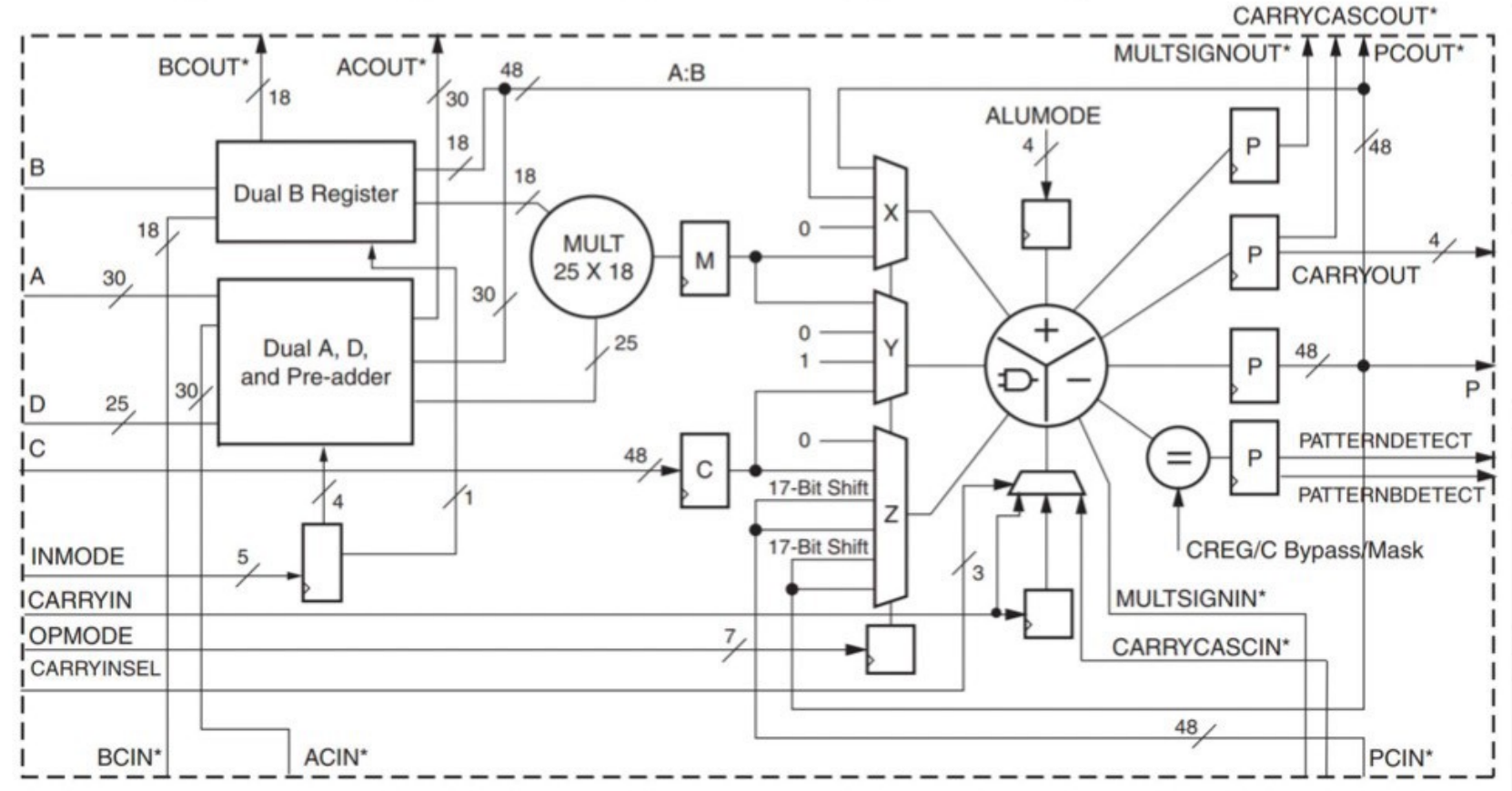
DSP Slice: Features

- 25b x 18b two's complement multiplier
  - Multiplies (A ± D) x B
- 48b accumulator
  - Can add data carry-in to multiplier output
  - Can also be stand-alone
- Pre-adder
  - Send A+D or A-D to multiplier

DSP Slice: Features (contd)

