European Organization for Nuclear Research

An introduction to FPGA architecture

Davide Falchieri

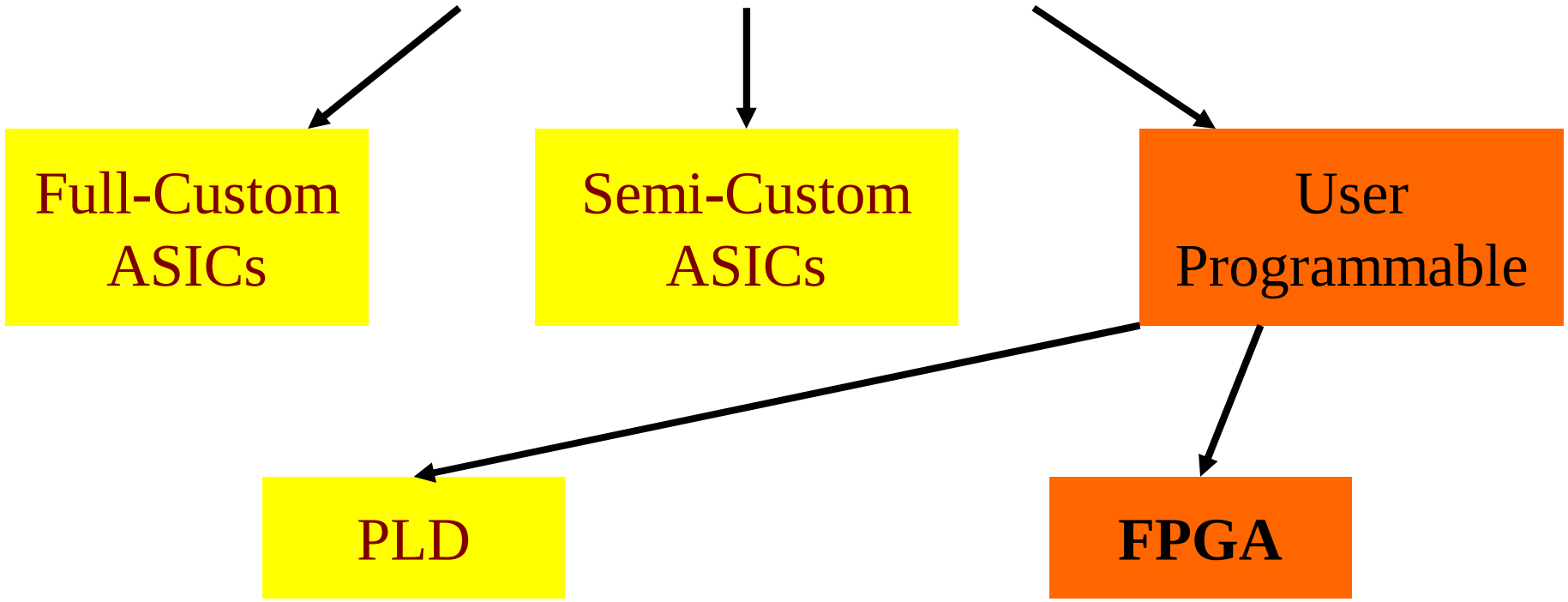
Data driven front-end electronics for highly segmented radiation detectors

Torino, 25-27 November 2013

Outline

- From CPLDs to FPGAs
- FPGA architecture

World of Integrated Circuits



ASIC
Application Specific Integrated Circuit

- designs must be sent for expensive and time consuming fabrication in semiconductor foundry
- designed all the way from behavioral description to physical layout

FPGA
Field Programmable Gate Array

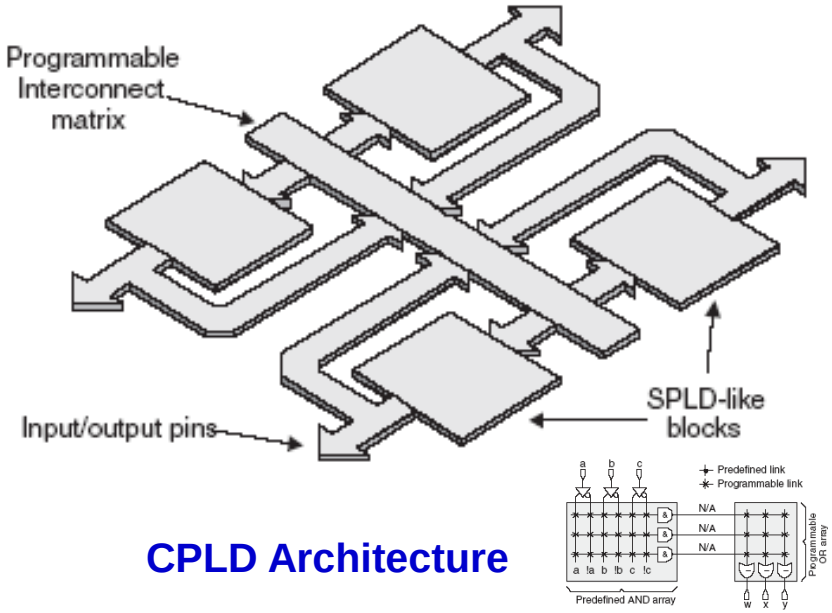
- small development overhead
- no NRE (non-recurring engineering) costs
- quick time to market
- no minimum quantity order
- reprogrammable

How can we make a “programmable logic”?

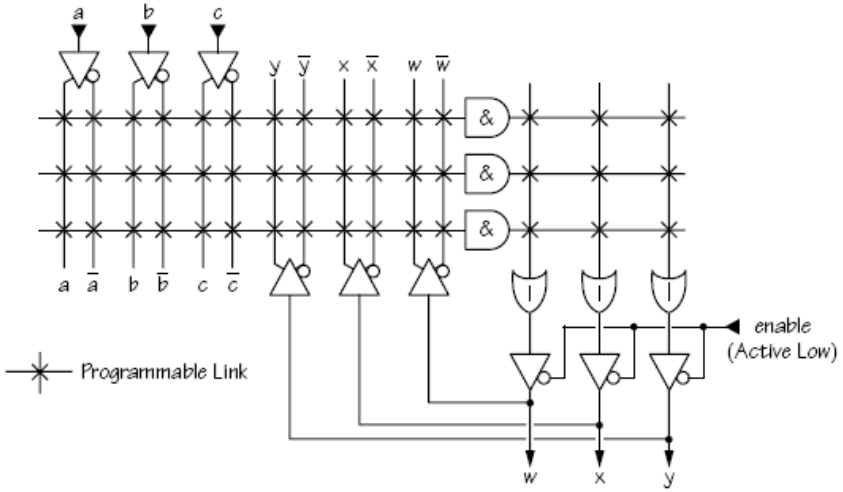
- One time programmable
  - Fuses (destroy internal links with current)
  - Anti-fuses (grow internal links)
  - PROM
- Reprogrammable
  - EPROM
  - EEPROM
  - Flash
  - SRAM (volatile)

Complex PLDs

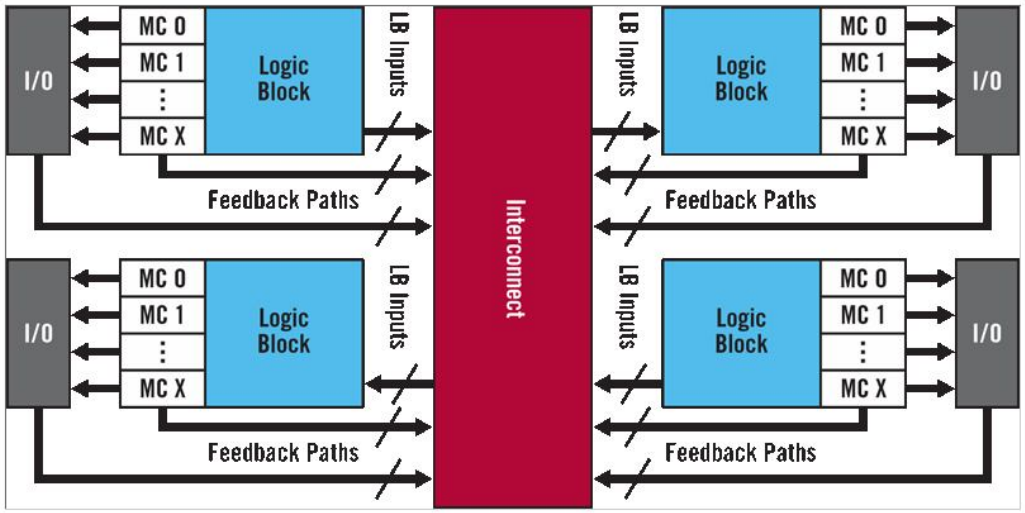
A **CPLD** is a combination of a fully programmable AND/OR array and a bank of macrocells. The AND/OR array is reprogrammable and can perform a multitude of logic functions. Macrocells are functional blocks that perform combinatorial or sequential logic, along with varied feedback paths.



CPLD Architecture



Feedback Outputs



Xilinx CPLDs features

Features	CoolRunner-II	XC9500XL
Core Voltage	1.8	3.3/2.5
Macrocells	32-512	36-288
I/Os	21-270	34-192
I/O Tolerance	1.5V, 1.8V, 2.5V, 3.3V	5.0V (XL), 3.3V, 2.5V
TPD / f max (fastest)	3.8/323	5/222
Ultra Low Standby Power	28.8µW*	Low power mode
I/O Standards	LVTTL, LVCMOS, HSTL, SSTL	LVTTL, LVCMOS

Benefits:

- reprogrammable devices
- low cost (~1-10 $)
- low power
- non volatile (FLASH based)

Top CPLDs applications

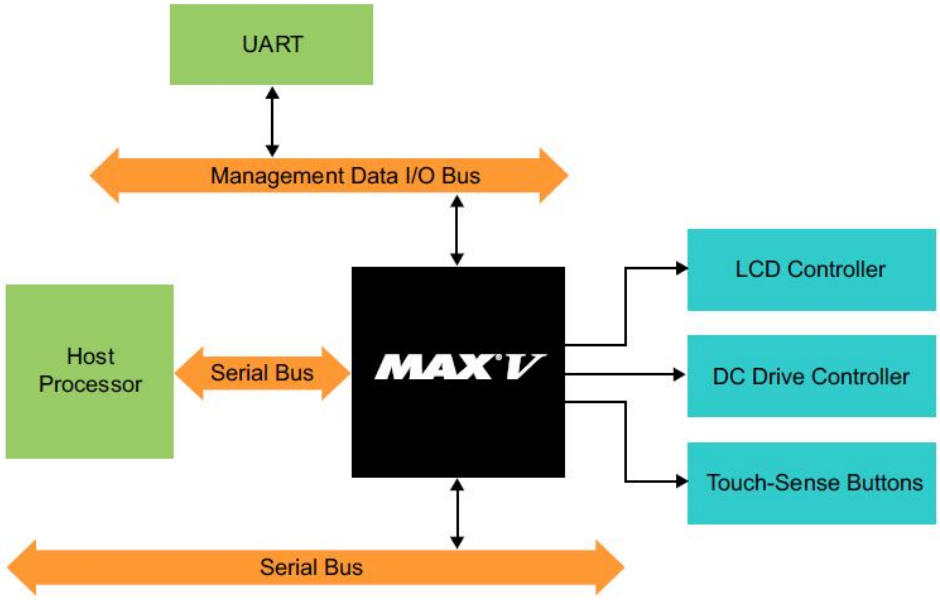
■ **I/O expansion**: performs I/O decoding, which increases the available I/O capability of another standard device with efficiency and at a low cost.

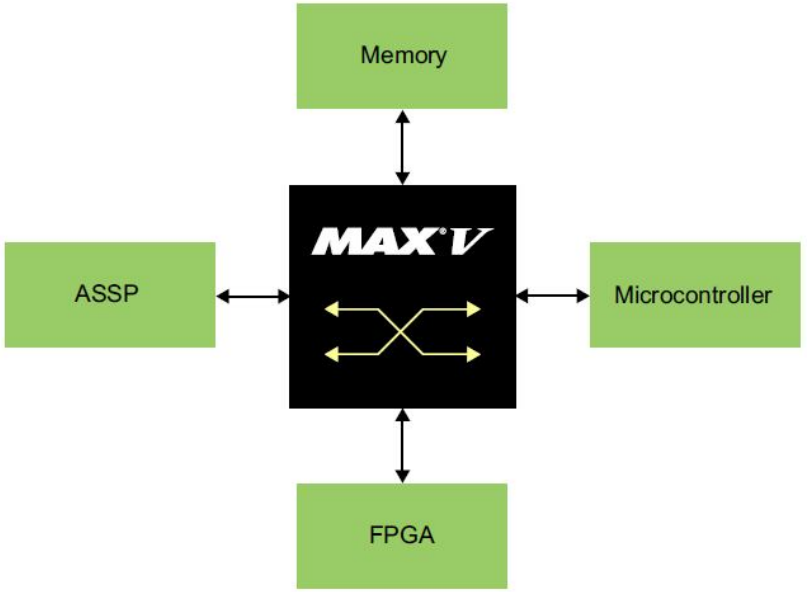
■ **interface bridging:** translates bus protocols and voltages between incompatible devices at the lowest possible cost.

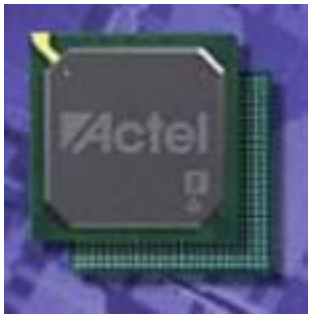
■ **power management:** manages the power-up sequencing and monitoring of other devices on the board.

■ **configuration and initialization:** controls the configuration or initialization of other devices on the board.

■ **analog control:** controls analog standard devices (light, sound, or motion) digitally via a pulse-width modulator (PWM), without needing a digital-to-analog converter (DAC).







FPGA: Field Programmable Gate Array

Born in the **1980s** from the CPLD the main manufacturers are:

Xilinx:	SRAM based devices
Altera:	SRAM based devices
Microsemi (ex Actel):	Antifuse + FLASH devices
Lattice:	SRAM based devices

Main focus will be given to Xilinx FPGAs

Short history

- First FPGA was released in **1985** by **Xilinx** with a mere 1,000 logic gates primarily used for interconnections, buses and other peripherals. By 2006 it had increased by over 10,000 times.

- FPGAs allow for **highly parallel processing** through inherent hardware nature.

- **Xilinx** and **Altera** are the two leaders in FPGA and hold over 90% of the market.

- Markets for intensive DSP applications include:
  - wireless communications,
  - video/image processing,
  - aerospace/defense industry,
  - high energy physics

Xilinx technology evolution

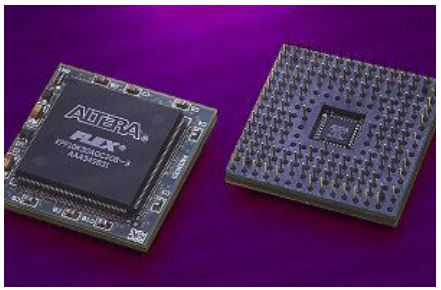
Family	Technology	Year	Logic cells
XC3000	0.7-0.5 µm	1995	1k-5k
XC4000	0.5-0.35 µm	1999	1k-7k
Virtex	0.22 µm		
Virtex-E	0.18 µm		
Virtex-II	0.15 µm		
Virtex-II PRO	0.13 µm	2004	3k-100k
Virtex4	0.09 µm		10k-200k
Virtex5	0.065 µm	2009	
Virtex6	0.040 µm	2010	100k-500k
Virtex7	0.028 µm	2012	300k-2M

Field Programmable Gate Arrays: FPGA

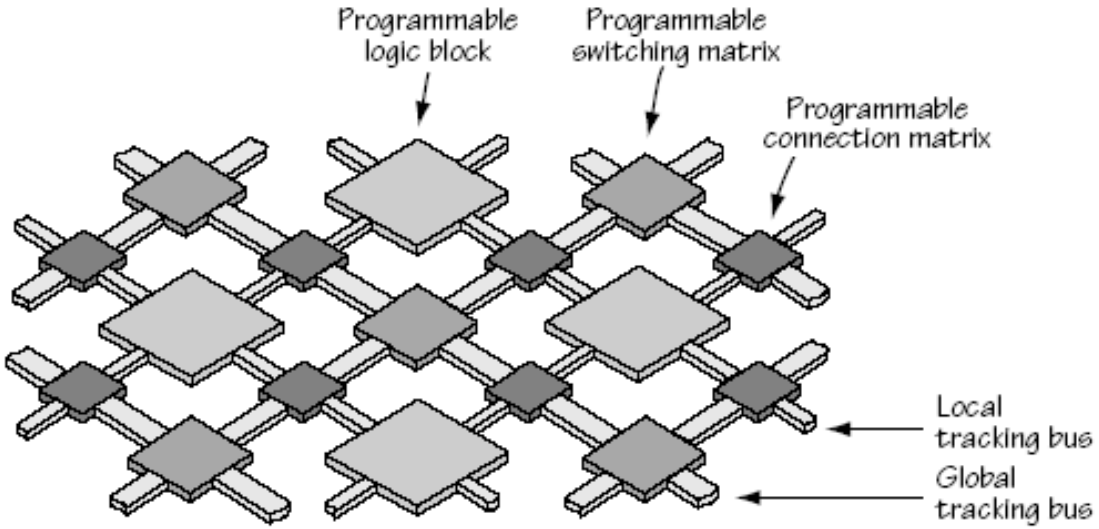
- Field Programmable Gate Array
  - New Architecture
  - ‘Simple’ Programmable Logic Blocks
  - Massive Fabric of Programmable Interconnects
  - Large Number of Logic Block ‘Islands’

  1,000 … 100,000+

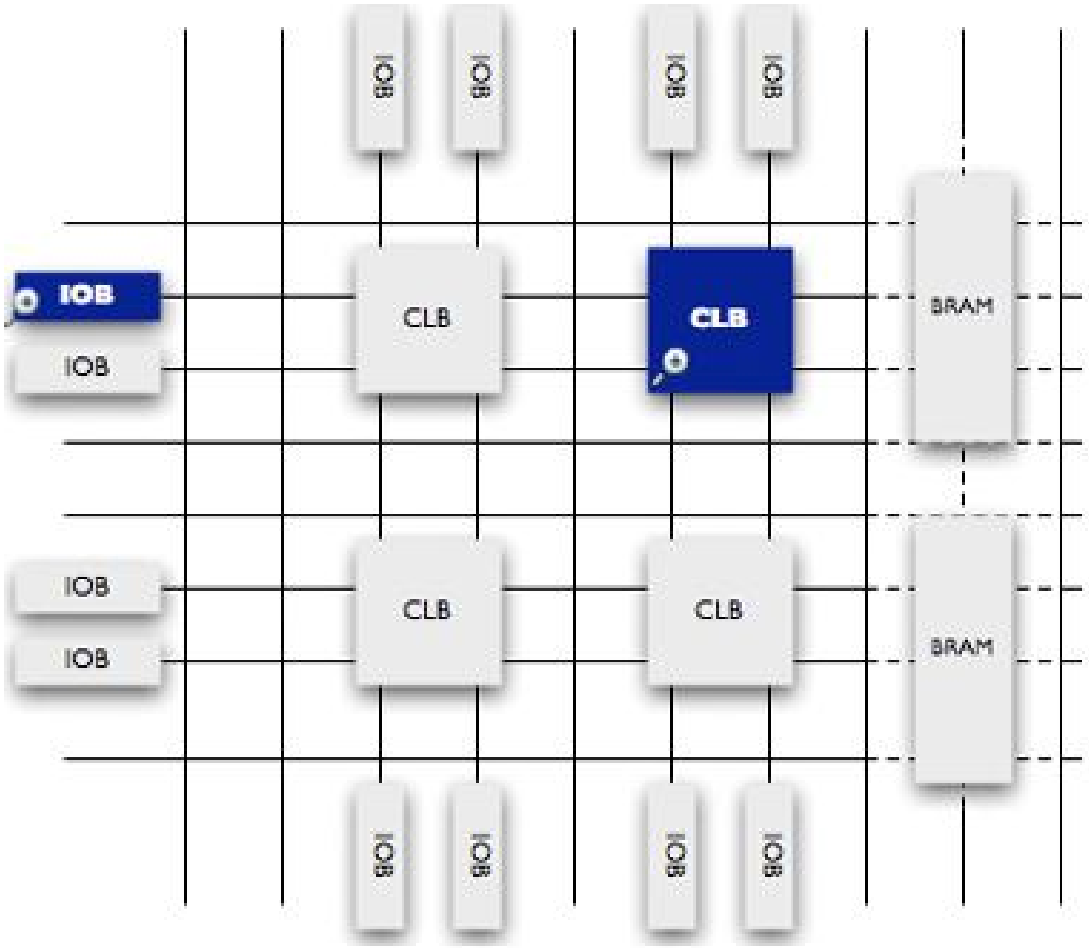
  in a ‘Sea’ of Interconnects

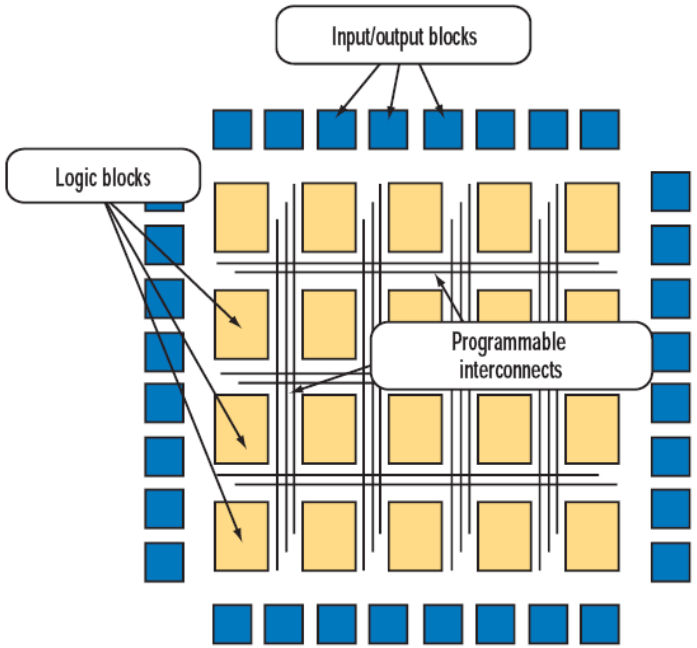


FPGA architecture



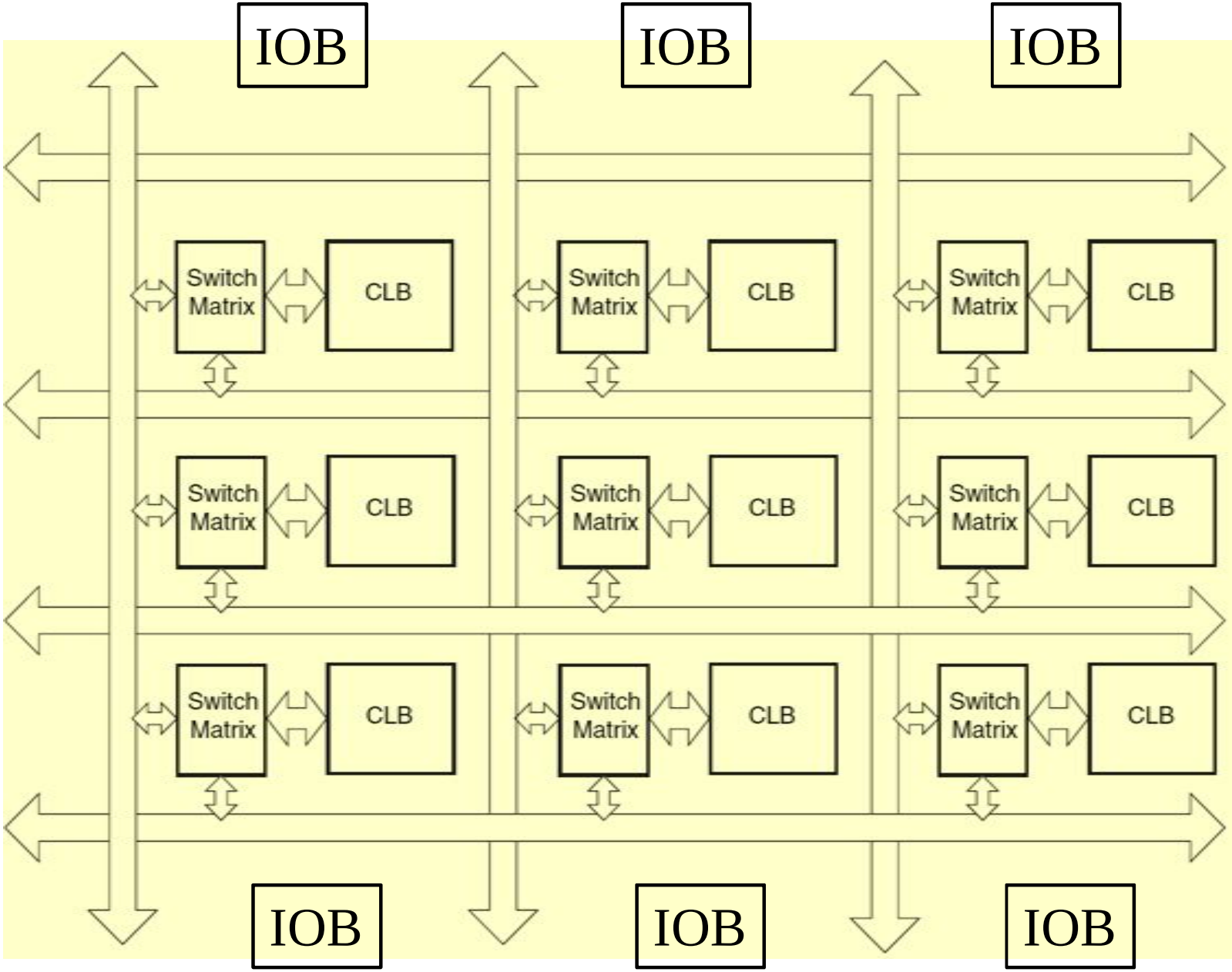
Architecture of a FPGA



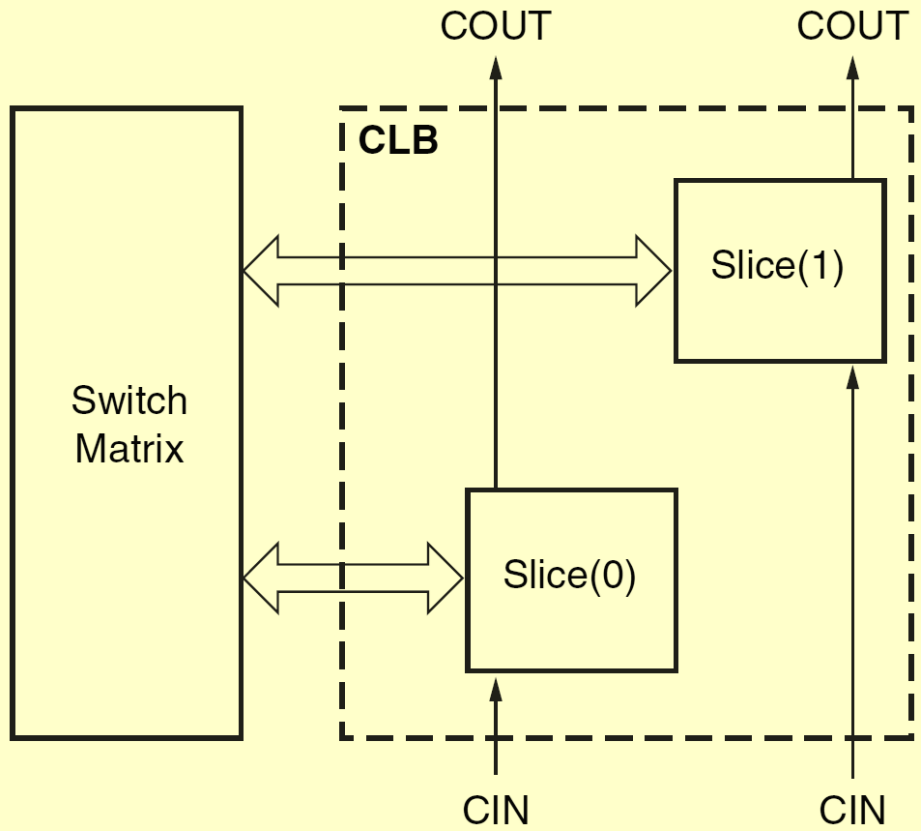


FPGA: it is a user-programmable matrix of logic blocks with programmable interconnections that can implement **any** logic function or algorithm.

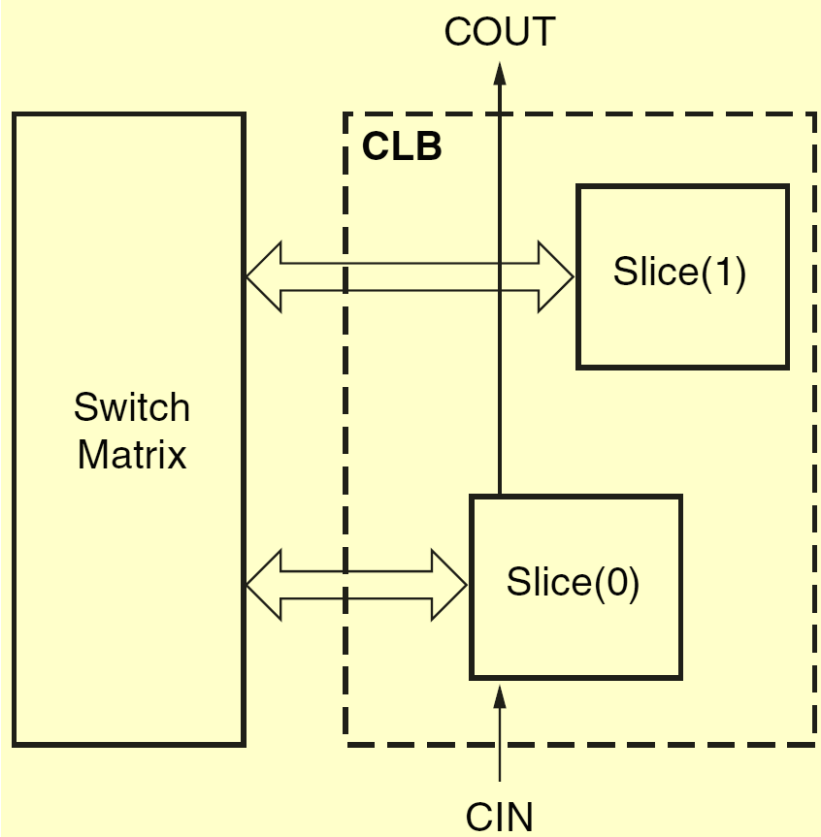
Architecture of a FPGA



The Configurable Logic Block (CLB): 2 slices



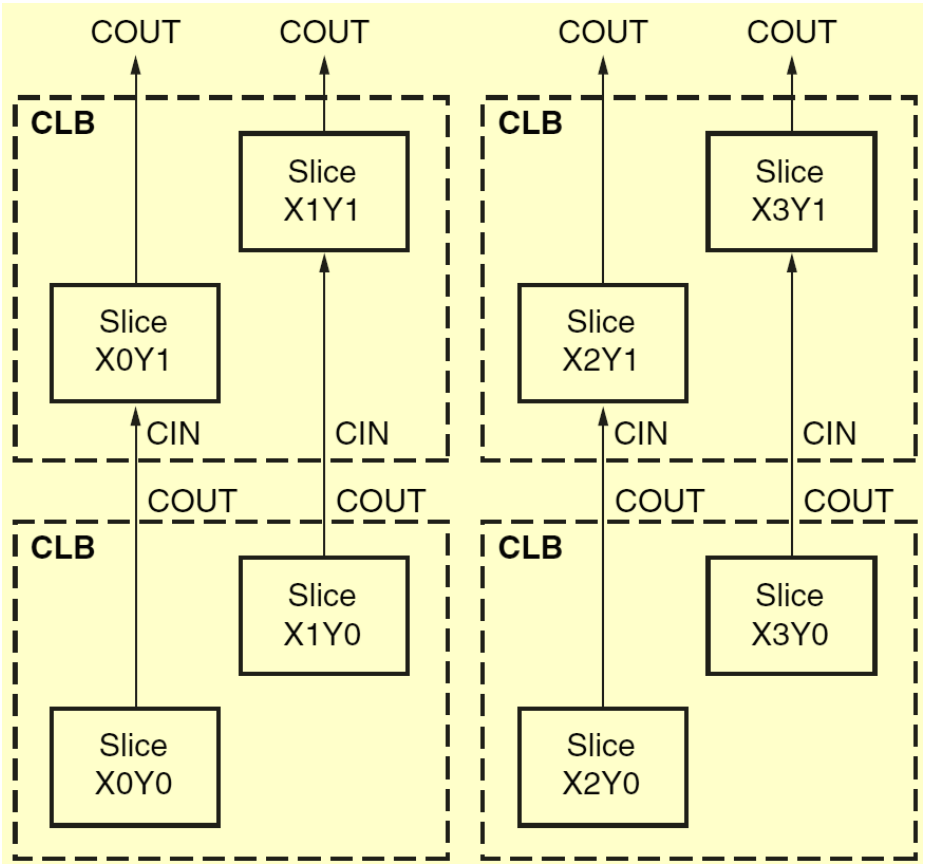
Virtex5 slice



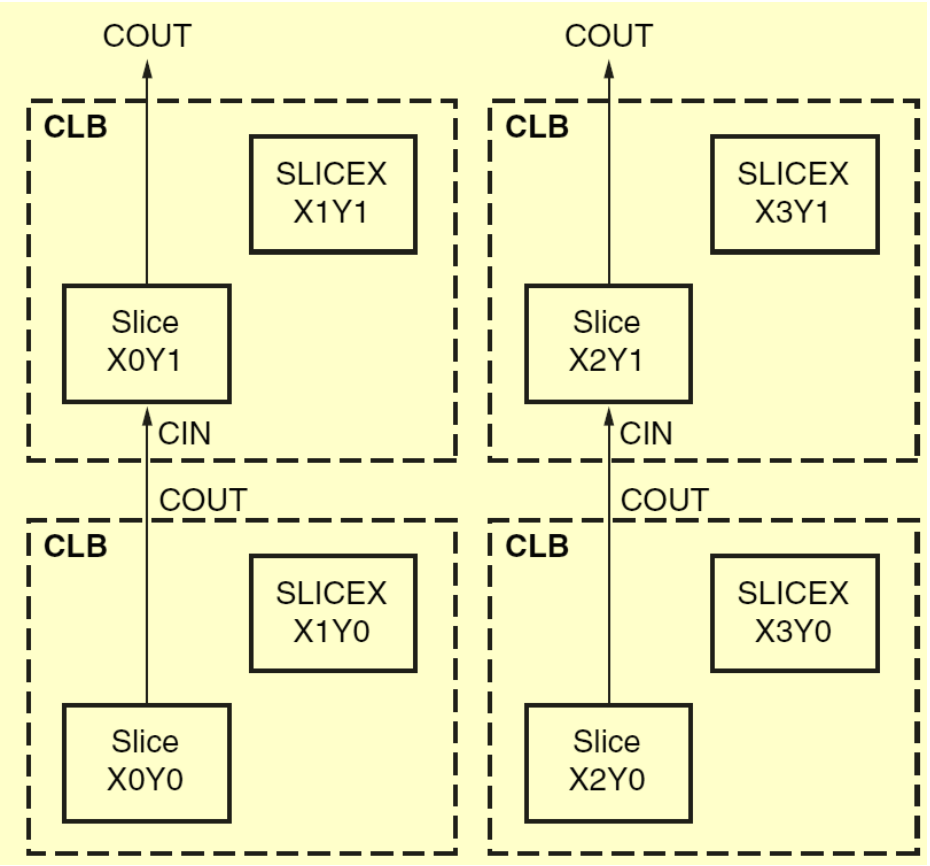
Spartan6 slice

Virtex provide carry logic chain on all slices,
while Spartan just one out of two

