Beginner Workshop for Intel FPGAs

Course Topics

- What is an FPGA?
- Introduction to Intel FPGAs
- Applications of FPGAs
- Quartus Prime design software tool flow
- Creating Projects
- Design entry
- Pin Planner
- Synthesis and Fitter
- Analyzing reports
- Implementation on FPGA Board
- Remote Hands free Labsland Setup

Course Outline

- FPGA Overview
  - FPGAS, What, Why, Where?
  - FPGA architecture
  - How an FPGA becomes what you want it to be
  - FPGA design flow

- Intel® Quartus® Software
  - Basics
  - Create a new project
  - Design Entry
  - Pin Planner & Assignment Editor
  - Compilation: Synthesis & Fitter
  - Programming the Device

Intel® FPGA Products

intel. ARRiA®

FPGA
Mid-range FPGAs
SoC & Transceivers

intel. STRATiX®

FPGA
Optimized for
High Bandwidth

intel. AGILEX™

FPGA
Optimized for
High Bandwidth

intel. CYCLONE®

FPGA
Cost/Power Balance
SoC & Transceivers

intel. MAX®

FPGA/CPLD
Lowest Cost,
Lowest Power

Resources

Embedded Soft and Hard Processors

Nios® II

Arm*

Design Software

Intel® Quartus® Prime
Design Software

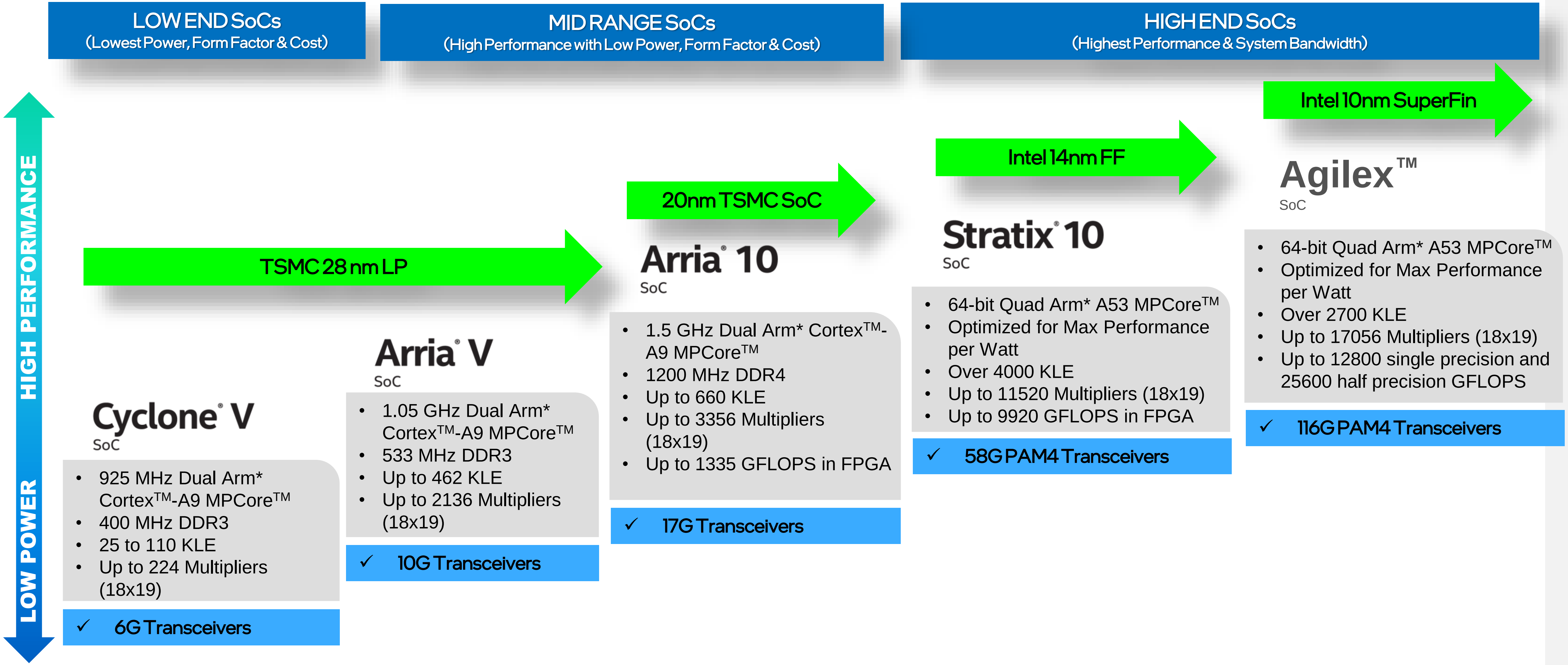
Intel® FPGA SDK for OpenCL™

Acceleration Boards & Development Kits

Intellectual Property (IP)

- Industrial
- Computing
- Enterprise

Intel SoC FPGA Portfolio



LOW END SoCs
(Lowest Power, Form Factor & Cost)

MID RANGE SoCs
(High Performance with Low Power, Form Factor & Cost)

HIGH END SoCs
(Highest Performance & System Bandwidth)

Cyclone® V
SoC