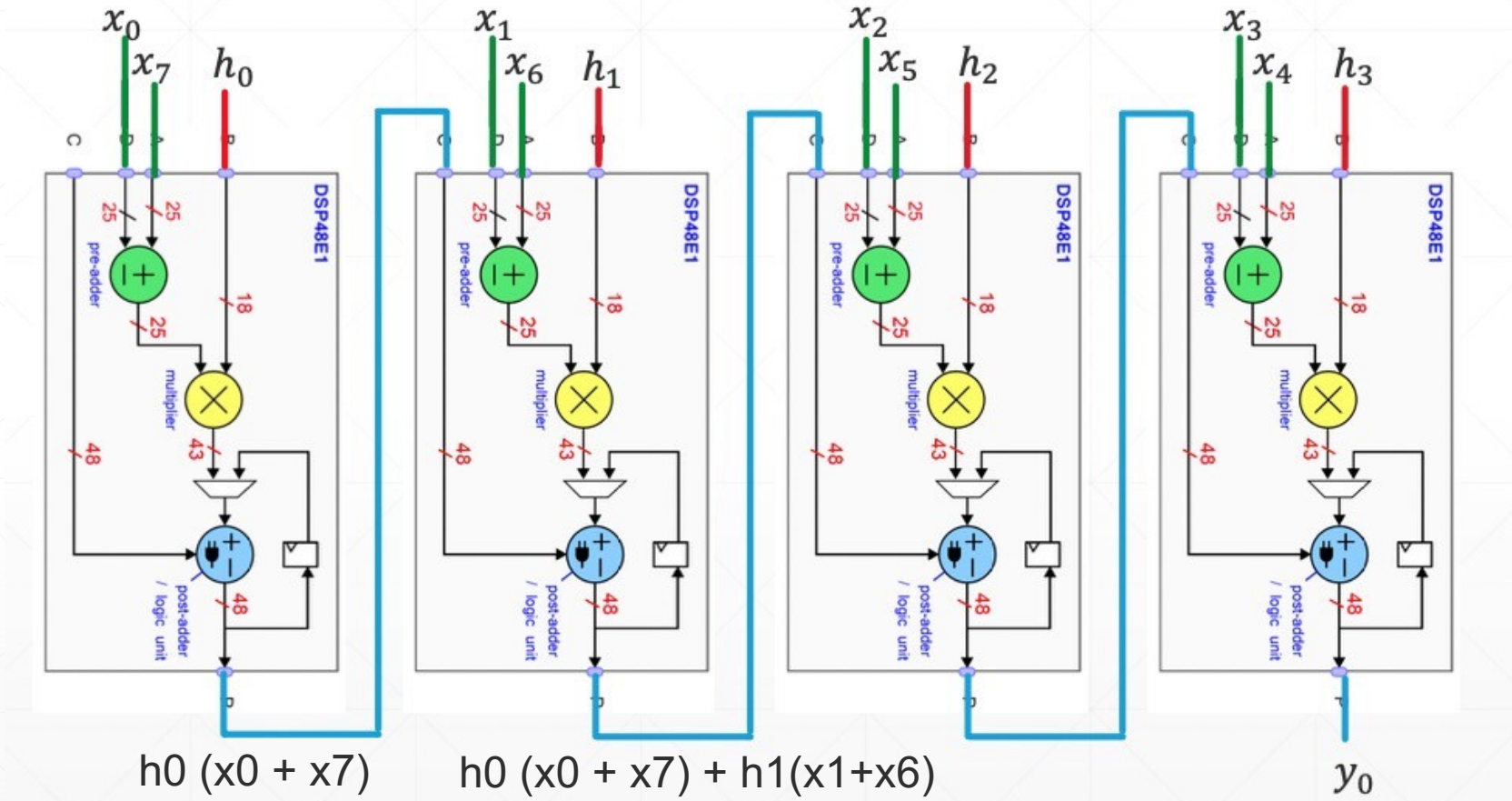
- Pattern detection
- Single Instruction Multiple Data (SIMD)
- Optional pipelining and cascading
- Dedicated memory access
- Can be clocked at device rate

DSP Slice: (for reference)



DSP Slice for FIR Filters

Y(n) = h0 (x0 + x7) + h1 (x1 + x6)
+ h2 (x2 + x5) + h3 (x3 + x4)