CLBs connected in slice columns

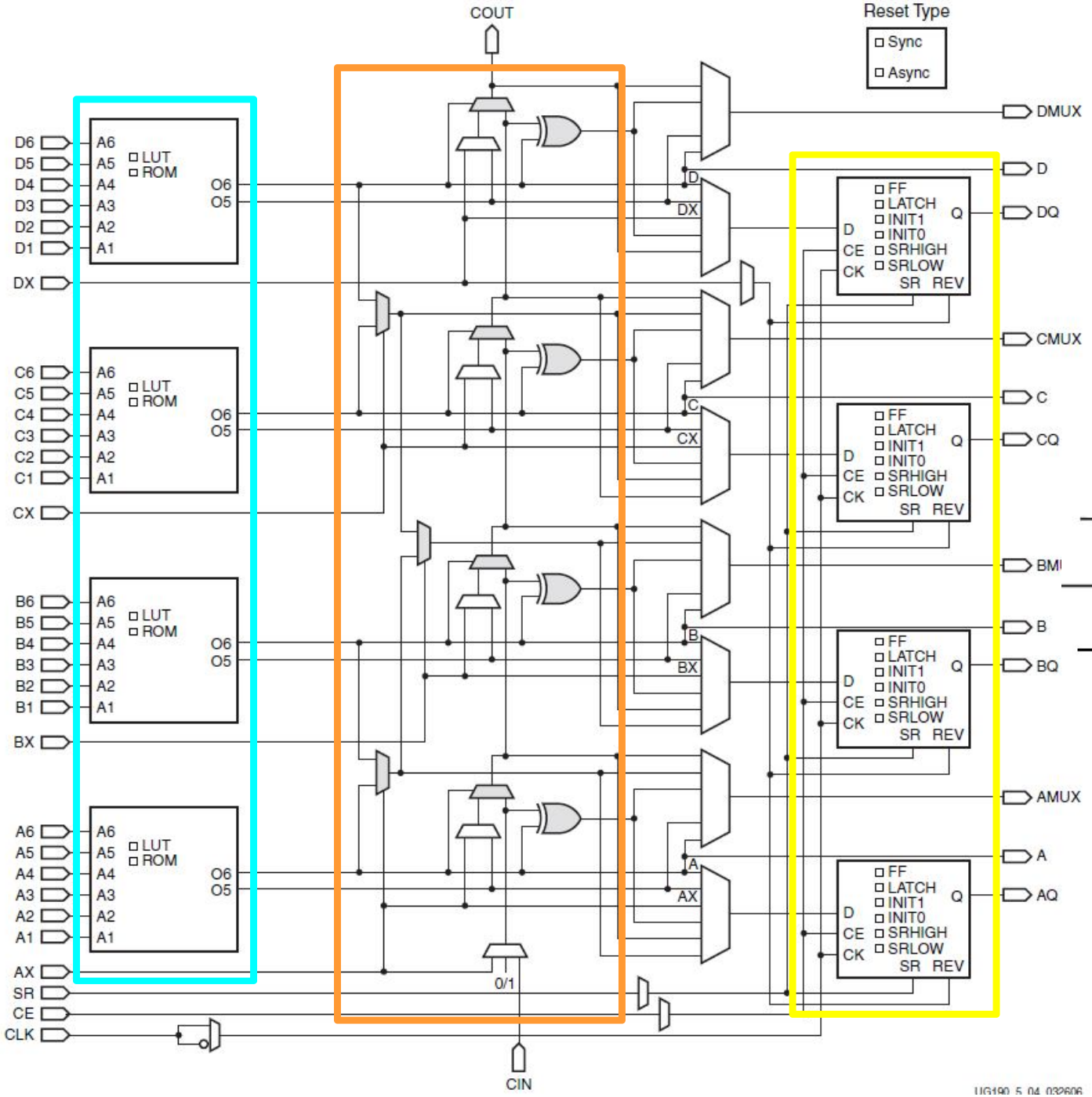


Virtex5 slices



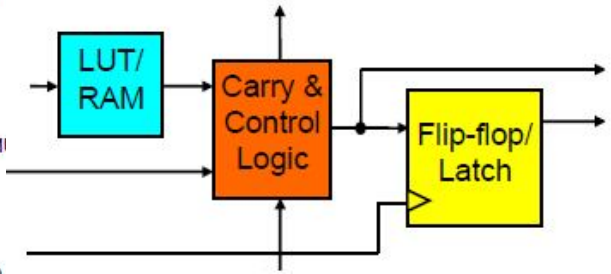
Spartan6 slices

Inside a Virtex5 slice



basic LUT structure:

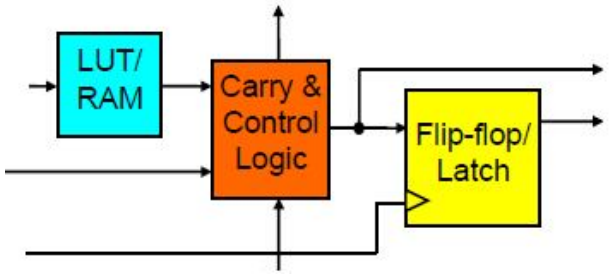
- LUT,
- carry logic,
- flip-flop



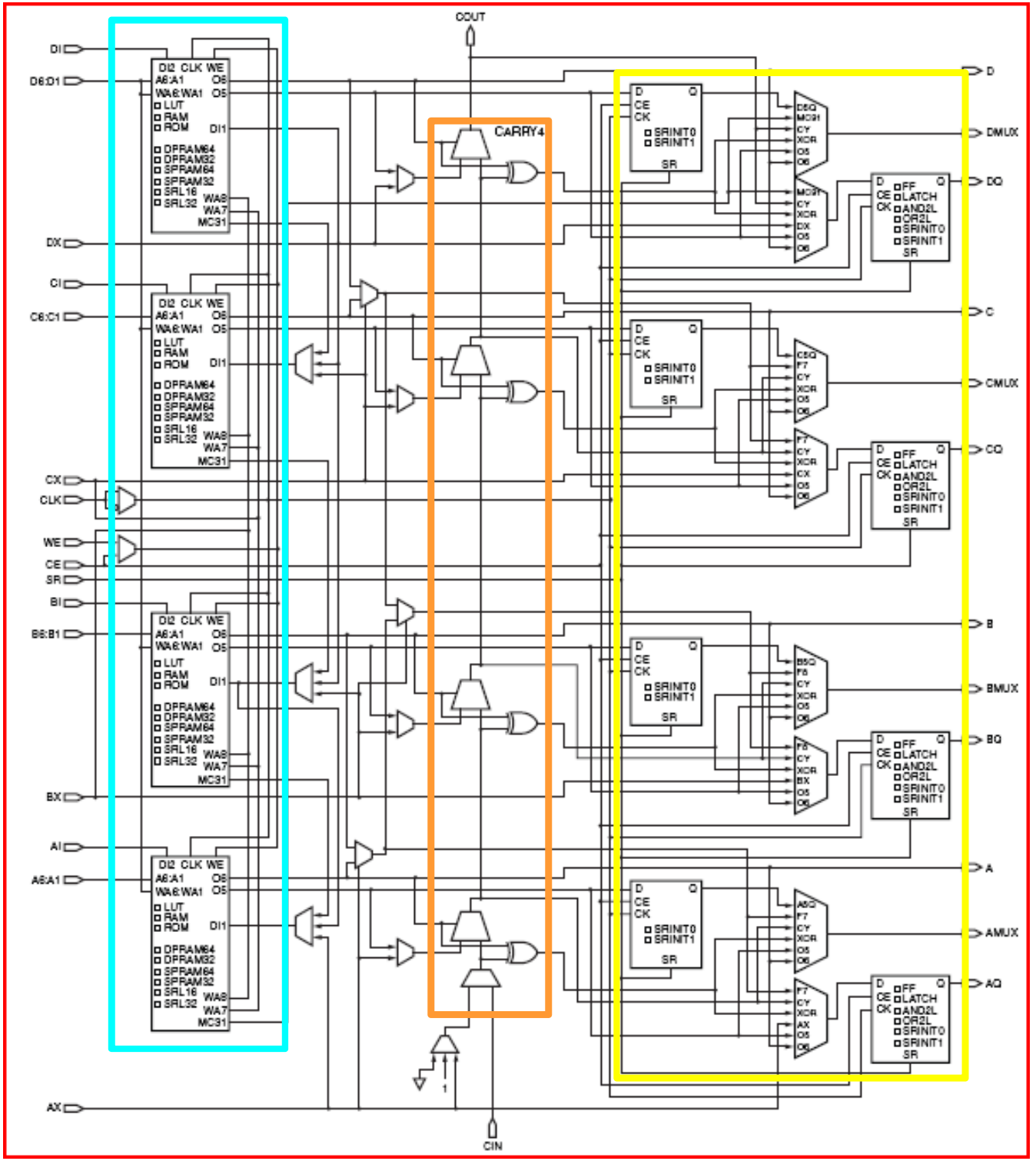
SliceM

Spartan6 features 3 types on slices:

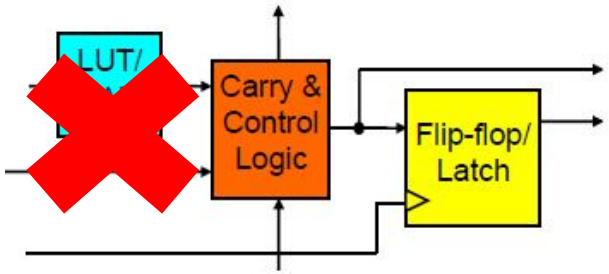
- Slice M
- Slice L
- Slice X



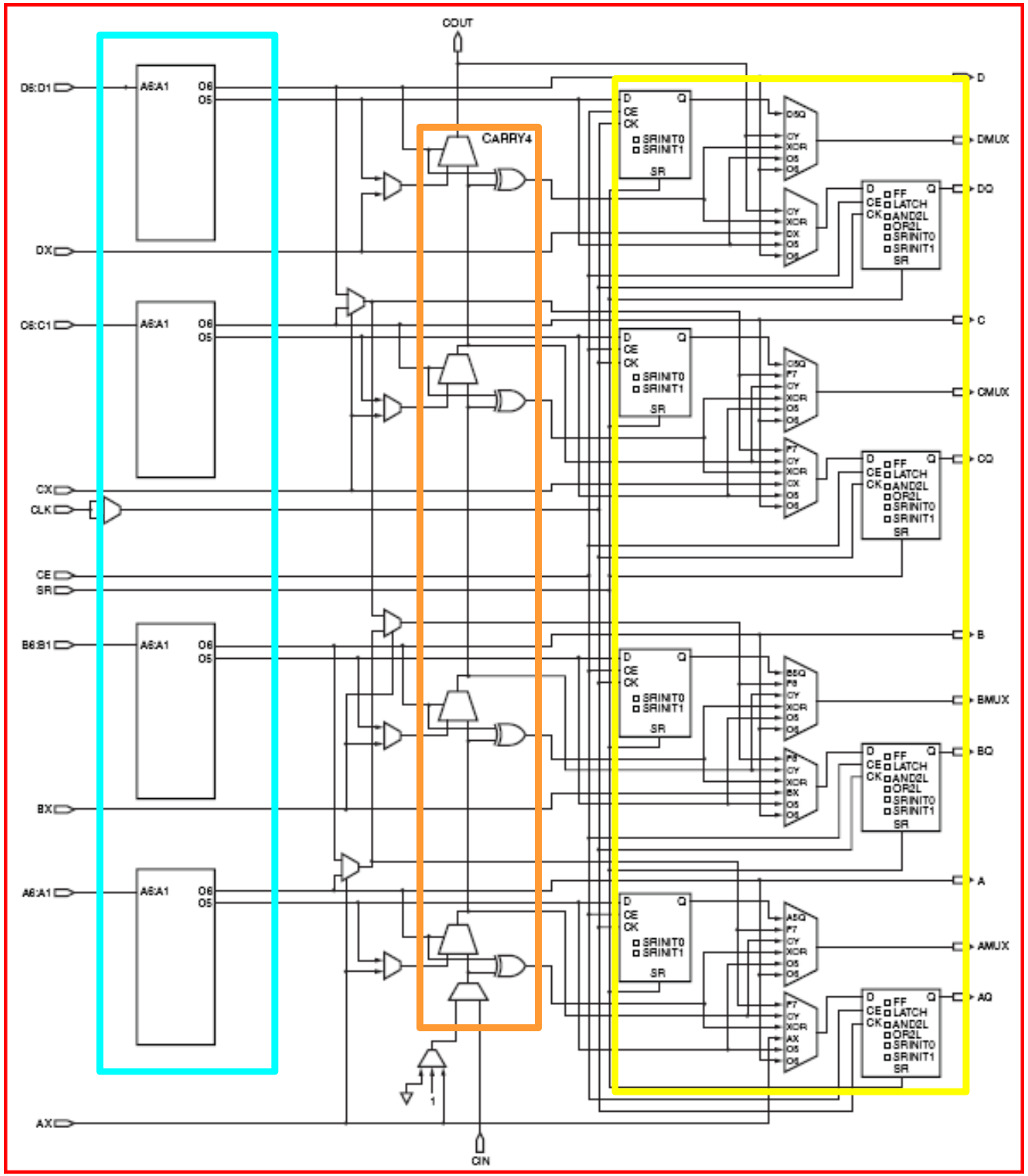
full featured



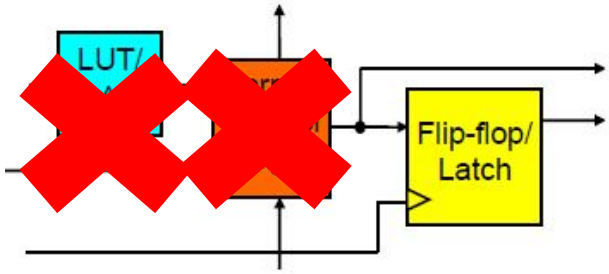
SliceL



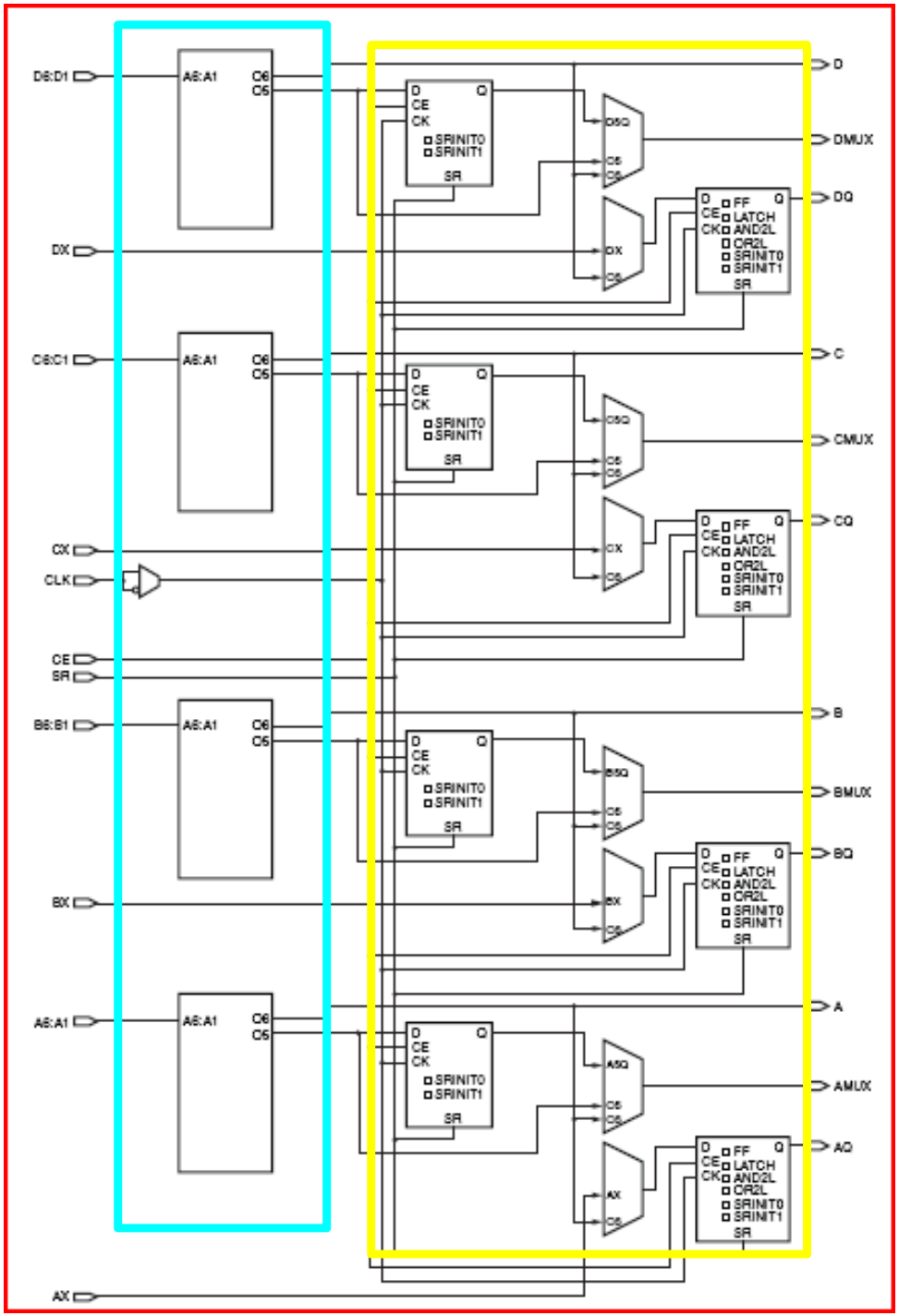
LUTs are not configurable as RAMs or SRLs



SliceX



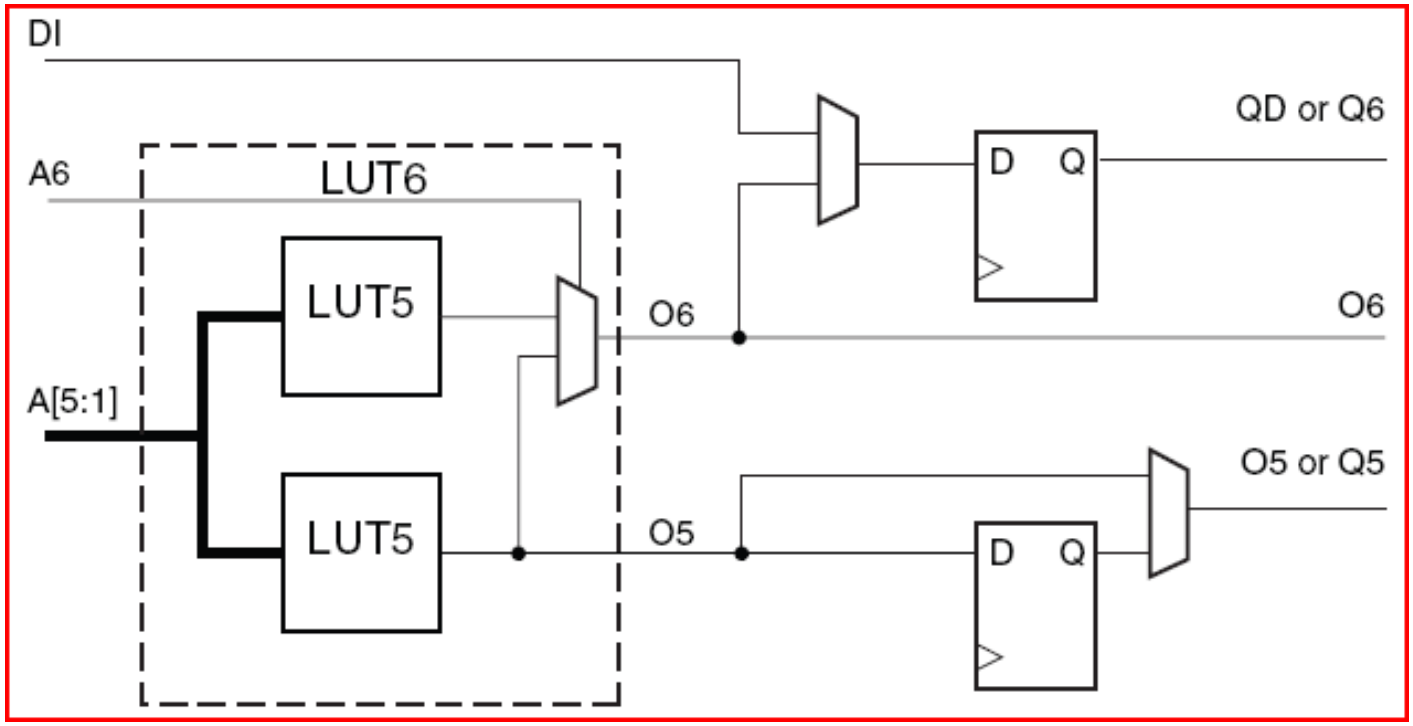
LUTs are not configurable as RAMs or SRLs and no carry logic



Spartan-6 FPGA Logic Resources

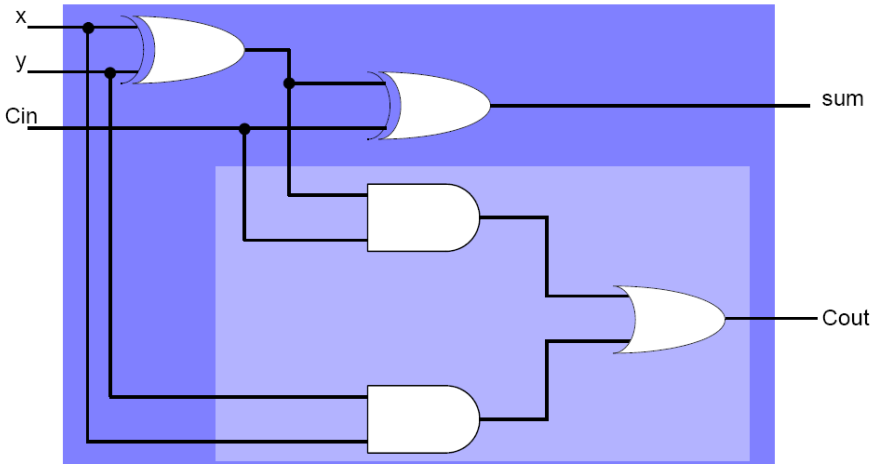
Device	Logic Cells	Total Slices	SLICEMs	SLICELs	SLICEXs	Number of 6-Input LUTs	Maximum Distributed RAM (Kb)	Shift Registers (Kb)	Number of Flip-Flops
XC6SLX4	3,840	600	300	0	300	2,400	75	38	4,800
XC6SLX9	9,152	1,430	360	355	715	5,720	90	45	11,440
XC6SLX16	14,579	2,278	544	595	1,139	9,112	136	68	18,224
XC6SLX25	24,051	3,758	916	963	1,879	15,032	229	115	30,064
XC6SLX45	43,661	6,822	1,602	1,809	3,411	27,288	401	200	54,576
XC6SLX75	74,637	11,662	2,768	3,063	5,831	46,648	692	346	93,296
XC6SLX100	101,261	15,822	3,904	4,007	7,911	63,288	976	488	126,576
XC6SLX150	147,443	23,038	5,420	6,099	11,519	92,152	1,355	678	184,304
XC6SLX25T	24,051	3,758	916	963	1,879	15,032	229	115	30,064
XC6SLX45T	43,661	6,822	1,602	1,809	3,411	27,288	401	200	54,576
XC6SLX75T	74,637	11,662	2,768	3,063	5,831	46,648	692	346	93,296
XC6SLX100T	101,261	15,822	3,904	4,007	7,911	63,288	976	488	126,576
XC6SLX150T	147,443	23,038	5,420	6,099	11,519	92,152	1,355	678	184,304

Using a LUT6 as a pair of LUT5



The function generators in Spartan-6 FPGAs are implemented as six-input look-up tables (LUTs). There are six independent inputs (A1 to A6) and two independent outputs (O5 and O6) for each of the four function generators in a slice (A, B, C, and D). The function generators can implement any arbitrarily defined six-input Boolean function. Each function generator can also implement two arbitrarily defined five-input Boolean functions, as long as these two functions share common inputs.

Using a LUT as a full adder



x	y	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

address

0	1	2	3	4	5	6	7
00	10	10	01	10	01	01	11

To implement a full-adder, we need a RAM with 8 2-bits words

SLICEM

Distributed RAM

Distributed RAM is fast, localized and ideal for small data buffers, FIFOs or register files. For larger memory requirements, consider using the 18Kb block RAM resources.

ROM

Each function generator can implement a 64 x 1-bit ROM. Three configurations are available: ROM64x1 (one LUT), ROM128x1 (two LUTs) and ROM256x1 (4 LUTs). ROM contents are loaded at each device configuration.

Shift register (SRL32)

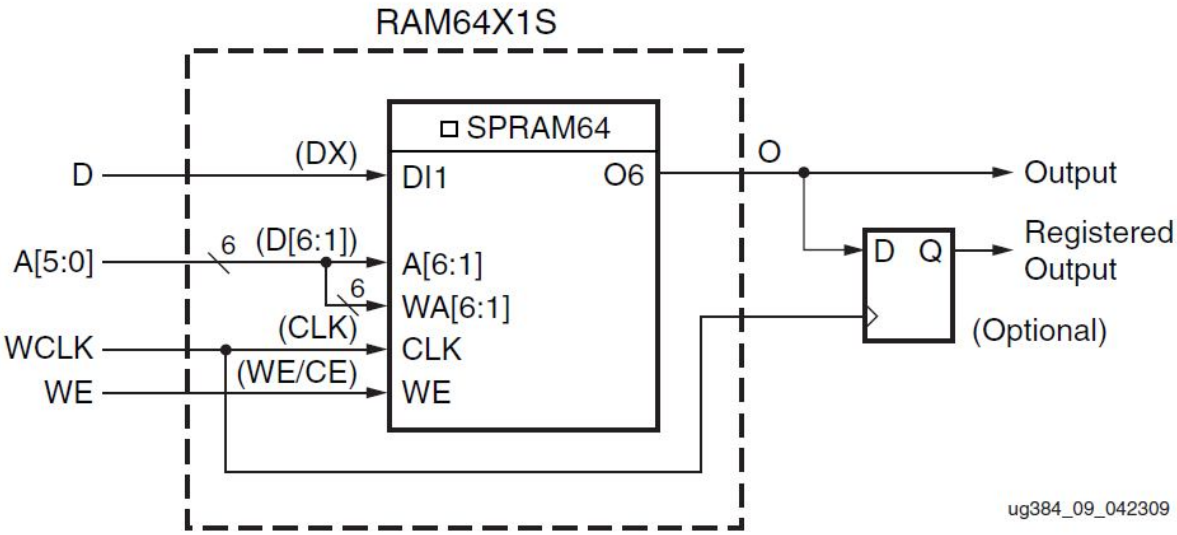
A SLICEM function generator can also be configured as a 32-bit shift register without using the flip-flops available in a slice. Used in this way, each LUT can delay serial data anywhere from one to 32 clock cycles.

Multiplexers

Function generators and associated multiplexers in SLICEL or SLICEM can implement the following:

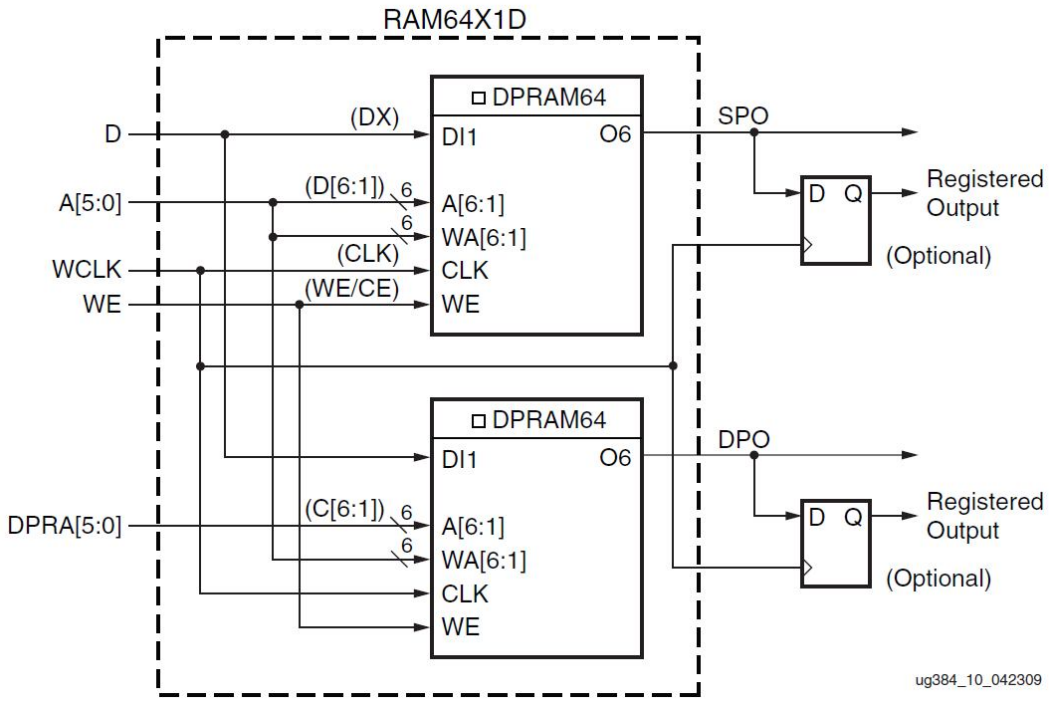
- 4:1 multiplexers using one LUT
- 8:1 multiplexers using two LUTs
- 16:1 multiplexers using four LUTs