- 925 MHz Dual Arm* Cortex™-A9 MPCore™
- 400 MHz DDR3
- 25 to 110 KLE
- Up to 224 Multipliers (18x19)

✓ 6G Transceivers

Arria® V
SoC

- 1.05 GHz Dual Arm* Cortex™-A9 MPCore™
- 533 MHz DDR3
- Up to 462 KLE
- Up to 2136 Multipliers (18x19)

✓ 10G Transceivers

Arria® 10
SoC

- 1.5 GHz Dual Arm* Cortex™-A9 MPCore™
- 1200 MHz DDR4
- Up to 660 KLE
- Up to 3356 Multipliers (18x19)
- Up to 1335 GFLOPS in FPGA

✓ 17G Transceivers

Stratix® 10
SoC

- 64-bit Quad Arm* A53 MPCore™
- Optimized for Max Performance per Watt
- Over 4000 KLE
- Up to 11520 Multipliers (18x19)
- Up to 9920 GFLOPS in FPGA

✓ 58G PAM4 Transceivers

Agilex™
SoC

- 64-bit Quad Arm* A53 MPCore™
- Optimized for Max Performance per Watt
- Over 2700 KLE
- Up to 17056 Multipliers (18x19)
- Up to 12800 single precision and 25600 half precision GFLOPS

✓ 116G PAM4 Transceivers

Section One

What is an FPGA?

What is an FPGA?

- First, let’s define the acronym
- It’s a Field Programmable Gate Array.

What is an FPGA?

- Field Programmable Gate Array
- Lego® bricks made to any design of choice
- A programmable chip

“Field Programmable Gate Array” (FPGA)

- “Gates” refers to transistors
    - These are the tiny pieces of hardware on a chip that make up the design
- “Array” means there are many of them manufactured on the chip
    - Many = Billions
    - They are arranged into larger structures as we will see
- “Field Programmable” means the connections between the internal components are programmable after deployment