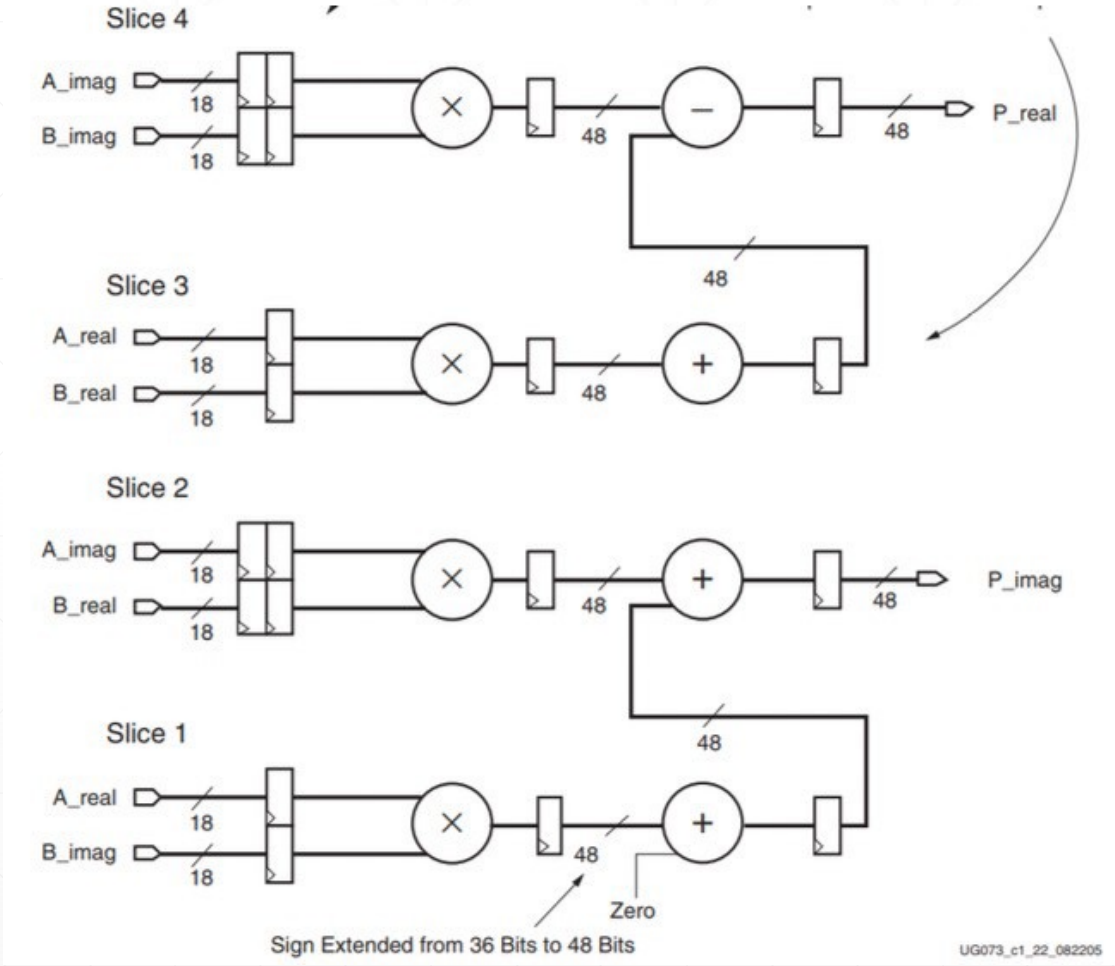


$$y_0 = h_0(x_0 + x_7) + h_1(x_1 + x_6) + h_2(x_2 + x_5) + h_3(x_3 + x_4)$$

DSP Slice for FIR Filters

- A,D take input signal samples
- B, multiplier input, takes filter tap value
- C takes output of previous stage to accumulate (MAC)
- Can fully implement filter without using any of the general fabric
  - Highly efficient implementation
  - Good for high performance applications

