Hands-on Tutorial

Rainer Hentges

Philipp Horn

Introduction to FPGAs

A. Straessner

Terascale Detector Workshop
Dresden
March 12-15, 2019

General Concepts

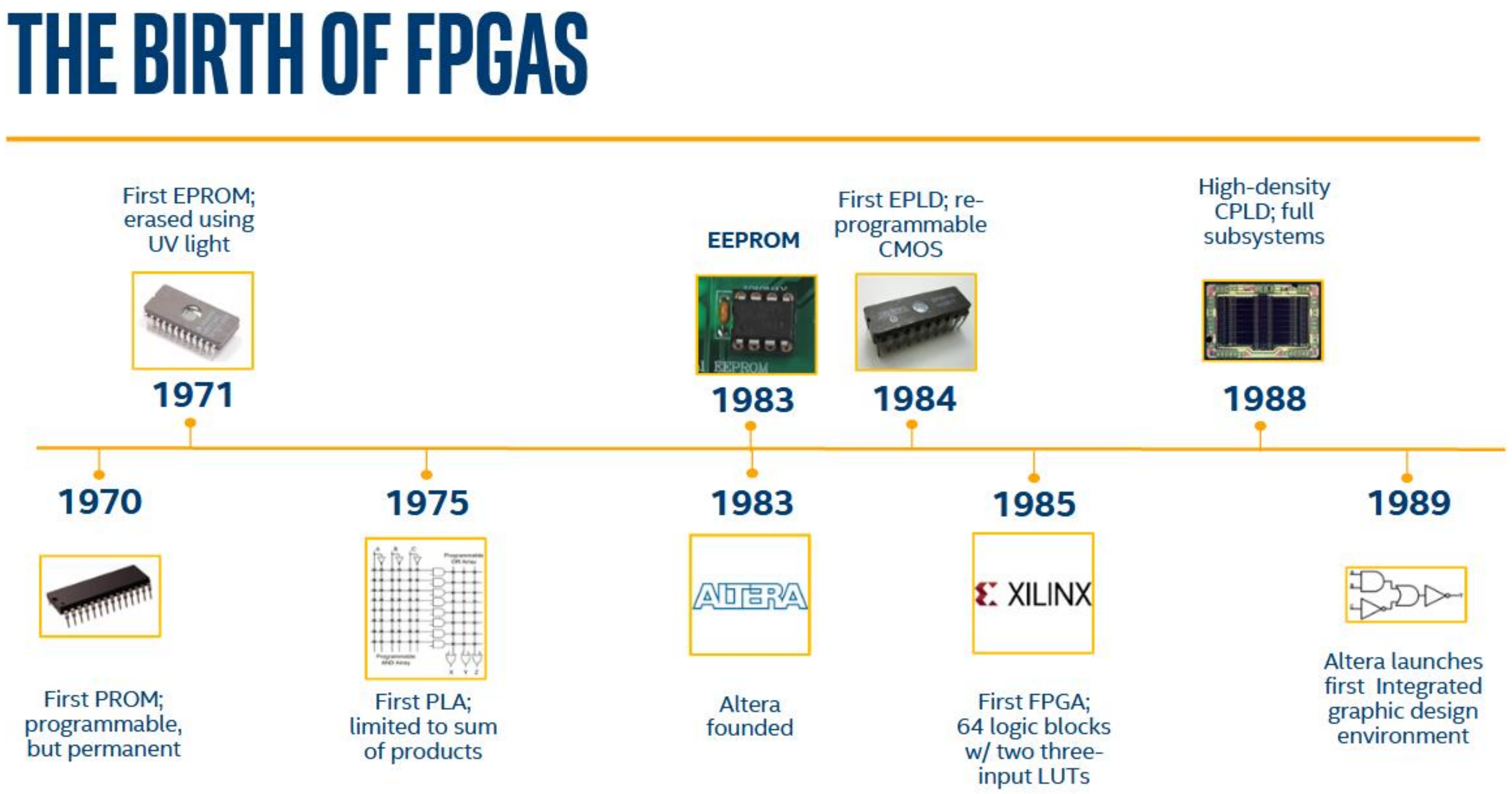
Field Programmable Gate Array

- Field Programmable Gate Array is an integrated electronic circuit
  - "field programmable" = can be programmed "in the field", i.e. not when it is fabricted but at the place and time of application
  - "gate array" = array of electronic logic gates - and much more

- You can "build" an electronic circuit which can be reconfigured any time
- Hardware Description Languages (HDL) and software tools help to realize your design:
  - VHDL = Very High Speed Integrated Circuit Hardware Description Language
  - Verilog
- High level languages: "C", OpenCL, ...
- Programming tools: graphical design tools, CAD, Labview, ...

→ FPGA firmware

FPGA History



THE BIRTH OF FPGAS

1970 — First PROM; programmable, but permanent

1971 — First EPROM; erased using UV light

1975 — First PLA; limited to sum of products

1983 — EEPROM

1983 — Altera founded

1984 — First EPLD; re-programmable CMOS

1985 — First FPGA; 64 logic blocks w/ two three-input LUTs

1988 — High-density CPLD; full subsystems

1989 — Altera launches first Integrated graphic design environment

- PROM = programmable read-only memory
- PLA = programmable logic array
- PLD = programmable logic device
- LUT = look-up table

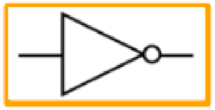
Digital Elements

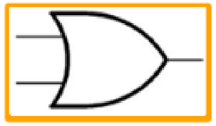
BACK TO THE BASICS

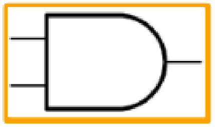
A	Z
0	1
1	0

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

A	B	Z
0	0	0
0	1	0
1	0	0
1	1	1

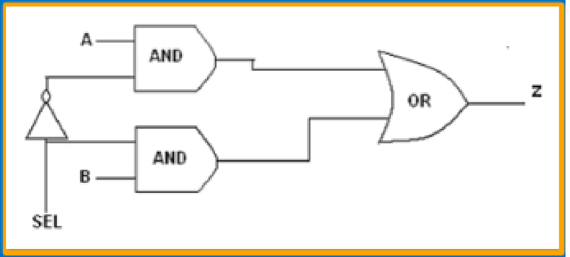

Inverter
Z = ~A


OR
Z = A | B


AND
Z = A & B

S	A	B	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

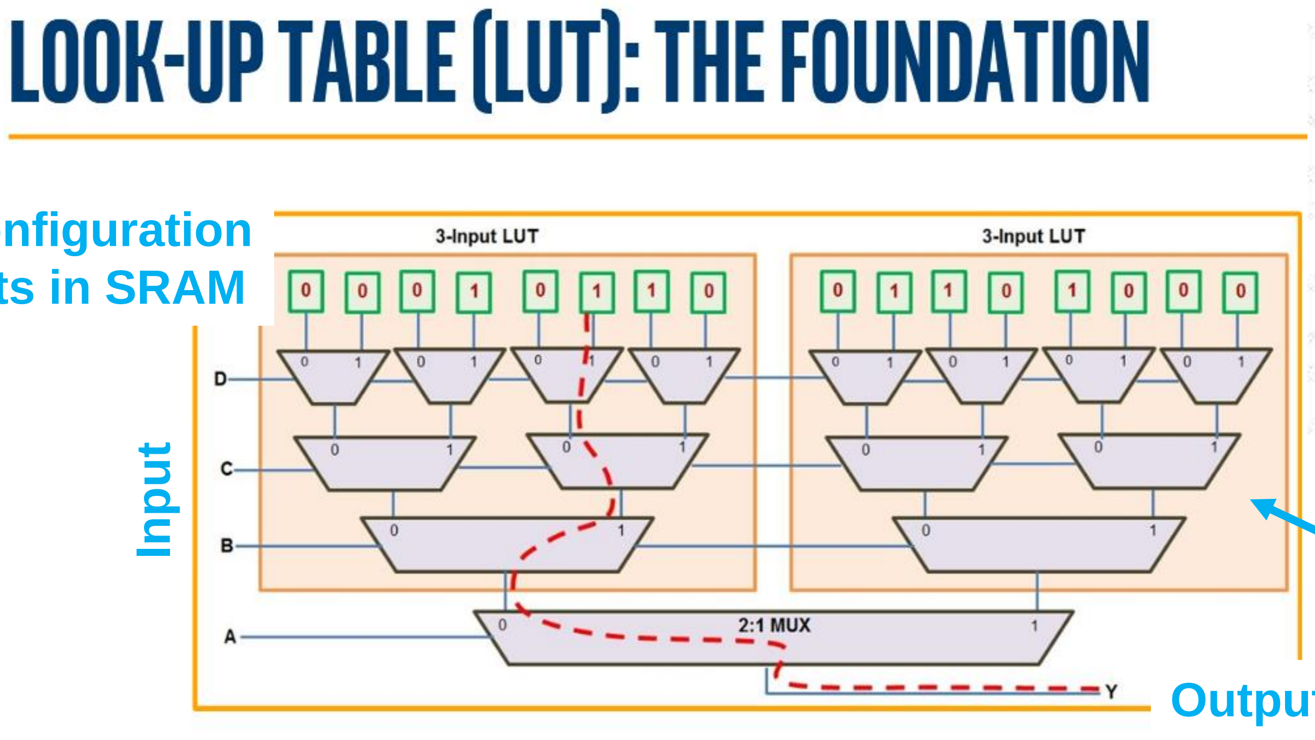
S = select



2:1 MUX
Z = (~S & A) | (S & B)

- MUX = multiplexer

Digital Logic

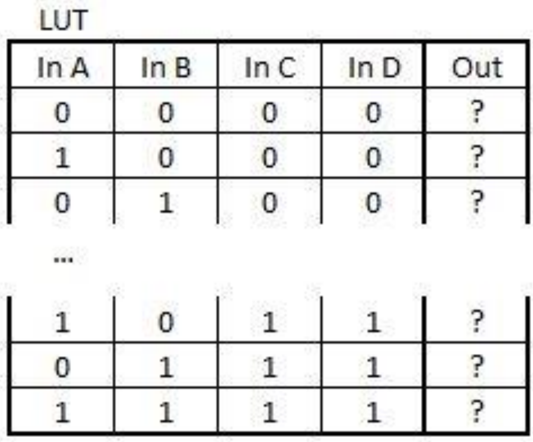


Configuration Bits in SRAM

Input

2:1 MUXes

Output



LUT

In A	In B	In C	In D	Out
0	0	0	0	?
1	0	0	0	?
0	1	0	0	?
...				
1	0	1	1	?
0	1	1	1	?
1	1	1	1	?

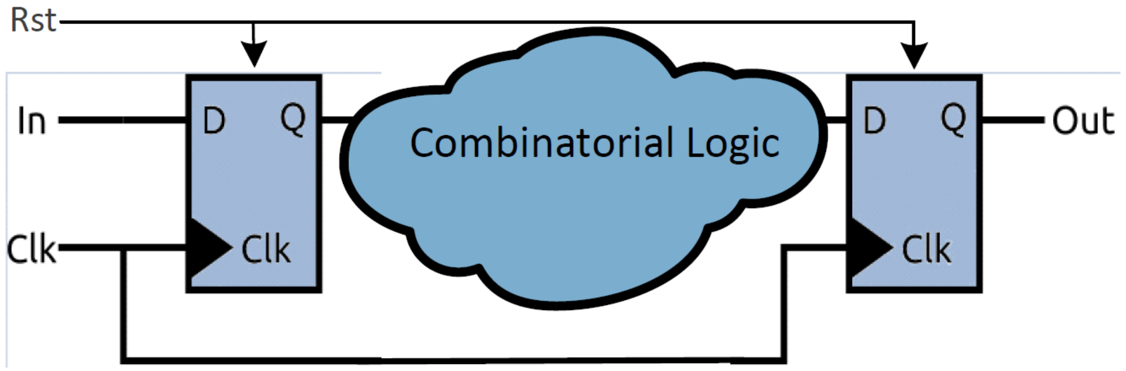
- A logic function can be implemented in a look-up-table (truth table)

- The input addresses bits in a configuration SRAM (static random access memory)
- SRAM of a LUT is programmed when the FPGA gets configured

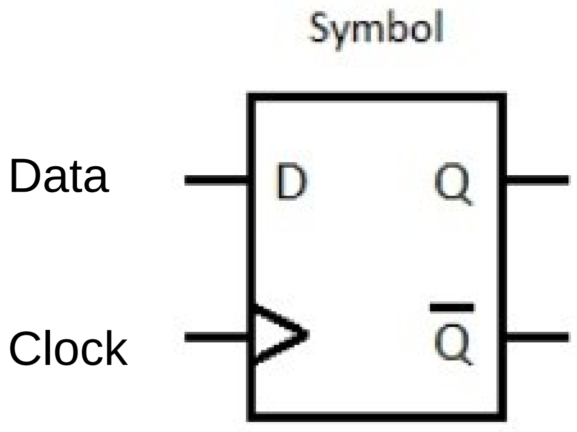
- Examples:
    - a 4-input LUT requires 16 bit SRAM and a tree of 2:1 multiplexers

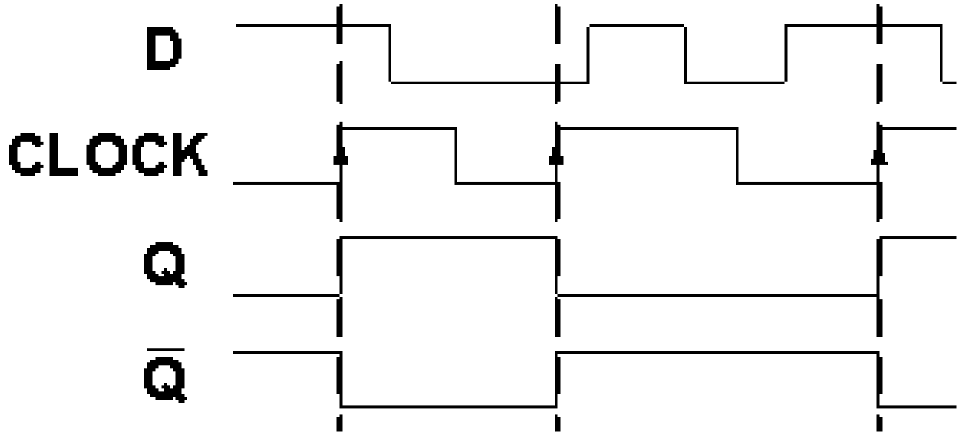
Registers and D-flip-flops

- In a typical FPGA building block the combinatorial logic is surrounded by registers
- A register samples and holds the input data (D) for a certain time and sends an output signal (Q) when a clock signal arrives = 1-bit temporal memory



- A register realized as a (synchronous) D-flip-flop (latch) clocked on the rising edge:

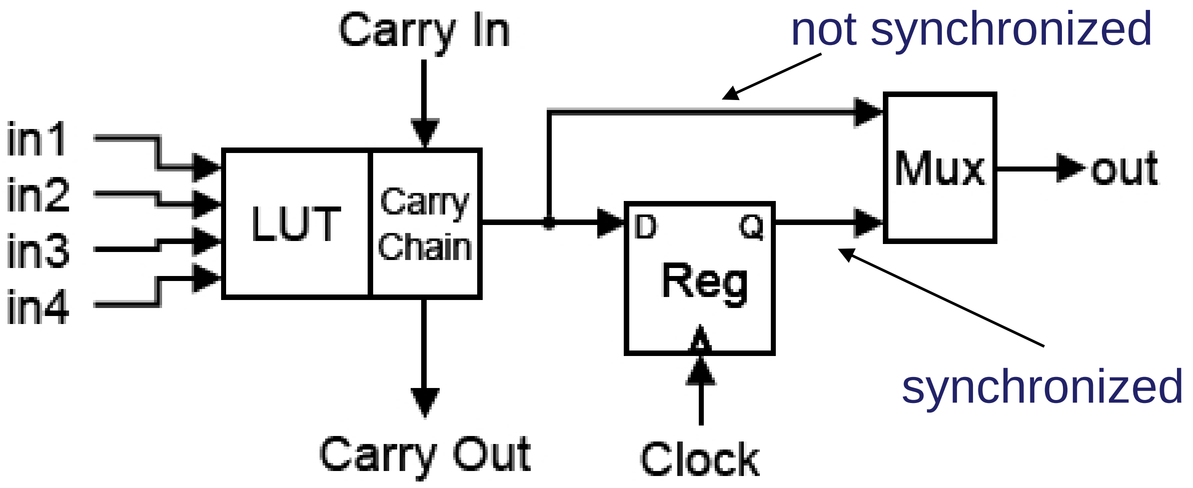




- Result: sequence of logic operations controlled by clock → synchronous design

The Building Blocks

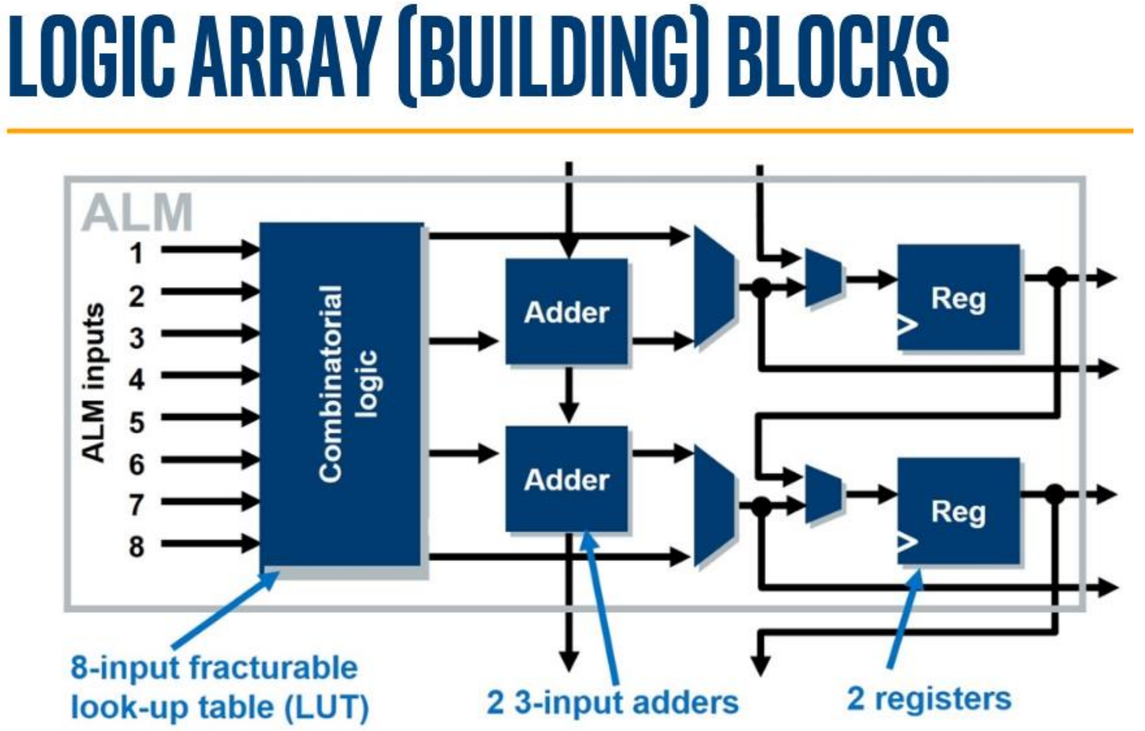
- Configurable Logic Block:
  - LUT
  - Carry chain
  - D-flip-flop / register
  - MUX