Using a LUT as a RAM



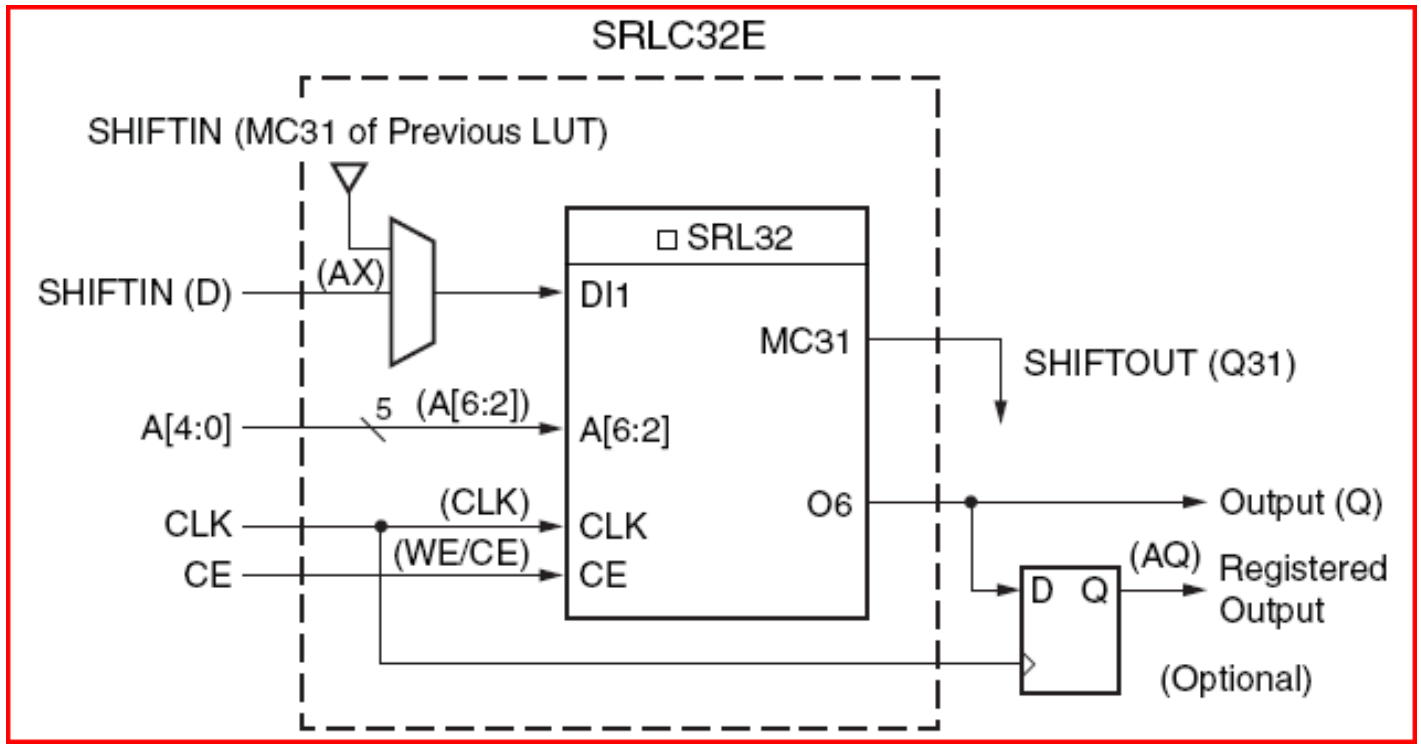
Single-Port
64 x 1-bit RAM

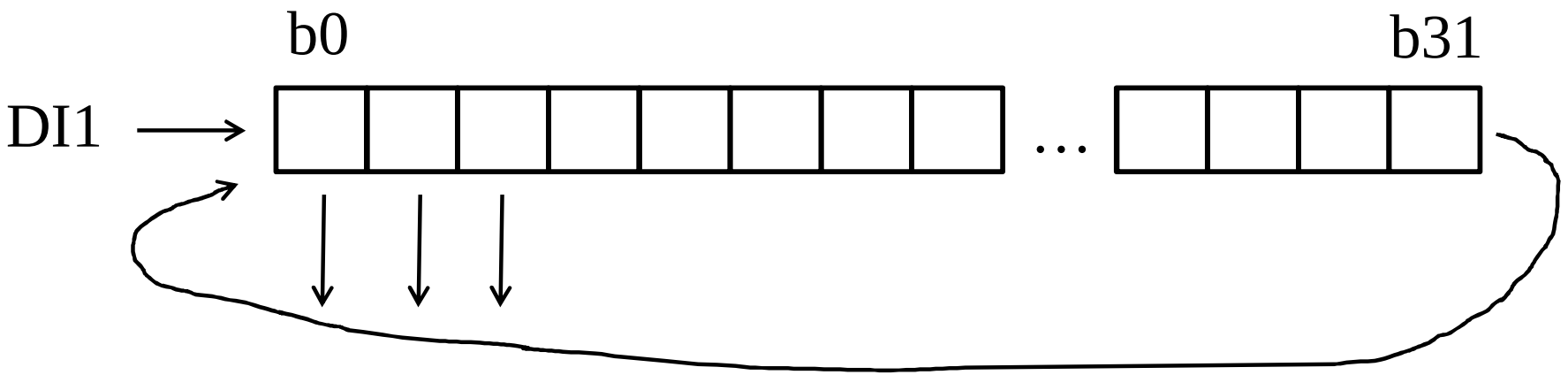
Dual-Port
64 x 1-bit RAM



Using a LUT as a shift register

LUT6

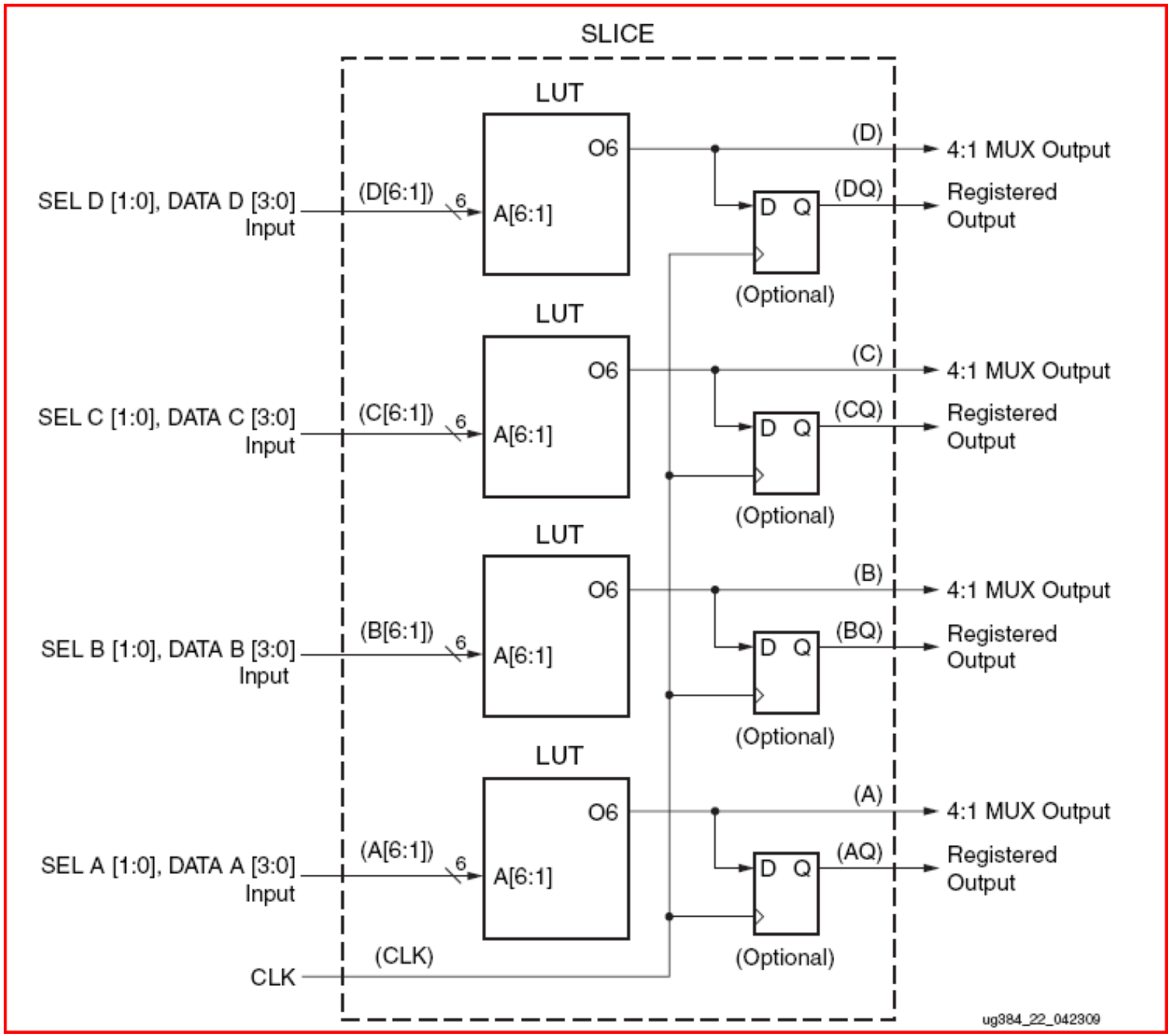




inserting a programmable latency on the input data using 1 LUT + 1 FF

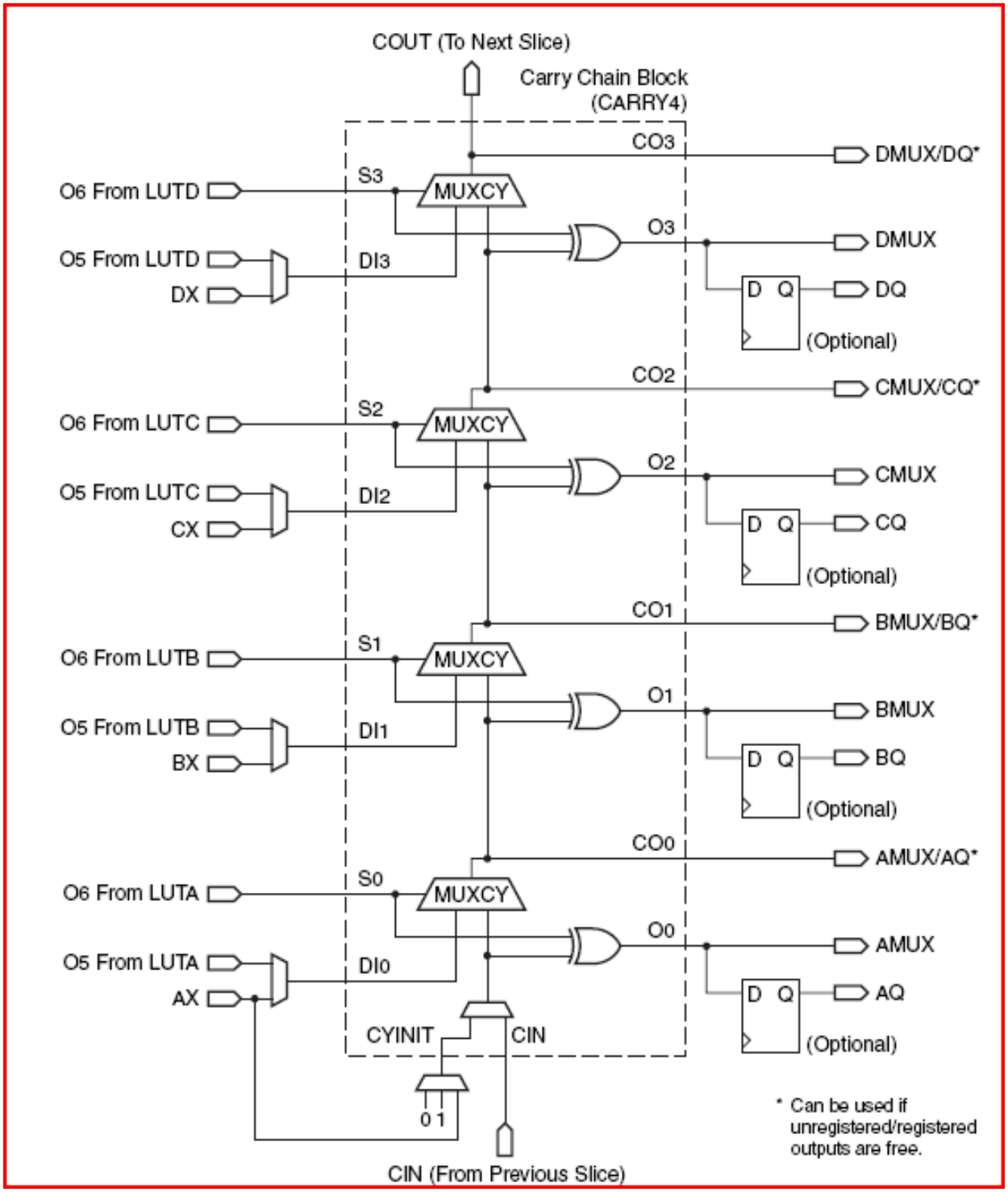
Using a LUT as a multiplexer

four 4:1 multiplexers in a slice



Fast Carry Logic Path

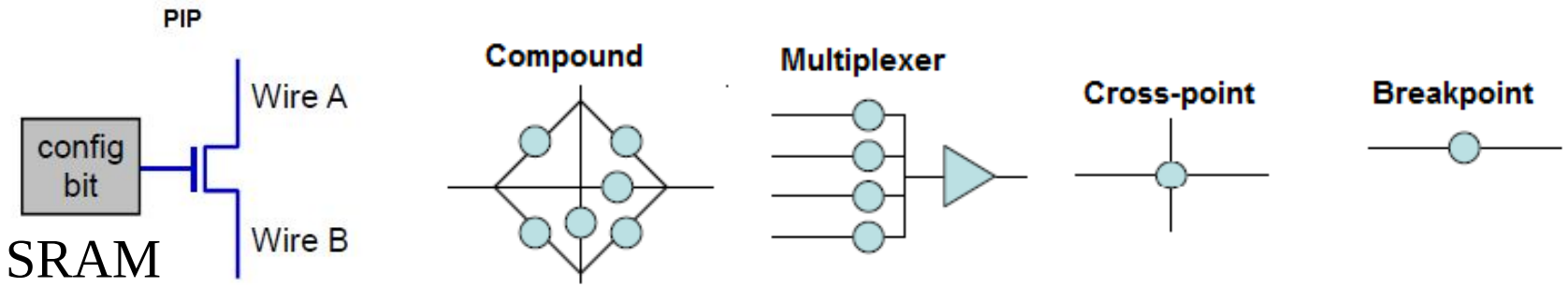
```
 1 1111  <--- Carry bits
  101111
+ 101101
--------
 1011100
```



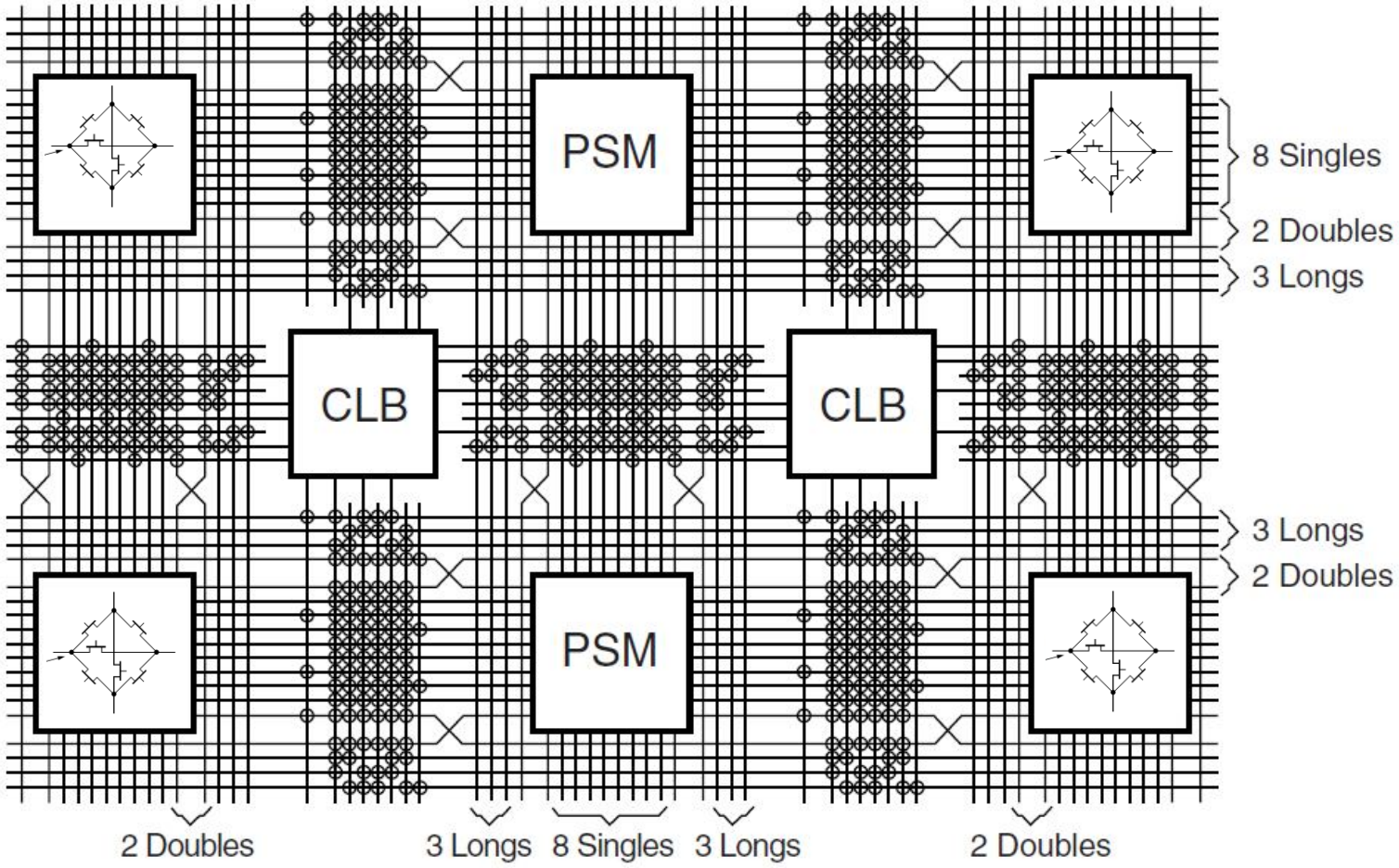
FPGA Programmable Interconnection Network

- Horizontal and vertical mesh of wire segments interconnected by programmable switches called programmable interconnect points (**PIPs**). These PIPs are implemented using a transmission gate controlled by a memory bits from the configuration memory.

- Consists of global routing connecting CLBs to I/O buffers, non-adjacent CLBs and other embedded components. Local routing connects CLBs to other adjacent CLBs and CLBs to global routing (done through a switch matrix)

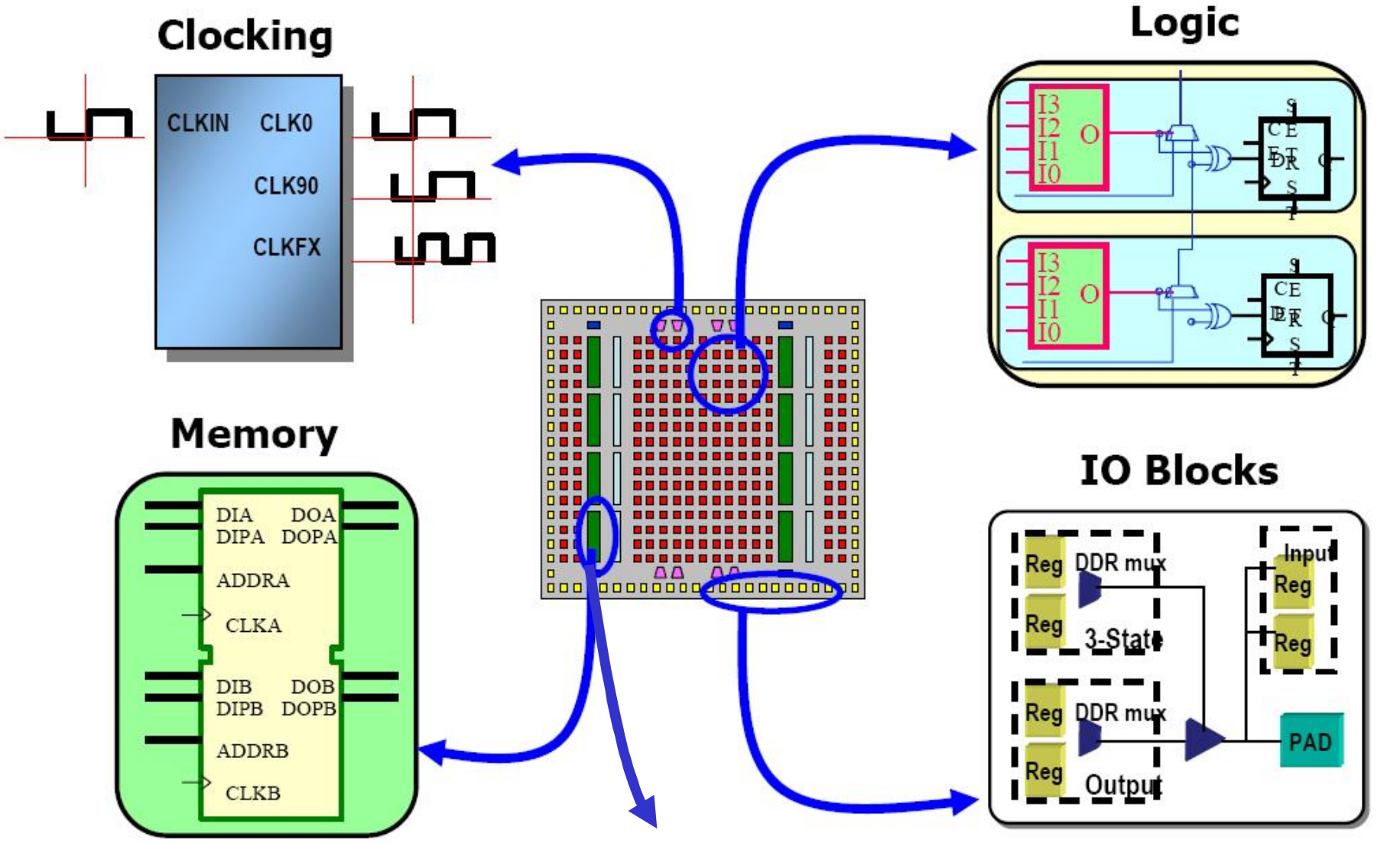


FPGA Programmable Interconnection Network



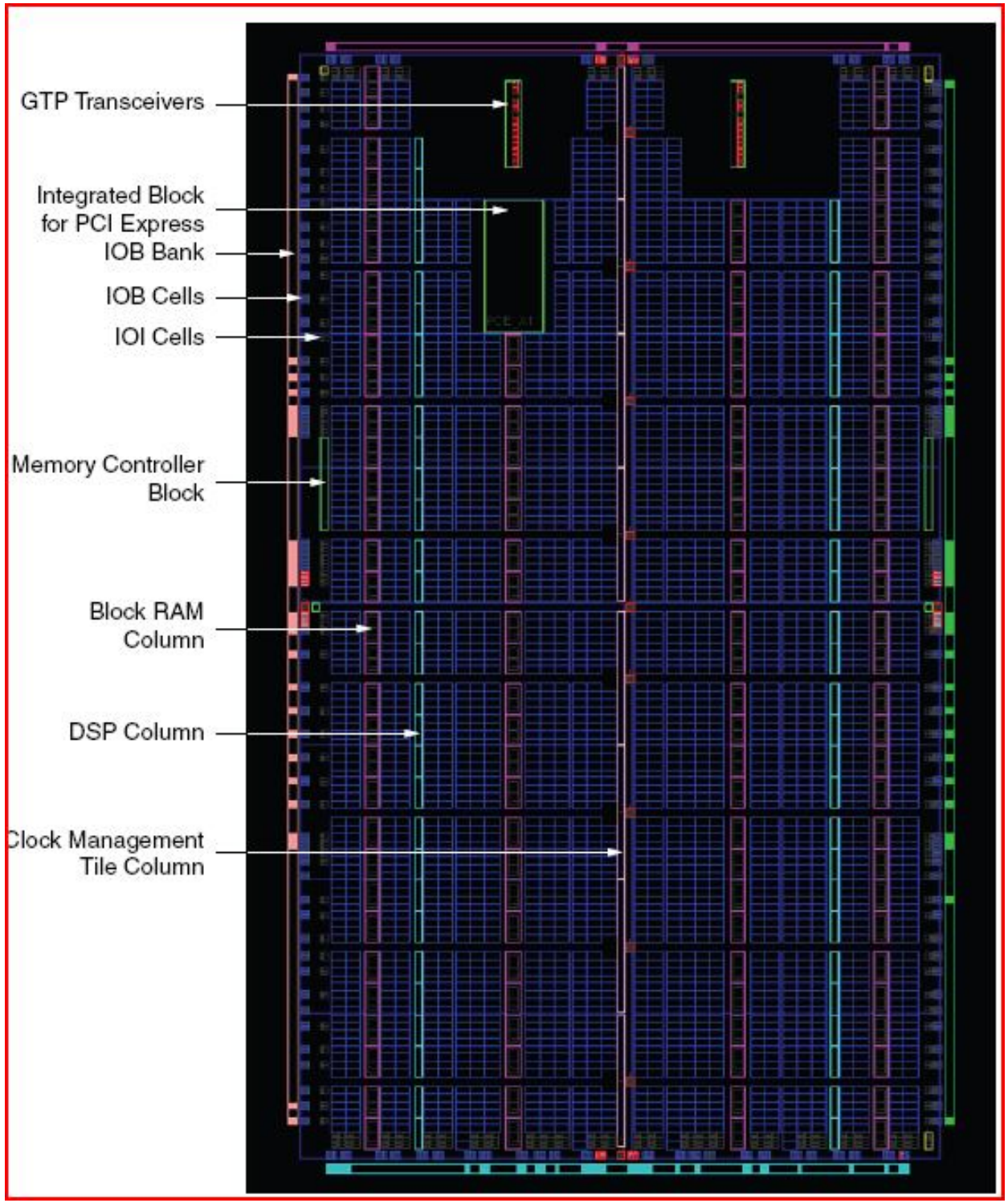
The horizontal and vertical single- and double-length lines intersect at a box called a programmable switch matrix (**PSM**). Each PSM consists of programmable pass transistors used to establish connections between the lines

Not only logic on larger and newer devices



DSP48E **MULTIPLIER + ACCUMULATOR**

Spartan6 XC6SLX45T floorplan view in PlanAhead



Global Networks

GSR: A separate Global Set/Reset line sets or clears each flip-flop during power-up, reconfiguration or when a dedicated Reset net is driven active. This global net does not compete with other routing resources; it uses a dedicated distribution network. GSR can be driven active from any user-programmable pin as a global reset input.

GTS: A separate Global 3-state line (GTS) forces all FPGA outputs to the high-impedance state. GTS does not compete with other routing resources; it uses a dedicated distribution network. GTS can be driven from any user-programmable pin as a global 3-state input.

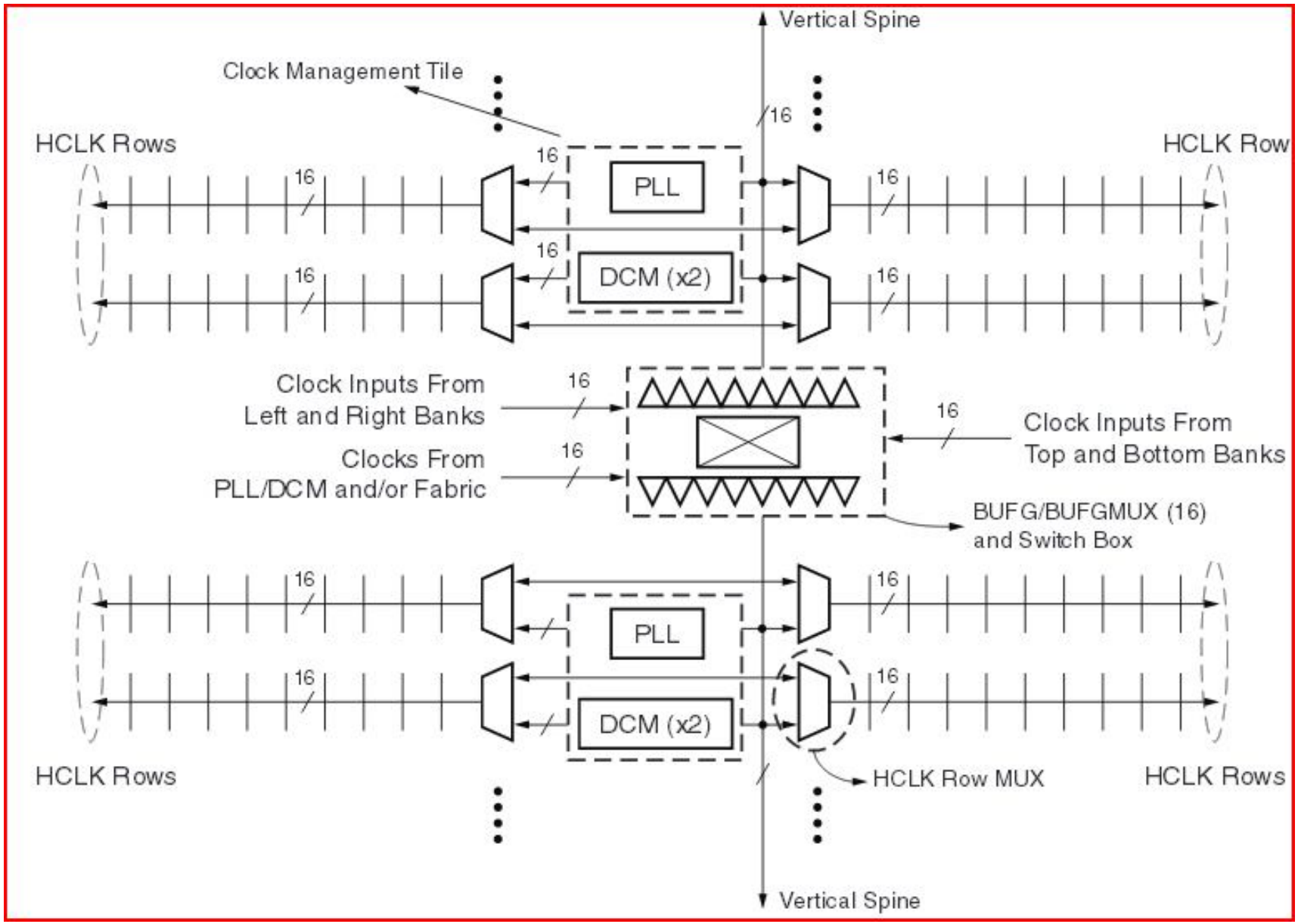
Clocks: see next slides

FPGA clock distribution

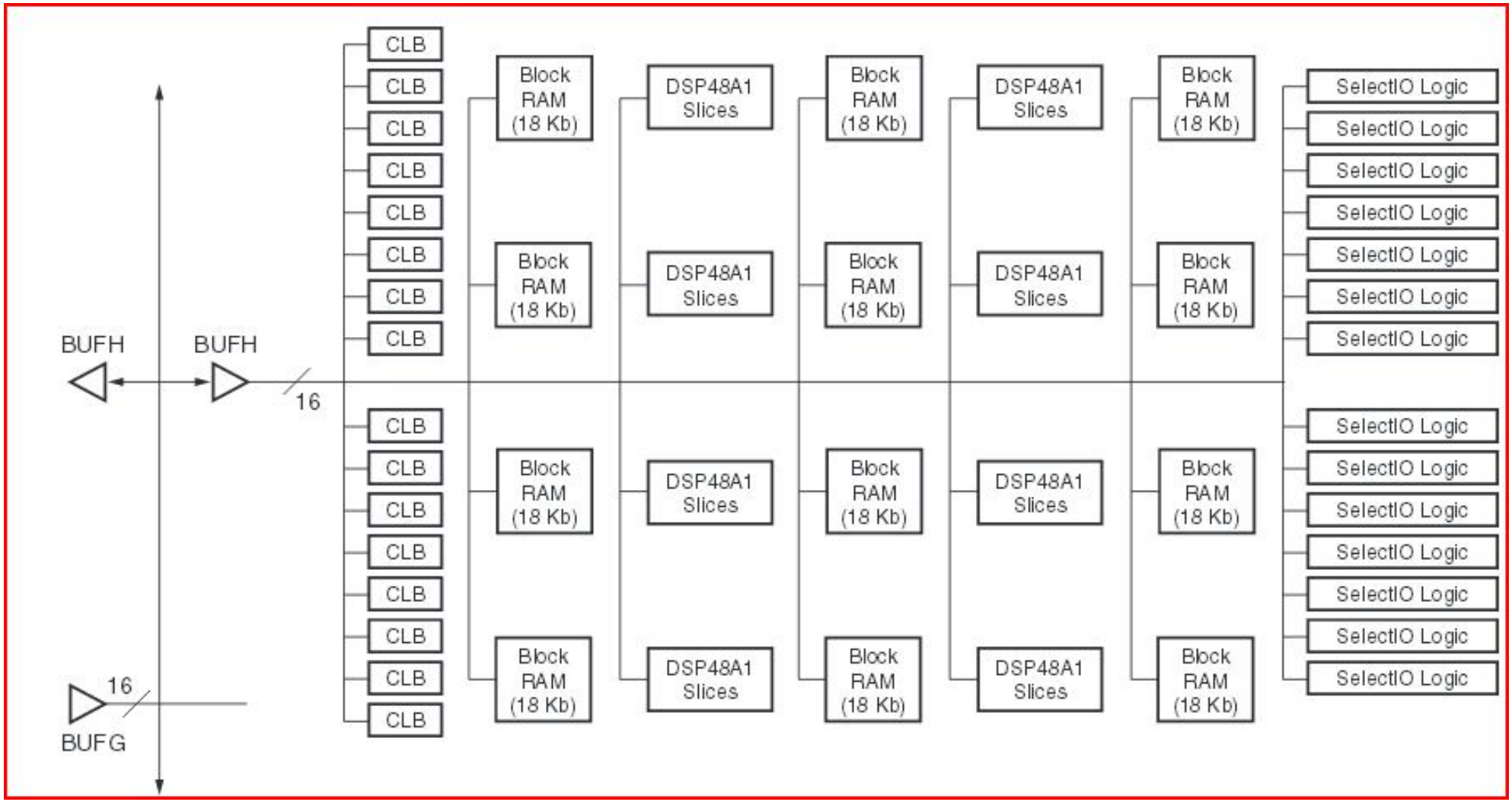
- **Global clock network:**
  - providing low-skew clock routing to the FPGA logic resources
  - each Spartan-6 FPGA device offers 16 high-speed, low-skew global clock resources to optimize performance
  - Global clock input pads (GCLK)
  - Global clock multiplexers (BUFG, BUFGMUX)
  - Horizontal clock routing buffers (BUFH)

- **Regional clock lines:**
  - providing high-performance low-skew clocking to the IO logic resources
  - each Spartan-6 FPGA provides 40 ultra high-speed, low-skew I/O regional clock resources (32 BUFIO2s and eight BUFPLLs) to serve localized I/O serializer/de-serializer (ISERDES and OSERDES) circuits.
  - I/O clock buffers (BUFIO2, BUFIO2_2CLK, BUFPLL)

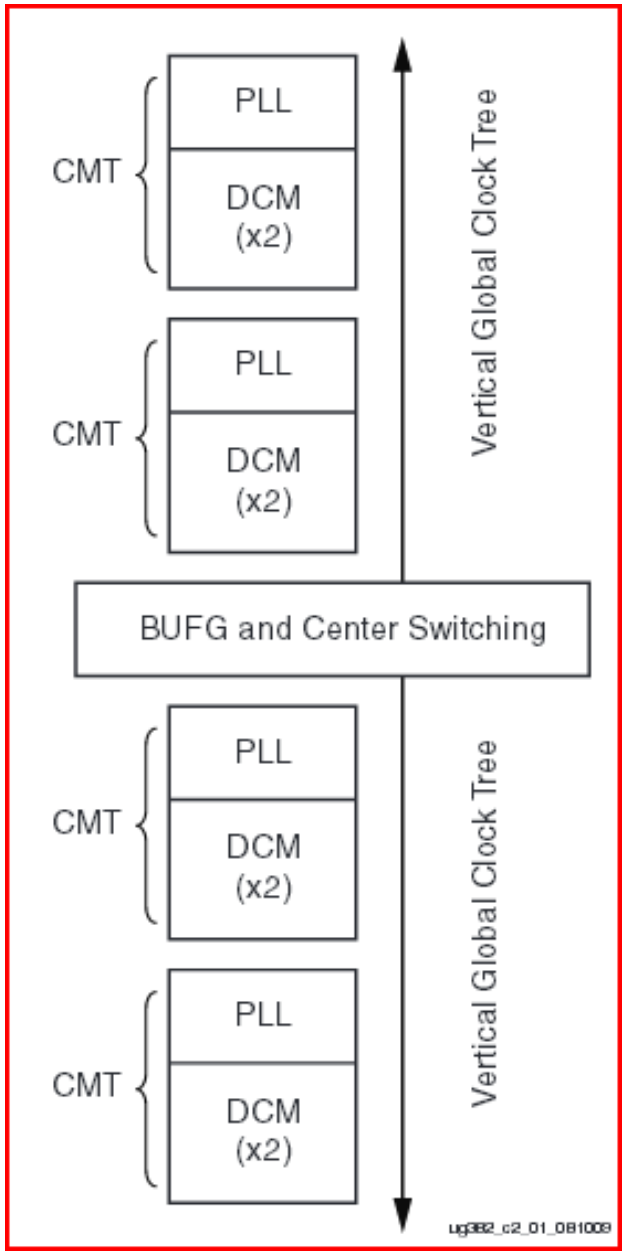
FPGA clock distribution



FPGA clocking resources



FPGA clocking resources

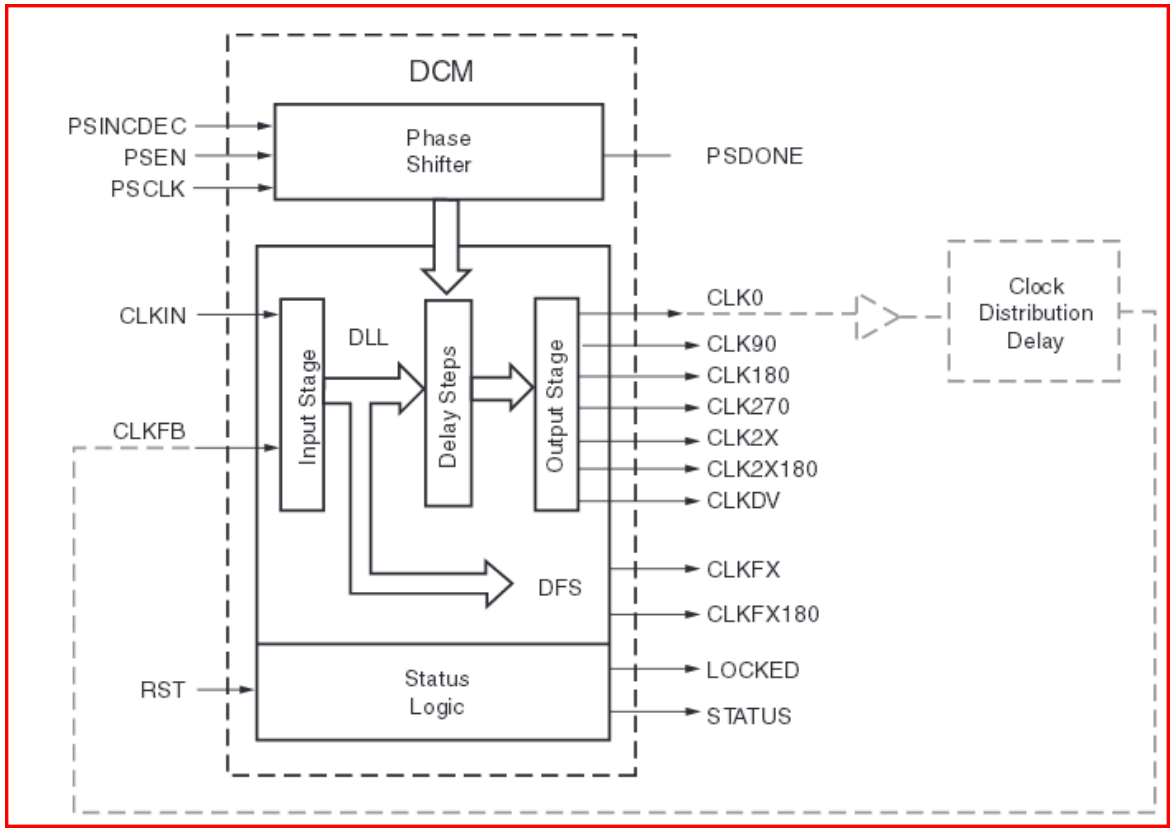


- **DCM**
  - Ease of design porting
  - Superior phase shift capability

- **PLL**
  - Higher operating frequency
  - Lower output jitter
  - Faster LOCK times

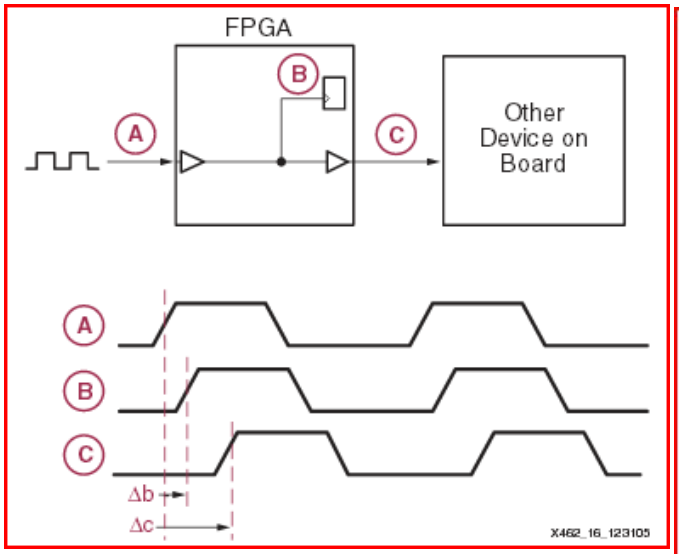
The DCM requires less area in silicon, thus we have more DCMs than PLLs. They are digital and greatly immune to power supply noise (as compared to PLLs). They are a state machine, so a "crummy clock" signal is NOT recommended, as a DCM will have a hard time, where a PLL is commonly used to "clean up" a "crummy clock" in order to make it useful at all. DCM will add totally random, white, broad band jitter, where a PLL will remove high frequency jitter, while adding low frequency jitter (due to its oscillator having very low Q on the silicon). It will also add as jitter any power supply noise present varies the VCO in the PLL.