DSP Slice for Complex Number Multiplication



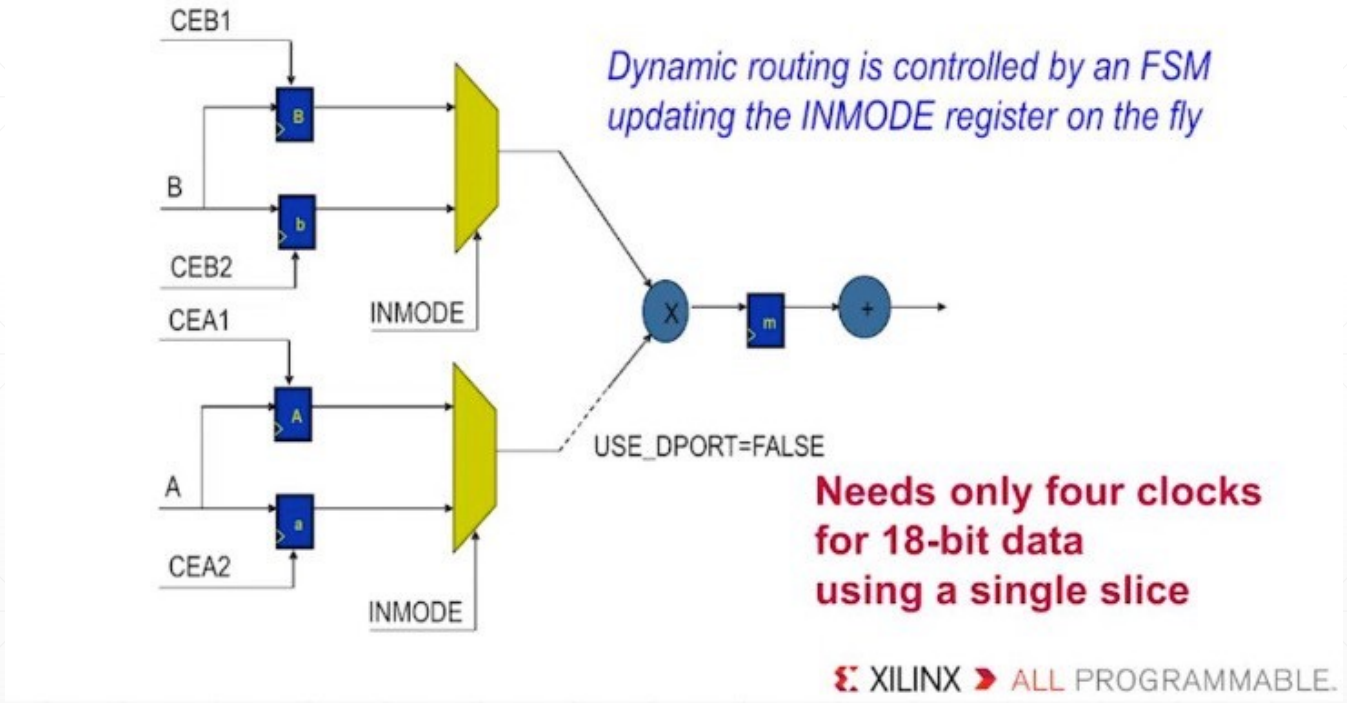
DSP Slice for Complex Number Multiplication

- A and B multiply pairs of numbers
- 4 multiplications to produce all complex number partial products
- 2 post adders are used to combine 2 sets of 2 partial products
  - `(A+jB) * (C + jD) = (AC – BD) + j(AD + BC)`
- Note that additional pipelining is used on DSP slices that will perform the post adding
  - Balances delay from register on slice3 multiplier output
  - Delays the output of slice3's multiplier relative to slice4's multiplier
  - Spend some time considering diagram to ensure you understand why this is

DSP Slice for Complex Number Multiplication

Complex Multiply

- (A + ai) * (B + bi) = (AB-ab) + (Ab+aB)i
- Use the two AB registers to locally store the real and imaginary parts of the operands
  - Read each component of the complex operands out of memory only once
  - Fewer memory reads because A, a, B, and b are then stored locally



https://www.xilinx.com/video/fpga/7-series-dsp-resources.html

DSP Slice: SIMD

- General term for system that can produce multiple data with a single instruction
- Single input signal
  - Input word is concatenation of multiple smaller words
- DSP Slice allows disabling of carries
  - Separate words in larger words are handled independently

Inferring the DSP Slice

- By default, synthesis will infer:
  - DSP slice for mult, mult-add, mult-sub and mult-accumulate
  - Logic implementation for adders, subtractors and accumulators
- `USE_DSP48` attribute forces usage
- Core generator and system generator useful for complex applications