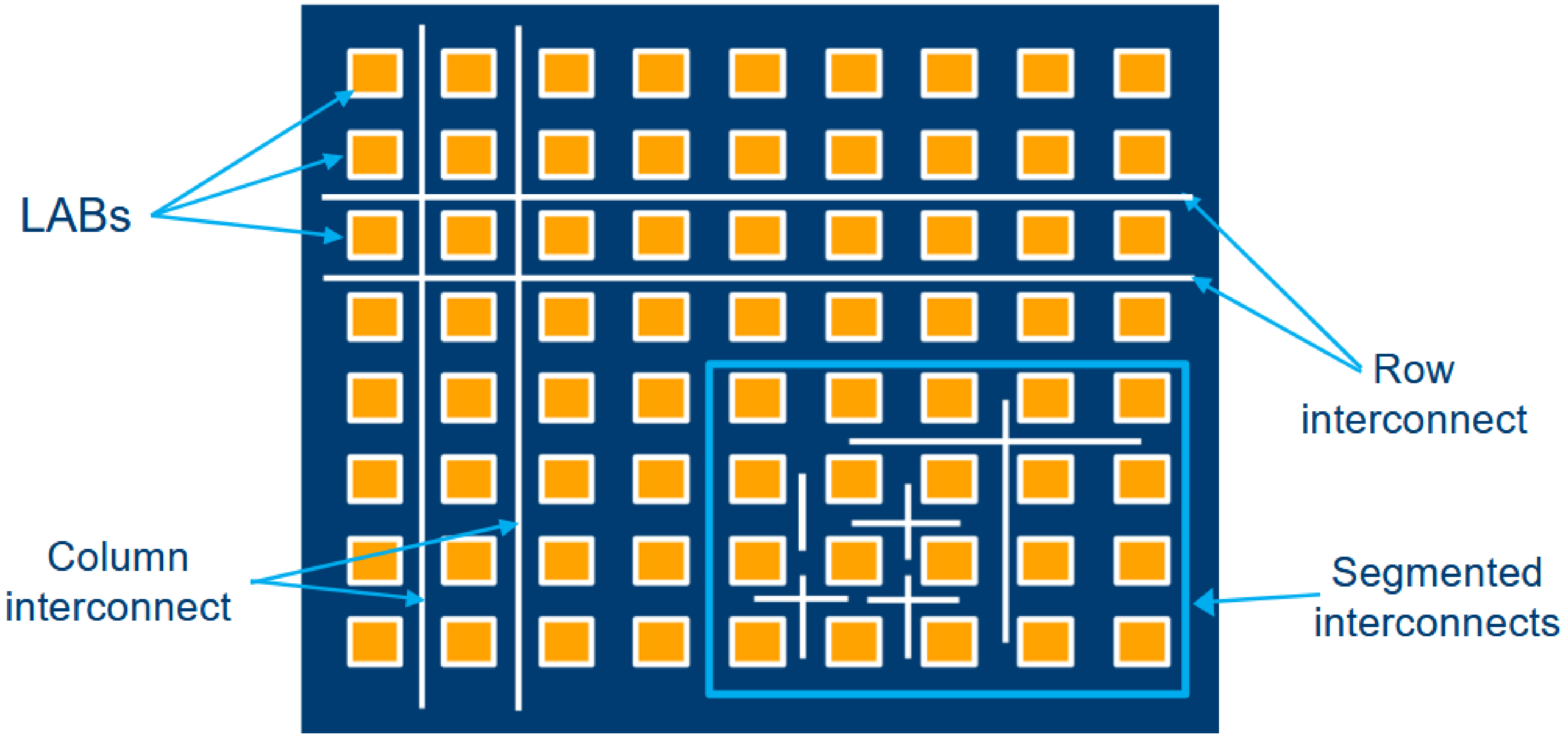
- Structure of logic building block depends on your FPGA model

- another example:
  - ALM = Adaptive Logic Module

- Usually, the design software takes care of implementing your design into logic blocks



The Array = Combinatorial Logic + Interconnects



- LAB = Logic Array Block

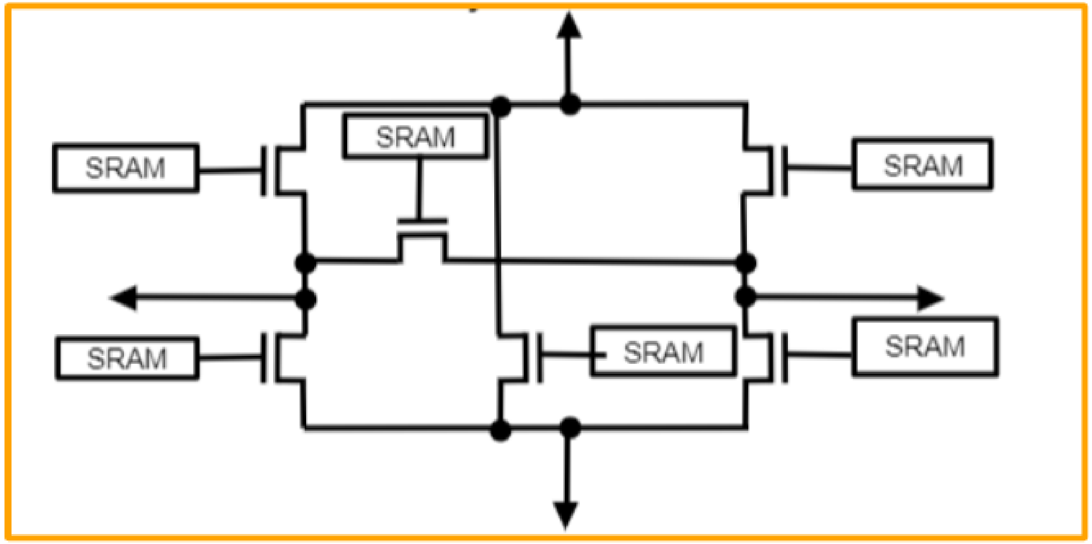
- Interconnects are crucial:
  - in large FPGAs there are hundreds of layers of configurable interconnects
  - it is a network of signal lines and programmable multiplexers

Programming the FPGA

- Configuration of FPGA corresponds to setting SRAM bits to define behaviour of LUTs, interconnects, etc.

- Example:

Row/Column Interconnect Junction

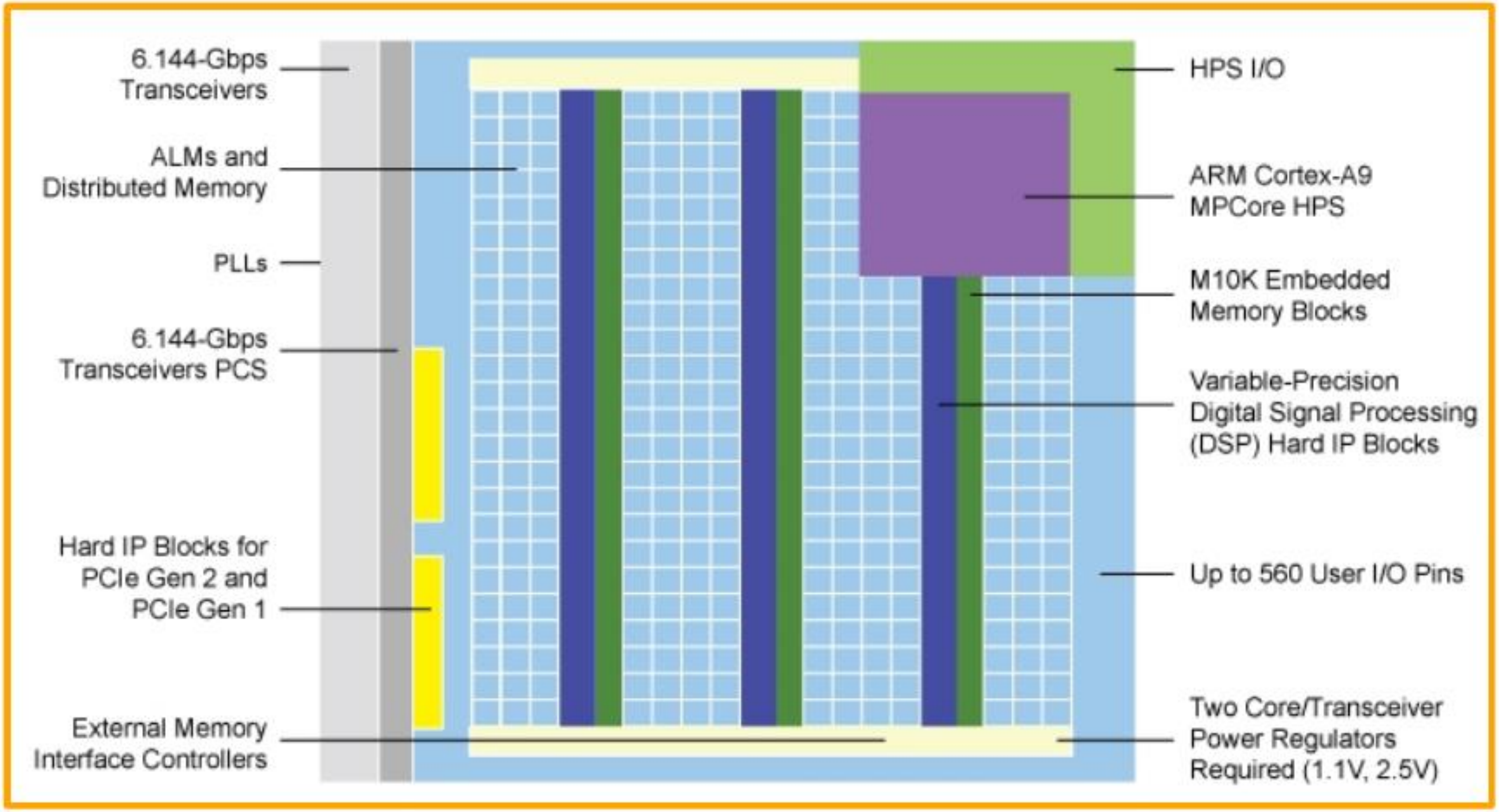


- SRAM is a volatile memory element

- Active configuration is typically performed at power-up sequence

- Programming information is stored in external non-volative device (e.g. EEPROM)

A More Complex FPGA


6.144-Gbps Transceivers
ALMs and Distributed Memory
PLLs
6.144-Gbps Transceivers PCS
Hard IP Blocks for PCIe Gen 2 and PCIe Gen 1
External Memory Interface Controllers
HPS I/O
ARM Cortex-A9 MPCore HPS
M10K Embedded Memory Blocks
Variable-Precision Digital Signal Processing (DSP) Hard IP Blocks
Up to 560 User I/O Pins
Two Core/Transceiver Power Regulators Required (1.1V, 2.5V)

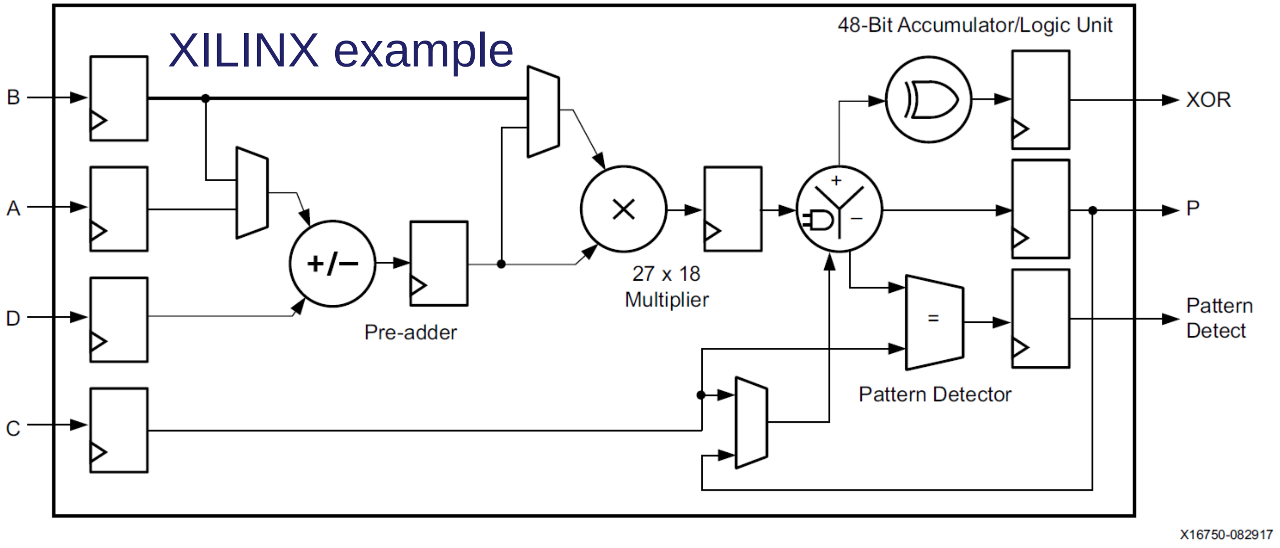
- ALM = Adaptive Logic Module
- PLL = phase locked loop - removes skew between external input clock and internal clock and is necessary for low-skew clock networks
- HPS = hard processor system
- DSP = digital signal processor

Digital Signal Processors (DSP)

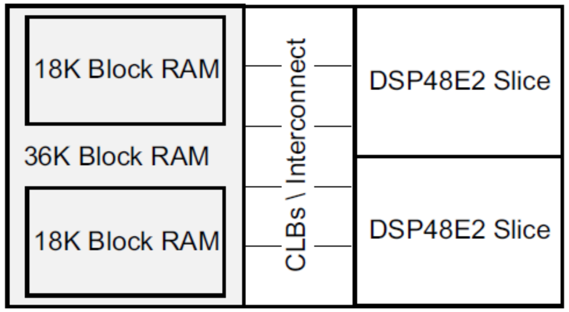
- DPSs are made for multiplying and adding integers

- Application e.g. in digital signal filters: $A = \sum_{i=1}^{n} a_i S_i$
- Floating point operation is usually fully supported, but may not be optimal

XILINX example



memory , logic block, DSP

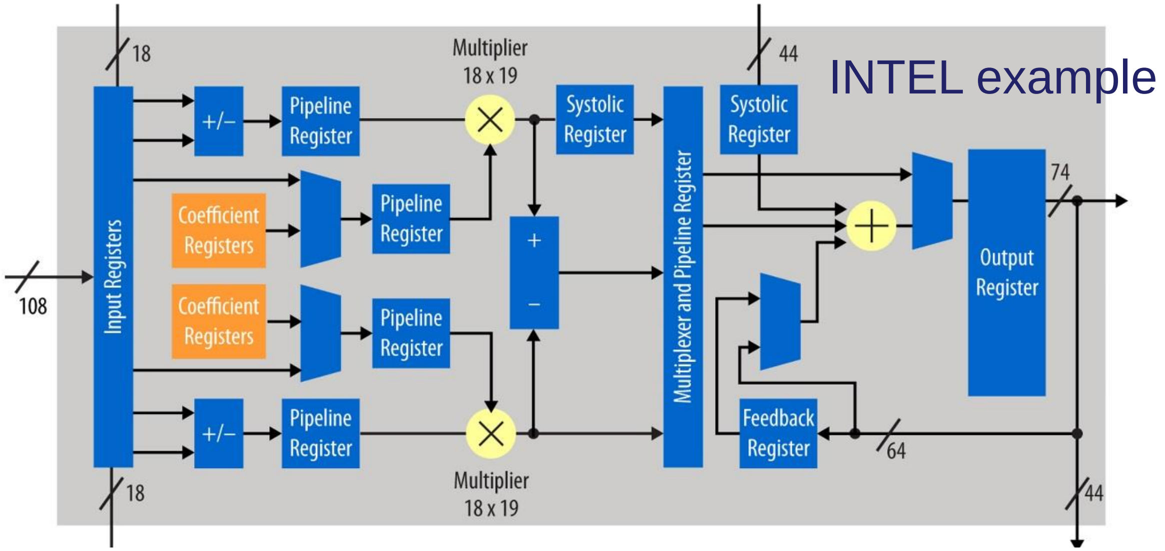


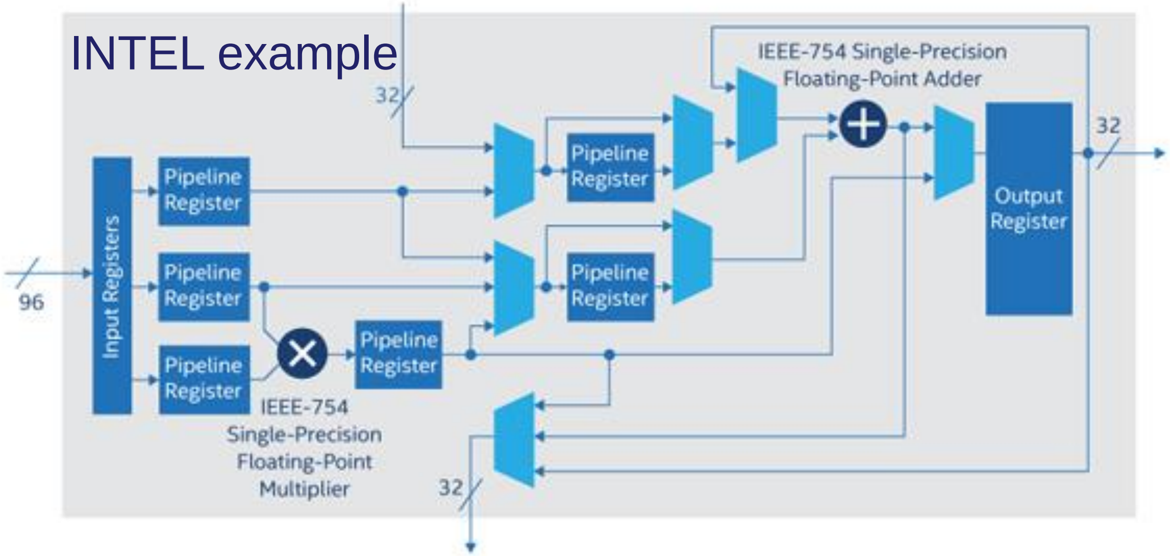
- Multiplier and accumulator: main operation
- Logic unit: bitwise AND, OR, NOT, NAND, NOR, XOR, and XNOR
- Pattern detector: terminal counts, overflow/underflow, rounding support