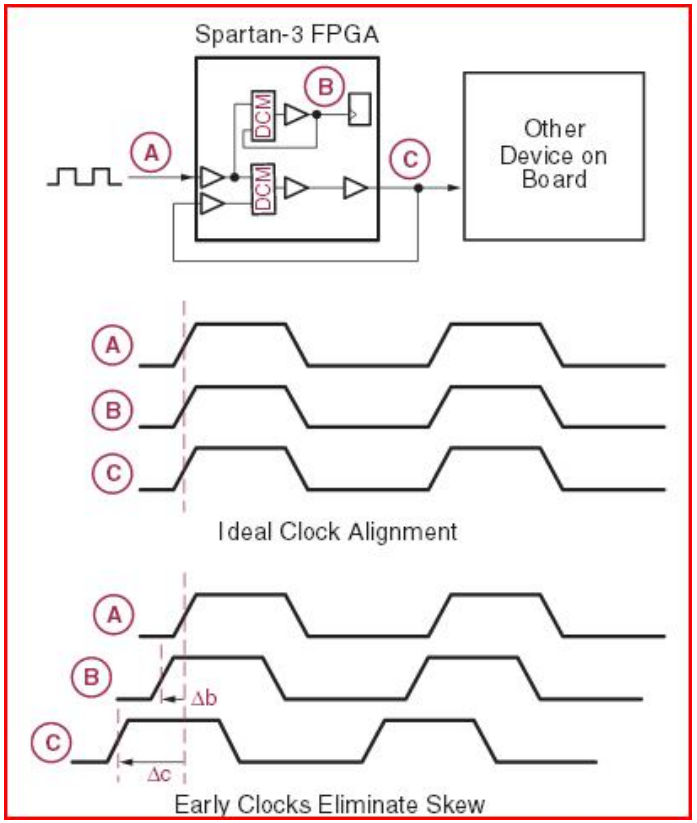
FPGA clocking resources: DCM

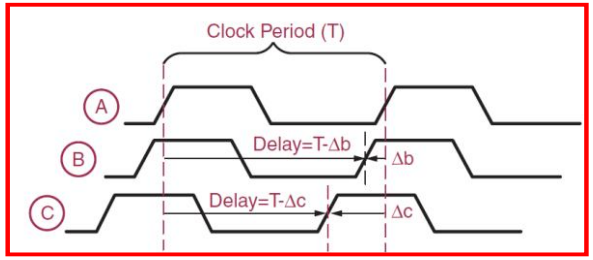


• **Eliminate clock skew**, either within the device or to external components, to improve overall system performance and to eliminate clock distribution delays.
• **Phase shift a clock signal**, either by a fixed fraction of a clock period or by incremental amounts.
• Multiply or divide an incoming clock frequency or synthesize a completely new frequency by a mixture of static or dynamic clock multiplication and division.
• Condition a clock, ensuring a clean output clock with a 50% duty cycle.
• Mirror, forward, or rebuffer a clock signal, often to deskew and convert the incoming clock signal to a different I/O standard. For example, forwarding and converting an incoming LVTTL clock to LVDS.
• Free-running oscillator

Eliminating clock skew with a DCM



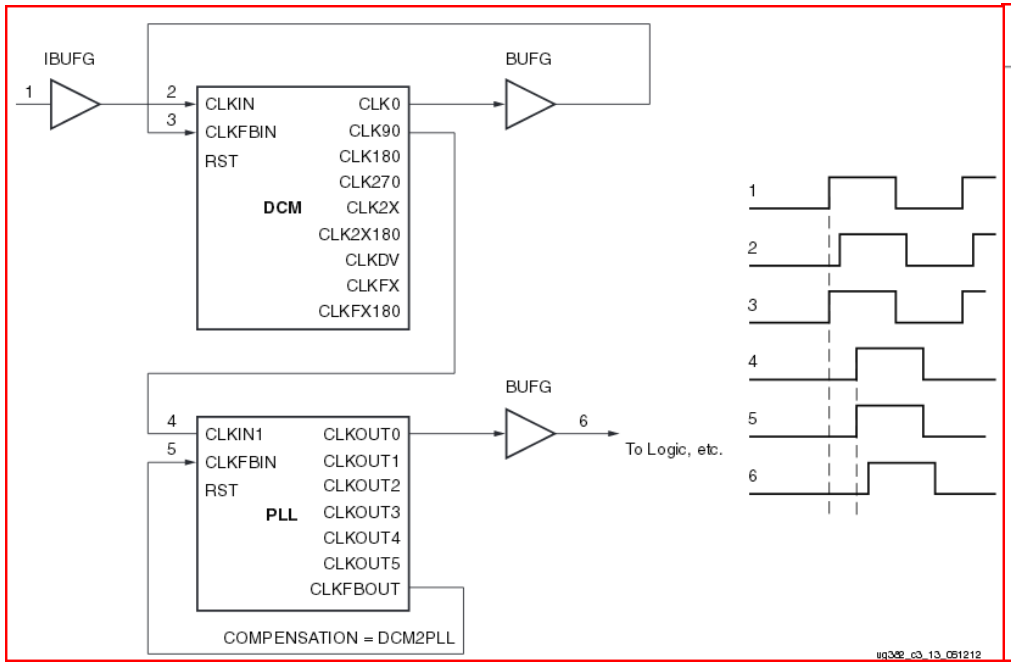




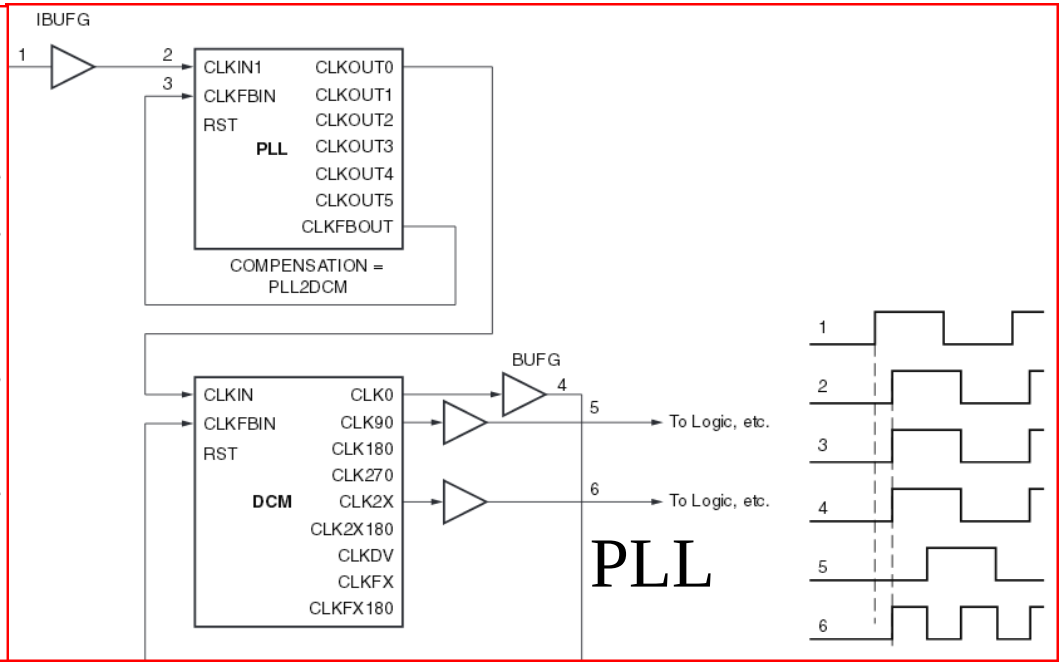
Two DCMs eliminate the clock skew: one DCM eliminates the skew for clocked items within the FPGA, the other DCM eliminates the skew when clocking the other device on the board. The result is practically ideal alignment between the clock at Points (A), (B), and (C). The Spartan6 DCM employs a Delay-Locked Loop (DLL) that constantly monitors the delay via a feedback loop. The DLL also constantly adapts to subtle changes caused by temperature and voltage.

FPGA clocking resources: PLL

The main purpose of a PLLs is to serve as a frequency synthesizer for a wide range of frequencies and to serve as a jitter filter for either external or internal clocks in conjunction with the DCMs

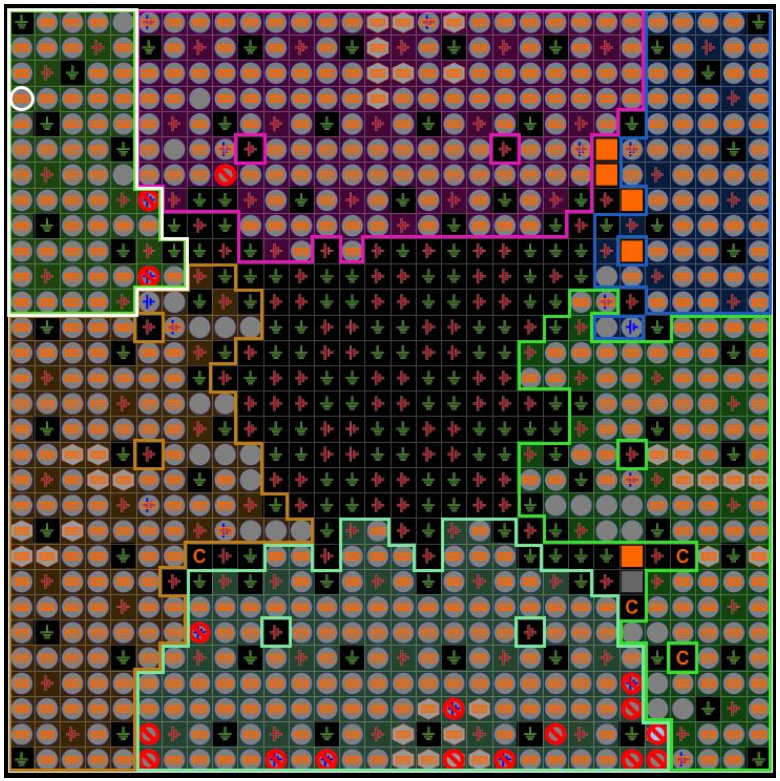


The PLL can be used to reduce the output jitter of one DCM clock output. The PLL is configured to not introduce any phase shift (zero delay through the PLL).

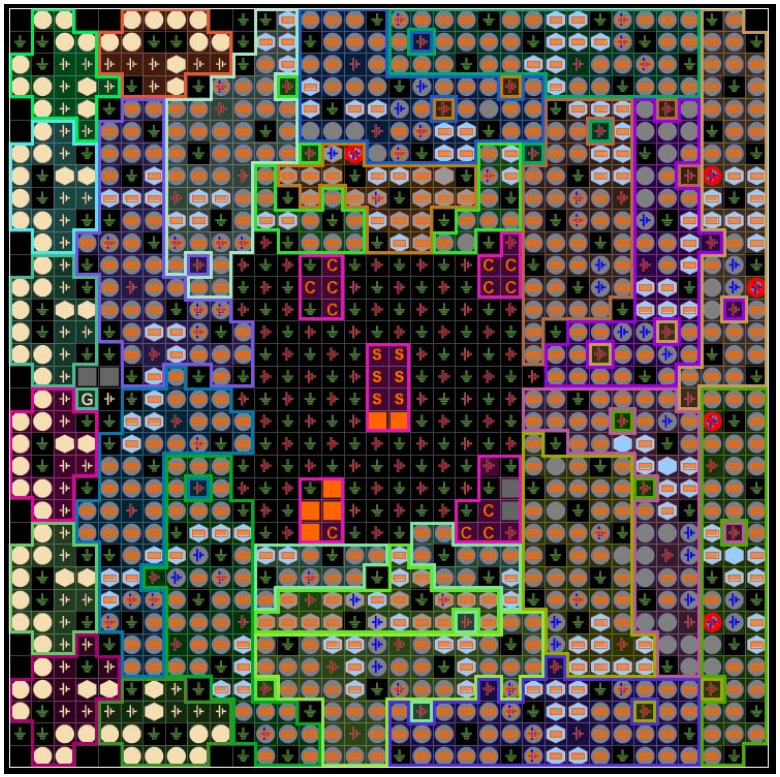


A second option for reduce clock jitter is to use the PLL to clean-up the input clock jitter before driving the DCM. This will improve the output jitter of all DCM outputs, but any added jitter by the DCM will still be passed to the clock outputs.

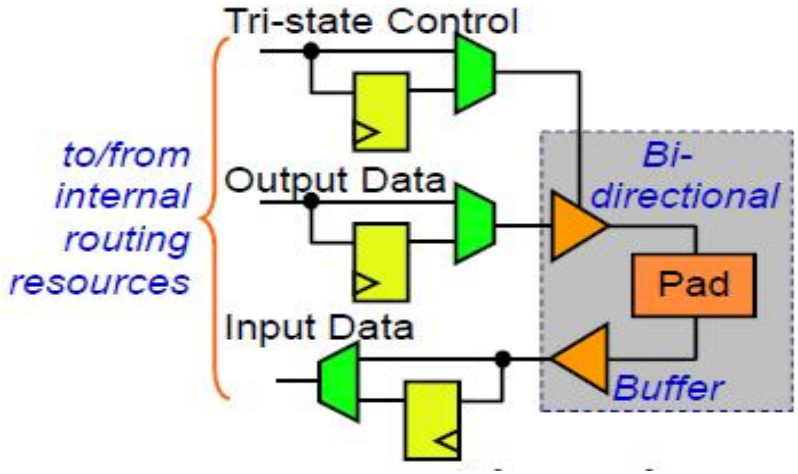
FPGA banks + IOBs



Spartan6: 6 banks



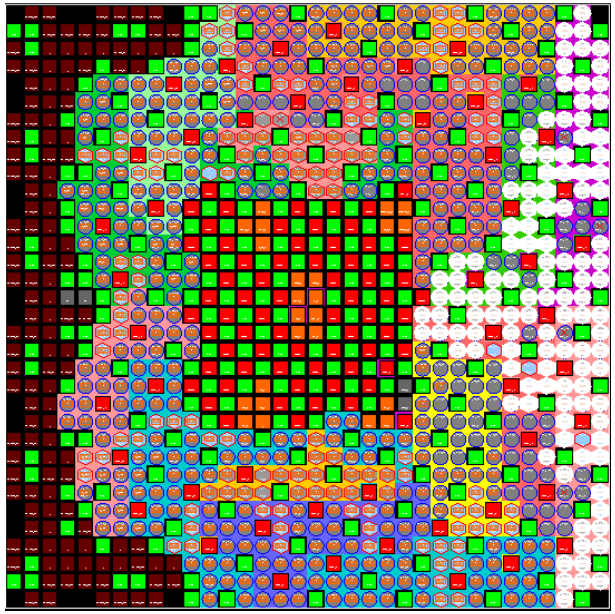
Virtex5: 19 banks



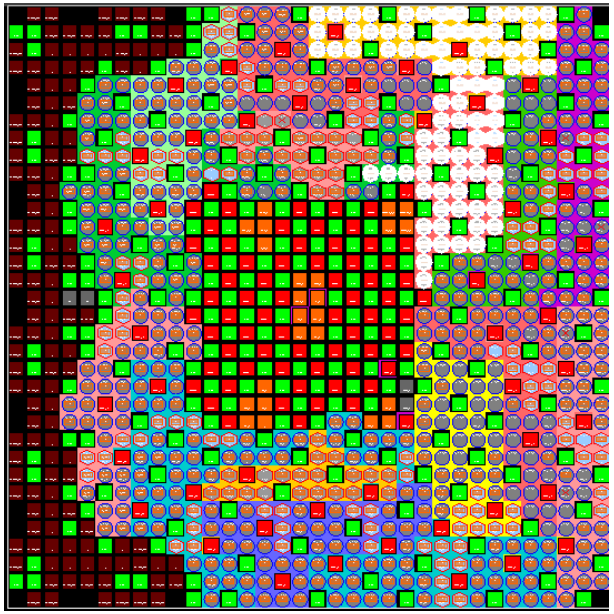
Each user I/O can be configured as either **input**, **output** or **bidirectional**

Virtex5 pin planning example

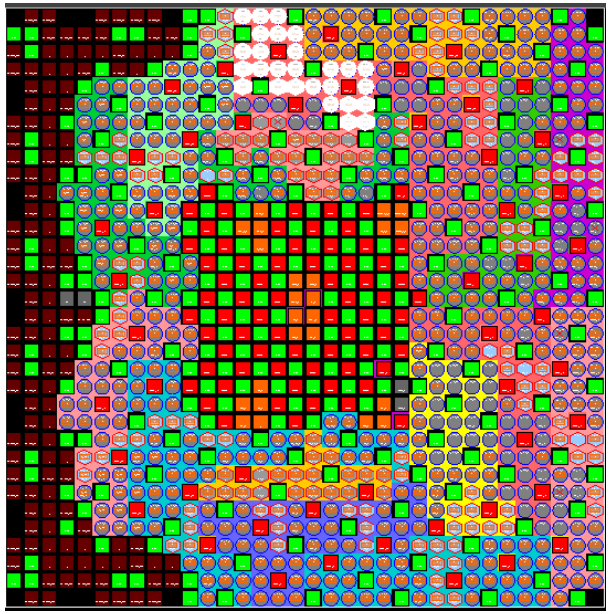
- Many small banks: 19 (20 or 40 pins each)
- Relaxed constraint on SSOs (Simultaneous switching Outputs)



SODIMM DDR2



DSP interface



ETH

Ground or power bounce occurs when a large number of outputs simultaneously switch in the same direction.

FPGA power supply

VCCINT
Internal core supply voltage. Supplies all internal logic functions, such as CLBs, block RAM, and DSP blocks. Powers input signals for most standards at 1.0V, 1.2V, 1.5V and 1.8V.

VCCAUX
Auxiliary supply voltage. Supplies clock management tiles (CMTs), some I/O resources, some dedicated configuration pins and JTAG interface. Powers input signals for most standards at 2.5V and 3.3V.

VCCO
Supplies the output buffers in I/O bank.

VREF
Input threshold voltage pins when HSTL/SSTL standards are used in the bank, otherwise user I/Os. When used as a reference voltage within a bank, all VREF pins within that bank must be connected.

IOBs

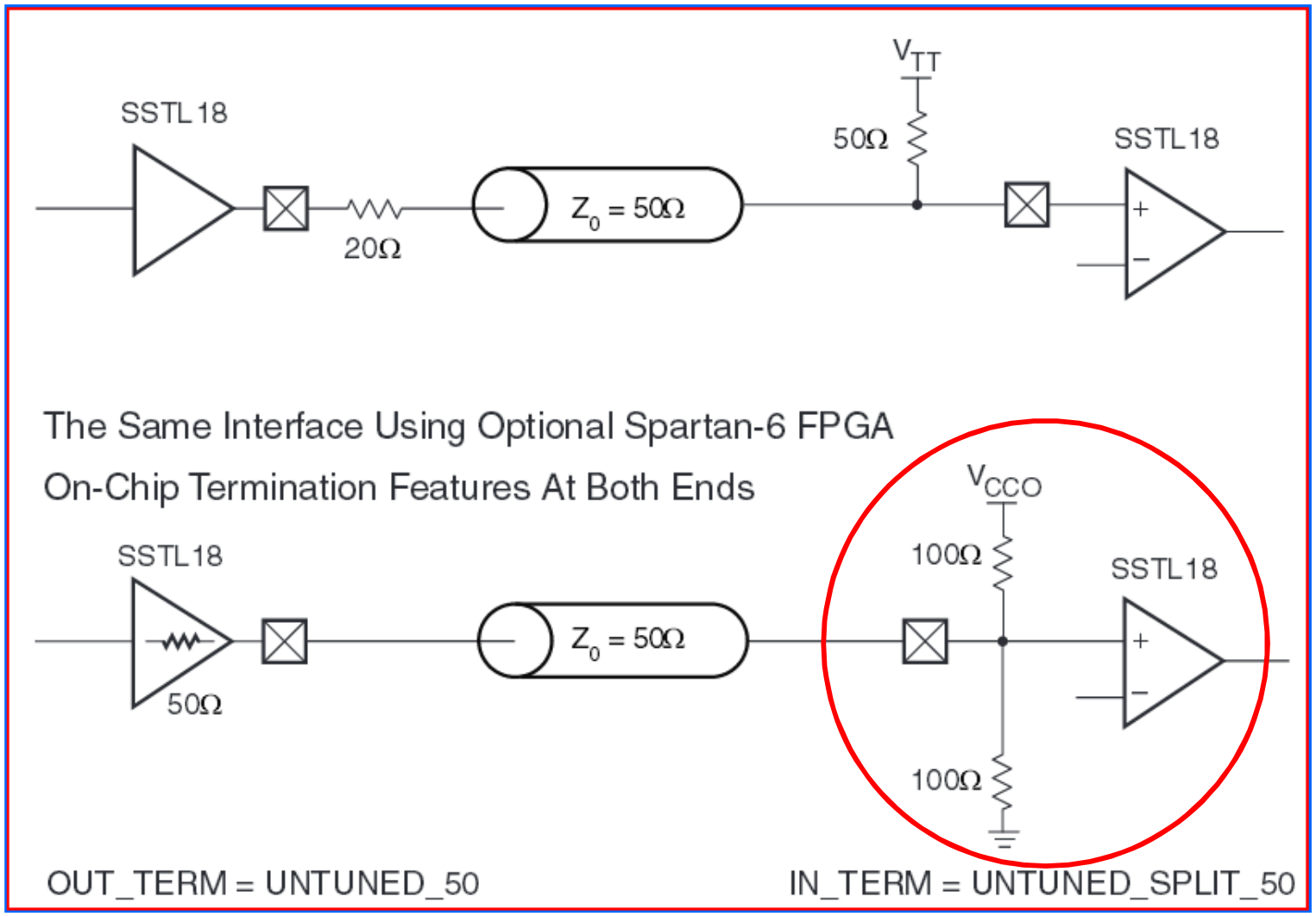
I/O electrical features: programmable control of output strength and slew rate and on-chip termination

Digital Controlled Impedance (DCI): The 3-state Digitally Controlled Impedance can control the output drive impedance (series termination) or can provide parallel termination of an input signal to VCCO or split (Thevenin) termination to VCCO/2. This allows users to eliminate off-chip termination. In addition to board space savings, the termination automatically turns off when in output mode or when 3-stated, saving considerable power compared to off-chip termination.

Input/Output Delay: any input and some outputs can be individually delayed by up to 32 increments of 78 ps or 52 ps each. Such delays are implemented as IDELAY and ODELAY. The number of delay steps can be set by configuration and can also be incremented or decremented while in use.

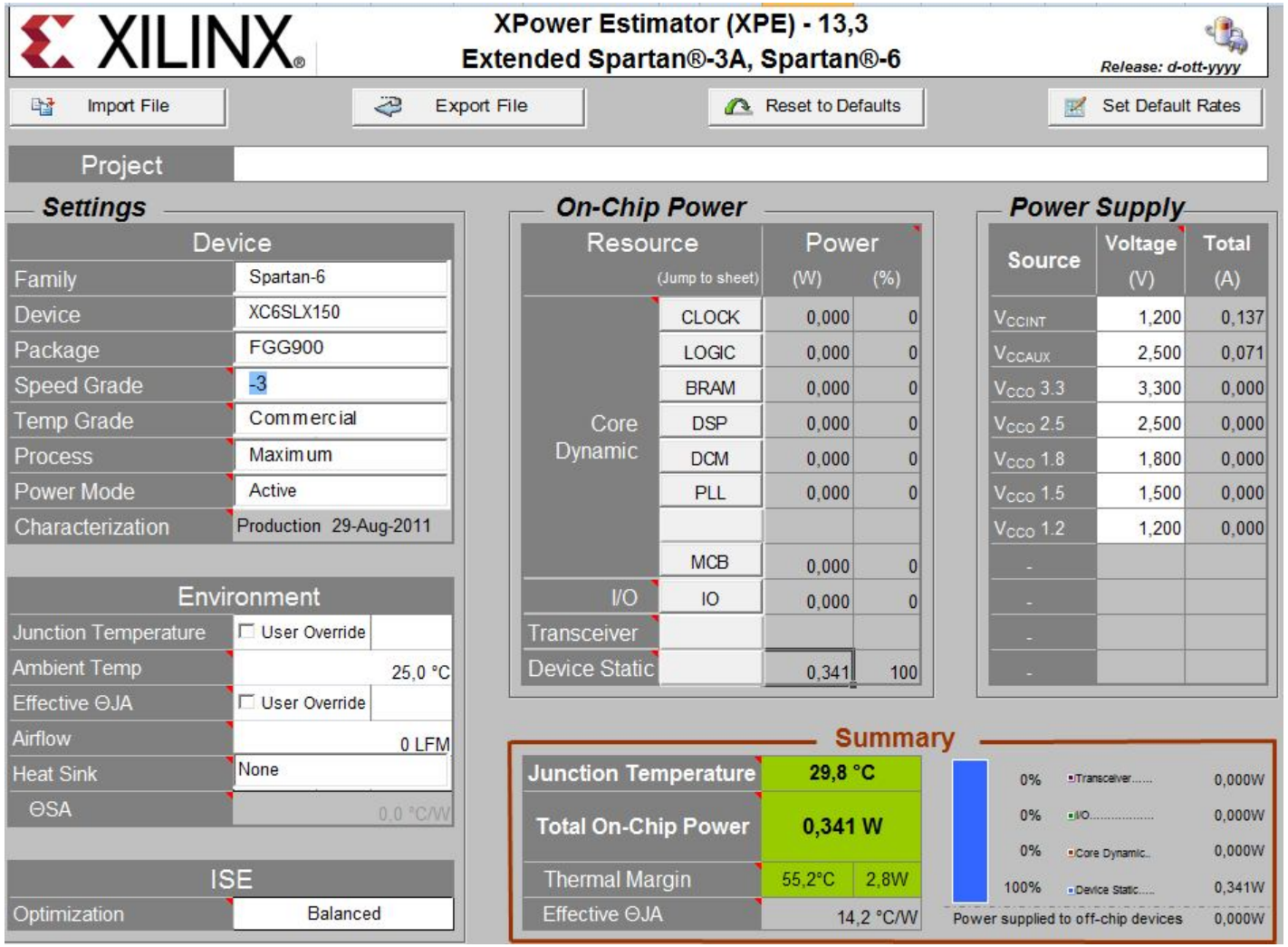
ISERDES / OSERDES: Many applications combine high-speed, bit-serial I/O with slower parallel operation inside the device. This requires a serializer and deserializer (SerDes) inside the I/O structure. Each I/O pin possesses an 8-bit IOSERDES (ISERDES and OSERDES) capable of performing serial-to-parallel or parallel-to-serial conversions with programmable widths of 2, 3, 4, 5, 6, 7, or 8 bits. By cascading two IOSERDES from two adjacent pins (default from differential I/O), wider width conversions of 10 and 14 bits can also be supported.

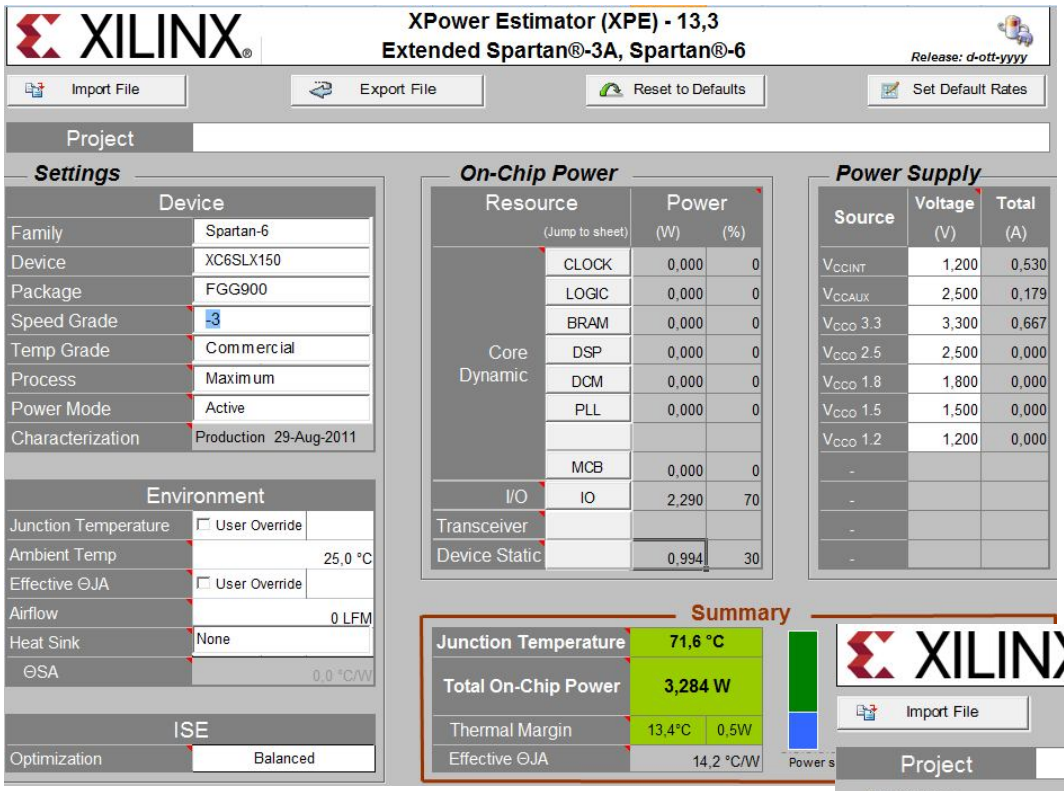
On chip termination



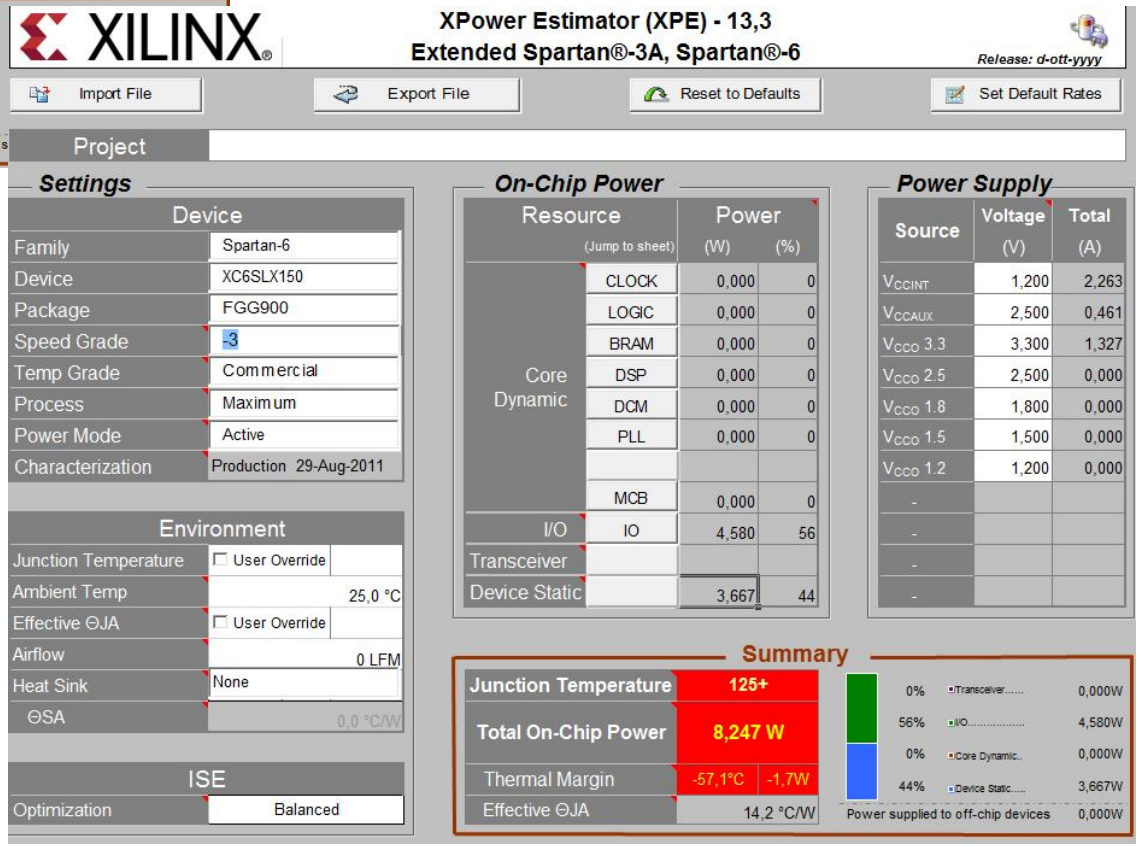
Pay attention to the static current increase with internal terminations!

Power estimation without using on chip termination





when using 40 on-chip terminated inputs

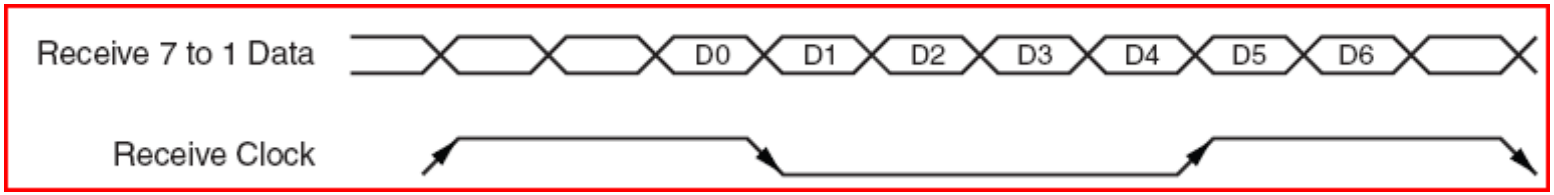


when using 80 on-chip terminated inputs … too much!

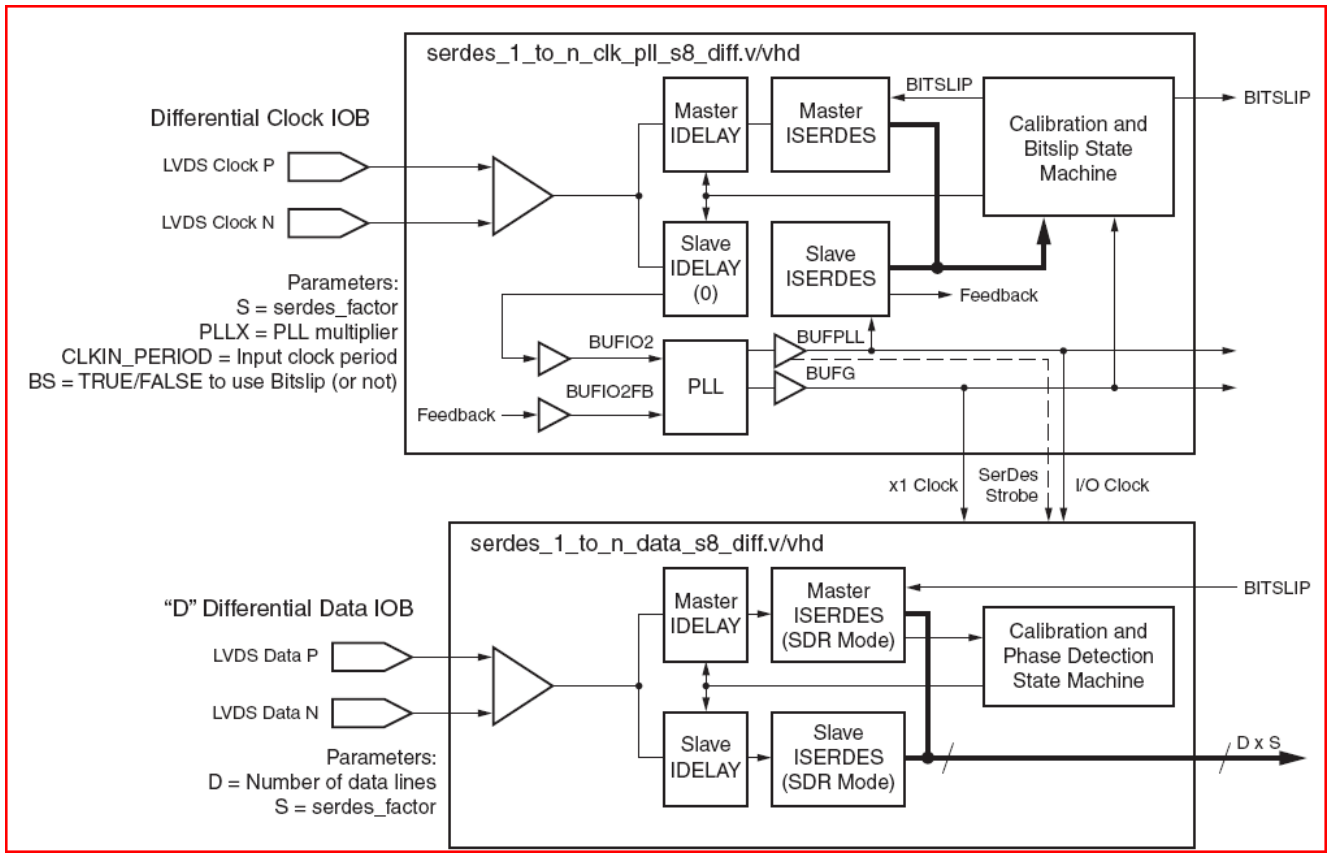
Die temperature should be lower than 85C for correct behavior, but never over 125C !!!

Data Reception Using PLL and BUFPLL (ISERDES)

Spartan-6 FPGAs perform in a wide variety of applications requiring various serialization and deserialization factors up to 16-to-1, at speeds up to 1050 Mb/s, depending on the application, speed grade, and package

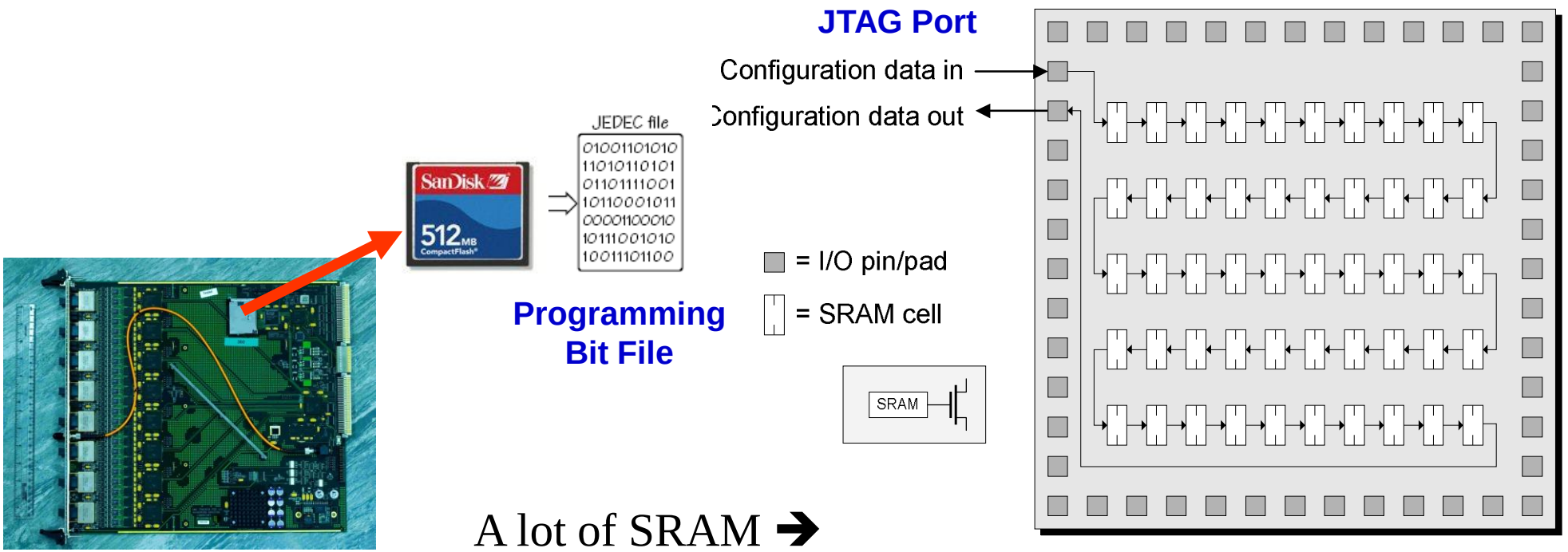


A 150 MHz input clock with accompanying 7:1 data requires the PLL and BUFPLL to operate at 1050 MHz. This high-speed capture clock is used to clock the receive data into the input deserializers and is capable of driving one whole edge of a device. Parallel data is then presented to the FPGA logic at the speed of the original incoming clock



Configuring an FPGA

- Millions of SRAM cells holding LUTs and Interconnect Routing
- SRAM is a volatile memory: it loses configuration when board power is turned off
- Keep Bit Pattern describing the SRAM cells in non-volatile memory, e.g. PROM or Digital Camera card
- Configuration takes ~ seconds after power-on



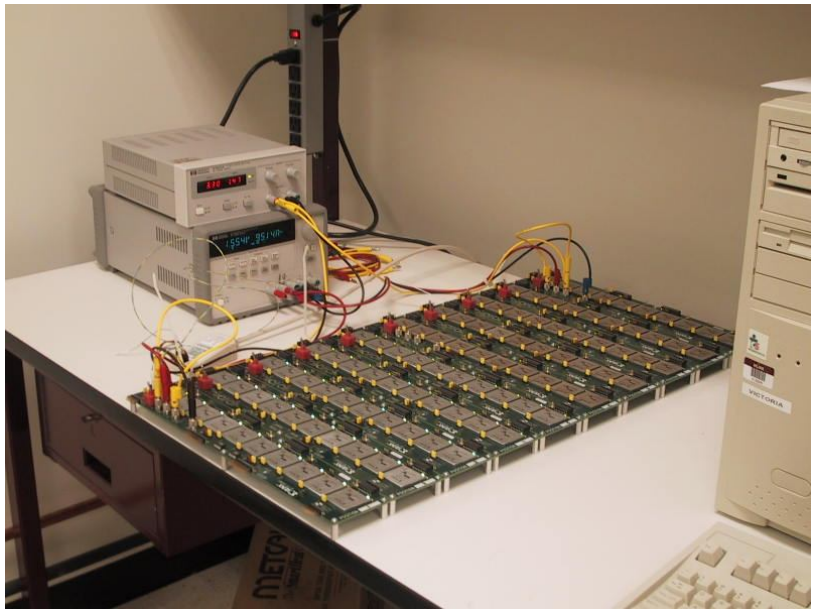
A lot of SRAM ➔

potentially dangerous in a radiation environment !