- **FPGA = Programmable Hardware**

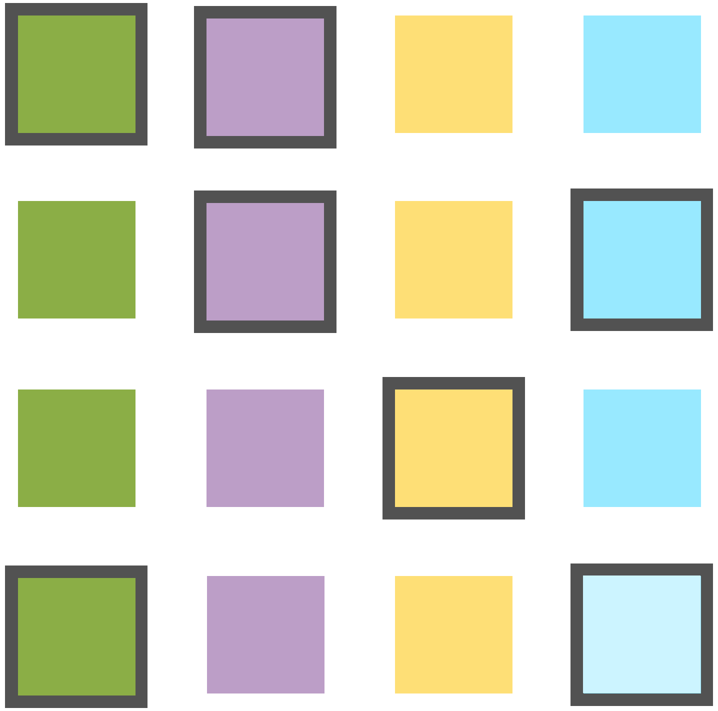
How an FPGA Becomes What You Want It To Be

- The FPGA is made up of small building blocks of logic and other functions

How an FPGA Becomes What You Want It To Be

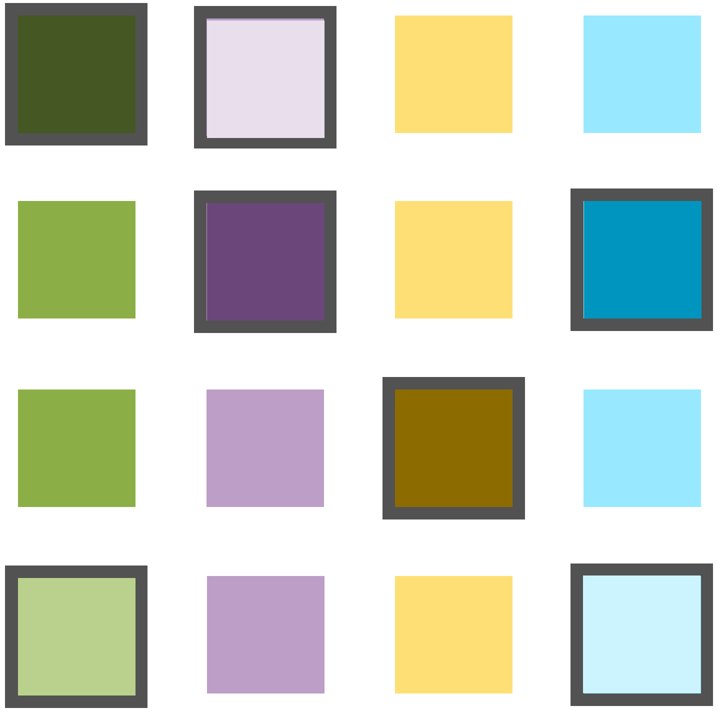
- The FPGA is made up of small building blocks of logic and other functions

- The building blocks you **Choose**



How an FPGA Becomes What You Want It To Be

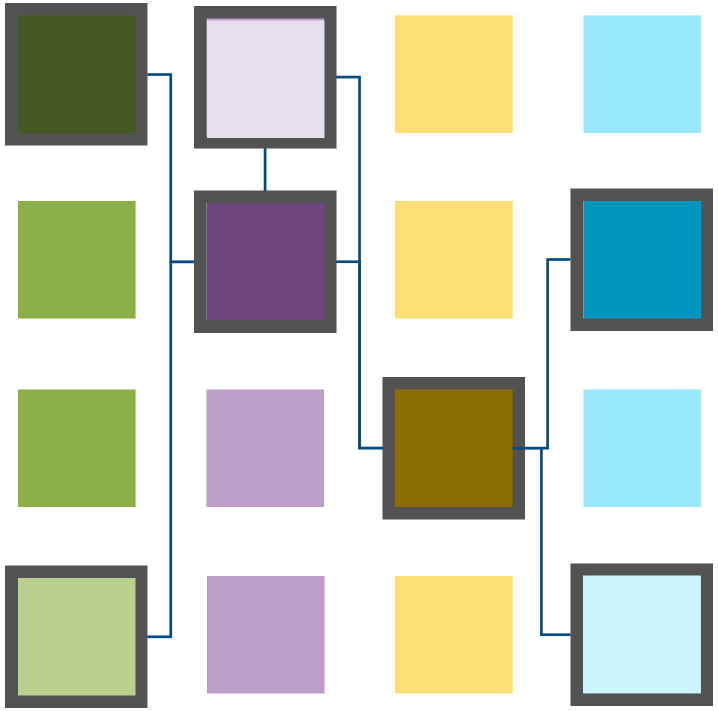
- The FPGA is made up of small building blocks of logic and other functions