- Full "DSP tile" comes with surrounding logic and memory

Digital Signal Processors (DSP)

- Some other architectures:
  - regular fixed point multiplier-adder
  - floating point multiplier-adder

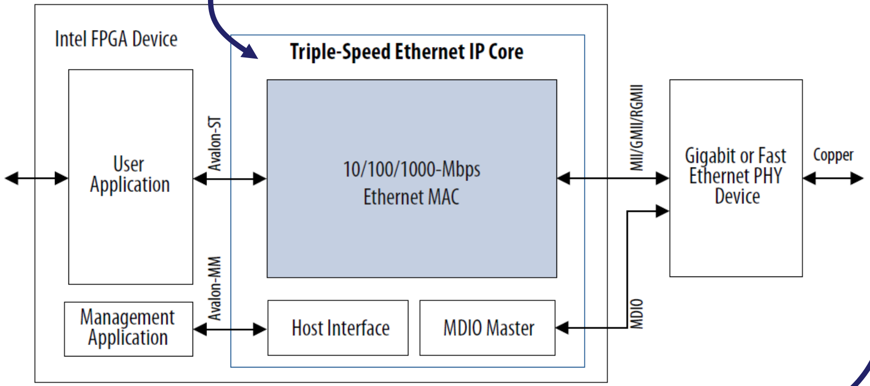


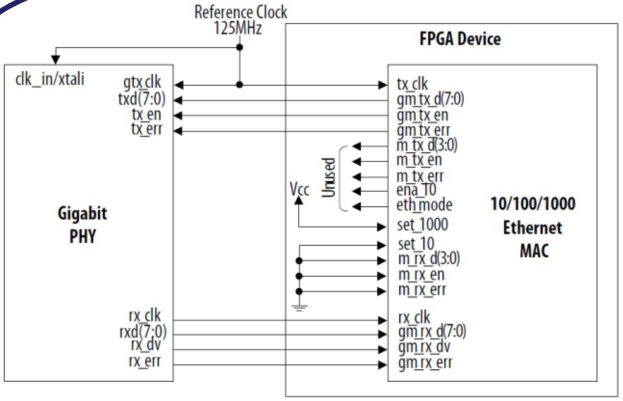


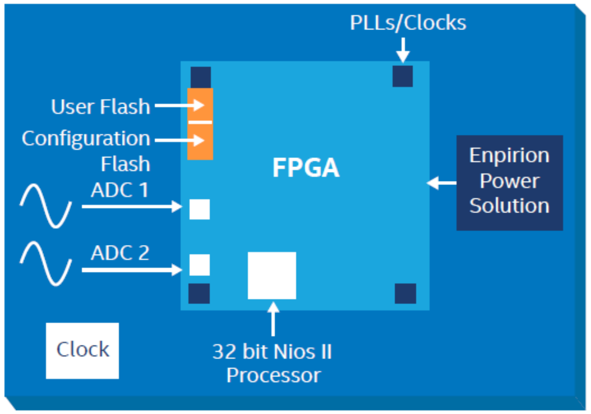
Intellectual Property (IP) Blocks

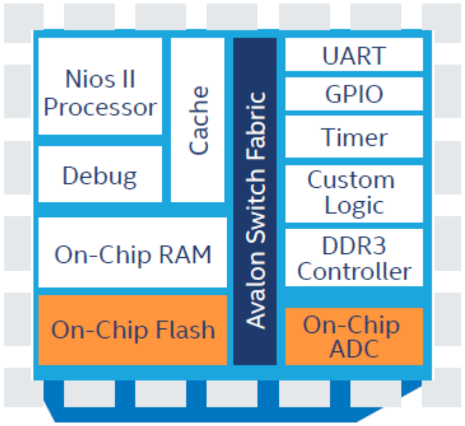
- IP = intellectual property
- IP block / IP core:
  - complete module, ready to use
  - developed by your colleagues or a company
  - you may need a licence even if it is for free, some products can be expensive

- Examples: Ethernet controller, PCIe controller, soft processor, multiplier functions, ...



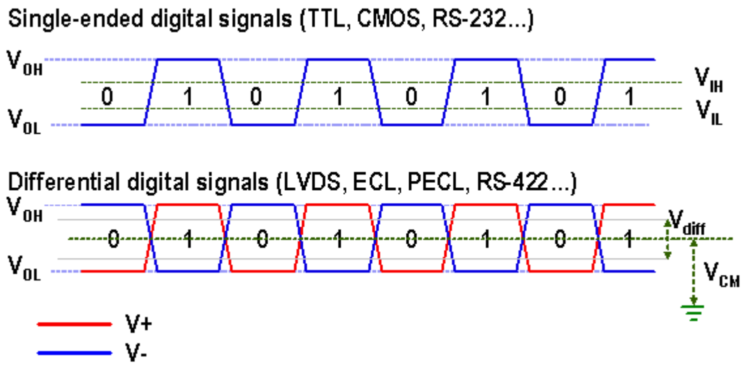




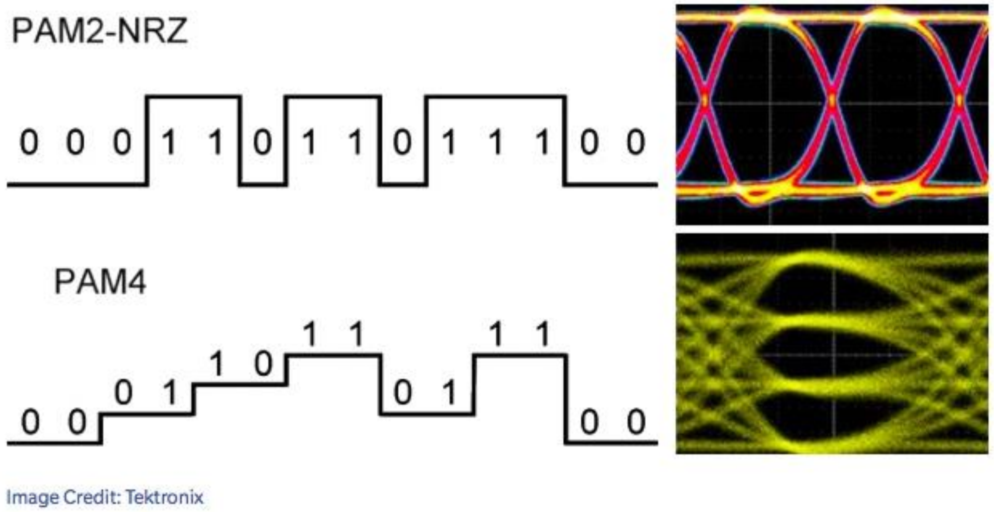


Input / Output

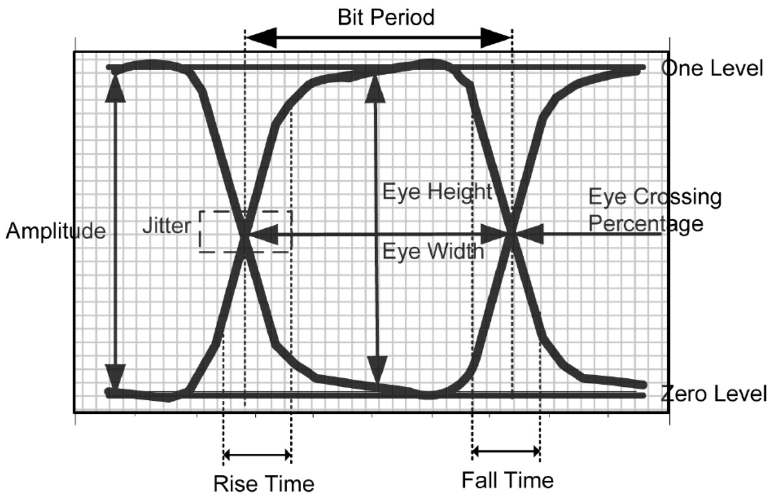
- Modern FPGAs have many I/O capabilities:
  - GPIO = general purpose I/O, single pin connection
  - Low Voltage Differential Signaling (LVDS) lines
    - few gigabit / second (Gbps)
  - multi-Gb transceivers
    - up to 30+ Gbps per I/O pair
    - PAM4 up to 60+ Gbps per I/O pair



Single-ended digital signals (TTL, CMOS, RS-232...)
V_{OH}
V_{OL}
V_{IH}
V_{IL}
0 1 0 1 0 1 0 1
Differential digital signals (LVDS, ECL, PECL, RS-422...)
V_{OH}
V_{OL}
V_{diff}
V_{CM}
0 1 0 1 0 1 0 1
V+
V-



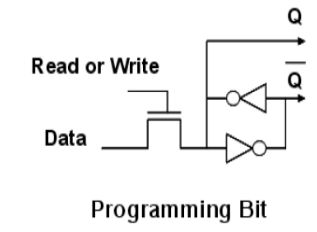
"eye diagram"



Technology

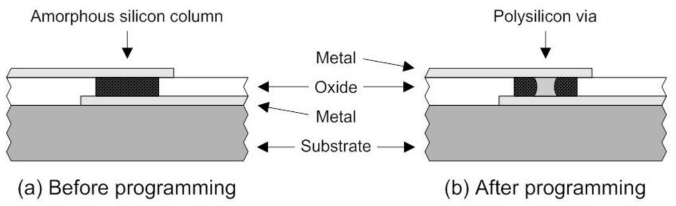
FPGA Technology

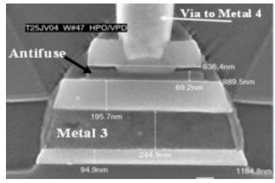
- most FPGAs are based on SRAM / CMOS technology:
  - configuration happens at power-on and is erased at power-off
  - configuration is sensitive to radiation
  - transistor sizes: smaller feature size → faster, lower power



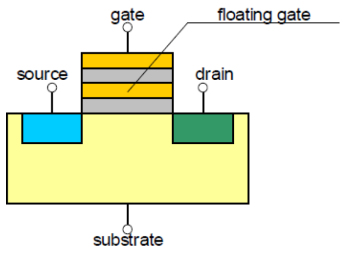
45 nm	28 nm	20 nm	16 nm	14 nm	28 nm	40 nm
SPARTAN 6	VIRTEX 7, KINTEX 7, ARTIX 7, SPARTAN 7	VIRTEX UltraSCALE, KINTEX UltraSCALE	VIRTEX UltraSCALE+, KINTEX UltraSCALE+	intel STRATIX 10 inside — Intel® Stratix® 10 Introduced in 2013, 14 nm Tri-Gate technology	intel STRATIX inside — Stratix® V Introduced in 2010, 28 nm technology	intel STRATIX inside — Stratix® IV Introduced in 2008, 40 nm technology

- anti-fuse technology:
  - contact is grown to make a connection: amorphous silicon or metal-to-metal connections, non-volatile
  - fast, one-time programmable
  - configuration is immune to radiation





- flash-memory technology:
  - flash-EEPROM
  - non-volatile, slow
  - configuration is immune to radiation

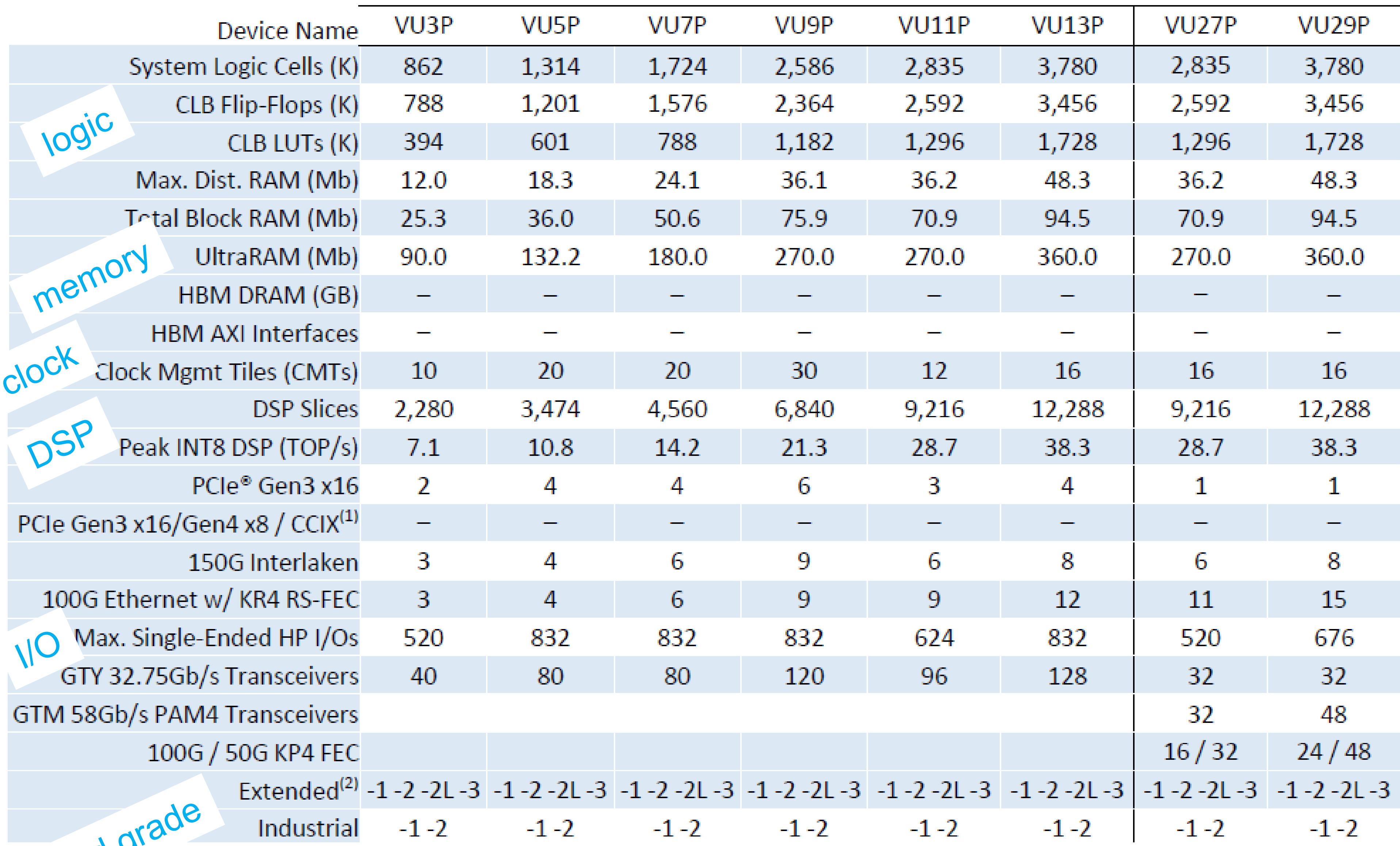


FPGA Producers and Costs

- Xilinx and INTEL/Altera: market leaders, SRAM-based FPGAs
- Microsemi (previously Actel, now Microchip), producing antifuse, flash-based, mixed-signal FPGAs
- Lattice Semiconductor: low-power SRAM-based FPGAs, non-volatile configuration
- QuickLogic: low-power, low-density SRAM-based FPGAs
- Achronix: SRAM-based FPGAS with fast 1.5 GHz fabric speed
- Aeroflex / Cobham: radiation tolerant SRAM-based FPGAs
- Nanoexplore: SRAM-based FPGAs
- FPGA cost range:

€ $\Rightarrow$ €€€€€

Example Product Overview



Device Name	VU3P	VU5P	VU7P	VU9P	VU11P	VU13P	VU27P	VU29P
System Logic Cells (K)	862	1,314	1,724	2,586	2,835	3,780	2,835	3,780
CLB Flip-Flops (K)	788	1,201	1,576	2,364	2,592	3,456	2,592	3,456
CLB LUTs (K)	394	601	788	1,182	1,296	1,728	1,296	1,728
Max. Dist. RAM (Mb)	12.0	18.3	24.1	36.1	36.2	48.3	36.2	48.3
Total Block RAM (Mb)	25.3	36.0	50.6	75.9	70.9	94.5	70.9	94.5
UltraRAM (Mb)	90.0	132.2	180.0	270.0	270.0	360.0	270.0	360.0
HBM DRAM (GB)	–	–	–	–	–	–	–	–
HBM AXI Interfaces	–	–	–	–	–	–	–	–
Clock Mgmt Tiles (CMTs)	10	20	20	30	12	16	16	16
DSP Slices	2,280	3,474	4,560	6,840	9,216	12,288	9,216	12,288
Peak INT8 DSP (TOP/s)	7.1	10.8	14.2	21.3	28.7	38.3	28.7	38.3
PCIe® Gen3 x16	2	4	4	6	3	4	1	1
PCIe Gen3 x16/Gen4 x8 / CCIX(1)	–	–	–	–	–	–	–	–
150G Interlaken	3	4	6	9	6	8	6	8
100G Ethernet w/ KR4 RS-FEC	3	4	6	9	9	12	11	15
Max. Single-Ended HP I/Os	520	832	832	832	624	832	520	676
GTY 32.75Gb/s Transceivers	40	80	80	120	96	128	32	32
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
Industrial	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2	-1 -2

logic

memory

clock

DSP

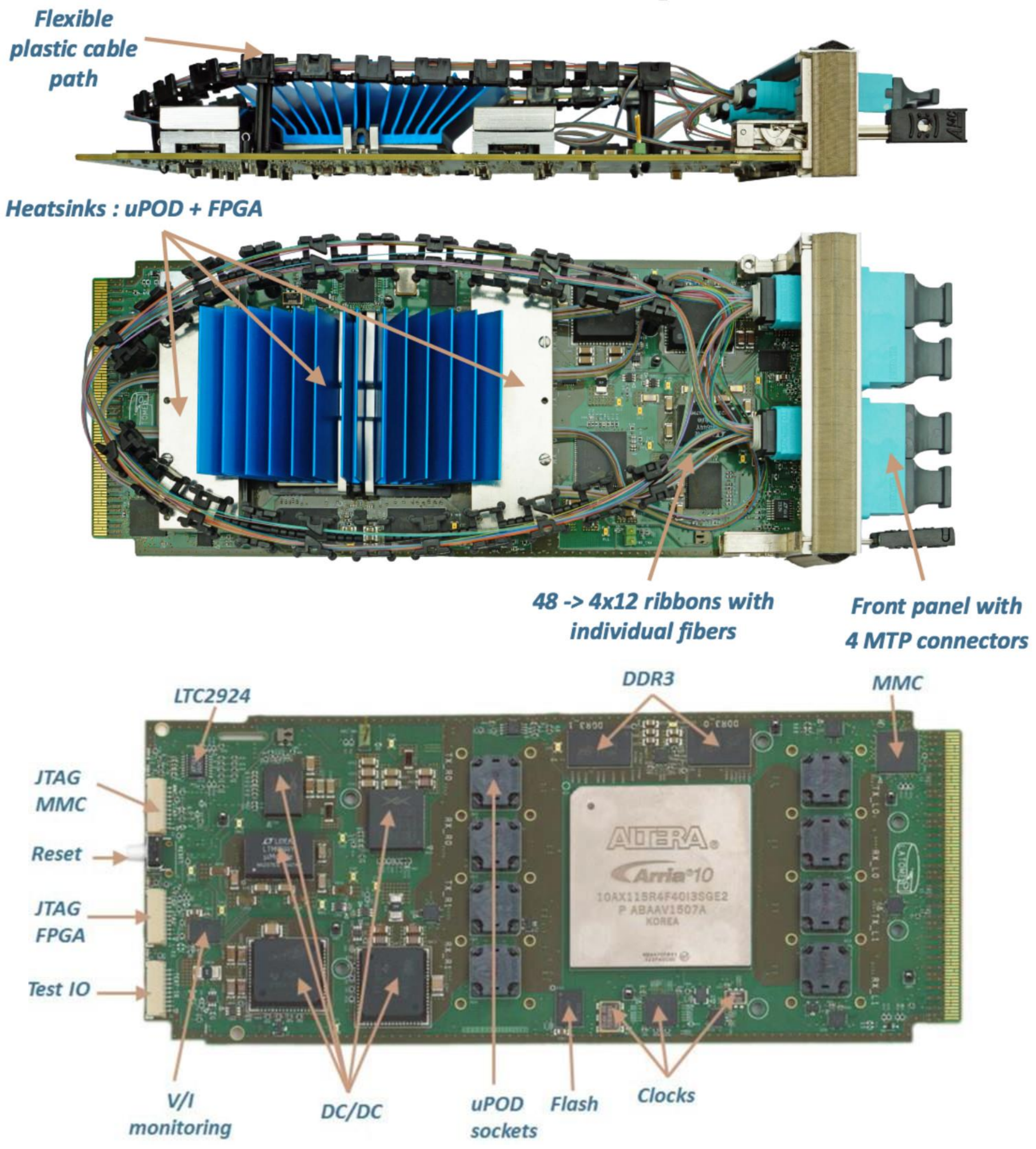
I/O

speed grade

Application Example

- Digital signal processing board for the ATLAS Liquid-Argon calorimeter trigger readout
- 320 detector channels at 40 MHz
- 2 digital filters per channel for energy and time measurement
- 2 x 48 transceivers at 5-12 Gpbs

€€€



Choose the best solution for your problem

- CPU (PC):
  - floating point and fixed point calculations
  - multi-core, clock speed: multi GHz
  - large command set, 64 bit+
  - only sequential operations possible (one or few Arithmetic Logical Units, ALUs)
  - I/O via bus systems
  - large memory, data storage
  - programmable with C/C++, Python, ...

- Graphics Processor Unit (GPU):
  - parallel processing
  - thousands of cores
  - clock speed: multi GHz
  - I/O limited
  - programmable in dedicated parallel processing frameworks

- Digital Signal Processor (DSP):
  - fast real-time floating point and fixed point calculations
  - clock speed: multi GHz
  - limited I/O capabilities
  - memory size OK
  - programmable with C/C++, special commands

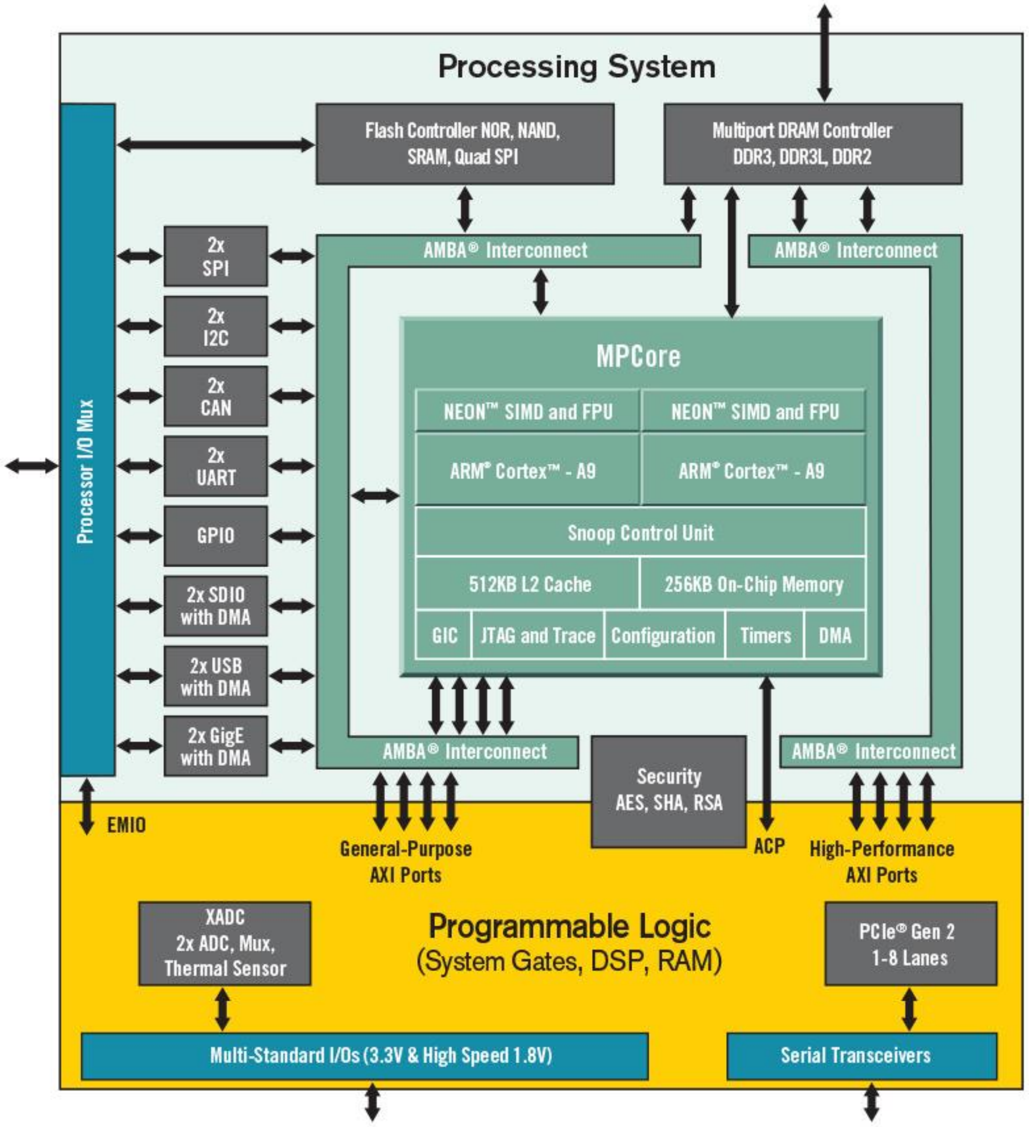
- FPGA:
  - real-time logic and signal operations
  - fixed/floating point calculations with integrated DSP slices
  - very good in parallel data processing in real time
  - very fast, multiple I/O data links
  - memory size OK
  - programmable with HDL, but also graphic CAD, System-C, C++, ...
  - radiation tolerant versions

- ASIC
  - combination of digital and analog circuits possible
  - highly integrated
  - custom design for large number of chips
  - fixed design reduces flexibility
  - ASIC/transistor technology allow optimized applications (e.g. radiation tolerance, ...)
  - design can be evaluated with FPGA and implemented into ASIC

System-on-Chip (SoC)

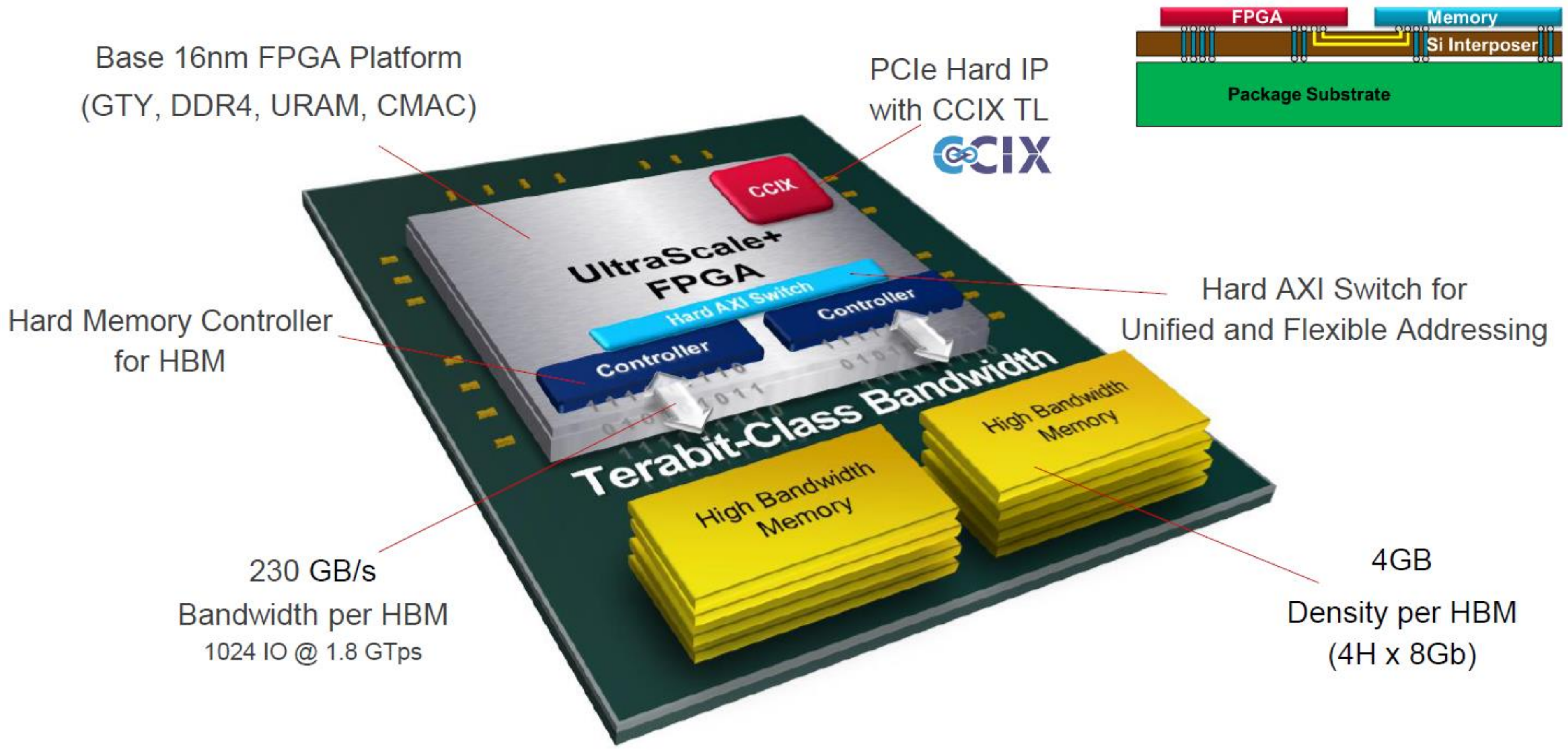
- FPGA and hard processor system (HPS) are combined in one chip: System-on-Chip

- typically ARM processors are used
- direct interconnect between FPGA fabric and processor
- some tasks are easier to handle by software/CPU:
  - high level system tasks
  - control and monitoring functionality
  - application interfaces
- FPGA takes care of fast, parallel data processing and fast I/O



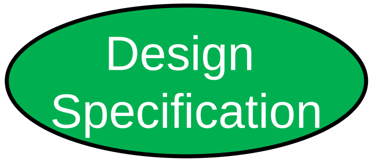
High Bandwidth Memory

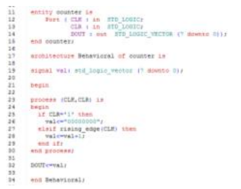
- In the past, FPGAs were limited in memory capabilities
- Recent FPGAs have larger resources internally (hundreds of Mb)
- Even more: high bandwidth memory



Project Design Flow

Classic FPGA Design Flow

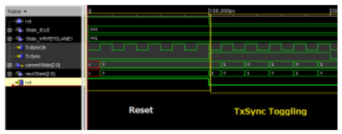




```
entity counter is
    Port ( CLK : in  STD_LOGIC;
           CLR : in  STD_LOGIC;
           DOUT : out  STD_LOGIC_VECTOR (7 downto 0));
end counter;

architecture Behavioral of counter is

signal val: std_logic_vector (7 downto 0);

begin

process (CLK,CLR) is
begin
  if CLR='1' then
    val<="00000000";
  elsif rising_edge(CLK) then
    val<=val+1;
  end if;
end process;

DOUT<=val;

end Behavioral;
```

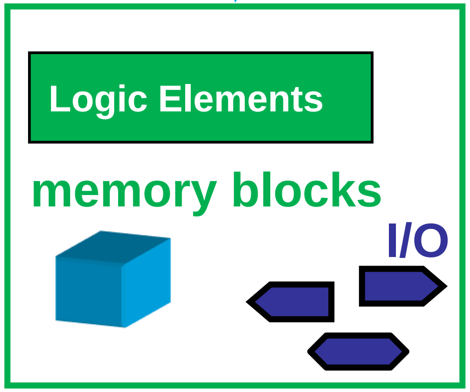
Design Entry/Coding

- behavioral or structural description of design



Register Transfer Level (RTL) Simulation

- Functional simulation
- Verify logic model & data flow (no timing delays)



Synthesis

- Translate design into device specific primitives
  - implementation in one specific FPGA type
- Optimization to meet required area & performance constraints

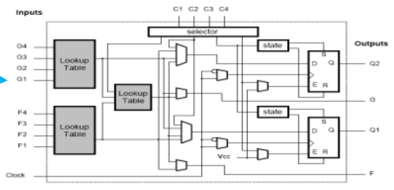


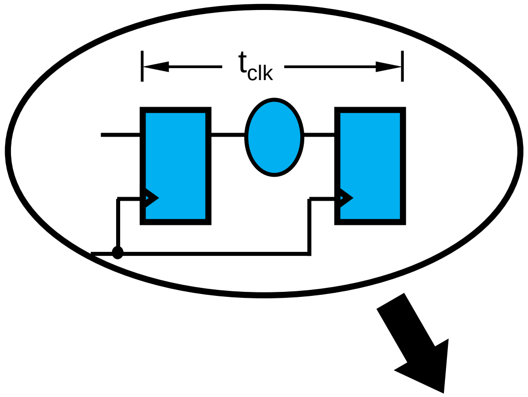
Figure 18 - *Xilinx XC4000 Configurable Logic Block (CLB).*

Place & Route

- Map primitives to specific locations inside target technology with reference to area & performance constraints
- Specify routing resources to be used

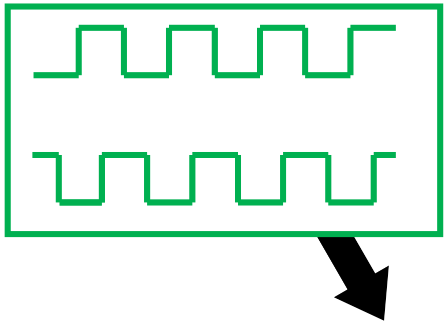
- modern, big FPGAs are complex: exploit design tools as much as possible

Classical FPGA Design Flow



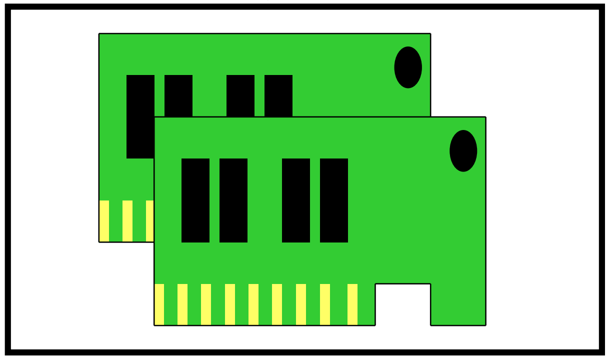
Timing Analysis

- Verify performance specifications were met
- Static timing analysis



Gate Level Simulation

- Timing simulation
- Verify design will work in target technology



Printed Circuit Board Simulation & Test

- Simulate board design
- Program & test device on board

VHDL example

```vhdl
entity counter is
    Port ( CLK : in   STD_LOGIC;
           CLR : in   STD_LOGIC;
           DOUT : out  STD_LOGIC_VECTOR (7 downto 0));
end counter;

architecture Behavioral of counter is

signal val: std_logic_vector (7 downto 0);

begin

process (CLK,CLR) is
begin
  if CLR='1' then
    val<="00000000";
  elsif rising_edge(CLK) then
    val<=val+1;
  end if;
end process;

DOUT<=val;

end Behavioral;
```