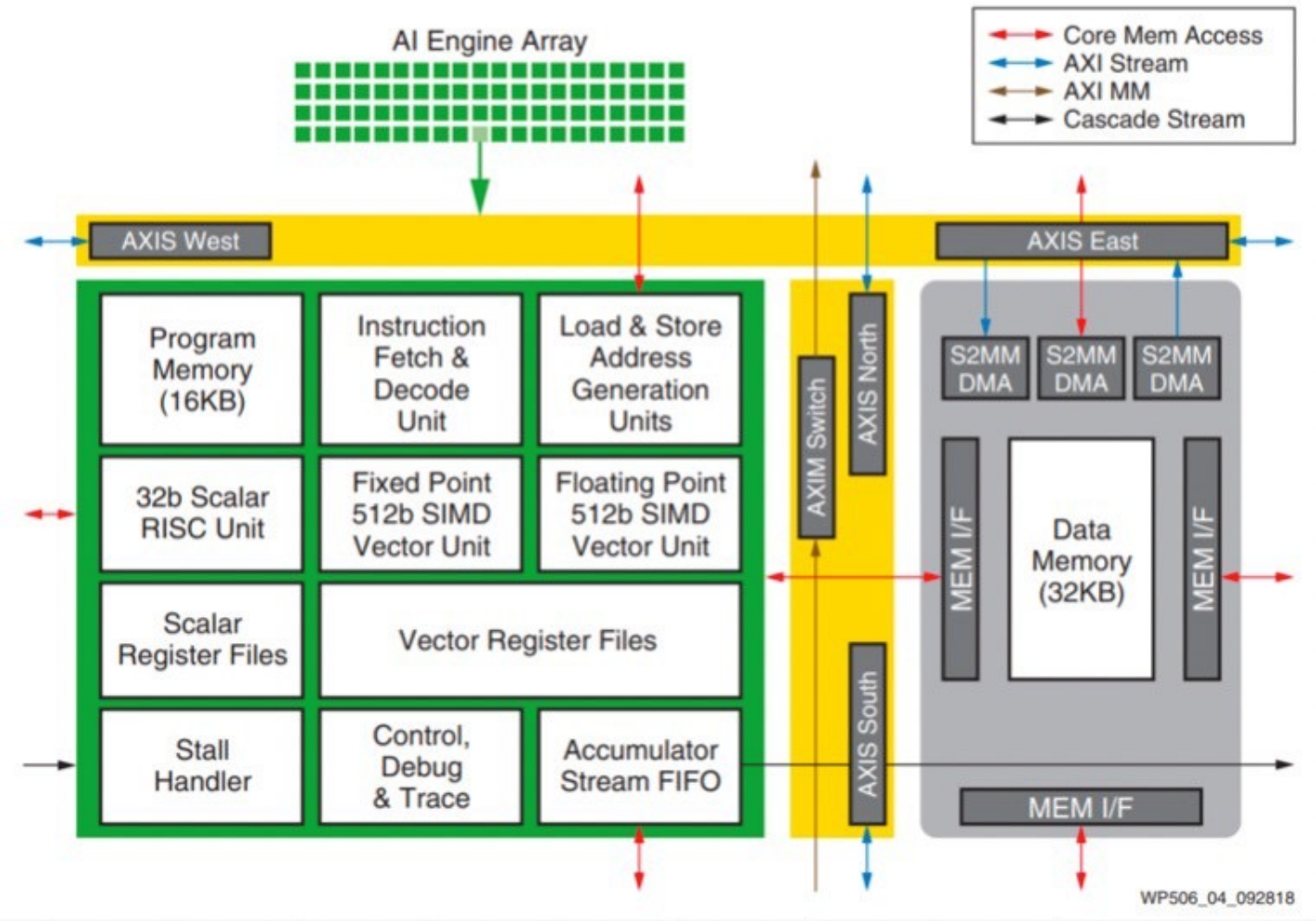
AI Engine

- Exponential growth of computational power from Moore's law has ended
  - But 5G/6G and AI demands faster computation
- AI Engine is specialised resource
  - Optimised for DSP and AI/ML computation
- Dedicated static memories for data and instruction
  - Deterministic timing, good for real-time