- The building blocks you **Choose**
- How you **Configure** them

How an FPGA Becomes What You Want It To Be

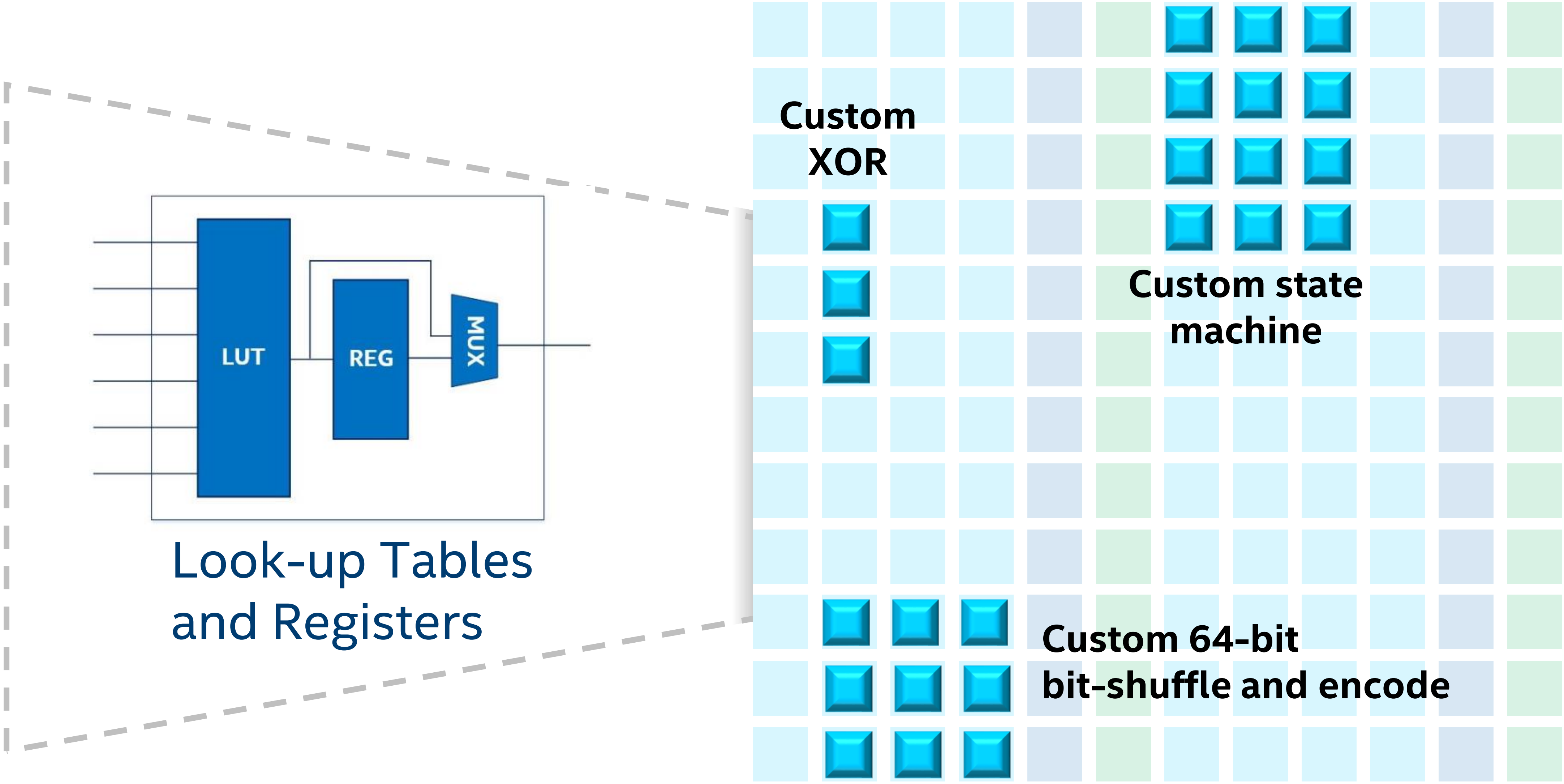
- The FPGA is made up of small building blocks of logic and other functions