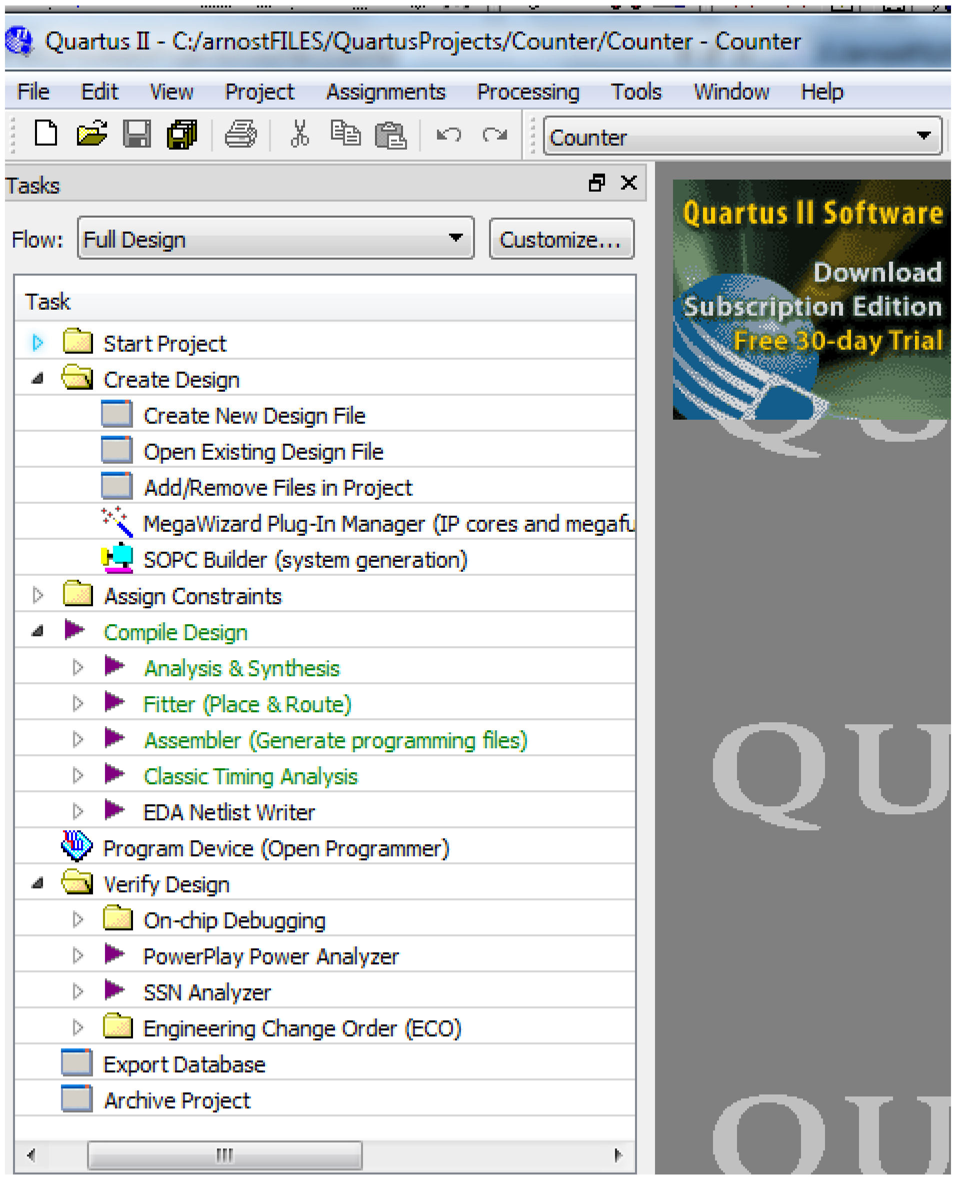
VHDL example

```vhdl
entity counter is
    Port ( CLK : in  STD_LOGIC;
           CLR : in  STD_LOGIC;
           DOUT : out  STD_LOGIC_VECTOR (7 downto 0));
end counter;

architecture Behavioral of counter is

signal val: std_logic_vector (7 downto 0);

begin

process (CLK,CLR) is
begin
  if CLR='1' then
    val<="00000000";
  elsif rising_edge(CLK) then
    val<=val+1;
  end if;
end process;

DOUT<=val;

end Behavioral;
```

- difference to "C"-Style programming:
  - these are electronic signals
  - the assignment is instantaneous
  - timing must be taken into account
  - clock management, signal synchronisation, registers, FIFO buffers, etc. are important tools

hardware description ≠ computer programming

Design software – Quartus II Web Edition – for INTEL FPGA



Quartus II - C:/arnostFILES/QuartusProjects/Counter/Counter - Counter

File Edit View Project Assignments Processing Tools Window Help

Counter

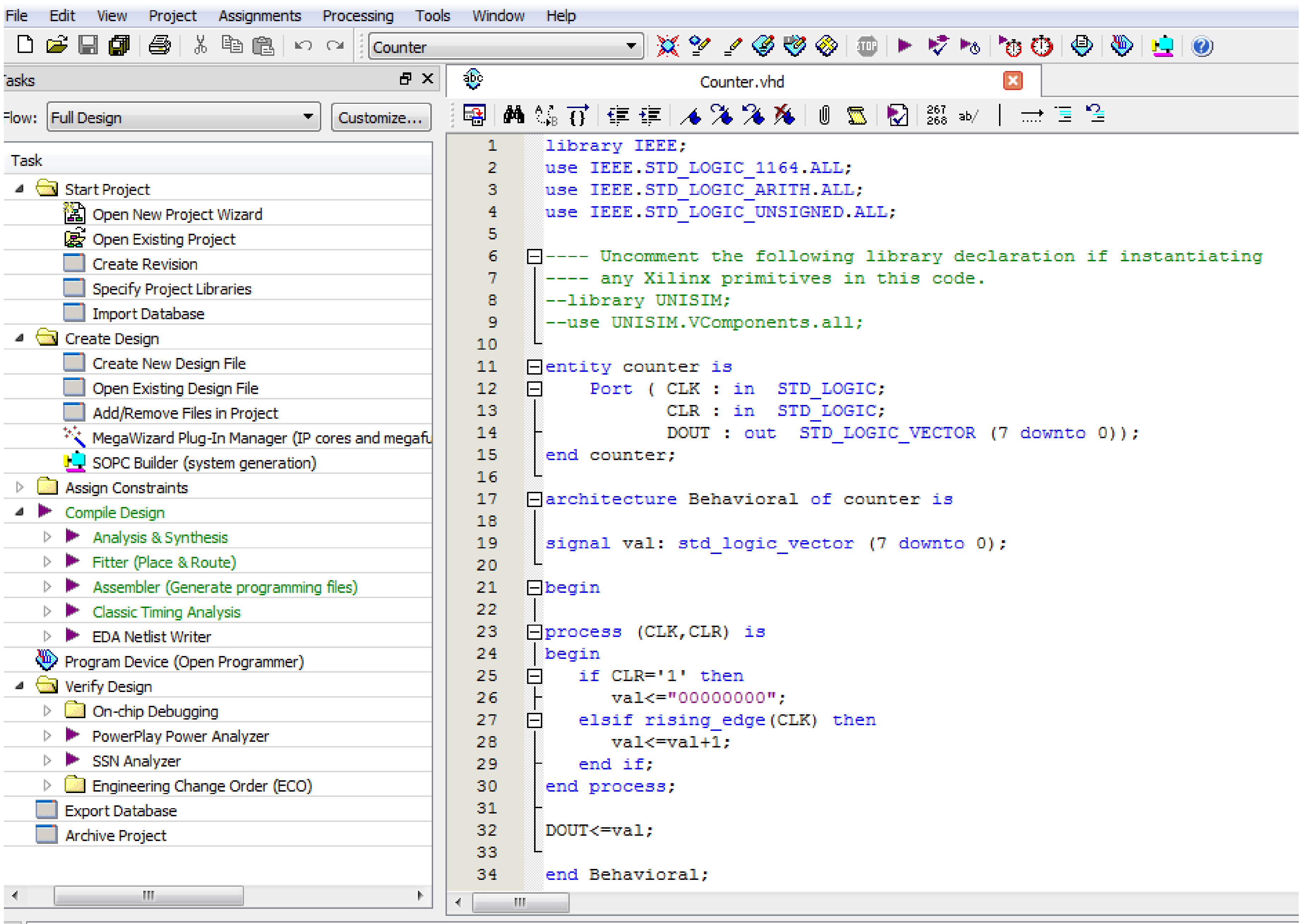
Tasks

Flow: Full Design Customize...

Task

- Start Project
- Create Design
  - Create New Design File
  - Open Existing Design File
  - Add/Remove Files in Project
  - MegaWizard Plug-In Manager (IP cores and megafu
  - SOPC Builder (system generation)
- Assign Constraints
- Compile Design
  - Analysis & Synthesis
  - Fitter (Place & Route)
  - Assembler (Generate programming files)
  - Classic Timing Analysis
  - EDA Netlist Writer
- Program Device (Open Programmer)
- Verify Design
  - On-chip Debugging
  - PowerPlay Power Analyzer
  - SSN Analyzer
  - Engineering Change Order (ECO)
- Export Database
- Archive Project

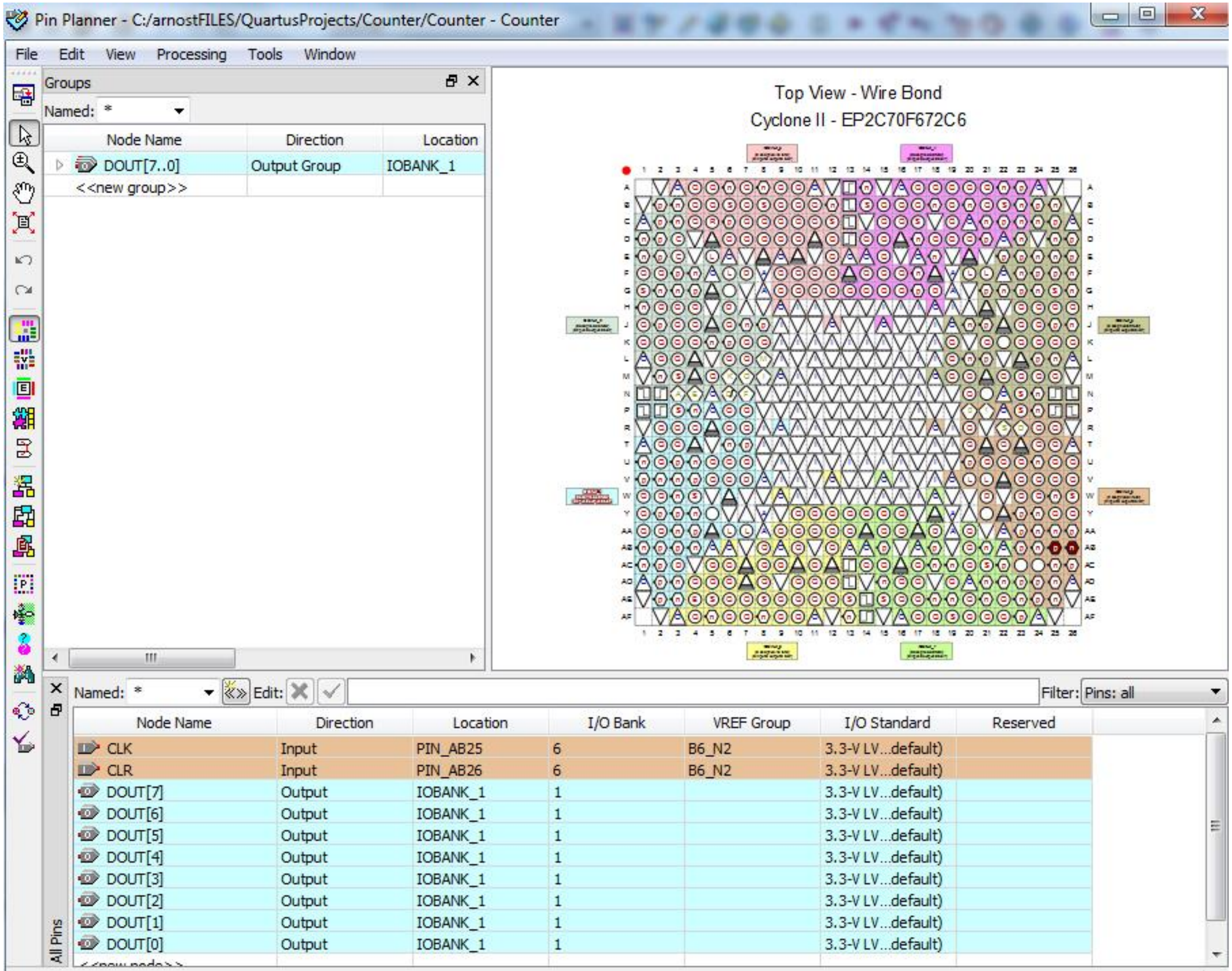
Design software – Quartus II Web Edition – for INTEL FPGA



Task
- Start Project
  - Open New Project Wizard
  - Open Existing Project
  - Create Revision
  - Specify Project Libraries
  - Import Database
- Create Design
  - Create New Design File
  - Open Existing Design File
  - Add/Remove Files in Project
  - MegaWizard Plug-In Manager (IP cores and megafu
  - SOPC Builder (system generation)
- Assign Constraints
- Compile Design
  - Analysis & Synthesis
  - Fitter (Place & Route)
  - Assembler (Generate programming files)
  - Classic Timing Analysis
  - EDA Netlist Writer
- Program Device (Open Programmer)
- Verify Design
  - On-chip Debugging
  - PowerPlay Power Analyzer
  - SSN Analyzer
  - Engineering Change Order (ECO)
- Export Database
- Archive Project

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

---- Uncomment the following library declaration if instantiating
---- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity counter is
    Port ( CLK : in  STD_LOGIC;
           CLR : in  STD_LOGIC;
           DOUT : out  STD_LOGIC_VECTOR (7 downto 0));
end counter;

architecture Behavioral of counter is

signal val: std_logic_vector (7 downto 0);

begin

process (CLK,CLR) is
begin
   if CLR='1' then
      val<="00000000";
   elsif rising_edge(CLK) then
      val<=val+1;
   end if;
end process;

DOUT<=val;