AI Engine

- Dedicated 16KB instruction memory and 32KB of RAM
- 32b RISC scalar processor
- 512b fixed-point and 512b floating-point vector processor with associated vector registers
- Synchronization handler
- Trace and debug

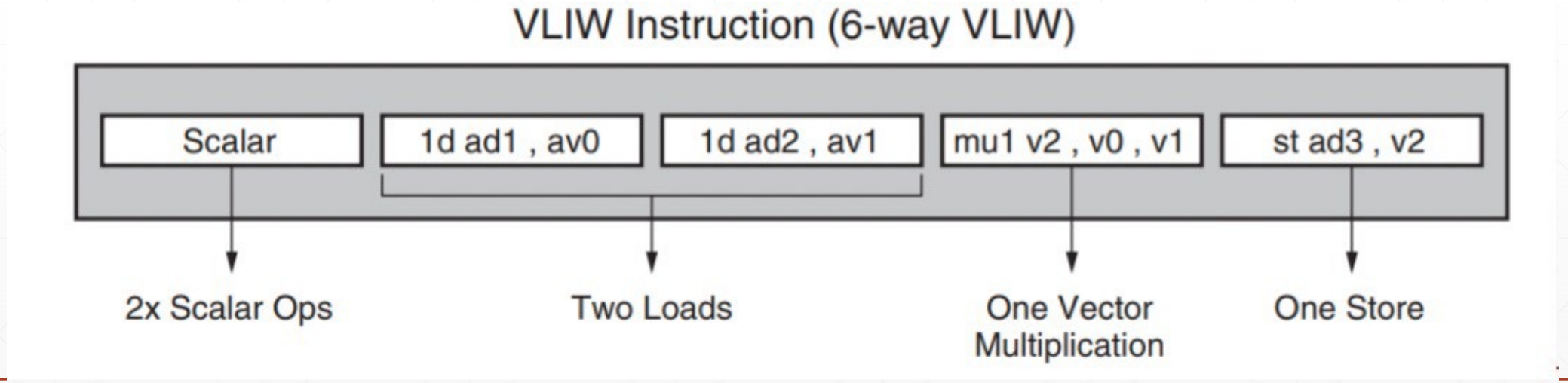
AI Engine Structure



https://www.xilinx.com/support/documentation/white_papers/wp506-ai-engine.pdf

AI Engine Instructions

- Similar idea to SIMD in DSP slice, AI engine uses Very Long Instruction Word (VLIW) strategy
- Exploits instruction level parallelism



Artix-7 FPGA Feature Summary

Table 4: **Artix-7 FPGA Feature Summary by Device**

Device	Logic Cells	Configurable Logic Blocks (CLBs)		DSP48E1 Slices[2]	Block RAM Blocks[3]			CMTs[4]	PCIe[5]	GTPs	XADC Blocks	Total I/O Banks[6]	Max User I/O[7]
		Slices[1]	Max Distributed RAM (Kb)		18 Kb	36 Kb	Max (Kb)						
XC7A12T	12,800	2,000	171	40	40	20	720	3	1	2	1	3	150
XC7A15T	16,640	2,600	200	45	50	25	900	5	1	4	1	5	250
XC7A25T	23,360	3,650	313	80	90	45	1,620	3	1	4	1	3	150
XC7A35T	33,280	5,200	400	90	100	50	1,800	5	1	4	1	5	250
XC7A50T	52,160	8,150	600	120	150	75	2,700	5	1	4	1	5	250
XC7A75T	75,520	11,800	892	180	210	105	3,780	6	1	8	1	6	300
XC7A100T	101,440	15,850	1,188	240	270	135	4,860	6	1	8	1	6	300
XC7A200T	215,360	33,650	2,888	740	730	365	13,140	10	1	16	1	10	500

Notes:

1. Each 7 series FPGA slice contains four LUTs and eight flip-flops; only some slices can use their LUTs as distributed RAM or SRLs.
2. Each DSP slice contains a pre-adder, a 25 x 18 multiplier, an adder, and an accumulator.
3. Block RAMs are fundamentally 36 Kb in size; each block can also be used as two independent 18 Kb blocks.
4. Each CMT contains one MMCM and one PLL.
5. Artix-7 FPGA Interface Blocks for PCI Express support up to x4 Gen 2.
6. Does not include configuration Bank 0.
7. This number does not include GTP transceivers.