SEU detection

Spartan-6 devices include a feature to perform continuous readback of configuration data in the background of a user design. This feature is aimed at simplifying detection of single event upsets (SEUs) that cause a configuration memory bit to flip. Detected failures appear either on a device pin (INIT_B) and/or on an internally accessible component, POST_CRC_INTERNAL. The clock source of the readback can be external or internally generated.

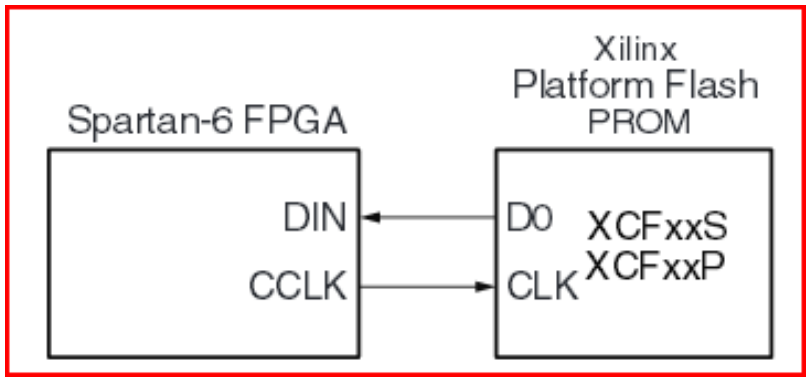
The expected “golden” CRC value is calculated by the software and written into the FPGA for later comparison. The subsequent scans of Readback CRC value are compared against the golden value.



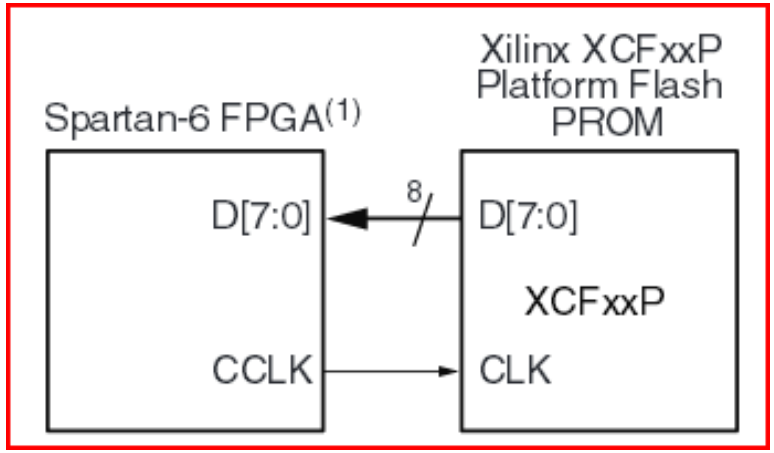
Measured mean time between SEUs in XC2V6000 at sea level is 18 to 23 years

(with 95% confidence)

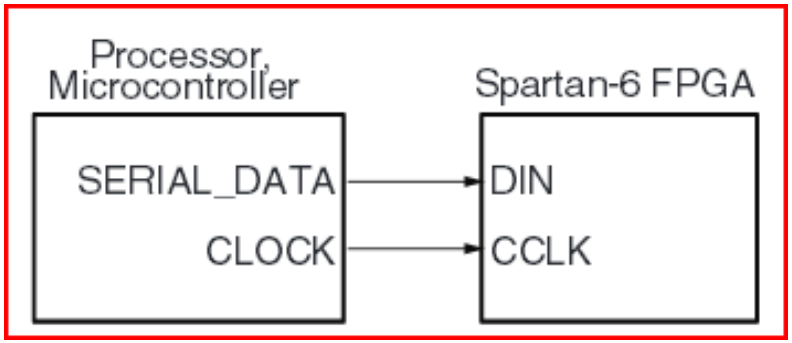
Configuration modes



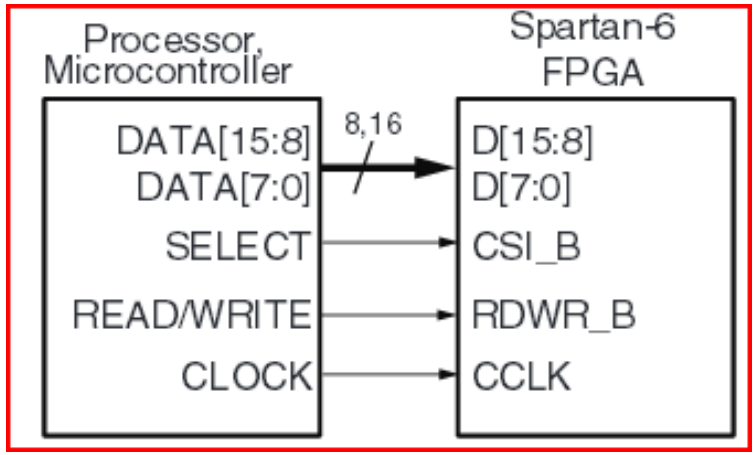
master serial mode



master selectMAP mode

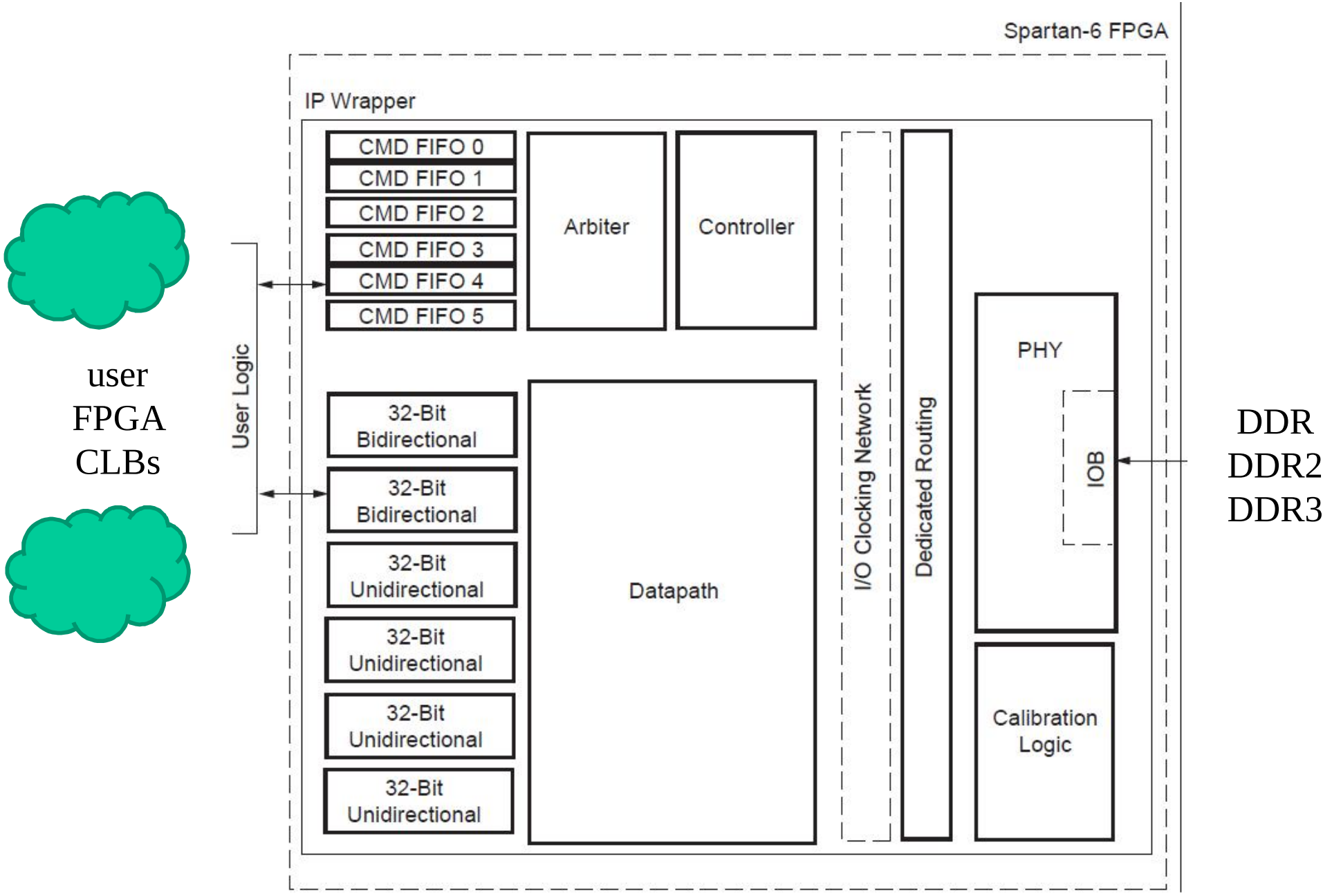


slave serial mode

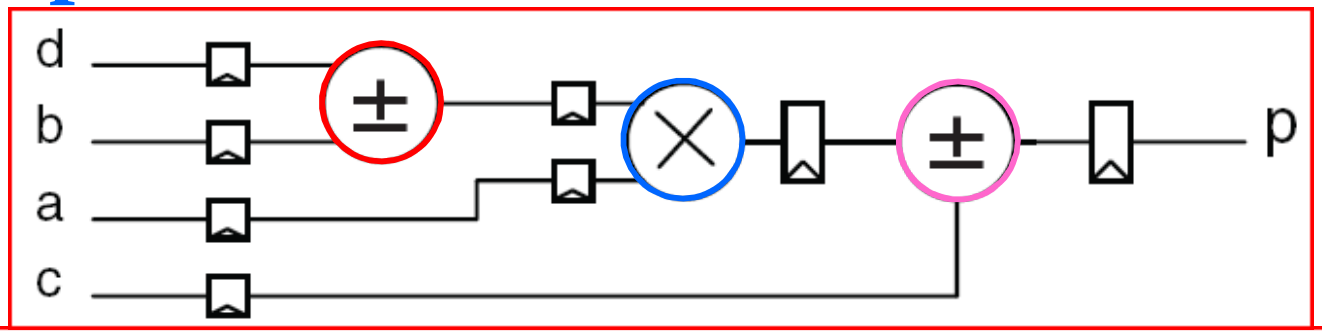


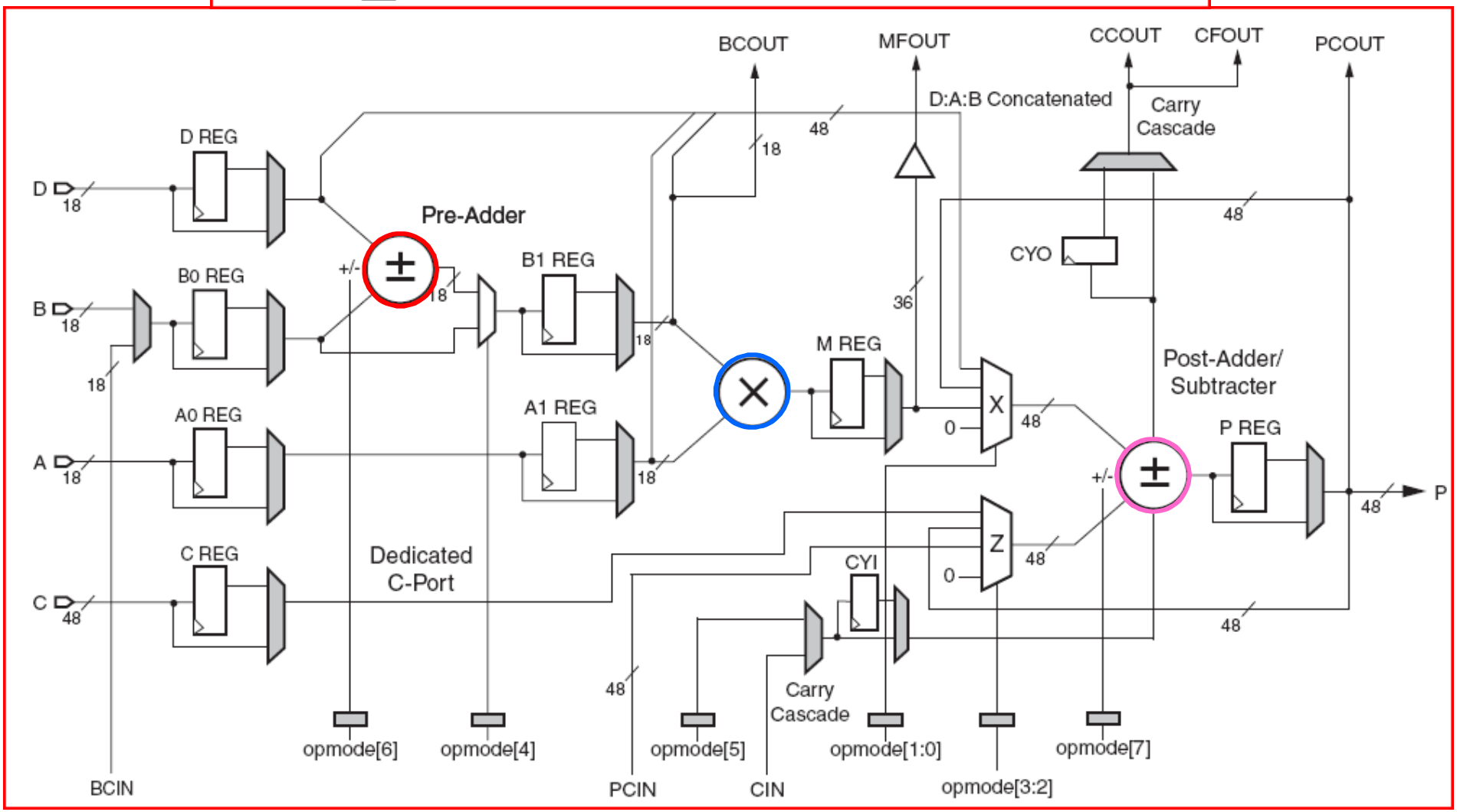
slave selectMAP mode

Spartan-6 FPGA Memory Controller Block



Spartan6 FPGA DSP48A1 slice





Spartan6 FPGA DSP48A1 slice

The DSP48A1 slices support many independent functions, including multiplier, multiplier-accumulator (MACC), pre-adder/subtracter followed by a multiply accumulator, multiplier followed by an adder, wide bus multiplexers, magnitude comparator, or wide counter. The architecture also supports connecting multiple DSP48A1 slices to form wide math functions, DSP filters, and complex arithmetic without the use of general FPGA logic.

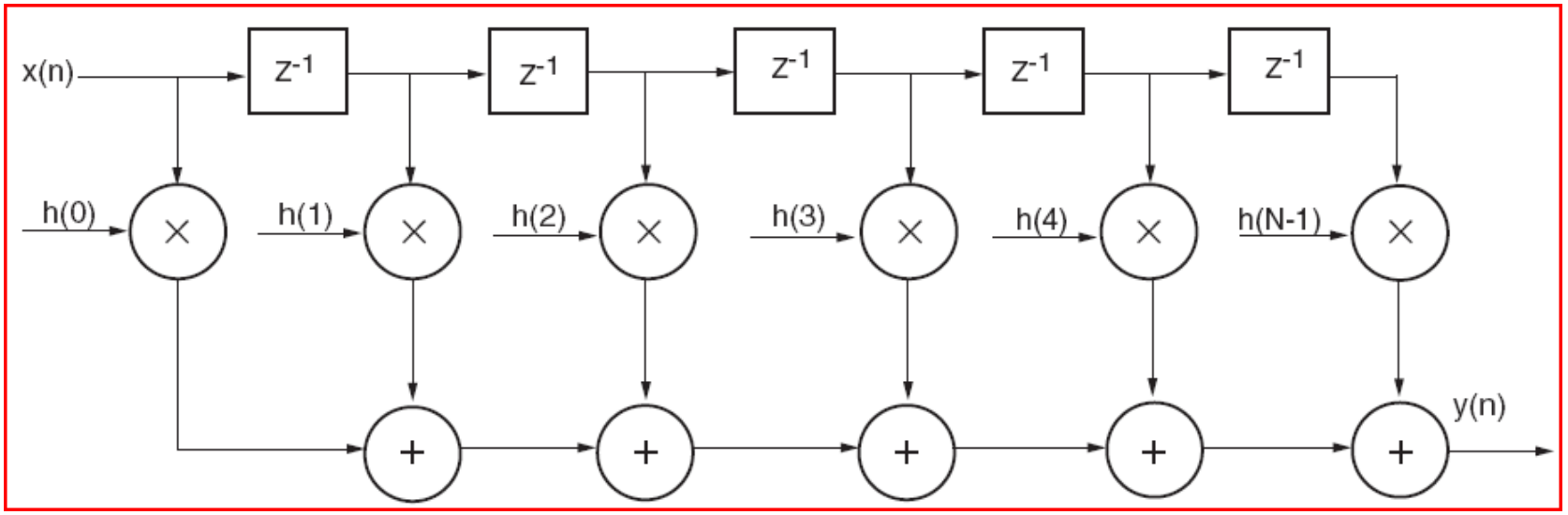
One of the most important features is the ability to cascade a result from one DSP48A1 slice to the next without the use of general fabric routing.

Device	Total DSP48A1 Slices per Device	Number of DSP48A1 Columns per Device
XC6SLX4	8	1
XC6SLX9	16	1
XC6SLX16	32	2
XC6SLX25	38	2
XC6SLX45	58	2
XC6SLX75	132	3
XC6SLX100	180	4
XC6SLX150	180	4

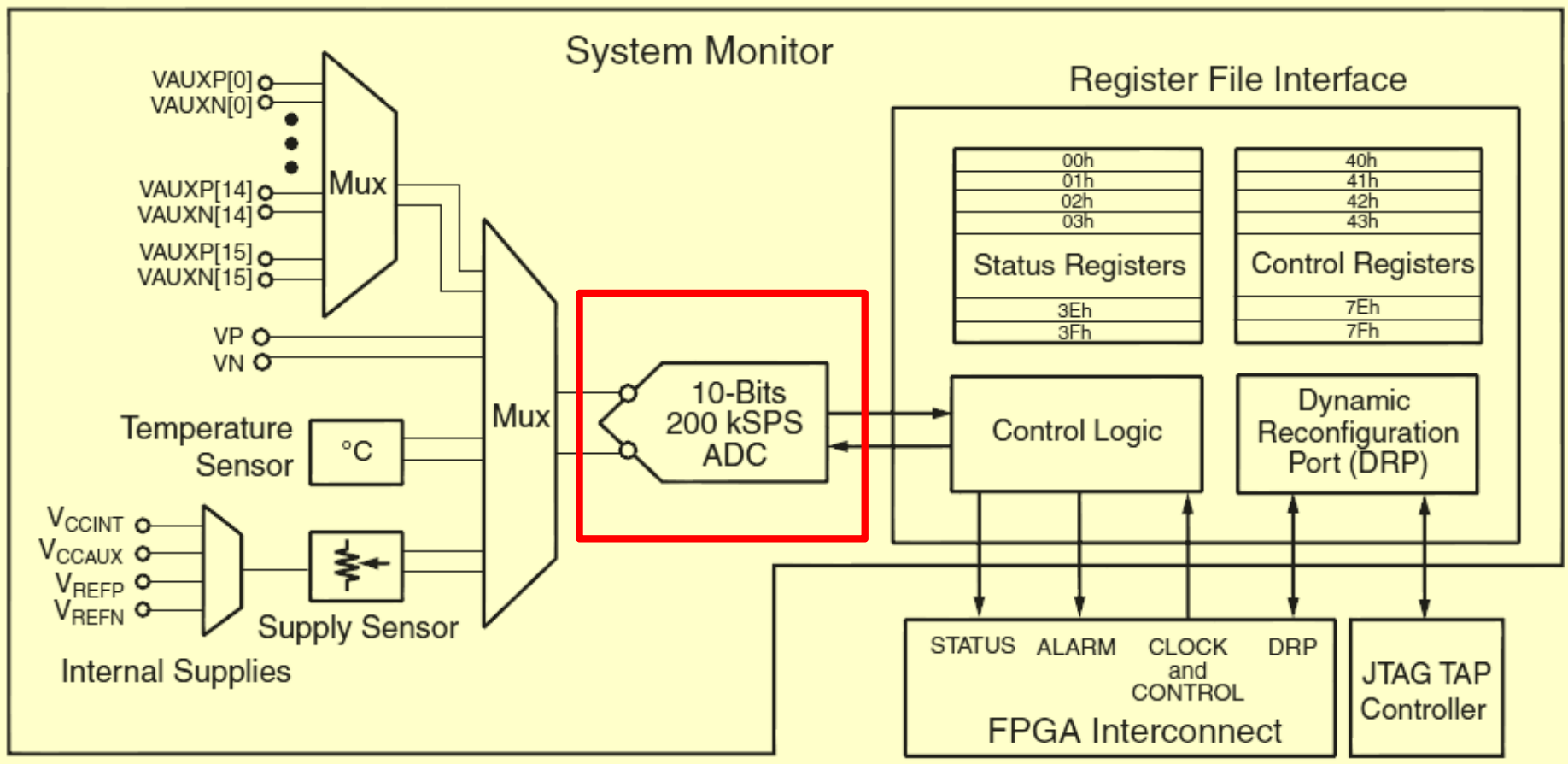
DSP for implementing digital filters

The main components used to implement a digital filter algorithm include adders, multipliers, storage and delay elements. The DSP48A1 slice includes all of the above elements, making it ideal to implement digital filter functions.

$$y(n) = \sum_{k=0}^{k=N-1} h(k)x(n-k)$$



Virtex5 FPGA system monitor:
ADC inside!



This feature may be very useful for avoiding damages to the FPGA! (not present in Spartan …)

Xilinx Virtex5 FXT family

Device	Array	Slices	DSP48E	Block RAM (Kb)	PowerPC	RocketIO	I/O banks	User I/O
FX30T	80x38	5120	64	2448	1	8	12	360
FX70T	160x38	11200	128	5328	1	16	19	640
FX100T	160x56	16000	256	8208	2	16	20	680
FX130T	200x56	20480	320	10728	2	20	24	840
FX200T	240x68	30720	384	16416	2	24	27	960

- **1 slice**: 4 LUTs and 4 flip-flops
- **1 DSP48E**: 1 25x18 multiplier, an adder and an accumulator
- **RocketIO** devices are designed to run from 150 Mb/s to 6.5 Gb/s

Cost may be an issue: FX70T price for instance is ~500 EUROs

Xilinx Virtex7 FXT family
(the newest and most expensive one)

Device[1]	Logic Cells	Configurable Logic Blocks (CLBs)		DSP Slices[3]	Block RAM Blocks[4]			CMTs[5]	PCIe[6]	GTX	GTH	GTZ	XADC Blocks	Total I/O Banks[7]	Max User I/O[8]	SLRs[9]
		Slices[2]	Max Distributed RAM (Kb)		18 Kb	36 Kb	Max (Kb)									
XC7V585T	582,720	91,050	6,938	1,260	1,590	795	28,620	18	3	36	0	0	1	17	850	N/A
XC7V1500T	1,465,920	229,050	16,163	1,620	1,938	969	34,884	18	3	36	0	0	1	17	850	3
XC7V2000T	1,954,560	305,400	21,550	2,160	2,584	1,292	46,512	24	4	36	0	0	1	24	1,200	4
XC7VX330T	326,400	51,000	4,388	1,120	1,500	750	27,000	14	2	0	28	0	1	14	700	N/A
XC7VX415T	412,160	64,400	6,525	2,160	1,760	880	31,680	12	2	0	48	0	1	12	600	N/A
XC7VX485T	485,760	75,900	8,175	2,800	2,060	1,030	37,080	14	4	56	0	0	1	14	700	N/A
XC7VX550T	554,240	86,600	8,725	2,880	2,360	1,180	42,480	20	2	0	80	0	1	16	600	N/A
XC7VX690T	693,120	108,300	10,888	3,600	2,940	1,470	52,920	20	3	0	80	0	1	20	1,000	N/A
XC7VX980T	979,200	153,000	13,838	3,600	3,000	1,500	54,000	18	3	0	72	0	1	18	880	N/A
XC7VX1140T	1,139,200	178,000	17,700	3,360	3,760	1,880	67,680	24	4	0	96	0	1	22	1,100	4
XC7VH290T	284,000	44,375	4,425	840	940	470	16,920	6	1	0	24	8	1	6	300	1
XC7VH580T	580,480	90,700	8,850	1,680	1,880	940	33,840	12	2	0	48	8	1	12	600	2
XC7VH870T	876,160	136,900	13,275	2,520	2,820	1,410	50,760	18	3	0	72	16	1	13	650	3

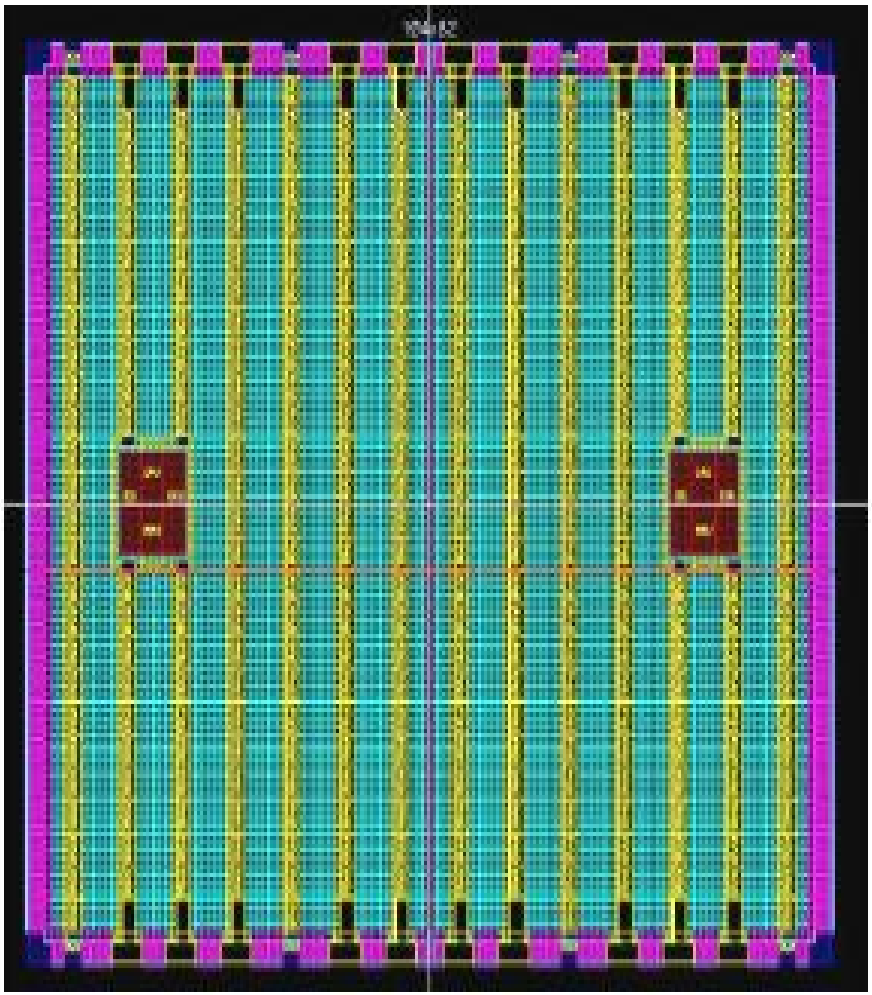
~4k EUROs for one XC7VX485T …

Xilinx Spartan6 family (cheap, but with some limitations)

Device	Logic Cells[1]	Configurable Logic Blocks (CLBs)			DSP48A1 Slices[3]	Block RAM Blocks		CMTs[5]	Memory Controller Blocks (Max)[6]	Endpoint Blocks for PCI Express	Maximum GTP Transceivers	Total I/O Banks	Max User I/O
		Slices[2]	Flip-Flops	Max Distributed RAM (Kb)		18 Kb[4]	Max (Kb)						
XC6SLX4	3,840	600	4,800	75	8	12	216	2	0	0	0	4	132
XC6SLX9	9,152	1,430	11,440	90	16	32	576	2	2	0	0	4	200
XC6SLX16	14,579	2,278	18,224	136	32	32	576	2	2	0	0	4	232
XC6SLX25	24,051	3,758	30,064	229	38	52	936	2	2	0	0	4	266
XC6SLX45	43,661	6,822	54,576	401	58	116	2,088	4	2	0	0	4	358
XC6SLX75	74,637	11,662	93,296	692	132	172	3,096	6	4	0	0	6	408
XC6SLX100	101,261	15,822	126,576	976	180	268	4,824	6	4	0	0	6	480
XC6SLX150	147,443	23,038	184,304	1,355	180	268	4,824	6	4	0	0	6	576
XC6SLX25T	24,051	3,758	30,064	229	38	52	936	2	2	1	2	4	250
XC6SLX45T	43,661	6,822	54,576	401	58	116	2,088	4	2	1	4	4	296
XC6SLX75T	74,637	11,662	93,296	692	132	172	3,096	6	4	1	8	6	348
XC6SLX100T	101,261	15,822	126,576	976	180	268	4,824	6	4	1	8	6	498
XC6SLX150T	147,443	23,038	184,304	1,355	180	268	4,824	6	4	1	8	6	540

~200 EUROs for one XC6SLX150

Hard Processor



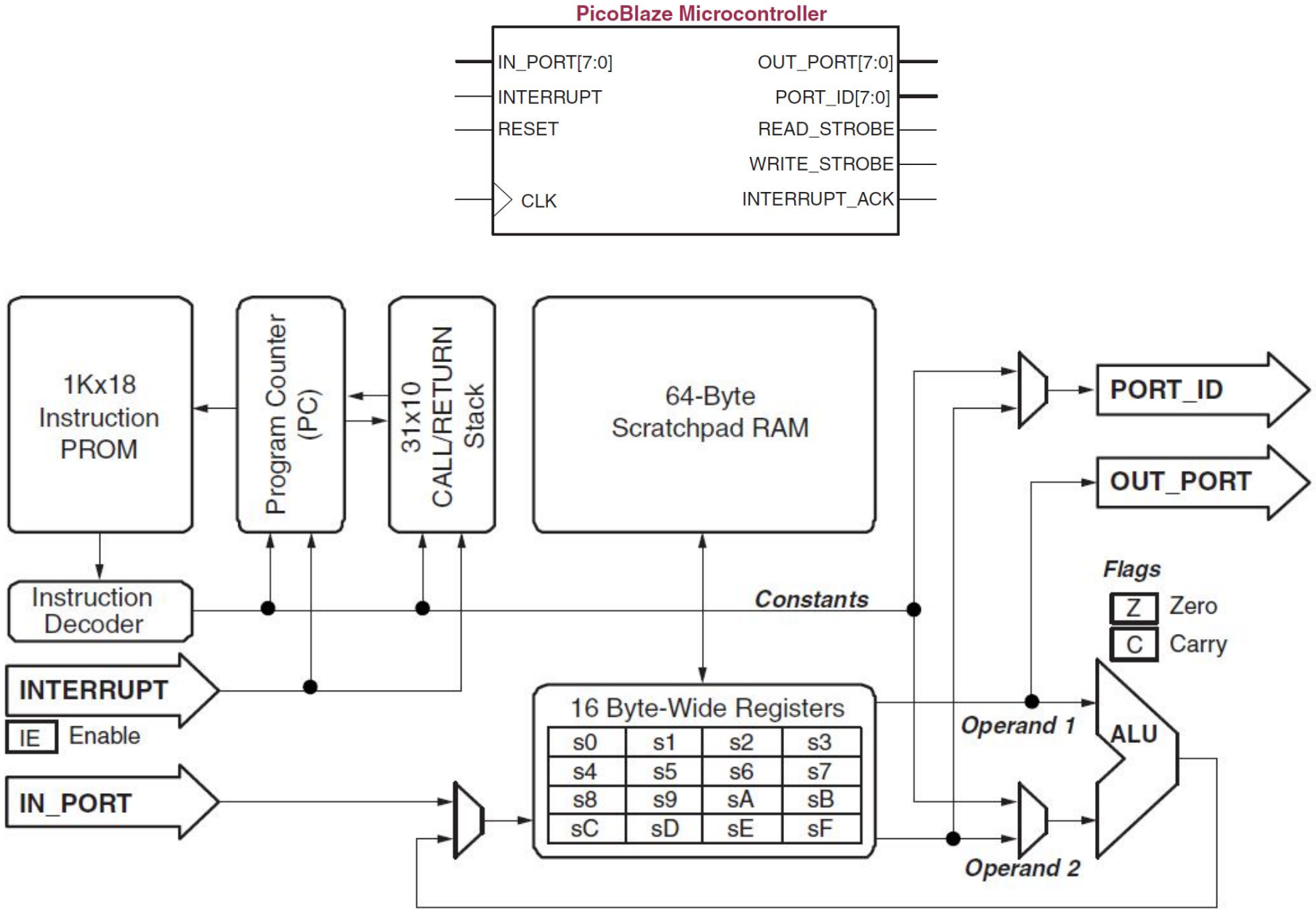
Xilinx Virtex II- PRO

- A processor built from dedicated silicon is referred to as a hard processor:
  - Such is the case for the ARM922T inside the Altera Excalibur family
  - The **PowerPC 405** inside the Xilinx Virtex-II Pro and Virtex4-Virtex5 families
  - Microsemi SmartFusion2 166 megahertz (MHz) ARM ® Cortex™-M3 processor

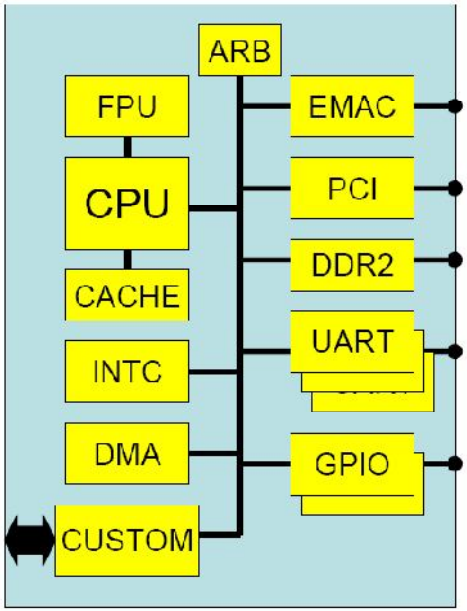
Soft Cores

- A soft processor is built using the FPGA's general-purpose logic.
- Unlike the hard processor, a soft processor must be synthesized and fit into the FPGA fabric.
- Xilinx Picoblaze & MicroBlaze
- Altera Nios

PicoBlaze Embedded Microcontroller Diagram



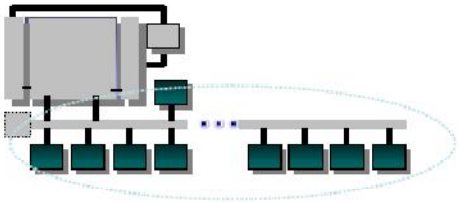
Why use embedded processors?