end Behavioral;
```

Input and Output Assignment – Pin Planner



Pin Planner - C:/arnostFILES/QuartusProjects/Counter/Counter - Counter

File Edit View Processing Tools Window

Groups

Named: *

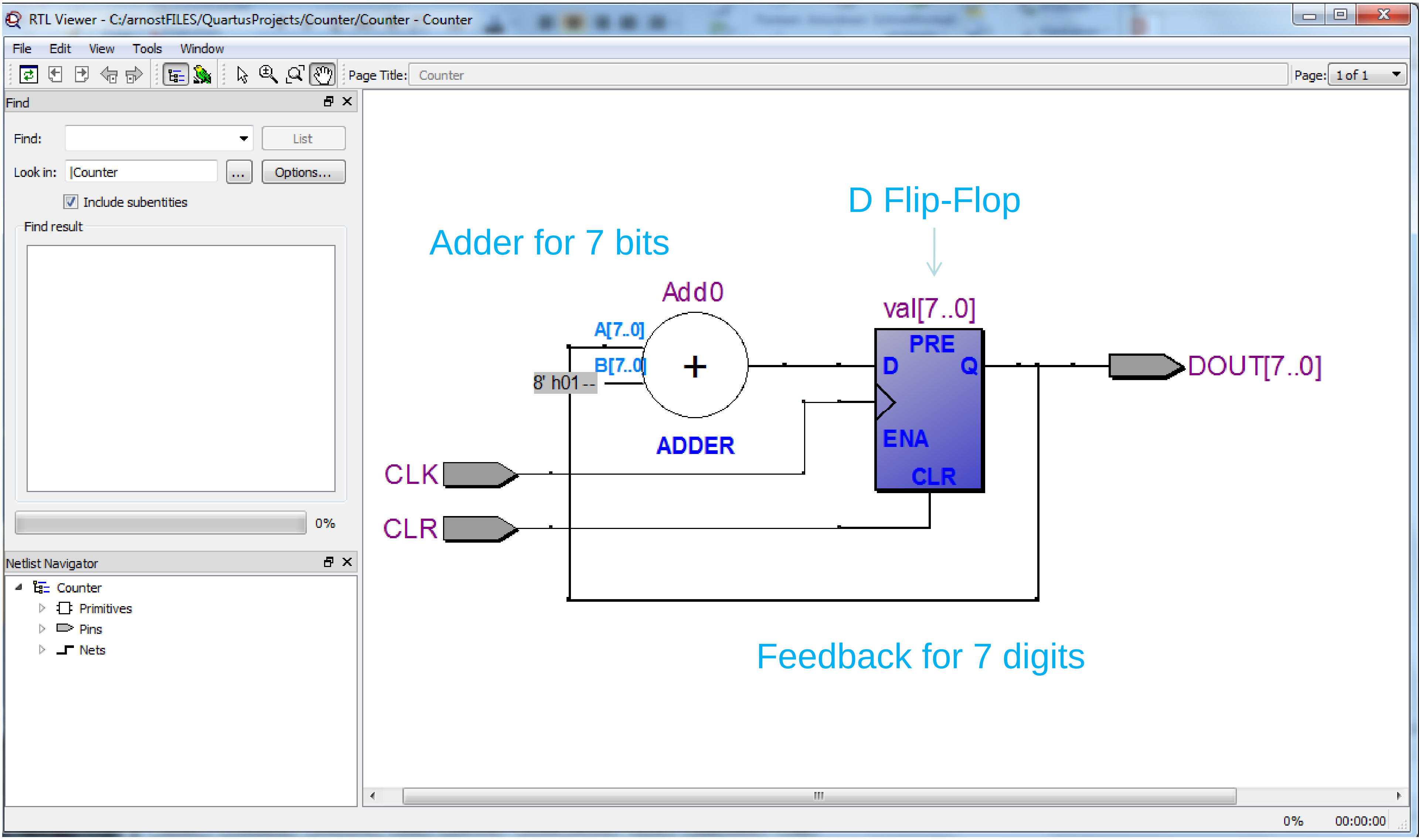
Node Name	Direction	Location
DOUT[7..0]	Output Group	IOBANK_1
<<new group>>		

Top View - Wire Bond
Cyclone II - EP2C70F672C6

Named: * Edit: Filter: Pins: all

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
CLK	Input	PIN_AB25	6	B6_N2	3.3-V LV...default)	
CLR	Input	PIN_AB26	6	B6_N2	3.3-V LV...default)	
DOUT[7]	Output	IOBANK_1	1		3.3-V LV...default)	
DOUT[6]	Output	IOBANK_1	1		3.3-V LV...default)	
DOUT[5]	Output	IOBANK_1	1		3.3-V LV...default)	
DOUT[4]	Output	IOBANK_1	1		3.3-V LV...default)	
DOUT[3]	Output	IOBANK_1	1		3.3-V LV...default)	
DOUT[2]	Output	IOBANK_1	1		3.3-V LV...default)	
DOUT[1]	Output	IOBANK_1	1		3.3-V LV...default)	
DOUT[0]	Output	IOBANK_1	1		3.3-V LV...default)	

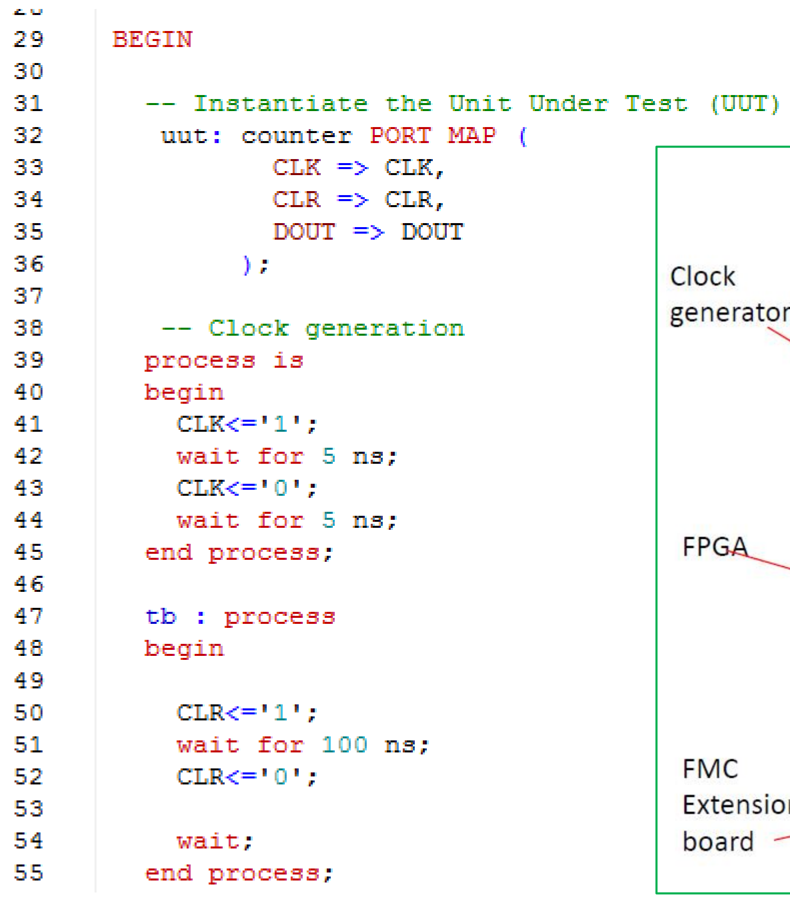
After Analysis & Synthesis: Netlist viewer



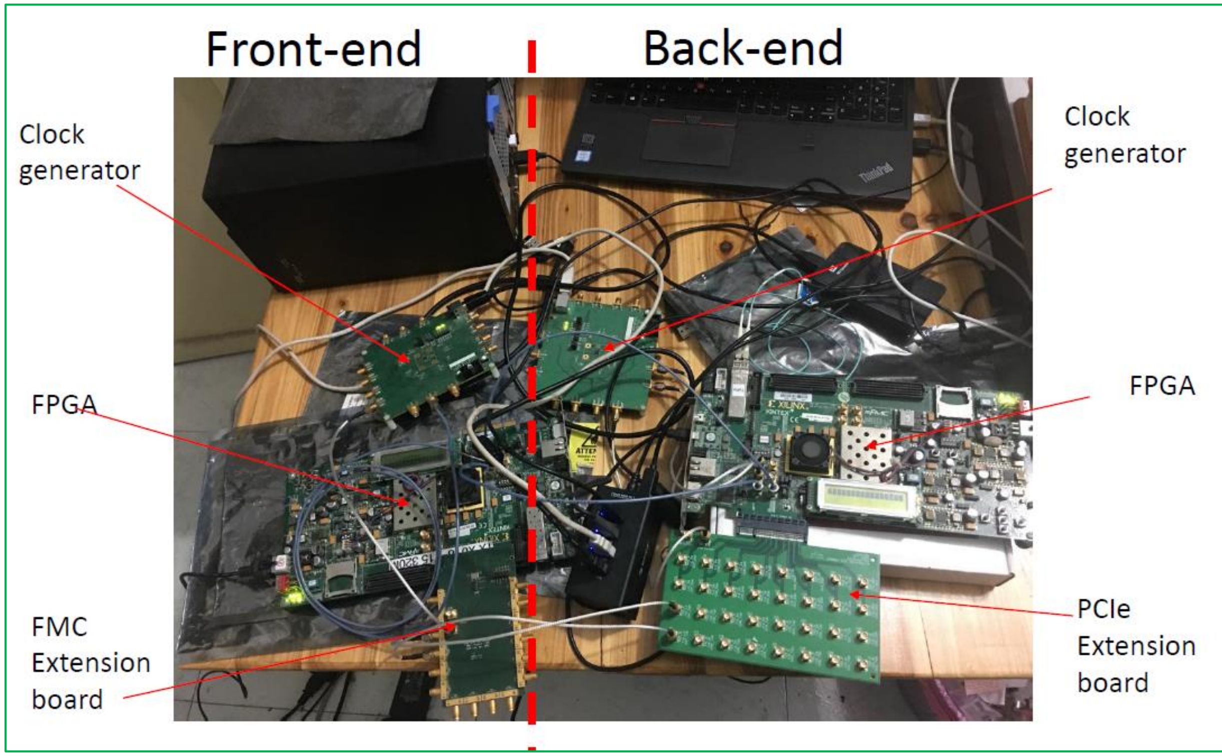
- RTL = register transfer level

Simulation of the Design: Testbench

- The logic circuit can only be tested if external signals are simulated
- The stimulus for the simulation is written in a separate VHDL code: the testbench



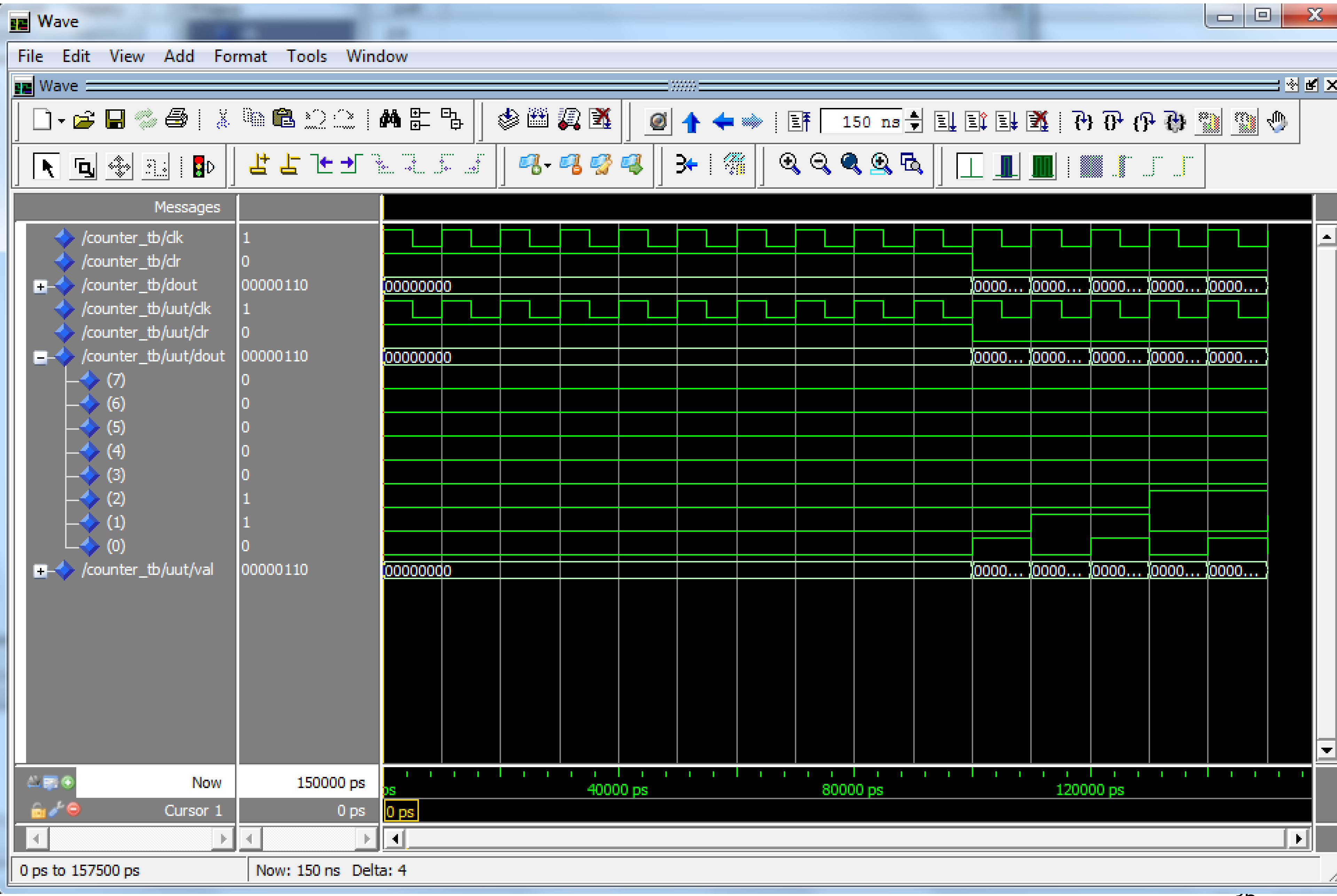
```
BEGIN

  -- Instantiate the Unit Under Test (UUT)
   uut: counter PORT MAP (
          CLK => CLK,
          CLR => CLR,
          DOUT => DOUT
        );

   -- Clock generation
  process is
  begin
    CLK<='1';
    wait for 5 ns;
    CLK<='0';
    wait for 5 ns;
  end process;

  tb : process
  begin

    CLR<='1';
    wait for 100 ns;
    CLR<='0';

    wait;
  end process;
```



simulated testbench

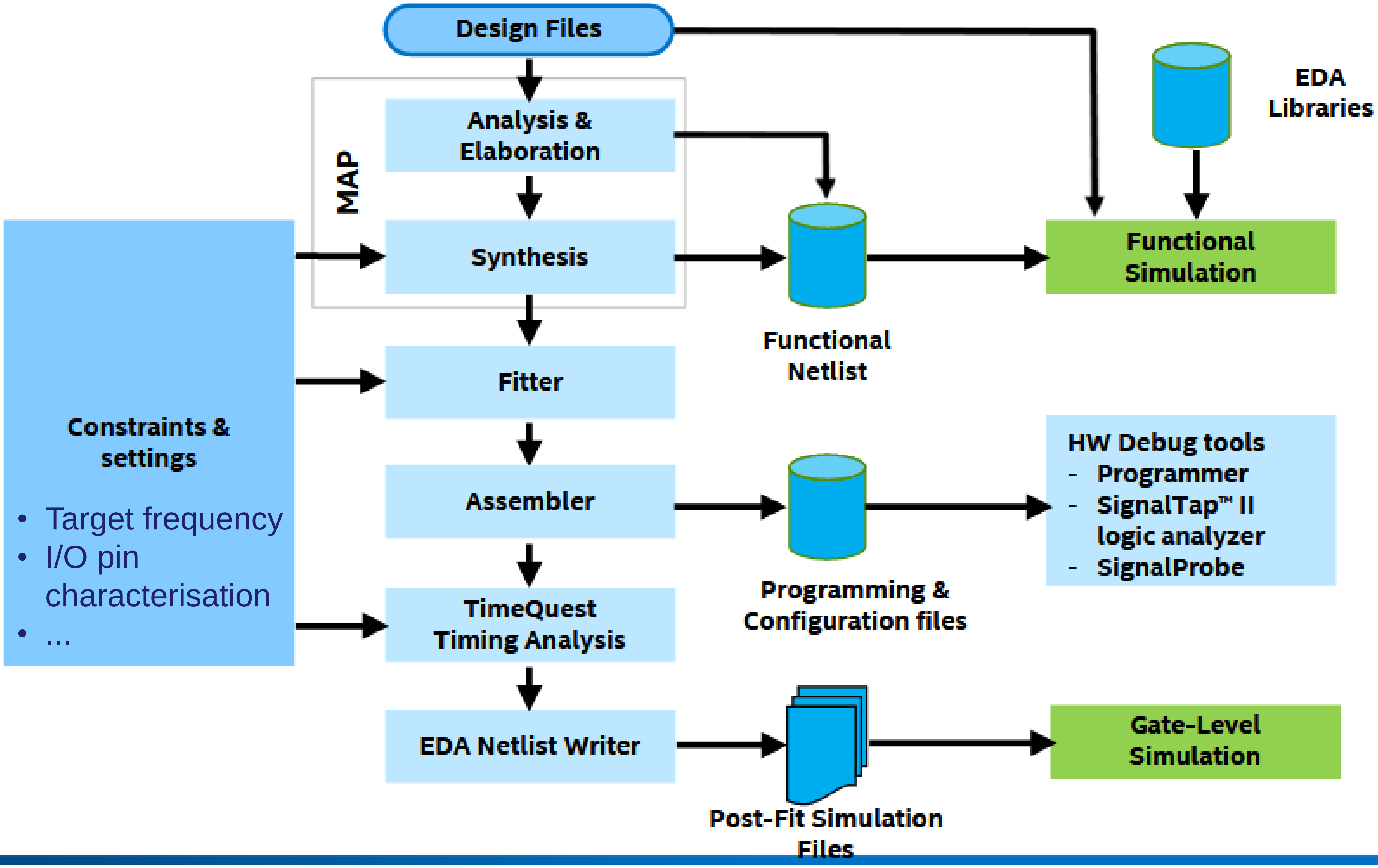
real testbench

Functional Simulation - Timing Diagram



```
/counter_tb/clk         1
/counter_tb/clr         0
/counter_tb/dout        00000110
/counter_tb/uut/clk     1
/counter_tb/uut/clr     0
/counter_tb/uut/dout    00000110
  (7)                   0
  (6)                   0
  (5)                   0
  (4)                   0
  (3)                   0
  (2)                   1
  (1)                   1
  (0)                   0
/counter_tb/uut/val     00000110