- The building blocks you **Choose**
- How you **Configure** them
- And how you **Connect** them

Determine what function the FPGA performs

Blocks Used to Build What You've Coded



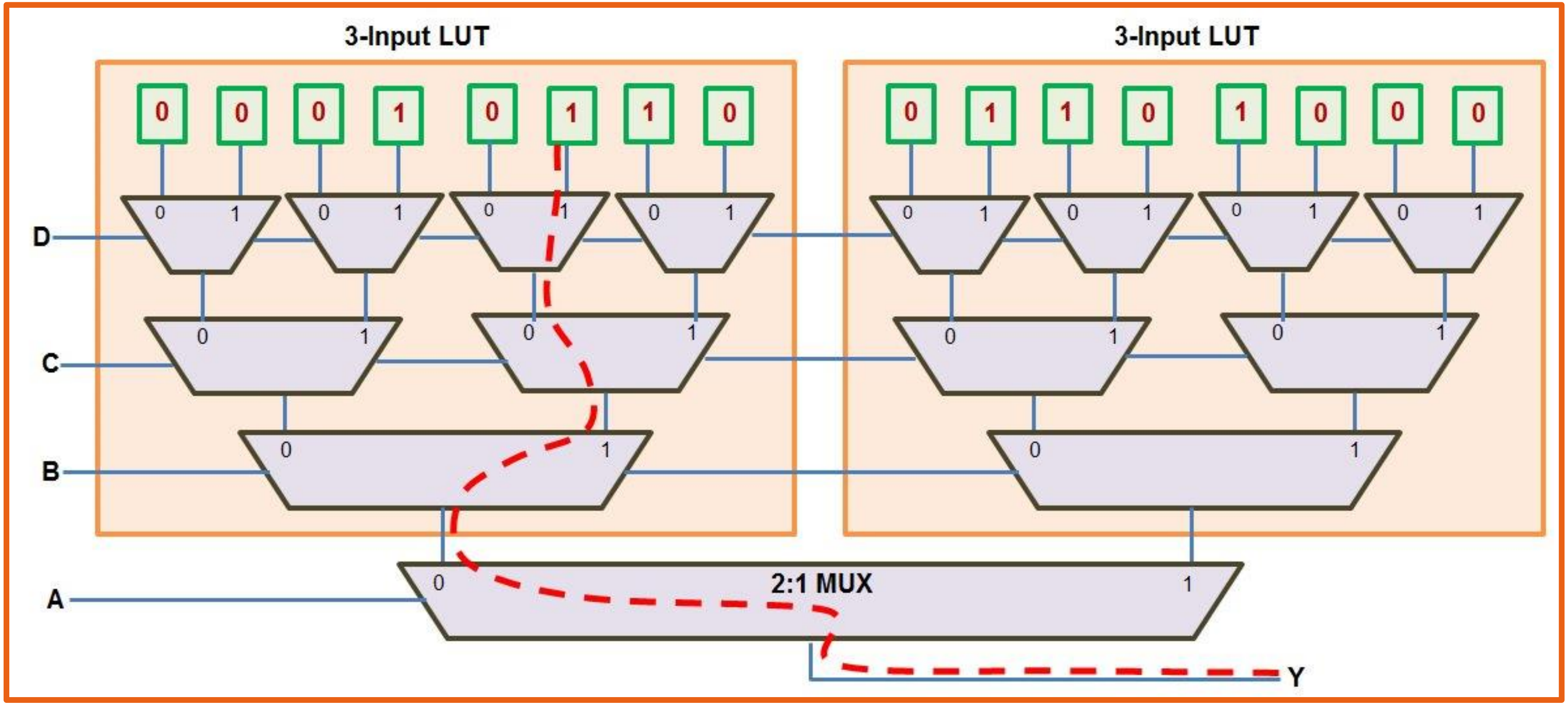
Look-up Tables and Registers

Custom XOR

Custom state machine

Custom 64-bit bit-shuffle and encode

LUT – The Foundation



LUT – The Foundation

- The green boxes at the top of this diagram correspond to various signals from other parts of our design. Here we have 16 of them and we can imagine them being numbered 0 to 15.
- Therefore, with just one 4-input LUT we can design a circuit that selects any one of those signals to get passed to our output, Y.
- Each entry is coded as 4-bit Boolean with A, B, C, and D representing each of the bits.
- The logic is called a lookup table because the output is selected by "looking up" the correct programmed level and routing it through the multiplexers based on the LUT input signals.

LUT – The Foundation

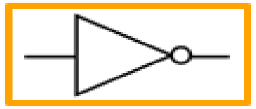
- If you follow the red path in reverse from bottom to top, you will see by setting bit A to 0, bit B to 1, bit C to 0 and bit D to 1, we get the 6th green box. So the input code 0101 corresponds to a 1 being passed to output Y.