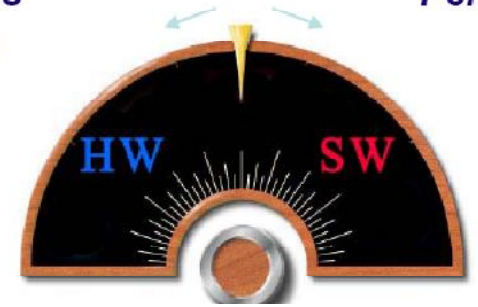


	Microcontroller	FPGA Logic
Strengths	• Easy to program, excellent for control and state machine applications • Resource requirements remain constant with increasing complexity • Re-uses logic resources, excellent for lower-performance functions	• Significantly higher performance • Excellent at parallel operations • Sequential vs. parallel implementation trade-offs optimize performance or cost • Fast response to multiple, simultaneous inputs
Weaknesses	• Executes sequentially • Performance degrades with increasing complexity • Program memory requirements increase with increasing complexity • Slower response to simultaneous inputs	• Control and state machine applications more difficult to program • Logic resources grow with increasing complexity

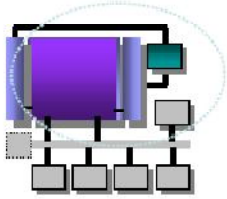
Customization: take only the peripherals you need and replicate them as many times as needed.
Create your own custom peripherals.

Performing some software tasks in hardware can be expensive



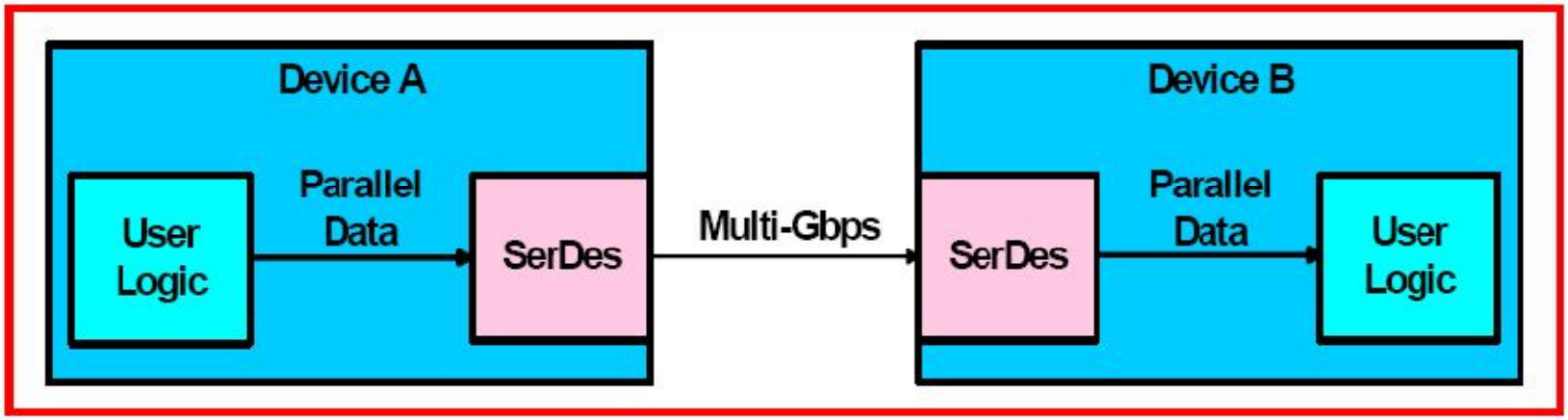


Performing some hardware tasks in software can be slow



Strike optimum balance in system partitioning.

Serial signaling



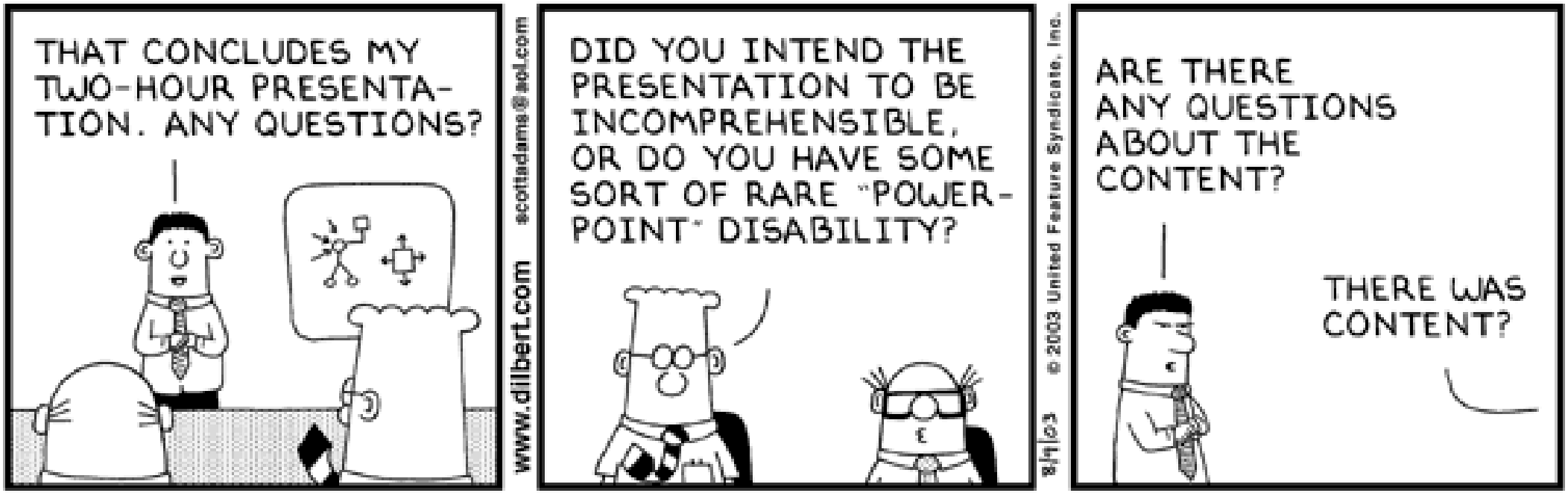
- Avoids clock/data skew by using embedded clock.
- Reduces EMI and power consumption.
- Simplifies PCB routing.

Conclusions

Up to date, FPGAs are very suitable devices for implementing digital signal processing. In fact:

- FPGAs feature enormous logic power with DSP optimized blocks (for MAC operations);
- FPGAs can manage in real time data throughput of the order of magnitude of GBit/s on several channels at the same time;
- FPGAs are flexible, they can be reprogrammed anytime
- The design flow is straight-forward (if you know VHDL)
- IP cores are provided by manufacturers and can easily be found over the Web.
- Price is an issue (large FPGAs are very expensive), but you can choose the one who is tailored to your needs.

THANKS FOR YOUR ATTENTION !



References

- For FPGA:
  - http://www.xilinx.com
  - http://www.altera.com
  - http://www.actel.com
  - http://www.latticesemi.com
  - http://www.atmel.com
- For DSP:
  - http://www.ti.com
  - http://www.analog.com

- "**The scientist and Engineer's Guide to Digital Signal Processing**" by Steven W. Smith, PhD
- "**Understanding Digital Signal Processing**" by Richard G. Lyons

Backup slides

Power management: suspend mode

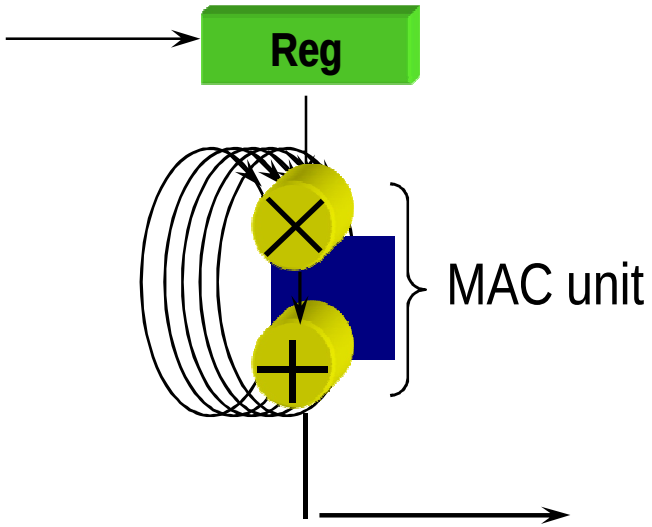
• Quickly and easily puts the FPGA into a static condition, eliminating most active current.
• Reduces quiescent current by 40% or more.
• Retains FPGA configuration data and the state of the FPGA application during suspend mode.
• Fast, programmable FPGA wake-up time from suspend mode.
• Individual control on each user-I/O pin to define pin behavior while in suspend mode.
• Activated externally by the system using a single dedicated control pin (SUSPEND).
• Indicates the present suspend mode status using the AWAKE pin.
• Awakens an FPGA in suspend mode using any of eight SUSPEND control pins (SCP).
• SUSPEND_SYNC primitive to acknowledge a ready state prior to entering suspend mode.

Why FPGAs for digital signal processing ?

Why FPGAs for DSP?

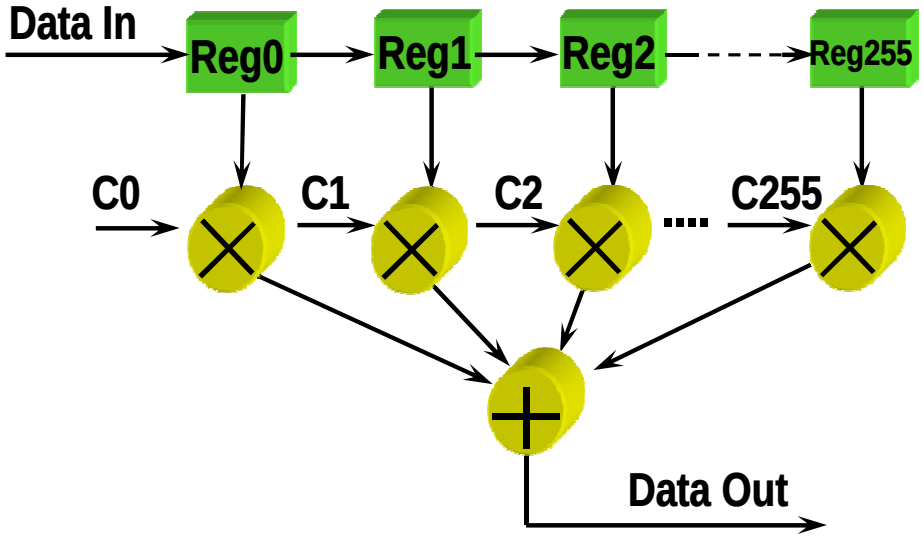
Reason 1: FPGAs handle high computational workloads

Conventional DSP Device

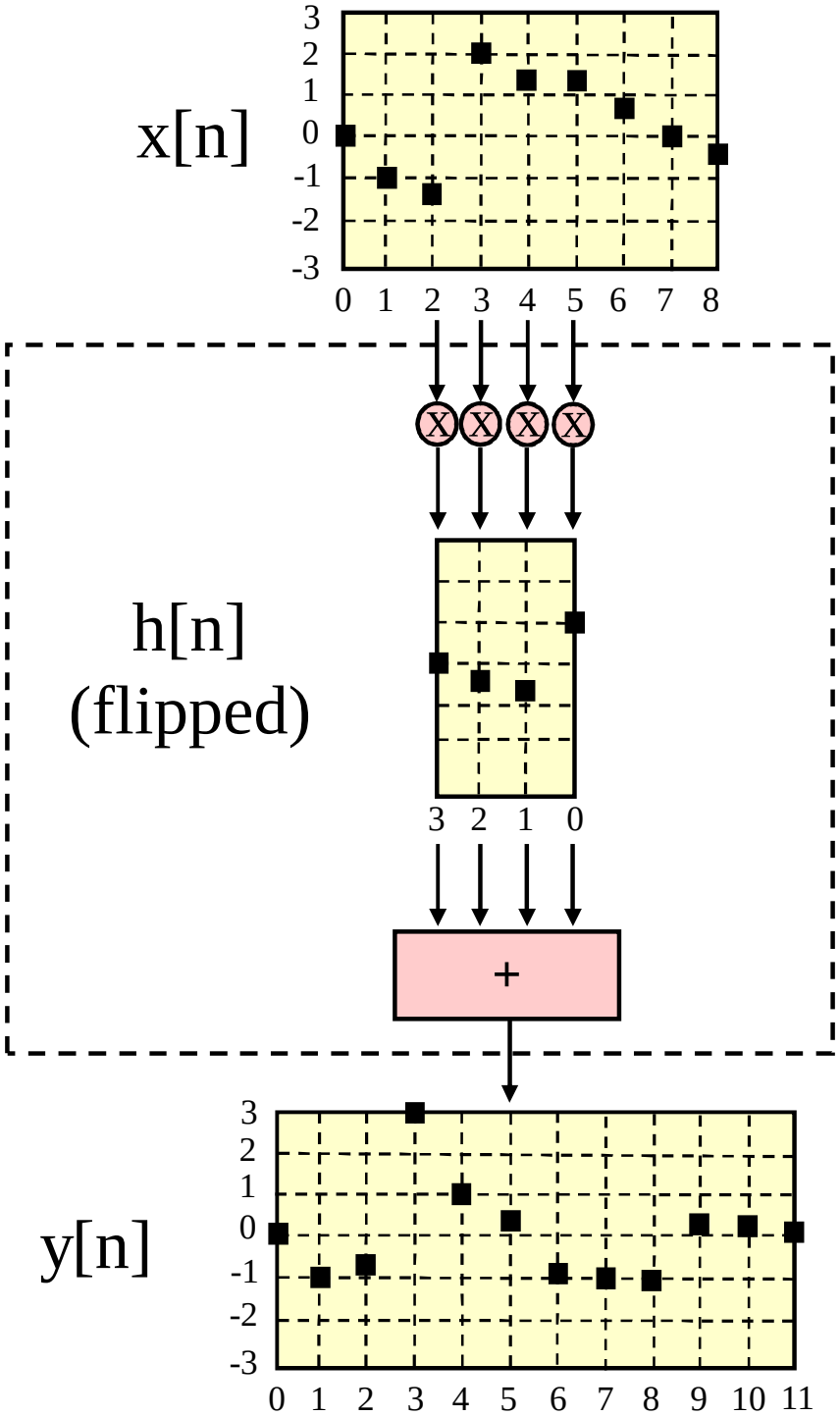


256 Loops needed to process samples

FPGA



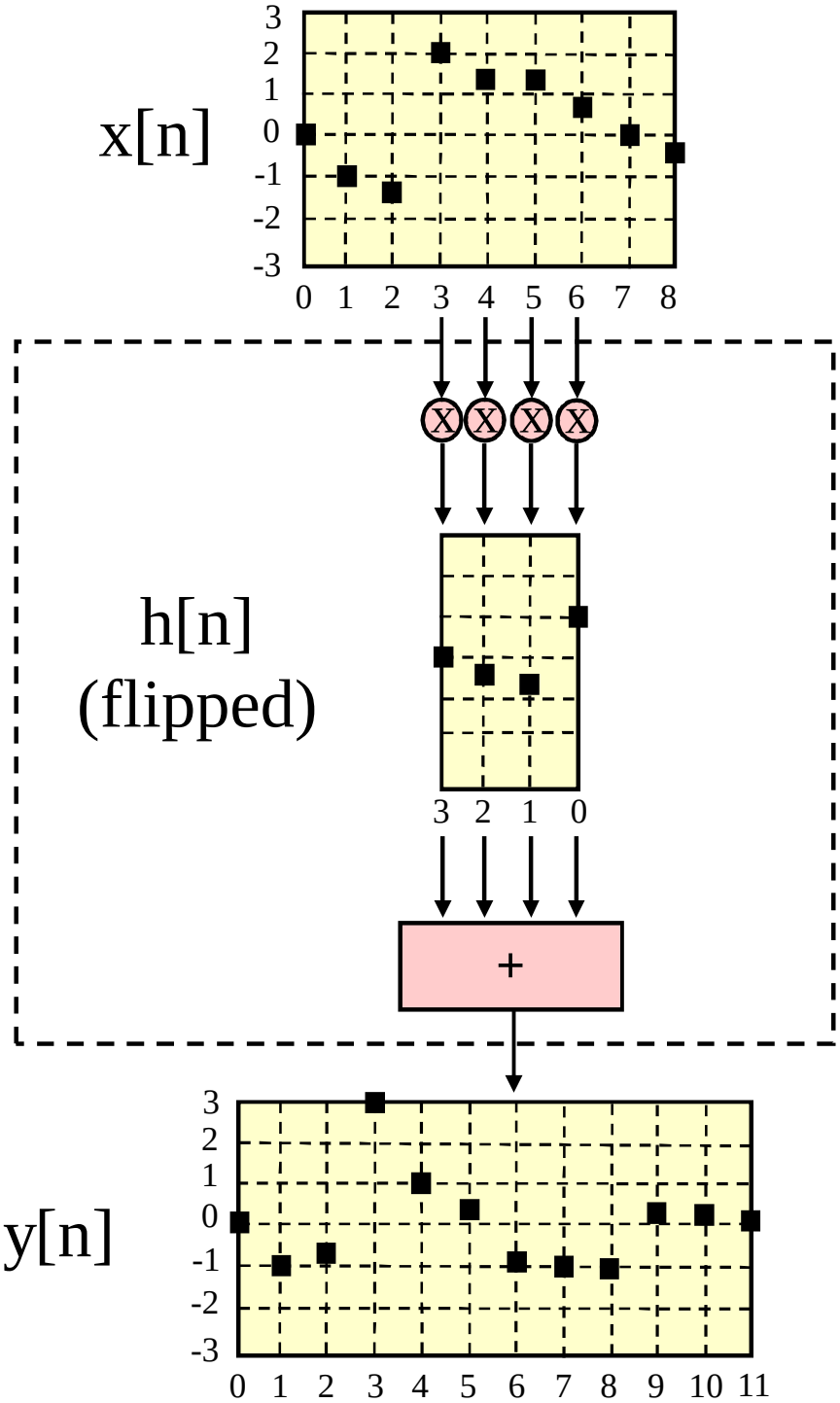
All 256 MAC operations in 1 clock cycle



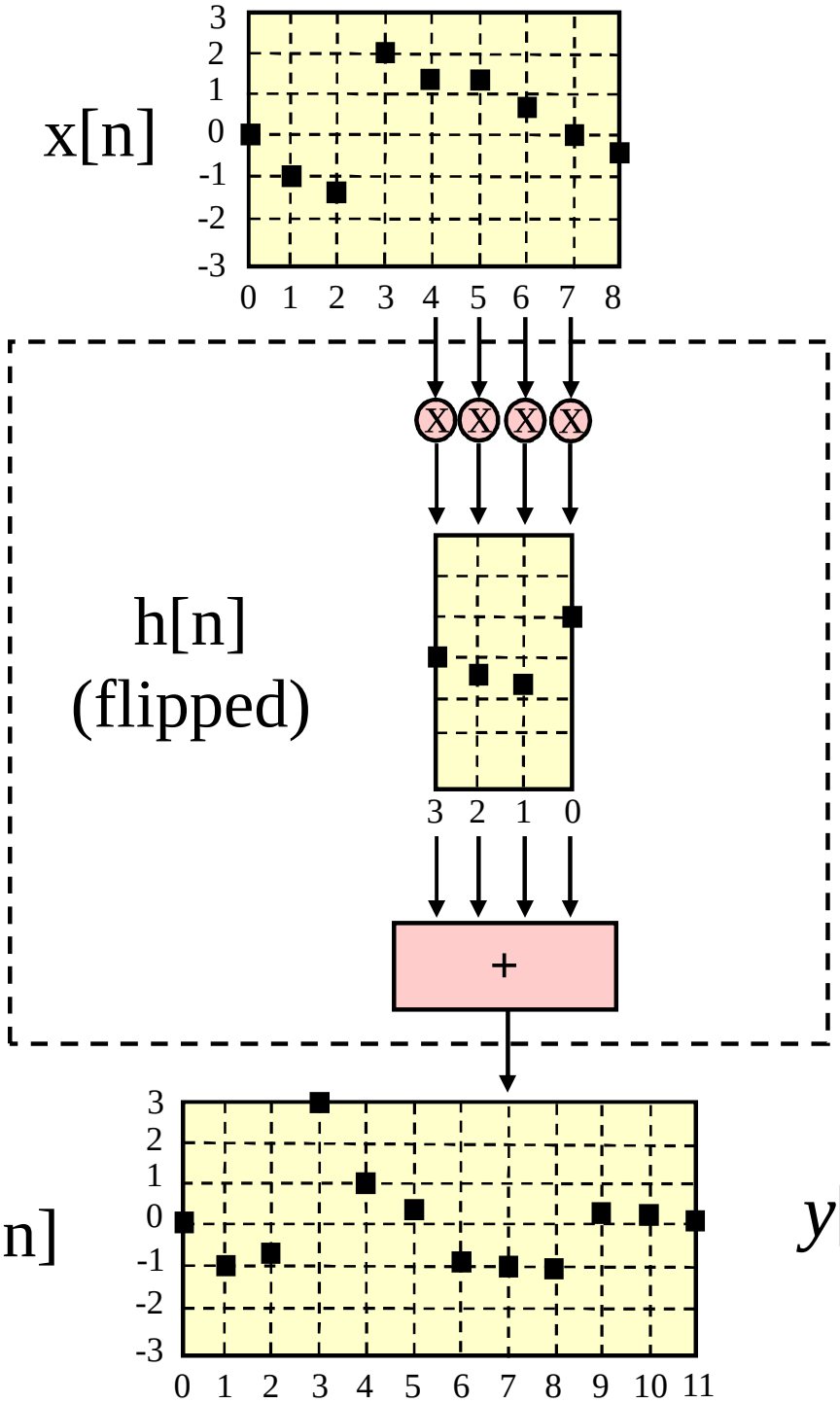
Example: FIR (Finite Impulse Response) digital filter

Each output sample is found by multiplying samples from the input signal by the **filter kernel** coefficients and summing the products.

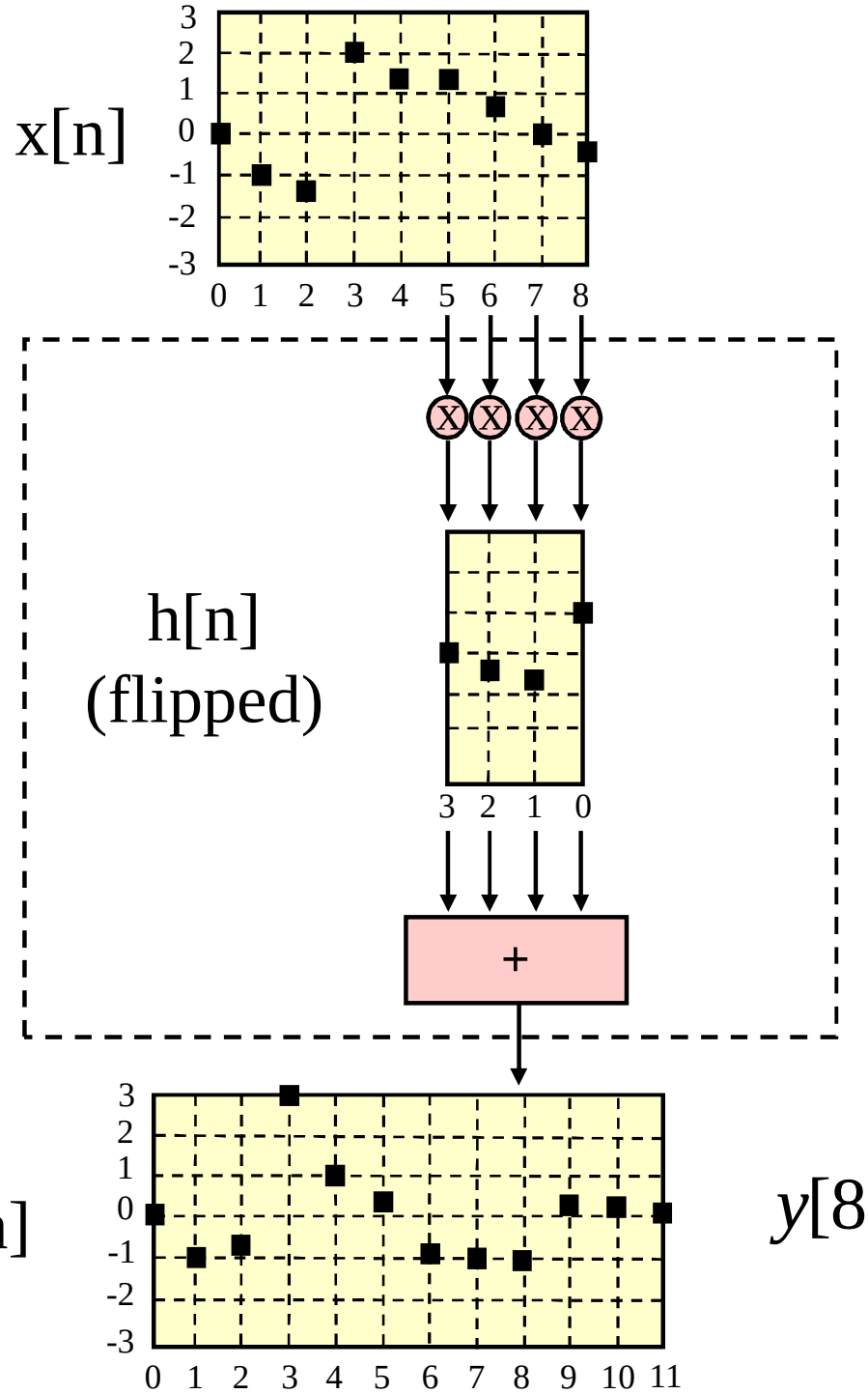
$$y[5] = \sum_{j=0}^{3} h[j]x[5-j]$$



$$y[6] = \sum_{j=0}^{3} h[j]x[6-j]$$

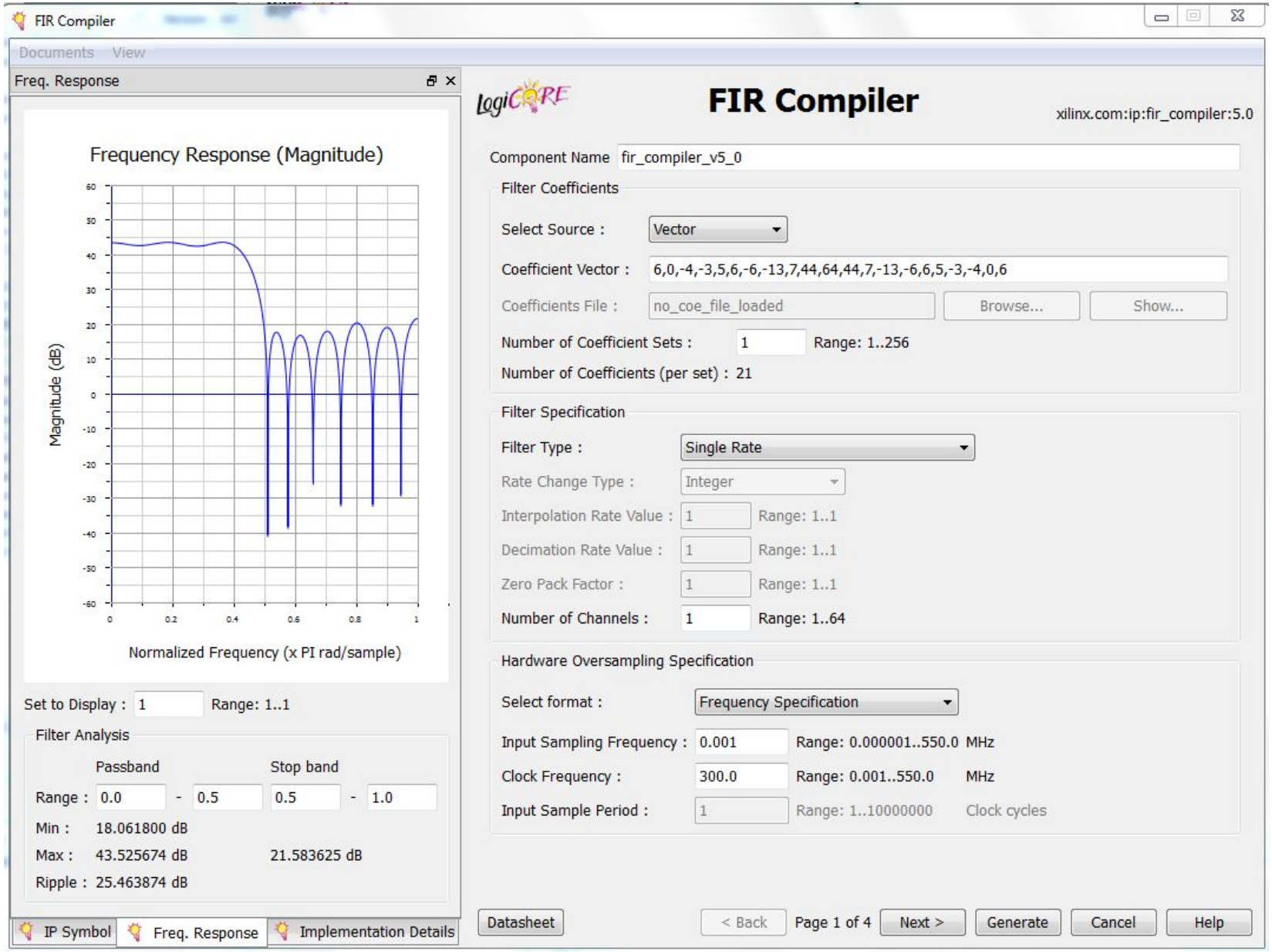


$$y[7] = \sum_{j=0}^{3} h[j]x[7-j]$$

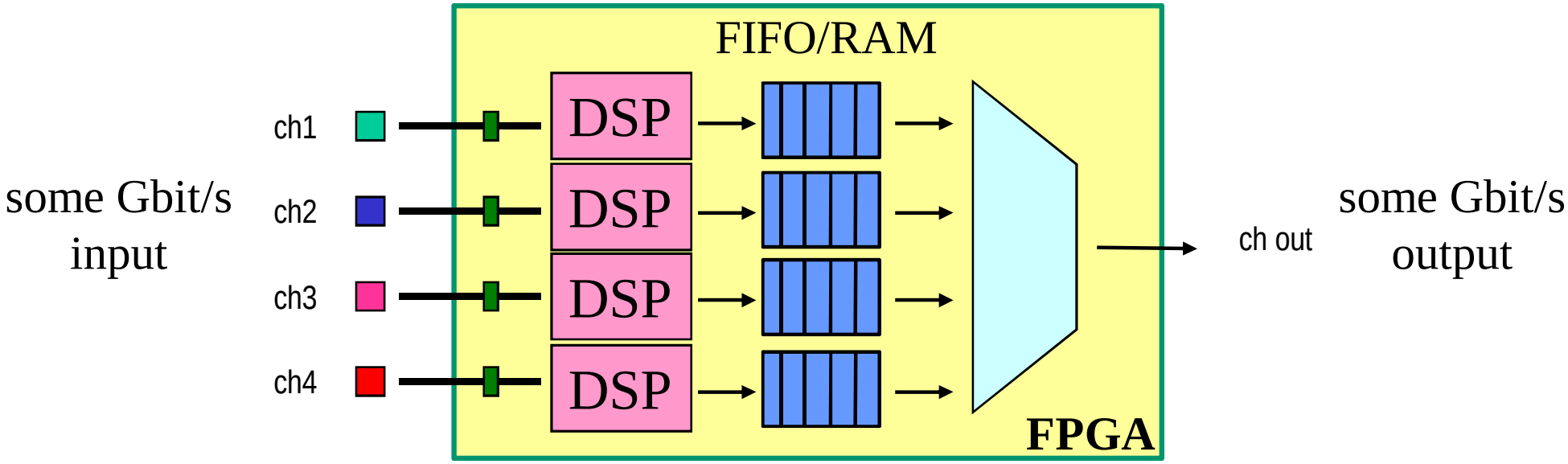


$$y[8] = \sum_{j=0}^{3} h[j]x[8-j]$$

From the Core generator library



FPGAs are ideal for multi-channel DSP designs



When sample rates grow above a few MHz, a DSP has to work very hard to transfer the data without any loss. This is because the processor must use shared resources like memory buses, or even the processor core which can be prevented from taking interrupts for some time.

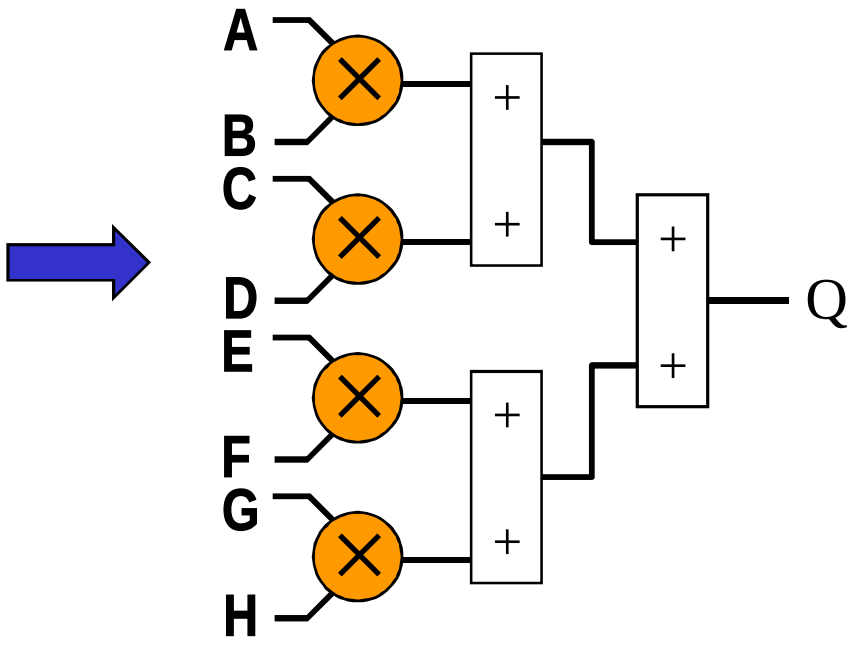
A FPGA on the other hand dedicates logic for receiving the data, thus maintaining high rates of I/O.

Why FPGAs for DSP?

Reason 2: Tremendous Flexibility

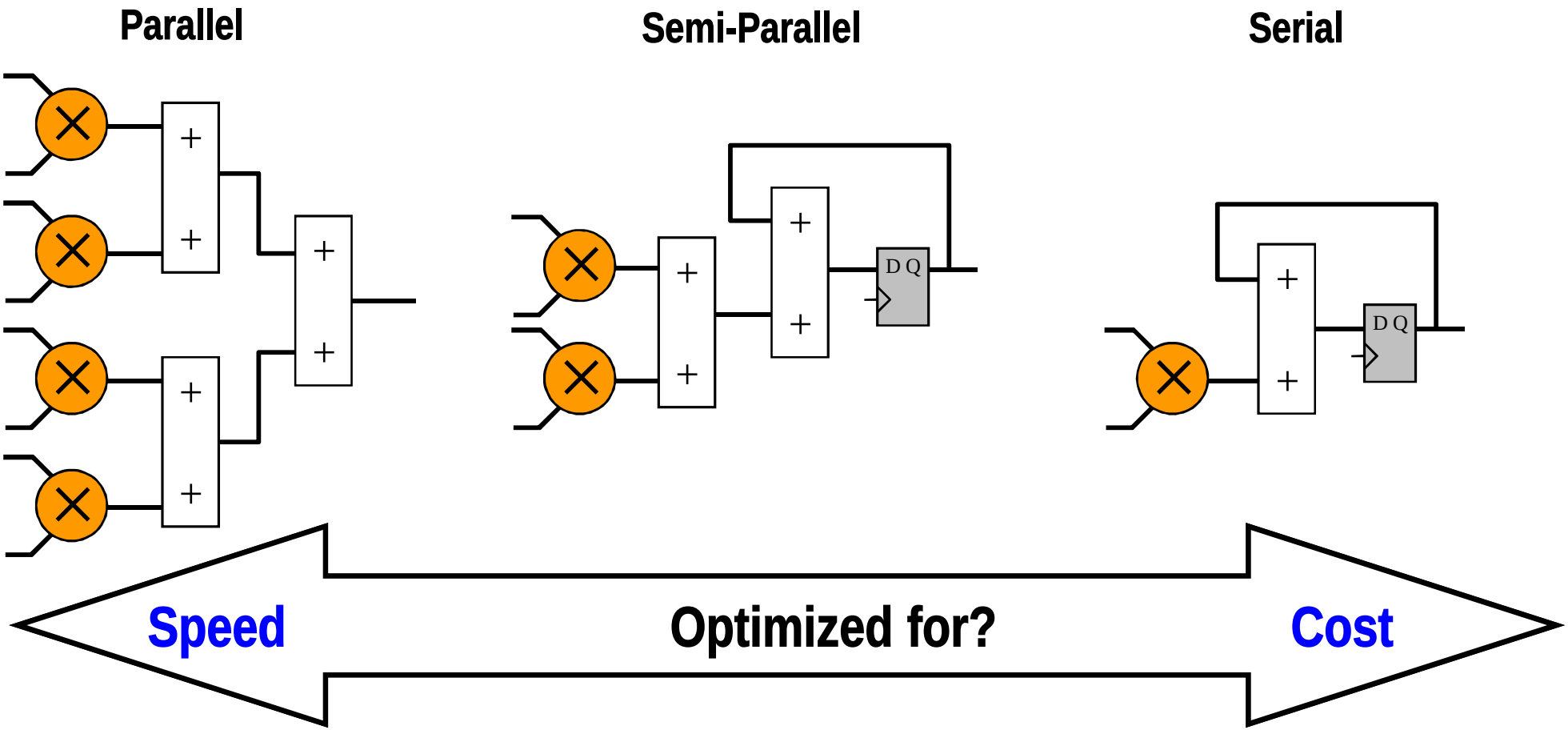
Q = (A x B) + (C x D) + (E x F) + (G x H)

can be implemented in parallel



But is this the only way in the FPGA?