Now       150000 ps
Cursor 1  0 ps

0 ps to 157500 ps     Now: 150 ns  Delta: 4
```

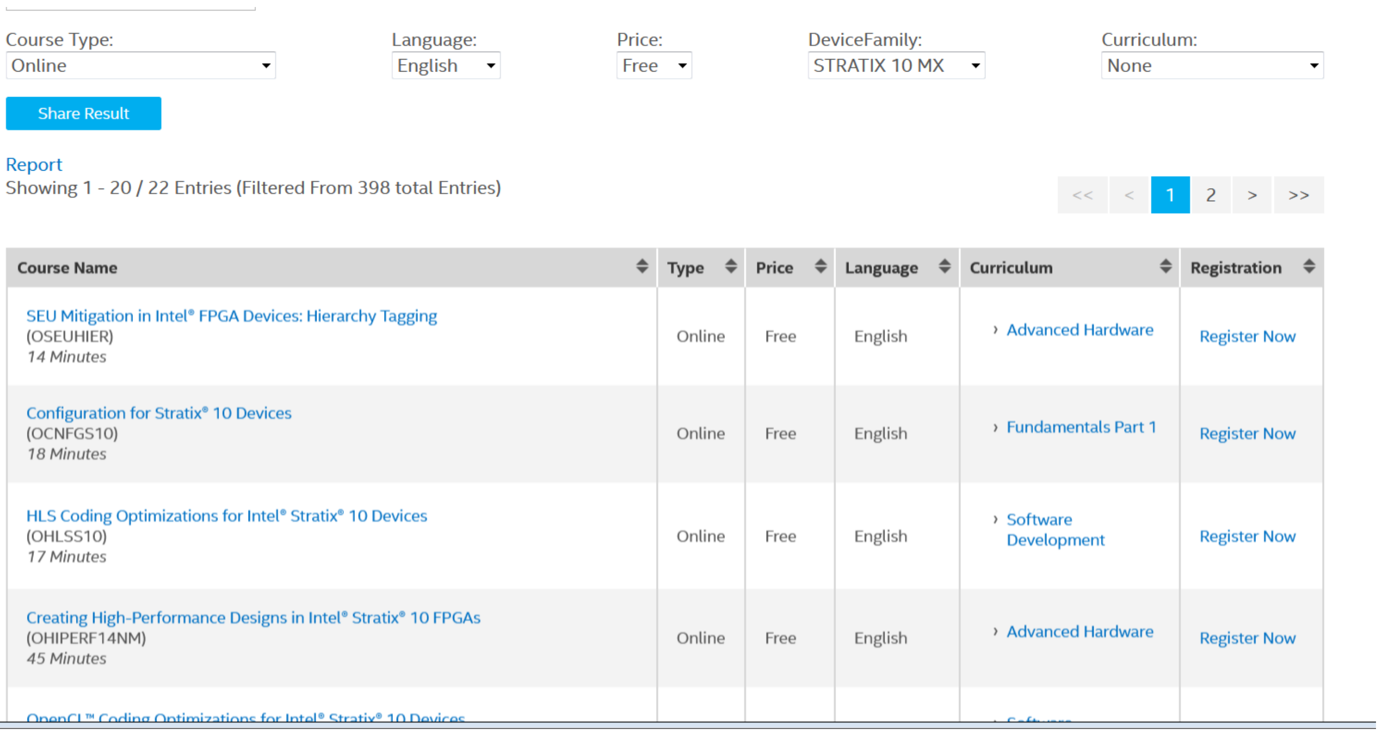
Tools Overview



- Target frequency
- I/O pin characterisation
- ...

More Information

- There are many interesting books on FPGA and firmware design
- There is a lot of material provided by FPGA producers:
  - Tutorials and documentation for all knowledge levels
  - Tutorials and documentation of device-specific features
  - Very useful source of information! Most of it is for free!



Course Type:	Language:	Price:	DeviceFamily:	Curriculum:
Online	English	Free	STRATIX 10 MX	None

Share Result

Report
Showing 1 - 20 / 22 Entries (Filtered From 398 total Entries)

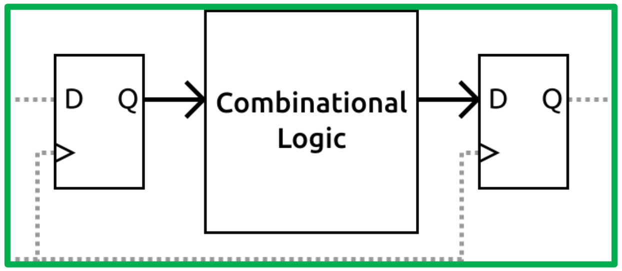
<< < 1 2 > >>

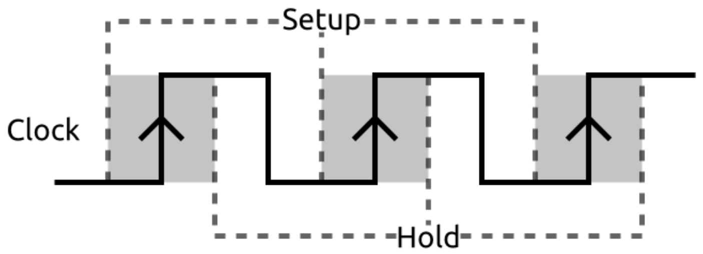
Course Name	Type	Price	Language	Curriculum	Registration
SEU Mitigation in Intel® FPGA Devices: Hierarchy Tagging (OSEUHIER) *14 Minutes*	Online	Free	English	› Advanced Hardware	Register Now
Configuration for Stratix® 10 Devices (OCNFGS10) *18 Minutes*	Online	Free	English	› Fundamentals Part 1	Register Now
HLS Coding Optimizations for Intel® Stratix® 10 Devices (OHLSS10) *17 Minutes*	Online	Free	English	› Software Development	Register Now
Creating High-Performance Designs in Intel® Stratix® 10 FPGAs (OHIPERF14NM) *45 Minutes*	Online	Free	English	› Advanced Hardware	Register Now
OpenCL™ Coding Optimizations for Intel® Stratix® 10 Devices				› Software	

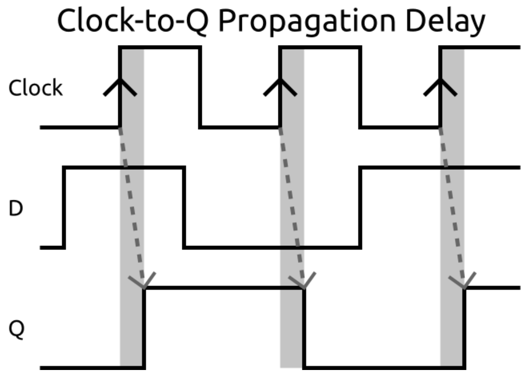
More Information

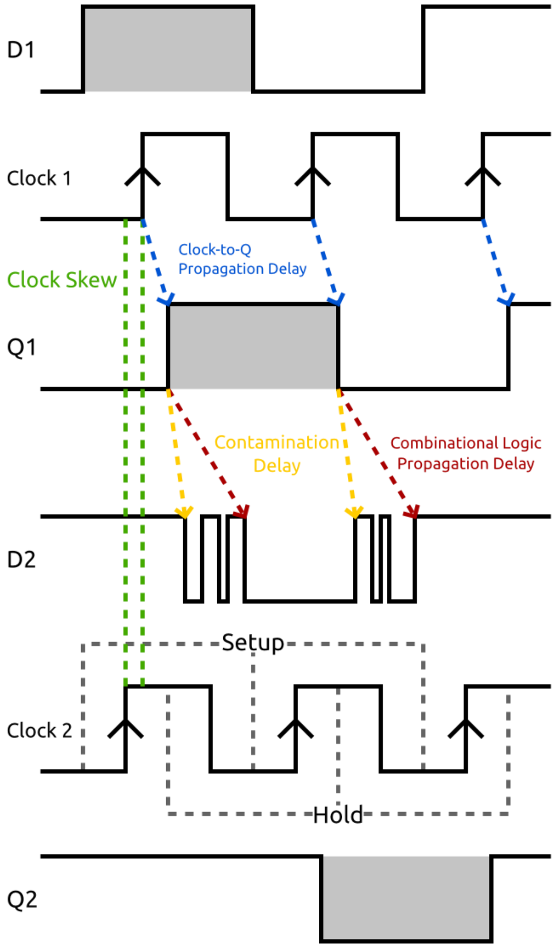
Timing

- The FPGA is an electronic circuit: signal timing can be an isssue
  - signals travel across the FPGA and need time to do so
  - clock skew = different arrival time of clock signal at different places in the FPGA
- Example:





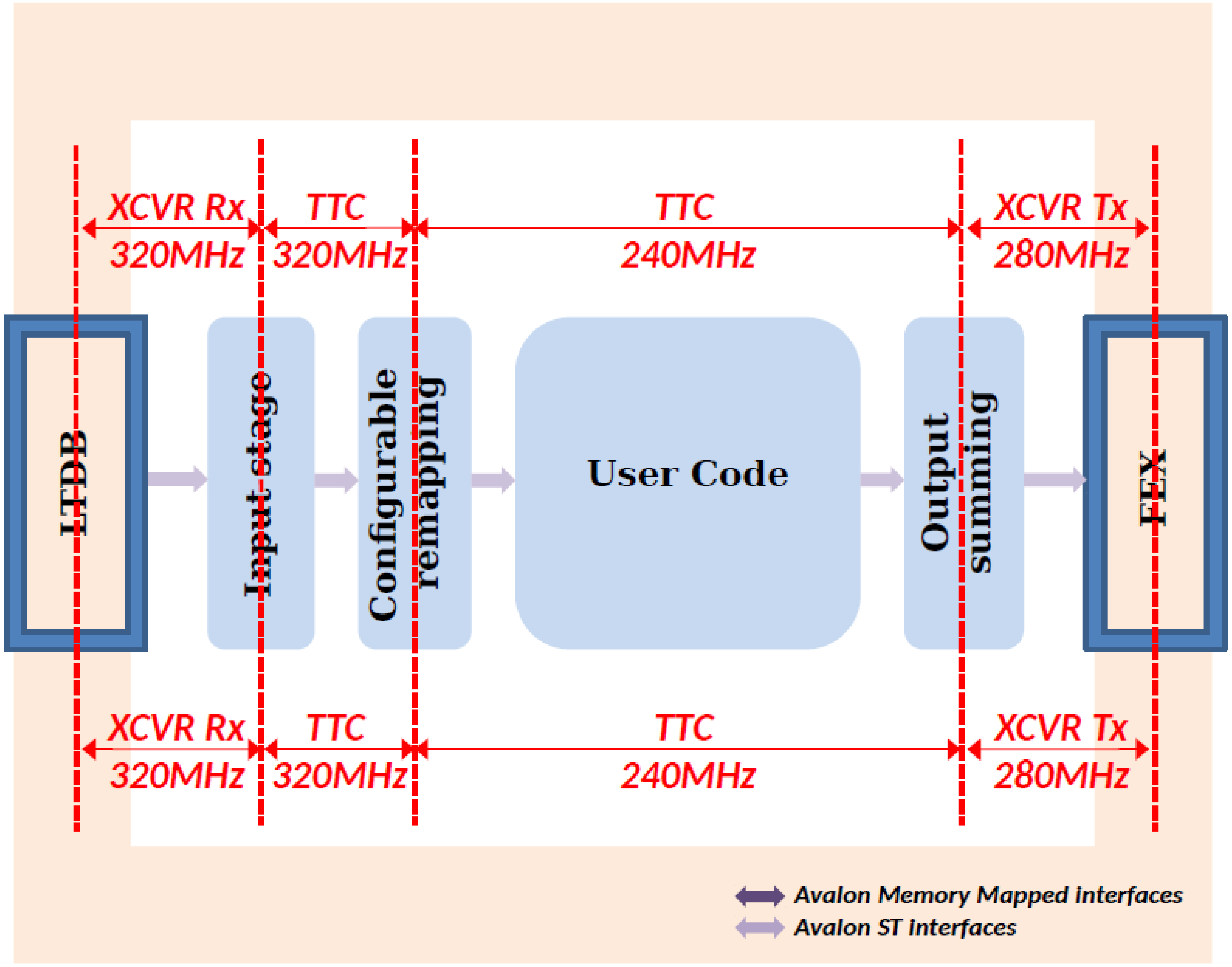



D1
Clock 1
Clock-to-Q Propagation Delay
Clock Skew
Q1
Contamination Delay
Combinational Logic Propagation Delay
D2
Setup
Clock 2
Hold
Q2

- Missed timing constraints lead to logic errors, which may look random
- A lower execution frequency and introduction of registers help to fulfill constraints

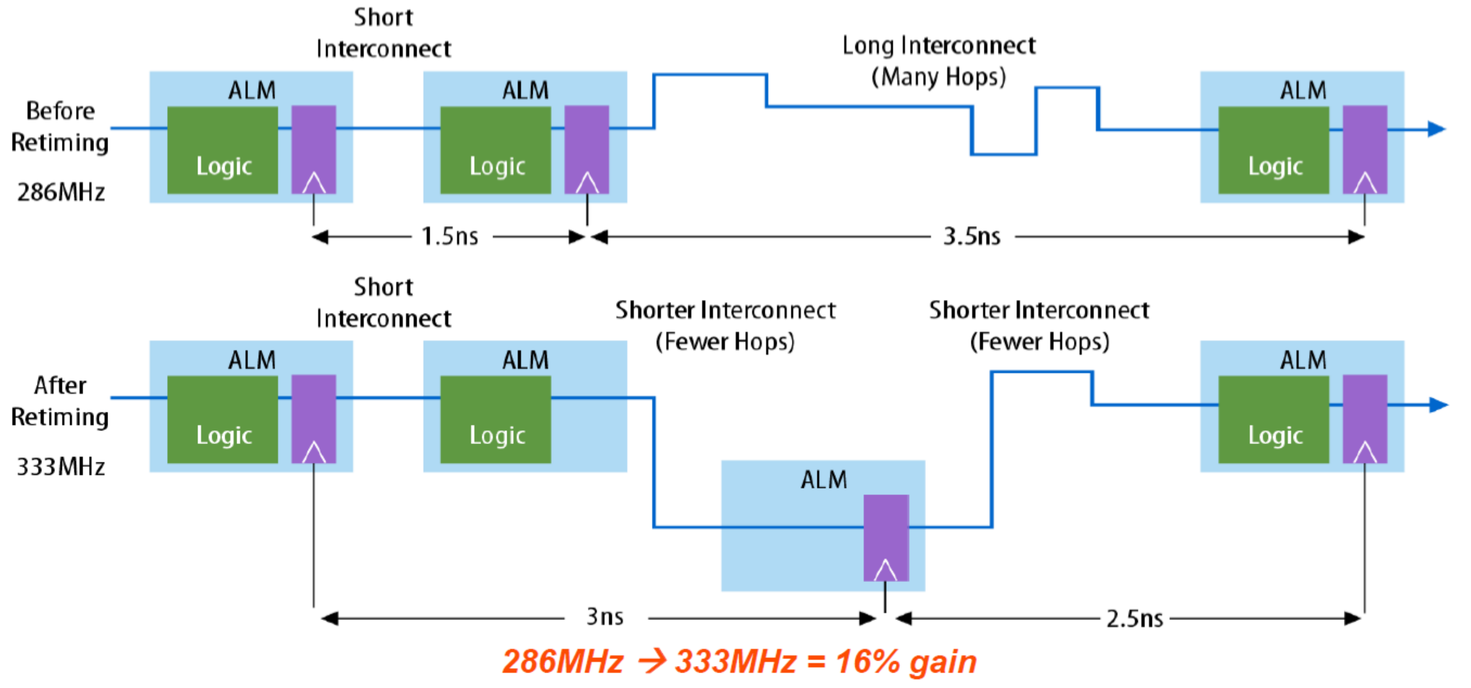
Example

- ATLAS LAr signal processing FPGA:



Improve Maximum Frequency

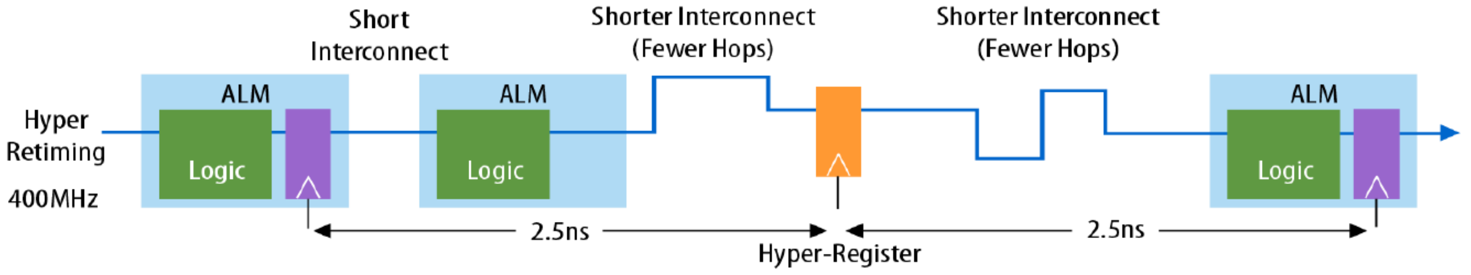
- in order to have well-defined signals at rising/falling edge of the clock, registers are placed in the signal path: pipeline stages



$$\frac{1}{3.5\ ns} = 286\ MHz$$

$$\frac{1}{3\ ns} = 333\ MHz$$

286MHz → 333MHz = 16% gain

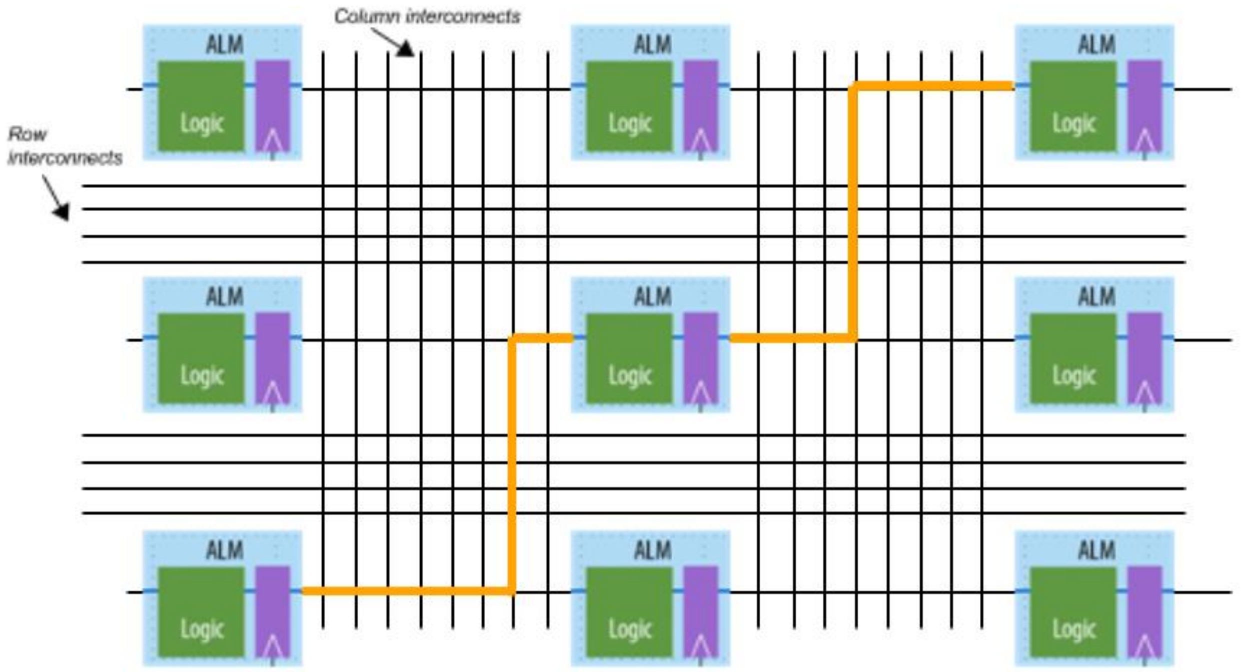


$$\frac{1}{2.5\ ns} = 400\ MHz$$

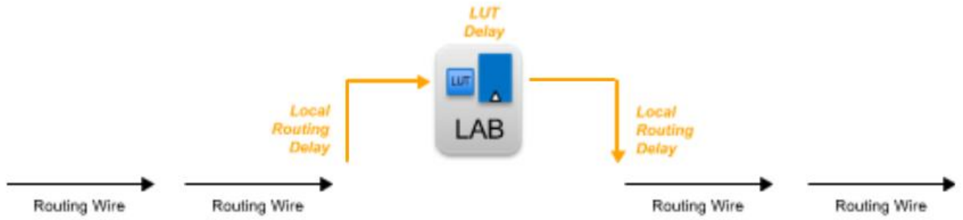
286MHz → 400MHz = 40% gain

Signal Routing and Register Distribution

Conventional FPGA Architecture - Interconnect



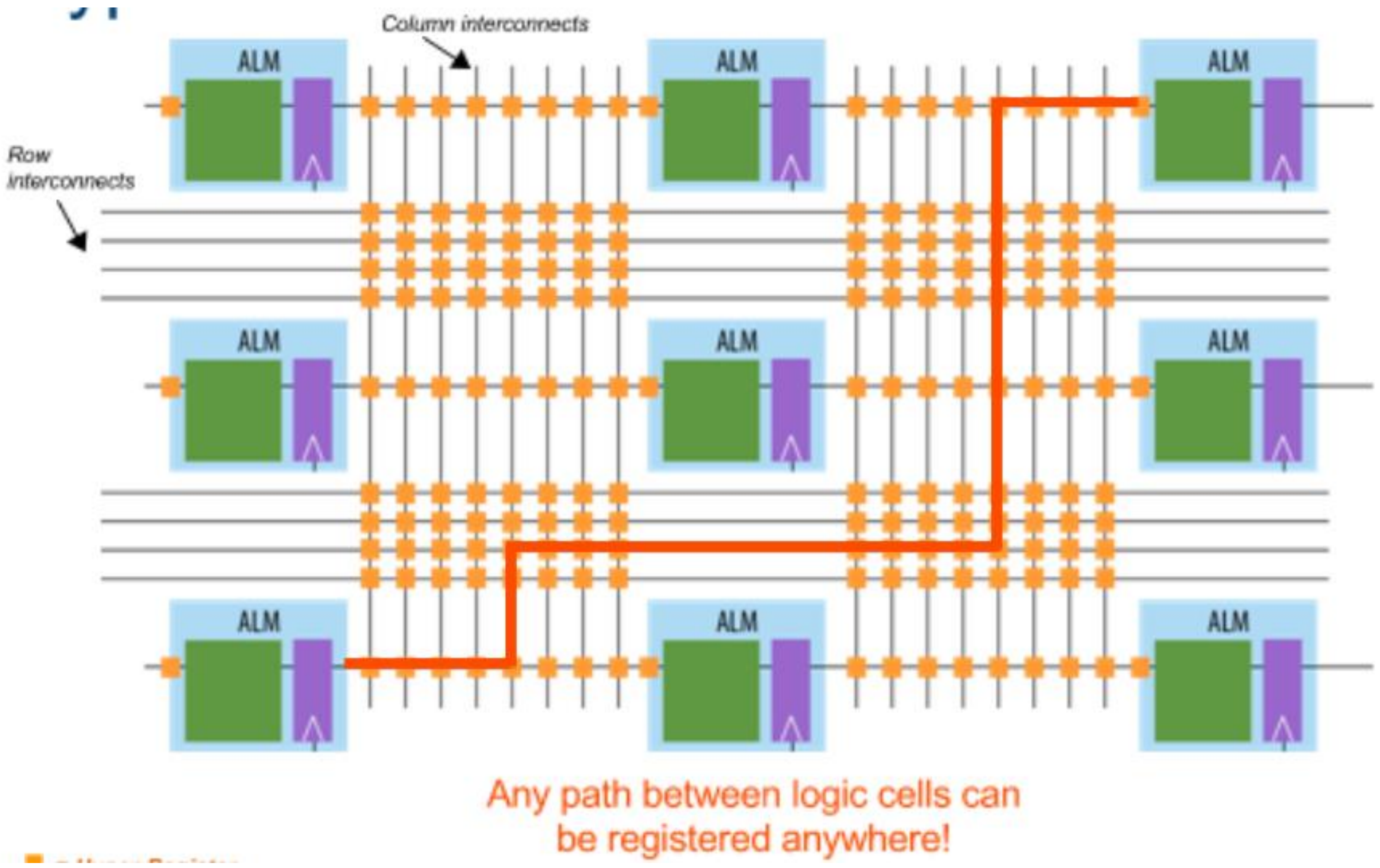
Conventional pipelined path between logic cells



Local Routing and LUT delay significantly increase delay

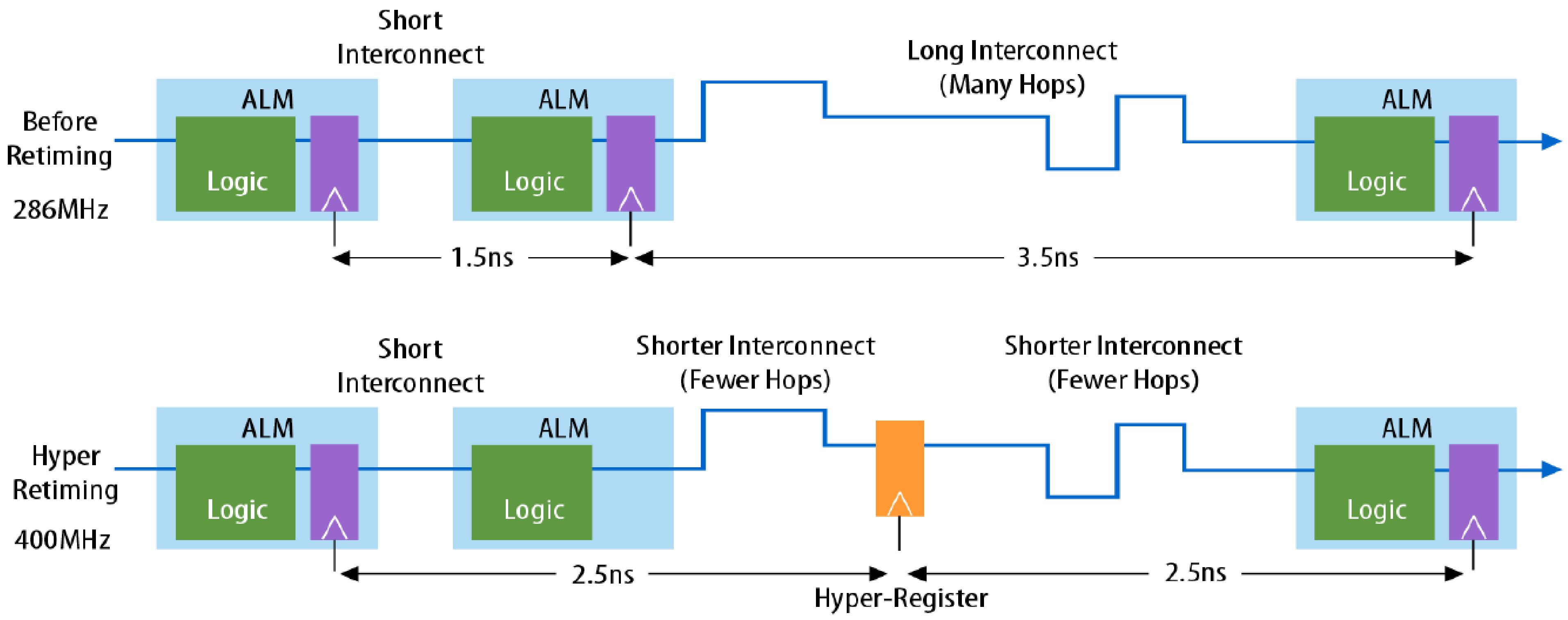
Signal Routing and Register Distribution

- Registers independent of logic blocks:



Signal Routing and Register Distribution

- If intermediate register can be placed right in the middle, delays are equalized



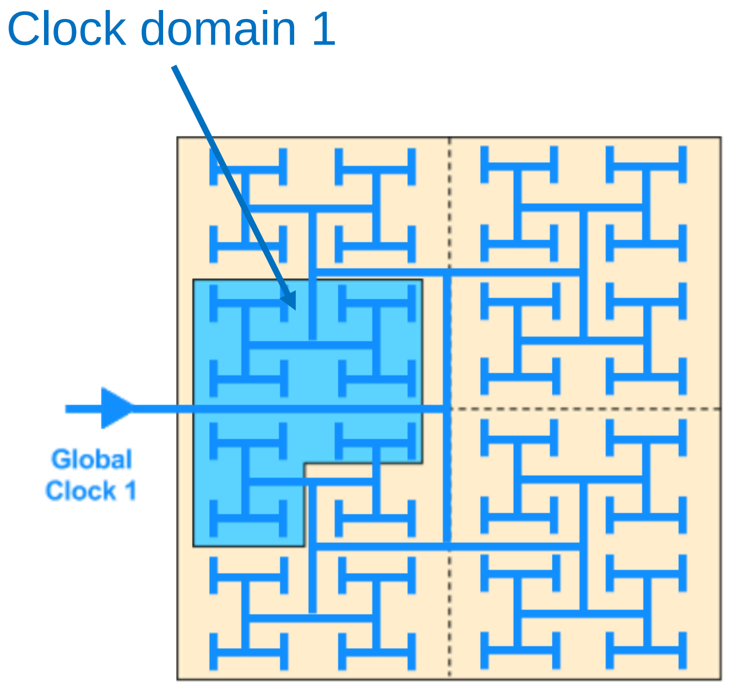
286MHz → 400MHz = 40% gain

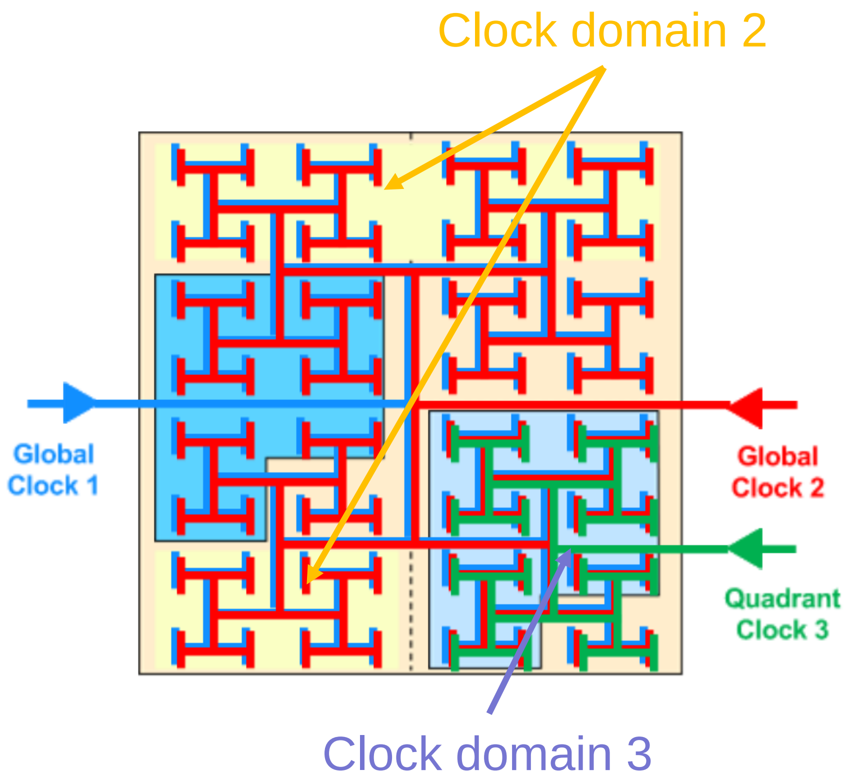
Conventional Clock Network

- Conventional example: FPGA divided into quadrants
- Global clock or quadrant clock possible

- If clock domain crosses more than one quadrant, only a global clock is possible

- Clock network consumes a lot of power: 10-20% of total FPGA power (charge/discharge of many capacitors)

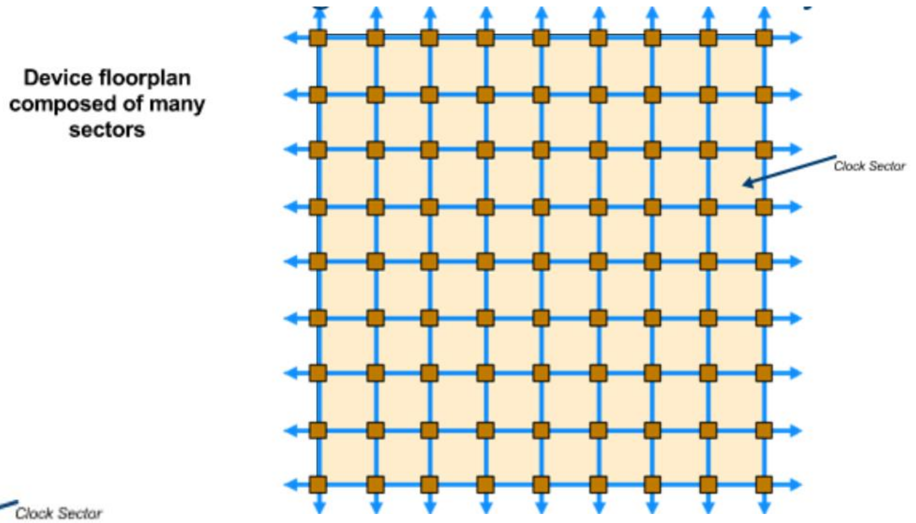


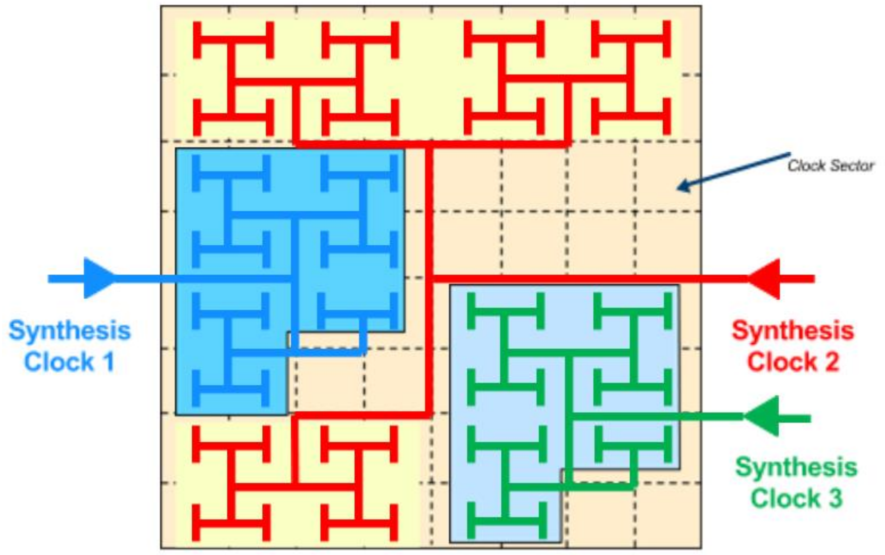


Flexible Clock Distribution

- Smaller clock sectors

- More detailed clock domain segmentation possible
- Reduced power consumption



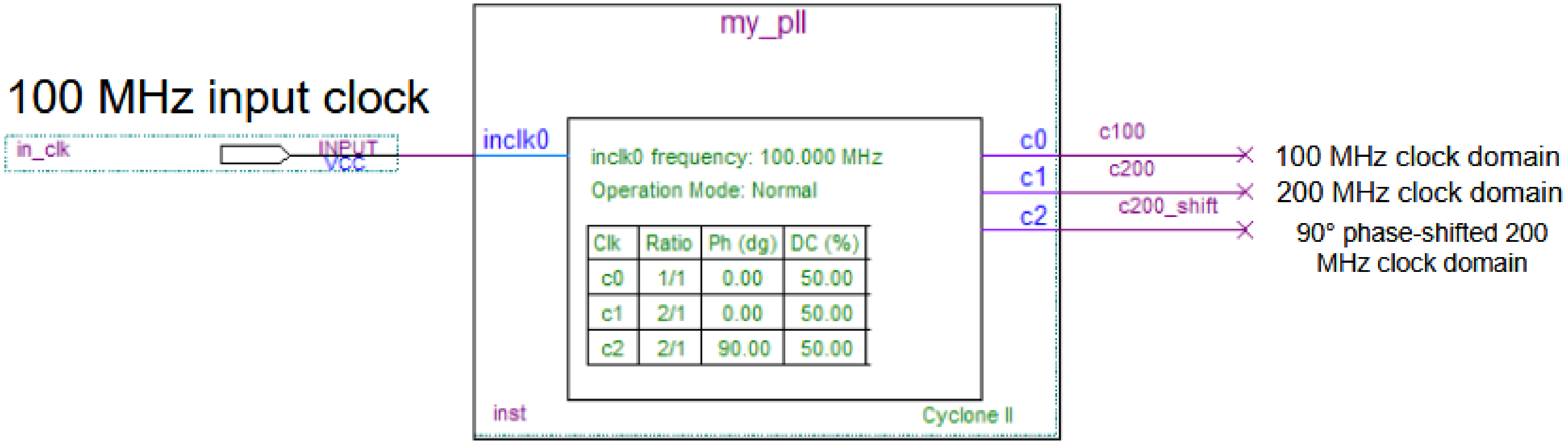


- If clock domains needs to be crossed a signal buffer needs to be implemented to pass over the data and signals

Phase-locked loop (PLL)

- Programmable block that taks input clock and can convert into more and different clock signals

- Example:



inclk0 frequency: 100.000 MHz
Operation Mode: Normal

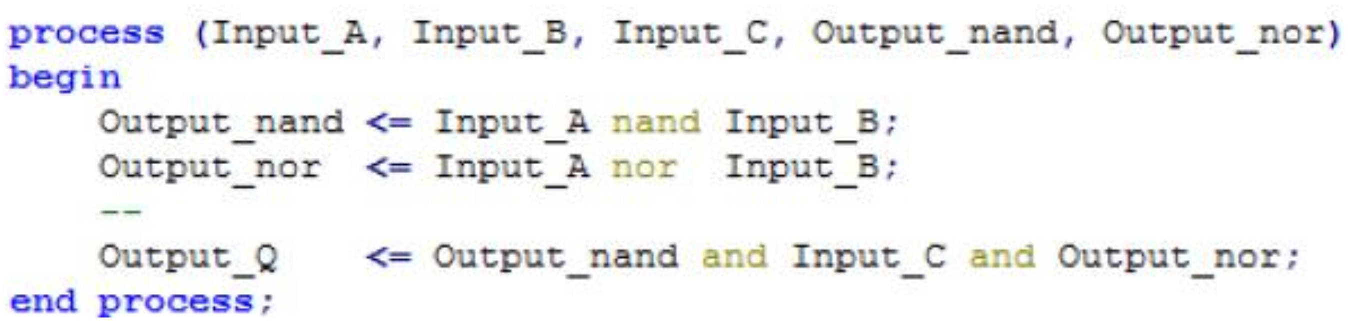
Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	0.00	50.00
c1	2/1	0.00	50.00
c2	2/1	90.00	50.00

- Typical FPGA clock frequencies: 100-400 MHz

- Modern FPGAs or high-frequency models reach 0.9 - 1.5 GHz

More design concepts

- Sensitivity list must include all input signals, otherwise outputs can be non-responsive under changes of inputs