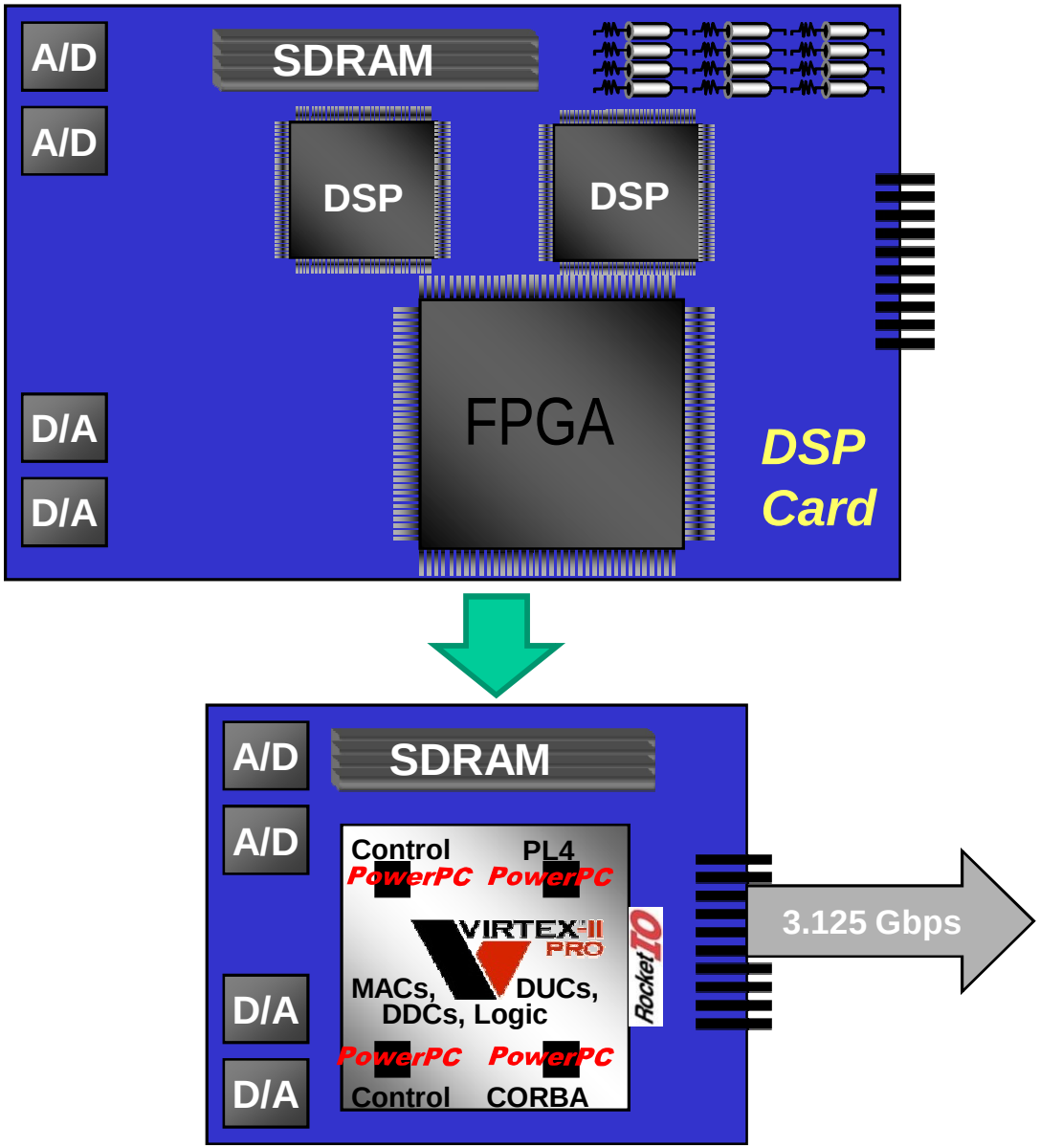
Customize architectures to suit your algorithms



FPGAs allow Area (cost) / Performance tradeoffs

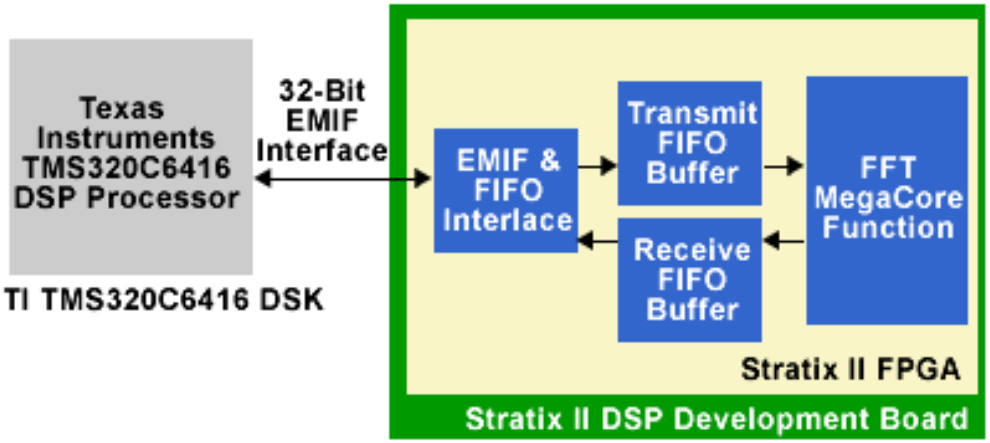
Why FPGAs for DSP?

Reason 3: Integration simplifies PCBs



FFT Co-Processing Example

- FFT co-processor implemented within an Altera Stratix FPGA and connected to a Texas Instruments DSP via the 32-bit external memory interface (EMIF)
- Develop co-processor from intellectual property (IP), such as FFT MegaCore



FFT implementation with only the TI DSP processor

TI DSP running at 720 MHz completed the 1024-point 16-bit FFT in **9.06 µs**

FPGA Co-processor implementation

At 278 MHz completed the transform in only **4.64 µs**

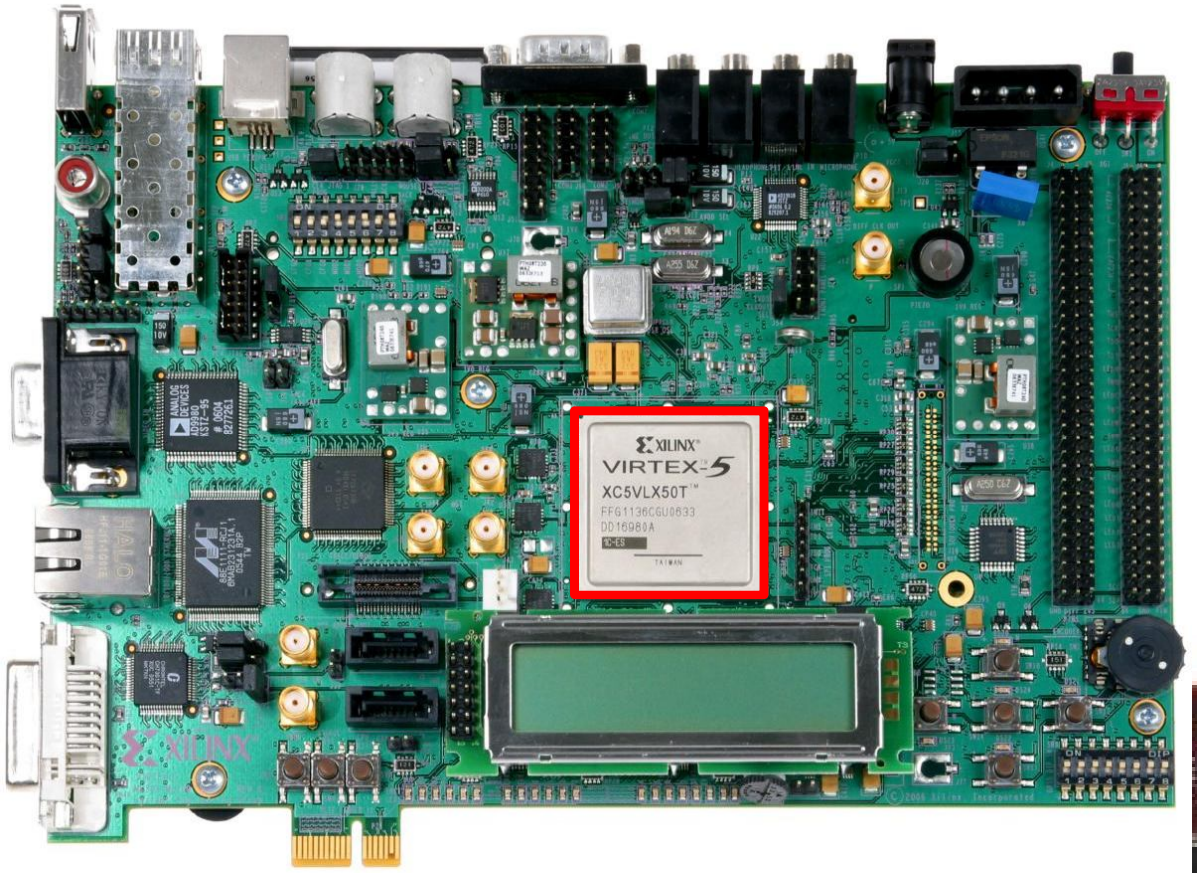
DSP versus FPGA

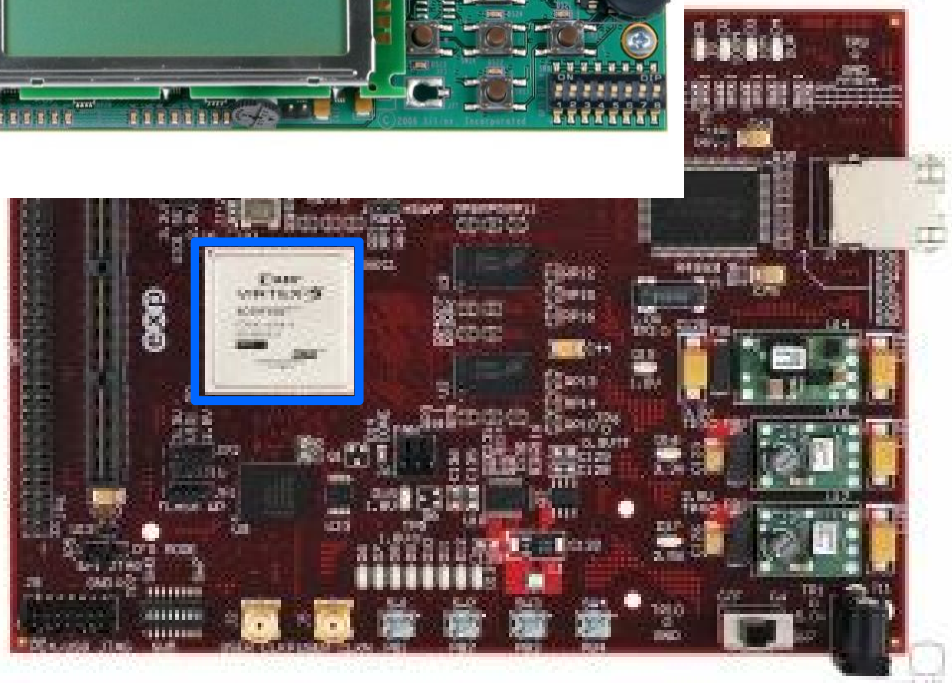
The **DSP** is a specialized microprocessor, typically programmed in C (or assembler for higher performance). It is well suited to extremely complex math-intensive tasks with conditional processing. It is limited in performance by the clock rate and the number of useful operations it can do per clock cycle. As an example, a TMS320C6201 has two multipliers and a 200MHz clock, so it can achieve **400M multiplies per second**.

An **FPGA** is an uncommitted "sea of gates". The device is programmed by connecting the gates together to form complex logic blocks, registers, adders and so forth, plus the blocks available in hardware, such as RAMs and multipliers. FPGA performance is limited by the number of gates available and the clock rate. A FX70T Virtex5 device has 128 multipliers that can operate at more than 100MHz. This gives a total of **12800M multiplies per second**.

Lower sampling rates and increased complexity suit the DSP approach; higher sampling rates combined with repetitive tasks suit the FPGA.

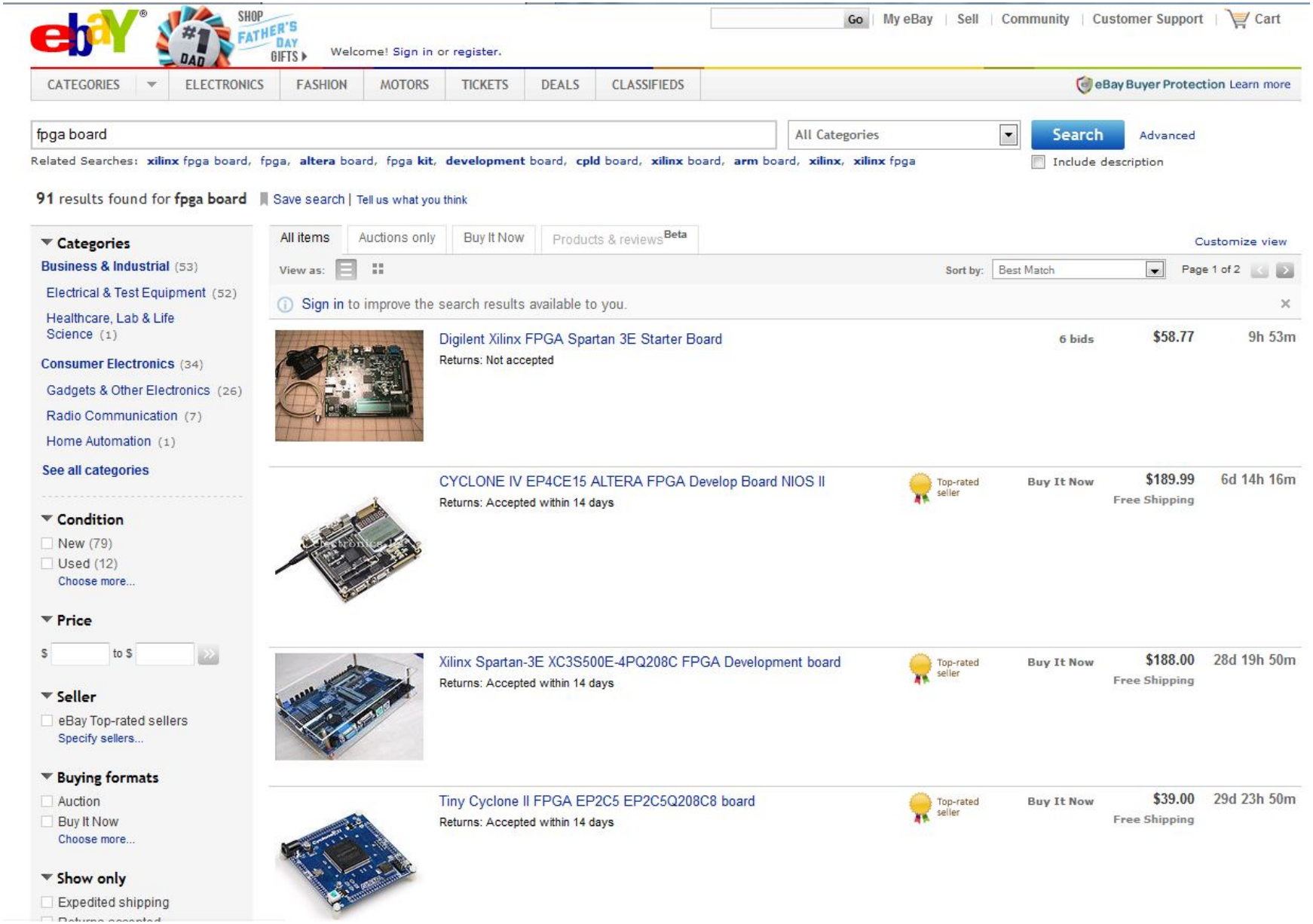
Starter boards





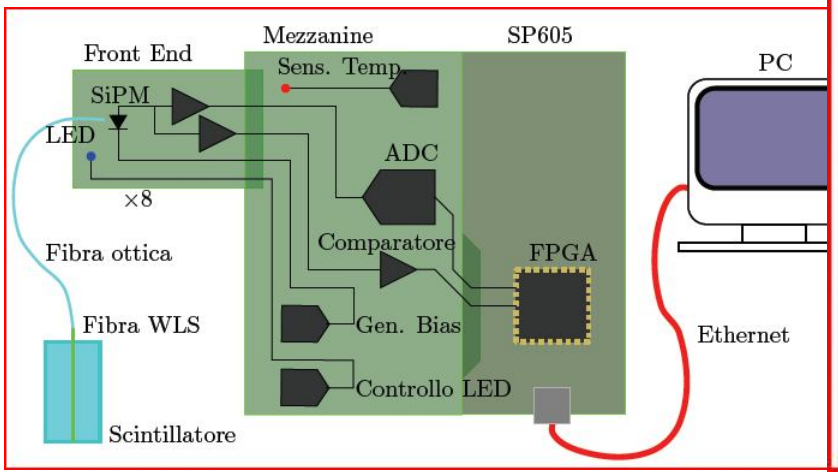
(Relatively) Low cost demo boards are ideal for beginning to work with FPGAs and for finding the right device

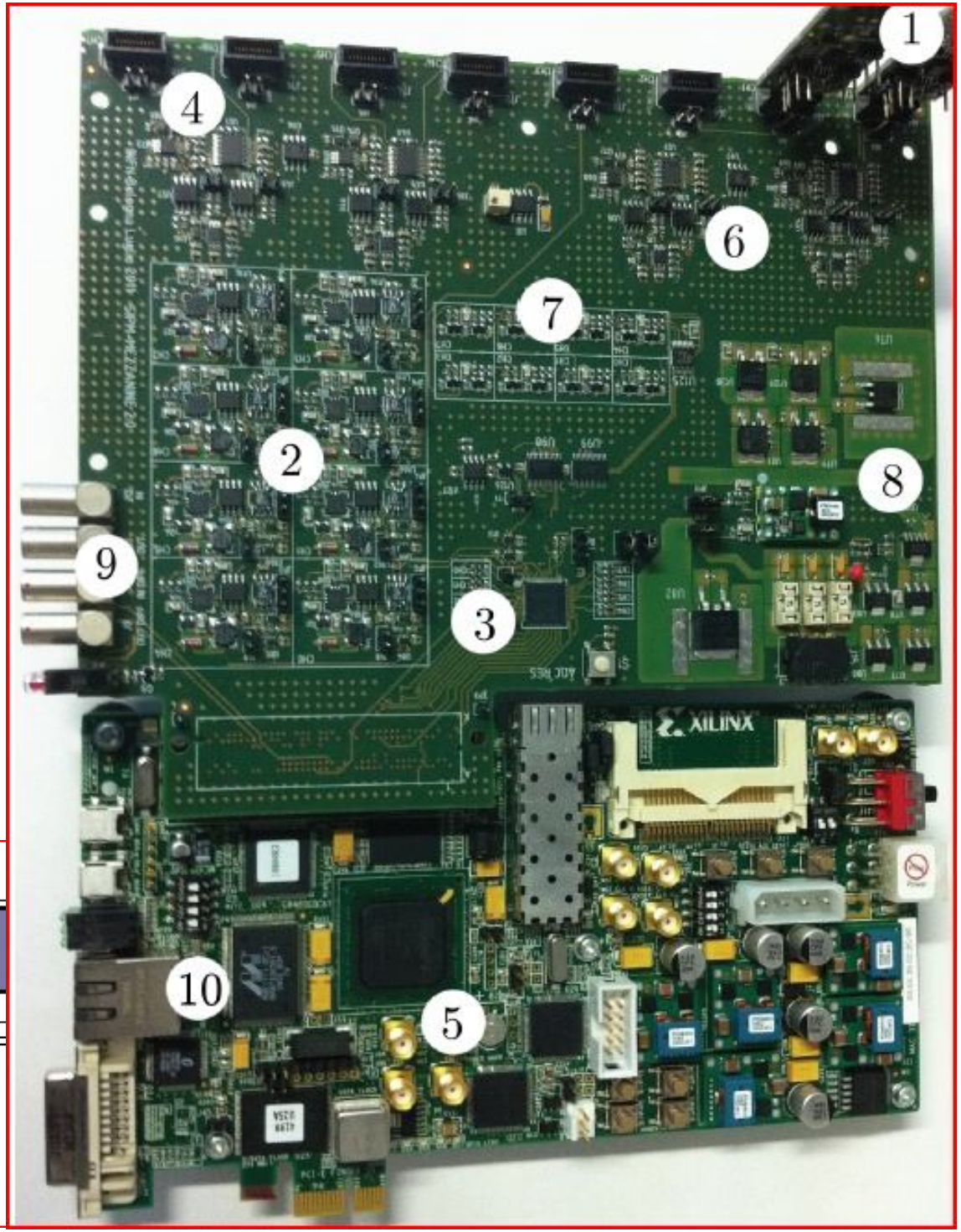
Starter boards on ebay



Demo boards can also be used for building simple detector readout systems, when connected to the related front end boards

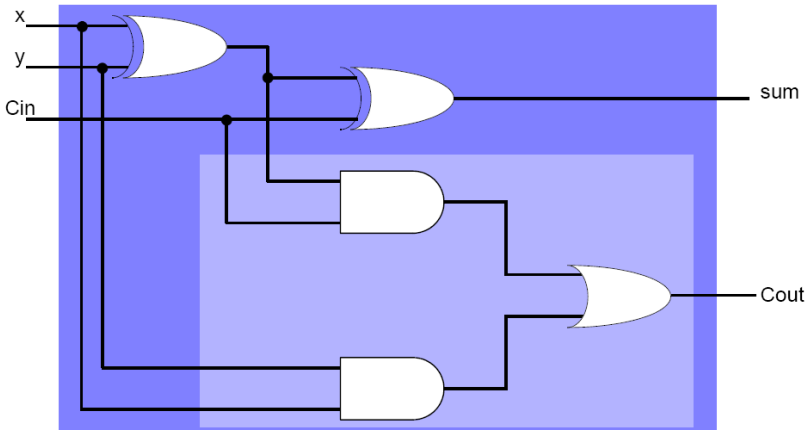
Courtesy:
A. Montanari (INFN, Bologna)

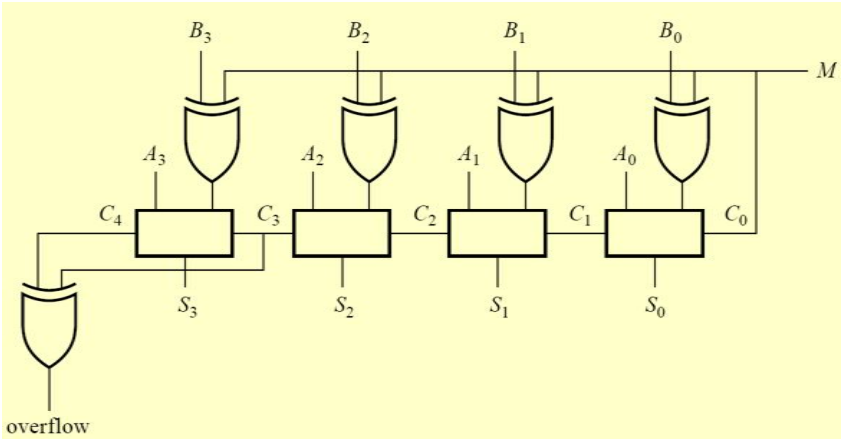




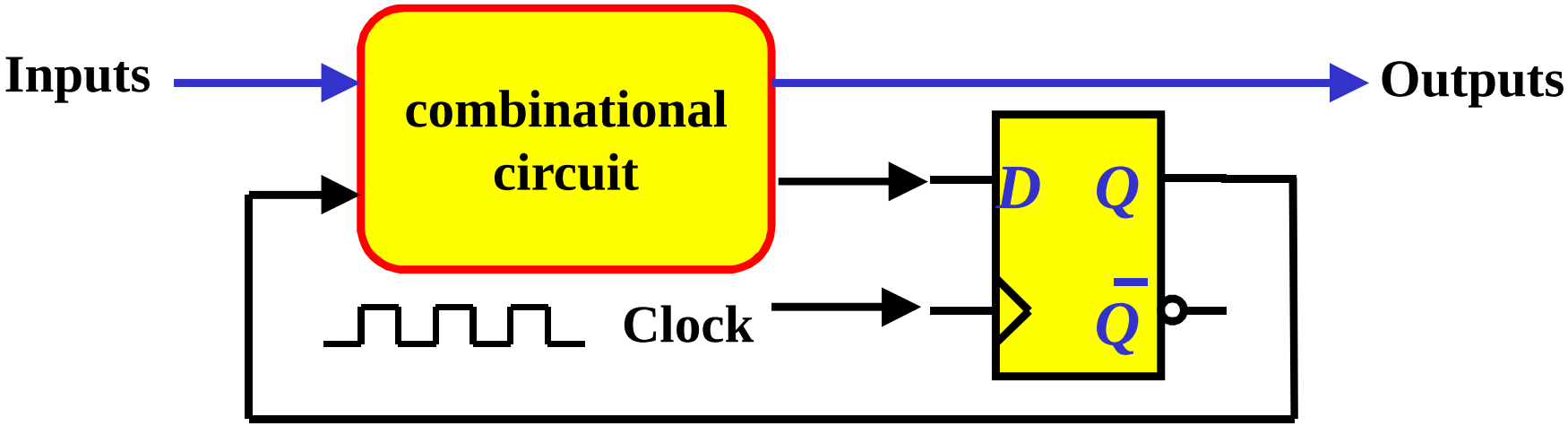
Digital design

- combinational logic



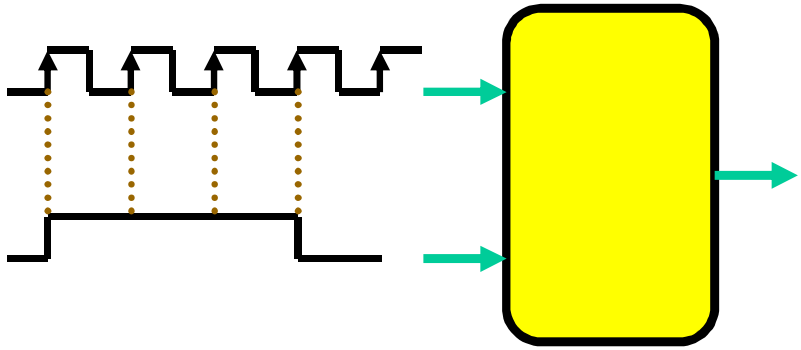


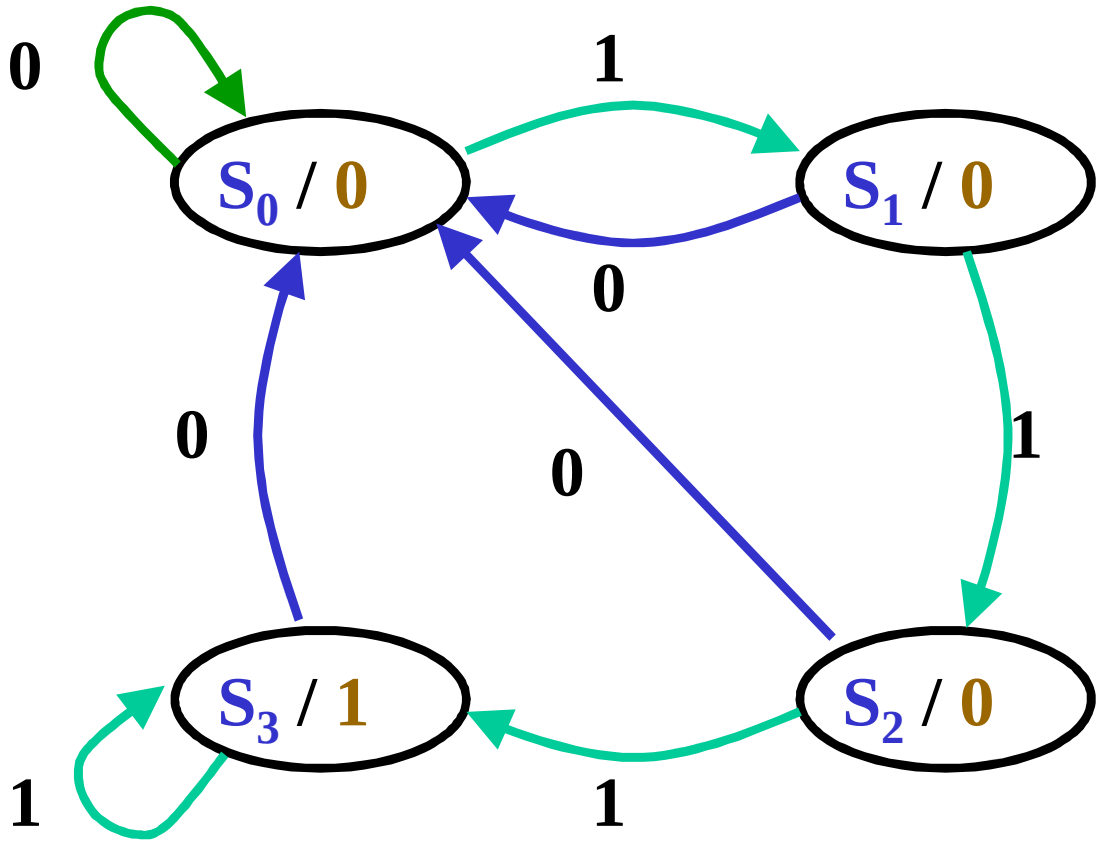
- sequential logic



State machines

Detect 3 or more consecutive 1's

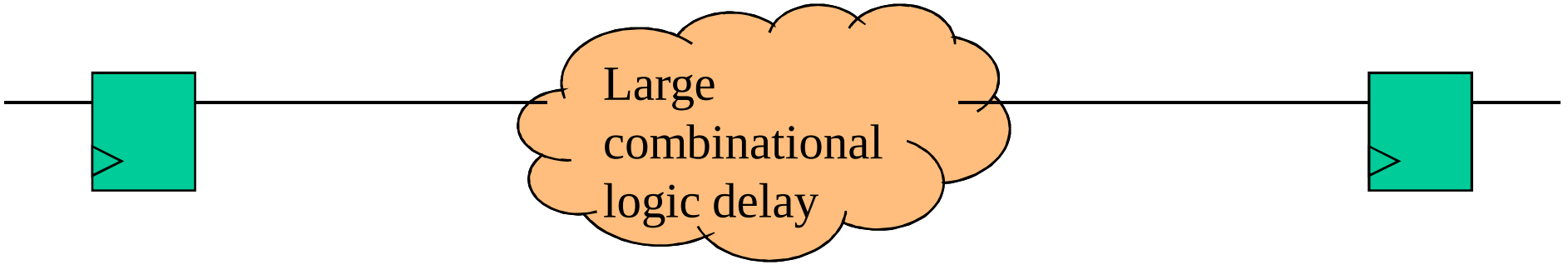




State	*A* *B*
S_0	0 0
S_1	0 1
S_2	1 0
S_3	1 1

Pipelining

Before:



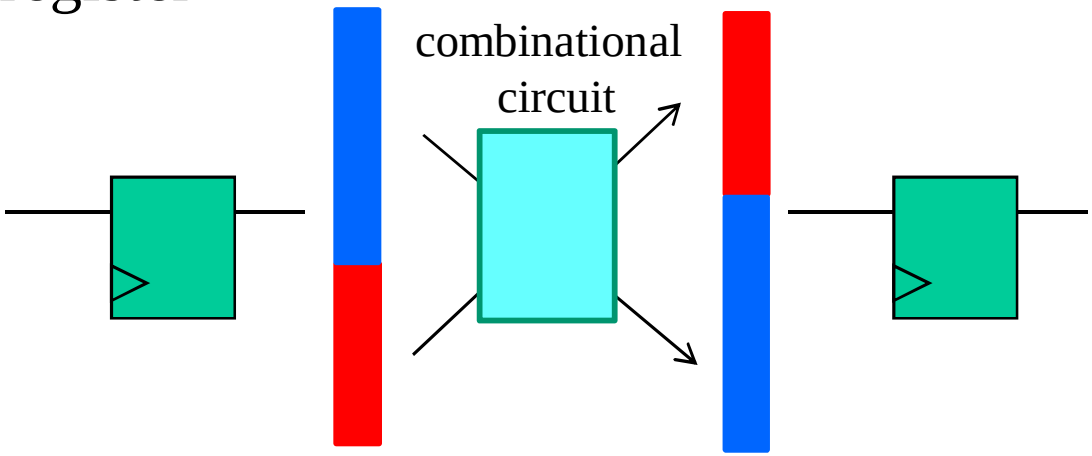
After:

Once the pipeline is full, a new result is produced every clock period

Digital design

Hardware implementation of a register bit-shift:

32-bit register

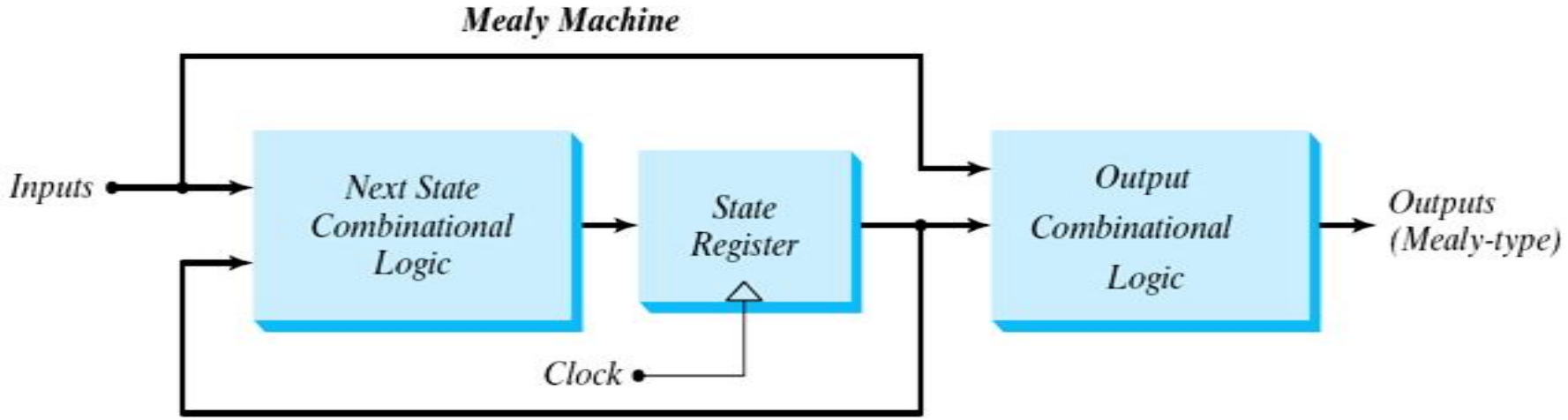


simple wiring
(*1 shift every clock cycle*)

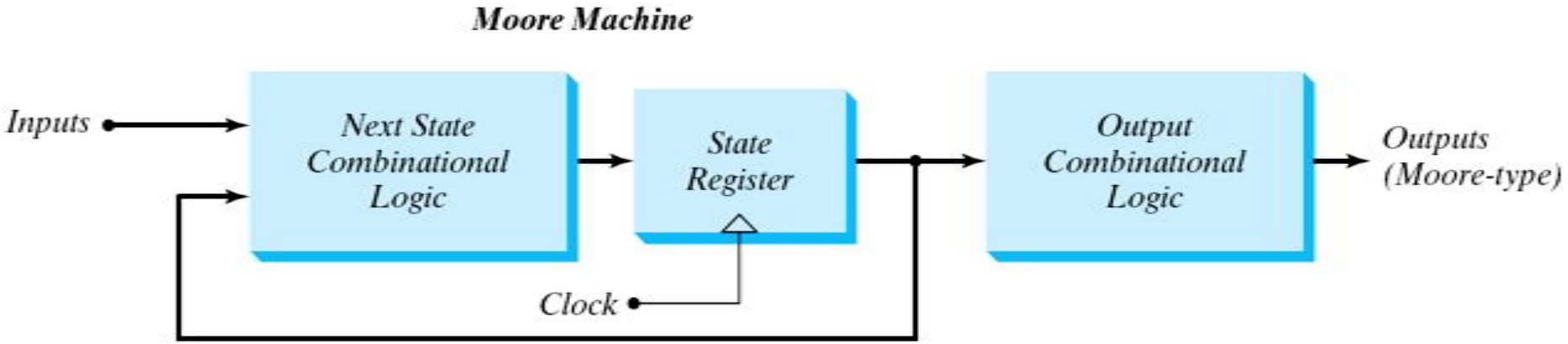
Software implementation of a register bit-shift:

```
UINT32 bitshift(UINT32 input)
{
UINT32 t1,t2;
        t1=input & 0x00000fff;
        t2=input & 0xfffff000;
        return (t1 << 20) | (t2 >> 12);
}
```

sequence of
instructions for the μP
(*1 shift every n clock cycles*)

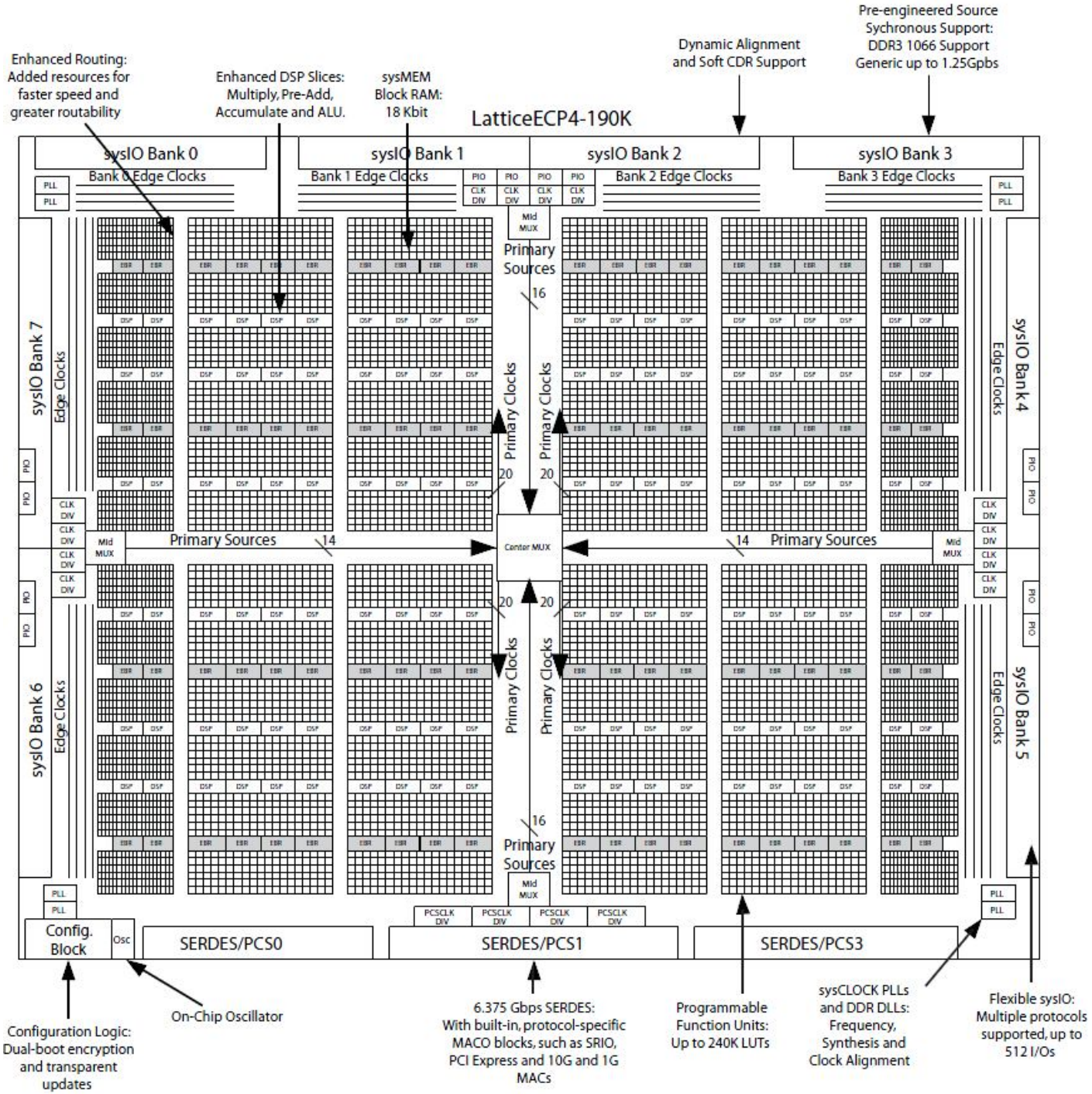

Mealy Machine
Inputs
Next State Combinational Logic
State Register
Clock
Output Combinational Logic
Outputs (Mealy-type)

(a)


Moore Machine
Inputs
Next State Combinational Logic
State Register
Clock
Output Combinational Logic
Outputs (Moore-type)

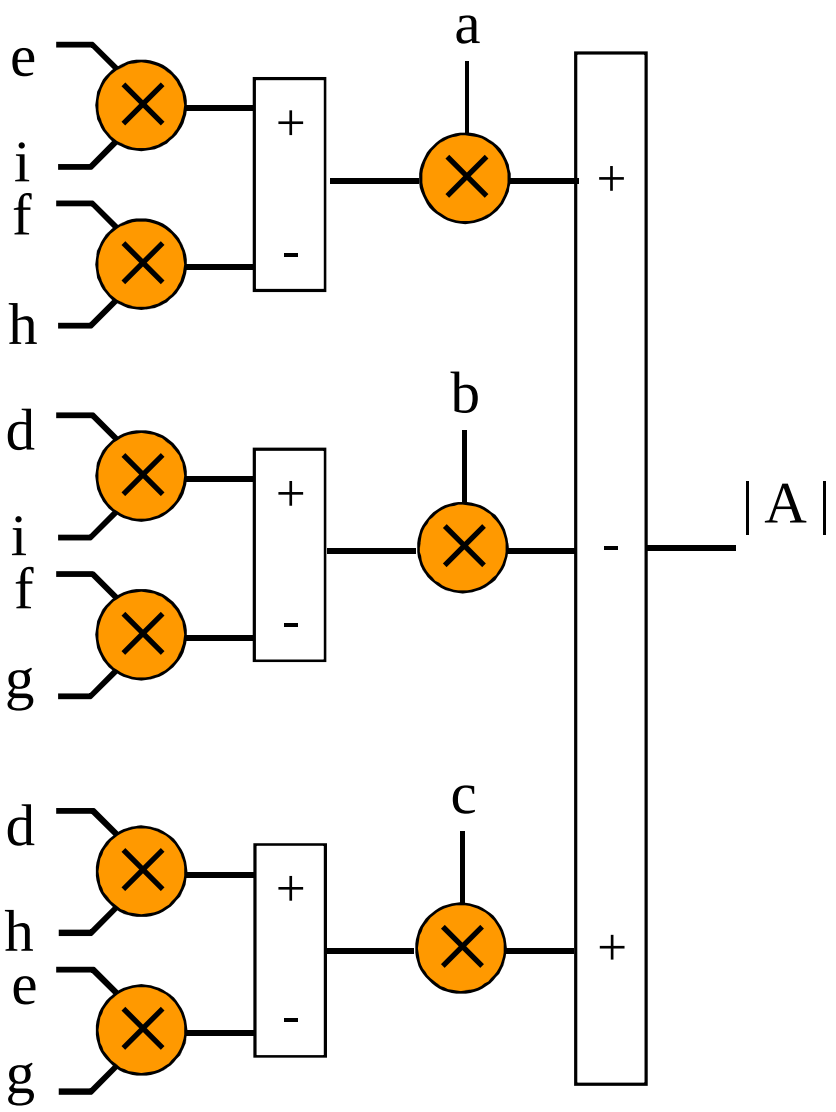
(b)

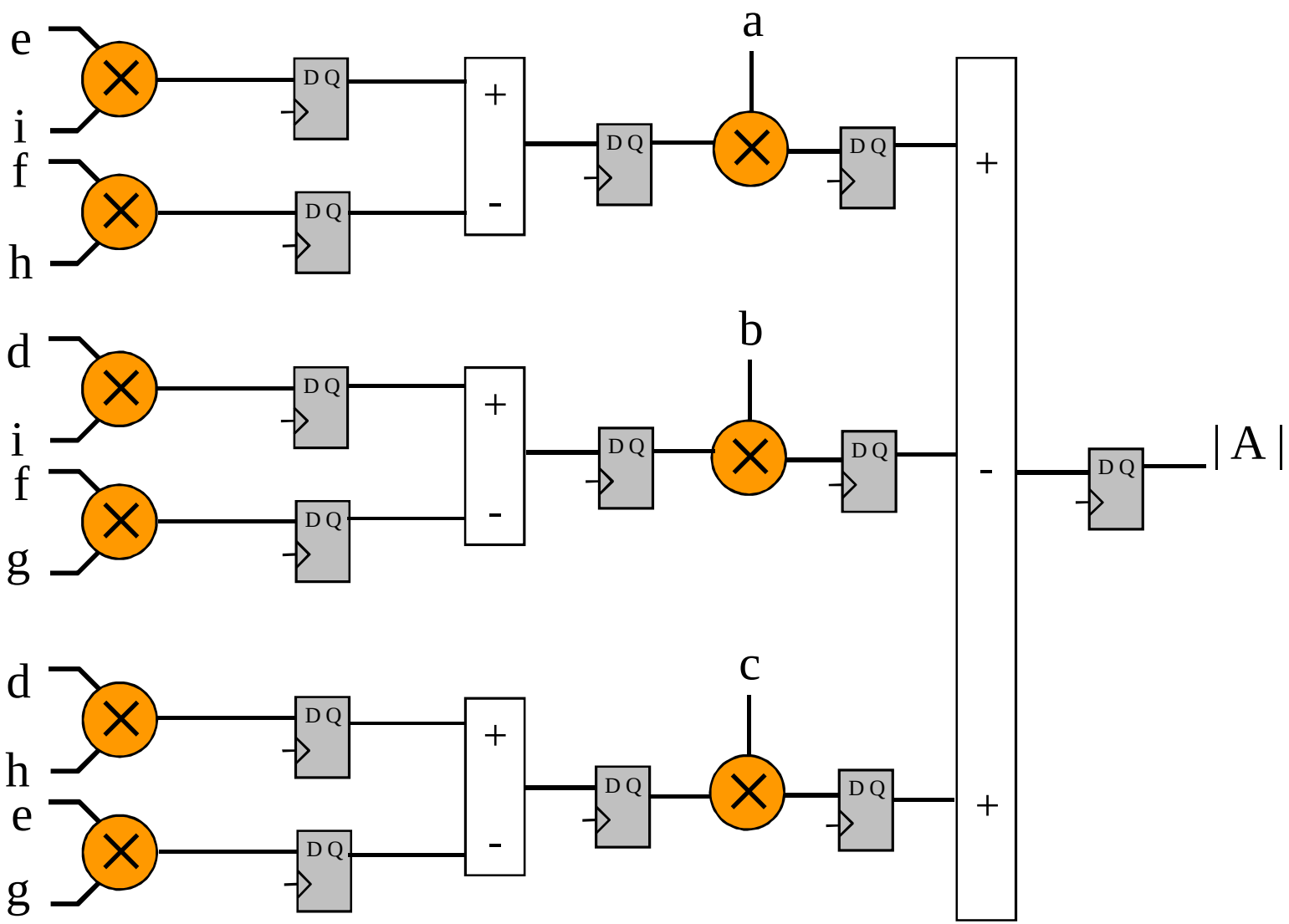
Lattice FPGA



Calculating the determinant of a 3x3 matrix

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$



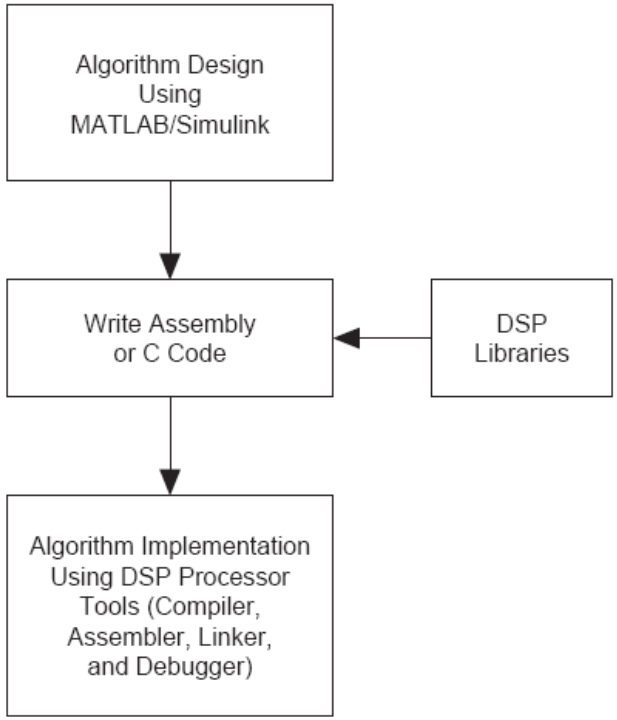

e
i
f
h
d
i
f
g
d
h
e
g
D Q
+
-
a
b
c
|A|

DSP Development

For the DSP-only approach, functions like FIR filters, FFTs and Correlators are available as pre-built, assembly optimized, C-callable library functions.

Challenges arise while trying to optimize the performance of a function for a particular DSP requiring an in-depth knowledge of the processor architecture.

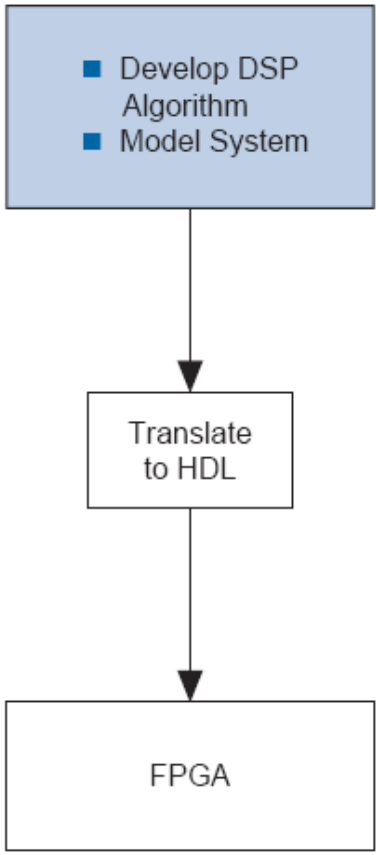
However, DSP designers in general are more comfortable with the DSP-only approach due to ease of implementation.



FPGA Development

FPGA co-processor approach requires hardware knowledge to assemble the various components of the FPGA co-processing system (EMIF/FIFO interface, transmit and receive FIFO buffers, and co-processing function).

The availability of architecturally optimized pre-built IP functions from FPGA vendors like Altera & Xilinx aid in the implementation of co-processors

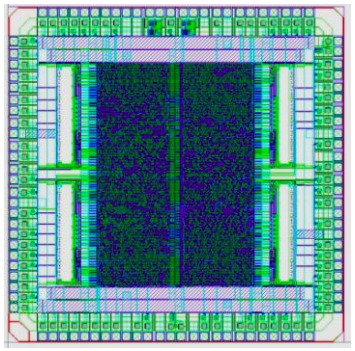


Hardware Description Language

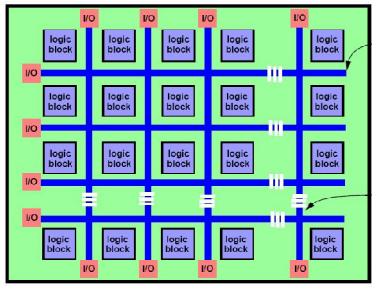
```
PROCESS (clk)
BEGIN
IF (clk'event and clk = '1') THEN
a_reg <= (a);
b_reg <= (b);
pdt_reg <= a_reg * b_reg;
adder_out <= adder_out + pdt_reg;
END IF;
END process;
accum_out <= (adder_out);
```

How to implement DPP algorithms for real time applications ?

DSP: Digital Signal Processor

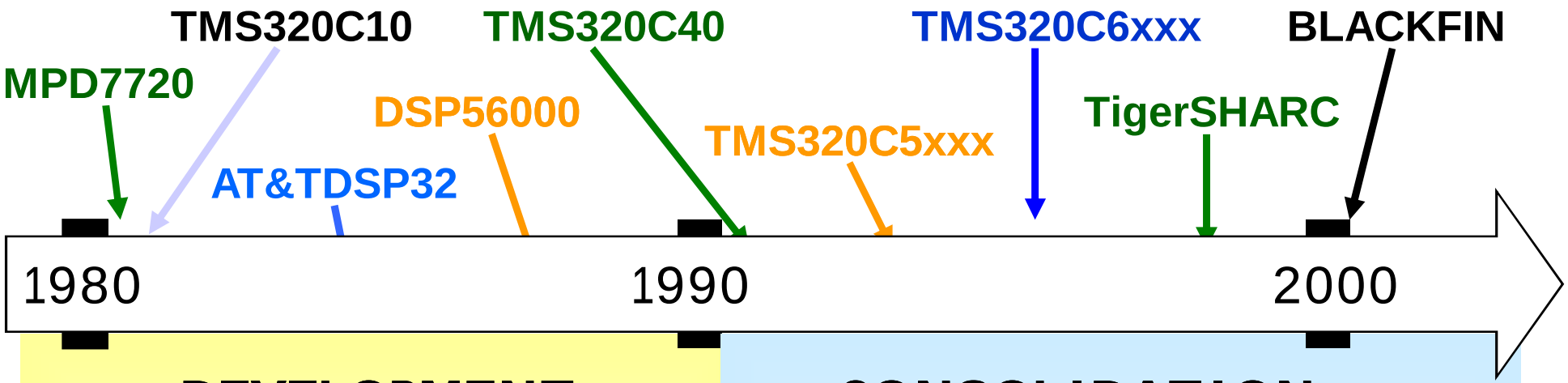


ASIC: Application Specific Integrated Circuit



FPGA: Field Programmable Gate Array

DSPs evolution: h/w features



DEVELOPMENT

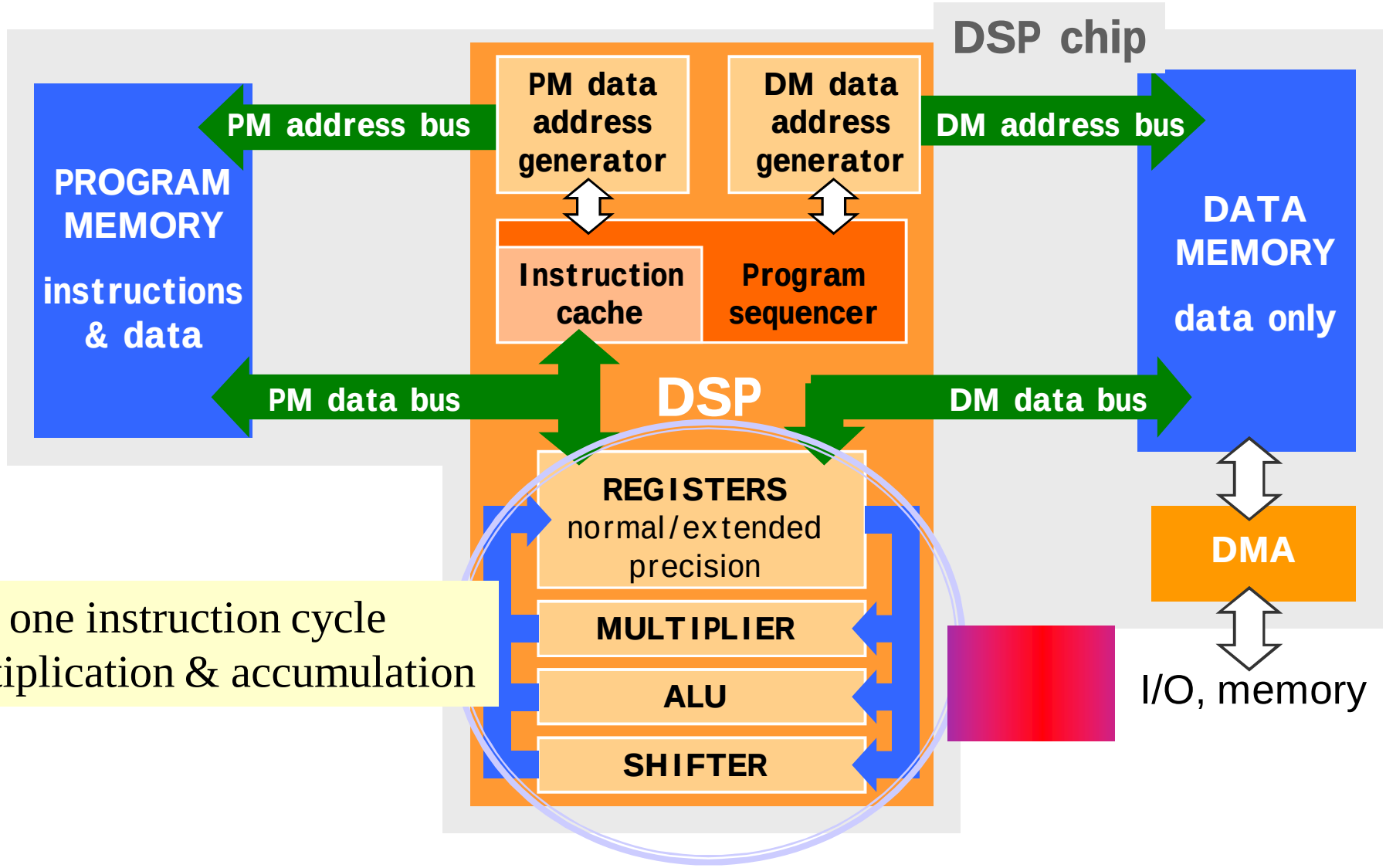
- Harvard architecture
- Data format:
  - early '80s: fixed point
  - late '80s: floating point (often *non* IEEE).
- DMA
- Fixed-width instruction set

CONSOLIDATION

- Parallel architectures
- Fewer manufacturers (TI & AD)
- Multiprocessing support

Fast computation

MAC (Multiply & Accumulate)-centered architecture



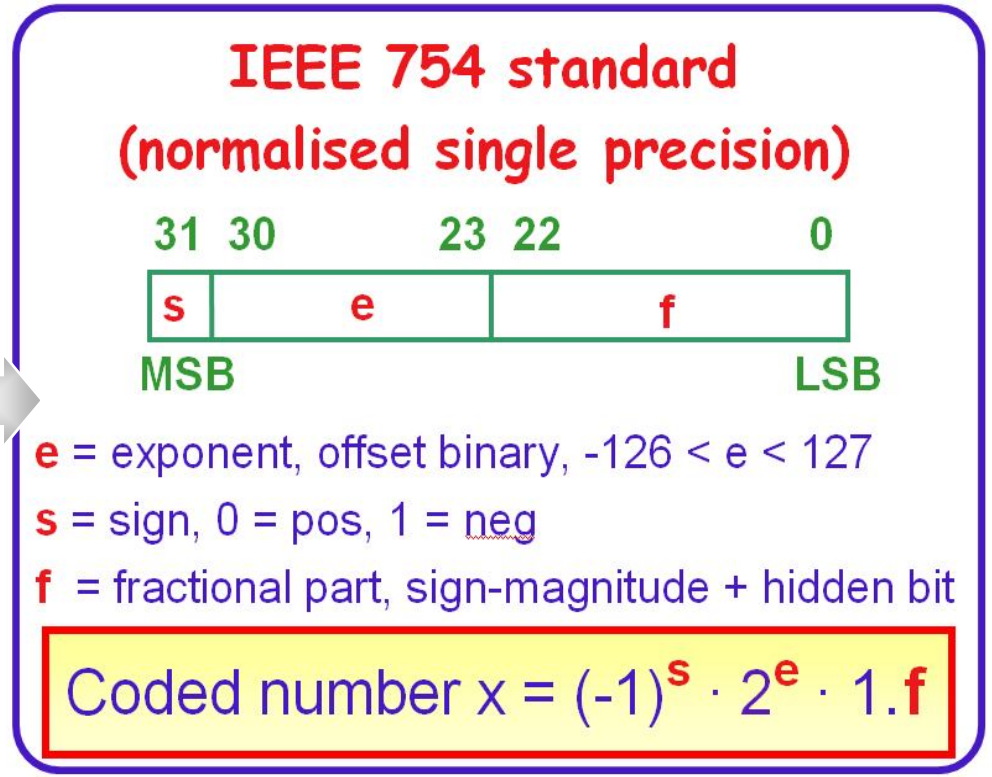
one instruction cycle
multiplication & accumulation

DSP programming

- DSPs: programmed by software.
- Languages:
  - Assembly
  - high-level languages (ANSI C, C extensions/dialects, C++, …)
- High-level software tools (ex. MATLAB, National Instruments …) to automatically generate files. → Rapid prototyping!
- Cross-compilation: code developed & compiled on different machine (PC, SUN…) then uploaded to DSP & executed.
- Code building tools from DSP manufacturers.
- Trend: more complex, powerful & user-friendly development tools.

Numerical fidelity

- Wide accumulators/registers for precision & overflow avoidance: *guard bits*.
- Overflow/underflow flags.
- Saturated arithmetic when overflowing.
- **Floating point arithmetic**: high dynamic range/precision.



$$\text{Dynamic range}_{dB} = 20 \log_{10}\left[\frac{\text{largest value}}{\text{smallest value}}\right]$$

Fixed point ~ 180 dB

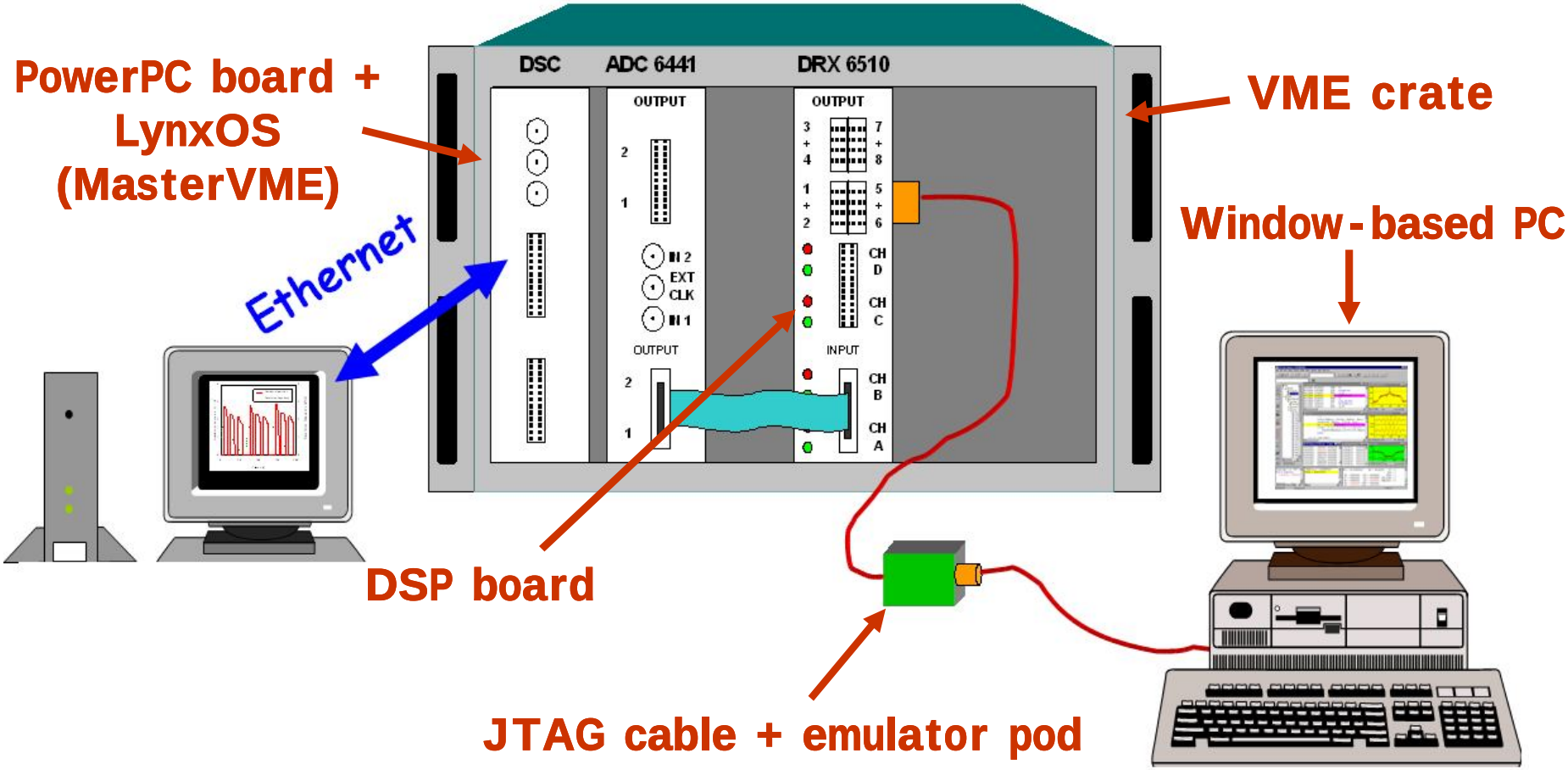
Floating point ~1500 dB

32 bits

Readout chain with DSPs

System use from Control Room

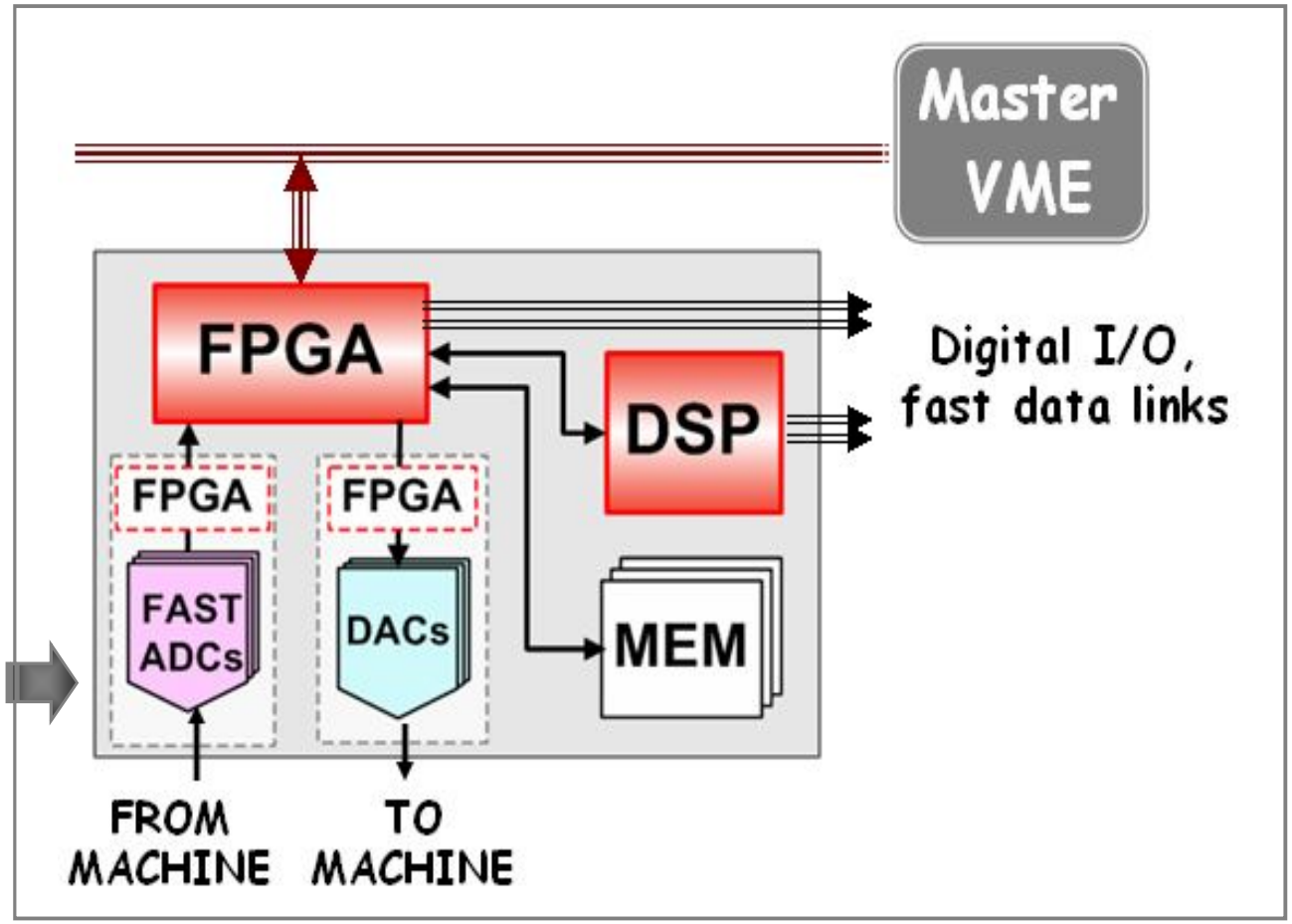
DSP code development/debugging



Code development setup. Example: AD beam intensity measurement (TI 'C40 DSP), CERN '98.

Readout chain with DSPs and FPGAs

Digital system: typical building blocks



Floating point designs

- To work in floating point you (potentially) need blocks to:
  - Convert from fixed point to floating point and back.
  - Convert between different floating point types.
  - Multiply.
  - Add/subtract (involves an intermediate representation with same exponent for both operands).
  - Divide.
  - Square root.
  - Compare 2 numbers.
- The main FPGA companies provide these in the form of IP cores. You can also roll your own.

Some performance figures (single precision)

Table 26: **Characterization of Single-Precision Format on Virtex-5 FPGA**

Operation	Resources				Maximum Frequency (MHz)[1]
	Embedded		Fabric		Virtex-5
	Type	Number	LUTs	FFs	-1
Multiply	DSP48E (max usage)	3	88	177	450
	DSP48E (full usage)	2	126	209	429
	DSP48E (medium usage)	1	294	390	375
	Logic	0	641	698	357
Add/Subtract	DSP48E (speed optimized, full usage)	2	267	375	410
	Logic (speed optimized, no usage)	0	429	561	395
	Logic (low latency)	0	536	625	372
Fixed to float	Int32 input		181	226	398
Float to fixed	Int32 result		218	237	373
Float to float	Single to double		44	101	466
Compare	Programmable		80	24	393
Divide	C_RATE=1		788	1,370	365
	C_RATE=26		227	233	316
Sqrt	C_RATE=1		542	787	398
	C_RATE=25		175	204	388

1. Maximum frequency obtained with map switches -ol high and -cm speed, and par switches -pl high and -rl high.

Fixed point (integers)

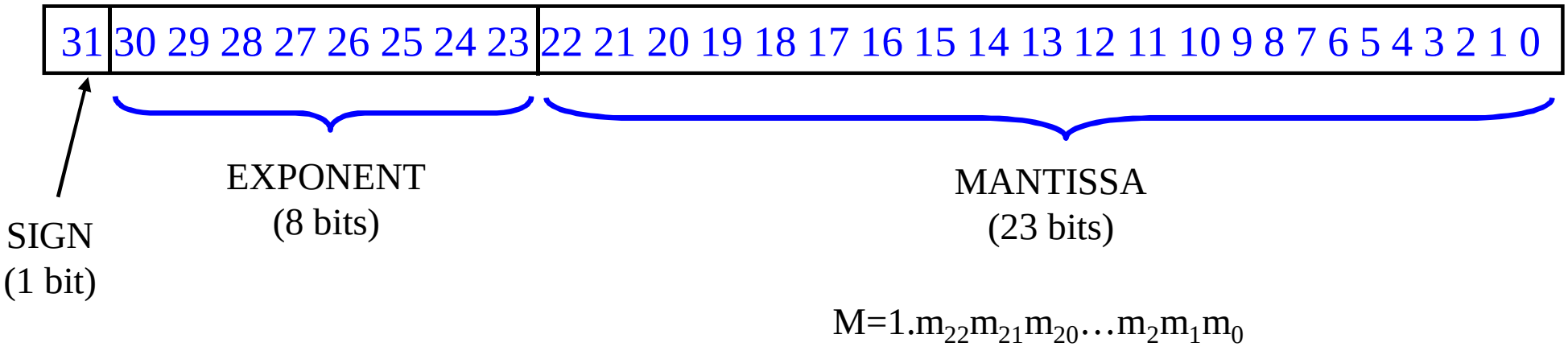
Unsigned integer		Sign & magnitude		2's complement	
Decimal	Bit pattern	Decimal	Bit pattern	Decimal	Bit pattern
15	1111	7	0111	7	0111
14	1110	6	0110	6	0110
13	1101	5	0101	5	0101
12	1100	4	0100	4	0100
11	1011	3	0011	3	0011
10	1010	2	0010	2	0010
9	1001	1	0001	1	0001
8	1000	0	0000	0	0000
7	0111	0	1000	-1	1111
6	0110	-1	1001	-2	1110
5	0101	-2	1010	-3	1101
4	0100	-3	1011	-4	1100
3	0011	-4	1100	-5	1011
2	0010	-5	1101	-6	1010
1	0001	-6	1110	-7	1001
0	0000	-7	1111	-8	1000

Fixed point (fractional numbers)

digit worth								decimal value
$-(2^2)$	2^1	2^0 •	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	
-4	**2**	**1** •	**0.5**	**0.25**	**0.125**	**0.0625**	**0.03125**	
0	0	0 •	0	0	0	0	1	0.03125
0	0	0 •	0	0	0	1	0	0.0625
1	0	1 •	0	0	0	0	0	-3.0
1	1	0 •	0	0	1	1	1	-1.78125
1	1	1 •	1	1	1	1	1	-0.03125

Example: 3 integer bits and 5 fractional bits

Floating point binary numbers



$$\text{Value} = (-1)^S \times M \times 2^{E-127}$$

Max value: $\pm (2-2^{23}) \times 2^{128} = \pm 6.8 \times 10^{38}$

Min value: $\pm 1.0 \times 2^{-127} = \pm 5.9 \times 10^{-39}$