```
process (Input_A, Input_B, Input_C, Output_nand, Output_nor)
begin
    Output_nand <= Input_A nand Input_B;
    Output_nor  <= Input_A nor  Input_B;
    --
    Output_Q    <= Output_nand and Input_C and Output_nor;
end process;
```

- All output signals must be assigned under all possible input conditions

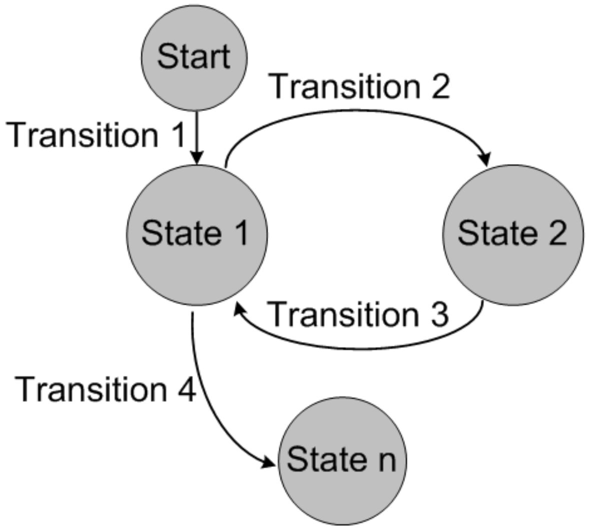
- No feedback from output to input signals

More design concepts

- Synchronous design is preferred, i.e. using signals synchronized to clock

- Reset: initialize register outputs to a know state
    - implement synchronous reset, i.e. synchronous to free-running clock

- State machines:
    - outputs may be assigned during states or state transitions
    - be careful with states that cannot be reached or are illegal



- Clock domains: avoid unnecessary clock domains
    - clock domain crossing require FIFOs/registers and proper treatment of signals (handshake, avoid FIFO overflow)

- Timing constraints are important: time-constrain all I/O signals, properly implement reset scheme, properly handle clock domain crossings