- If you are familiar with Boolean counting, you will recognize that 0101 represents the number 5, which is the 6th value in a number sequence beginning with 0.
- Similarly, to get the first box, we would have code 0000, and to get the last box we would have code 1111, which is 15 in Boolean. Therefore, by using just 4 select bits, you can create any logic you need.
- The path show here corresponds to “not A” and B and “not C” and D.

- As a designer using the Quartus Prime tools, you will be able to see which logic function Quartus Prime assigned to a particular lookup table in your design.

LUT – The Foundation

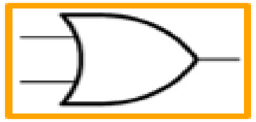
BACK TO THE BASICS

A	Z
0	1
1	0



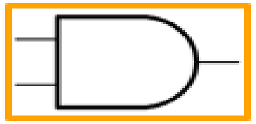
Inverter
Z = ~A

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



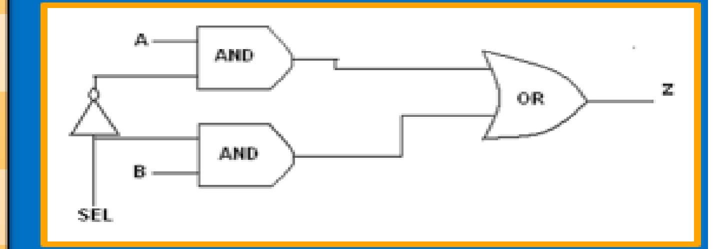
OR
Z = A | B

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



AND
Z = A & B

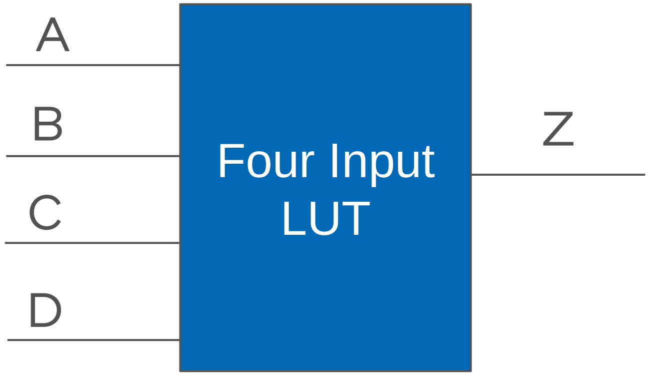
S	A	B	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



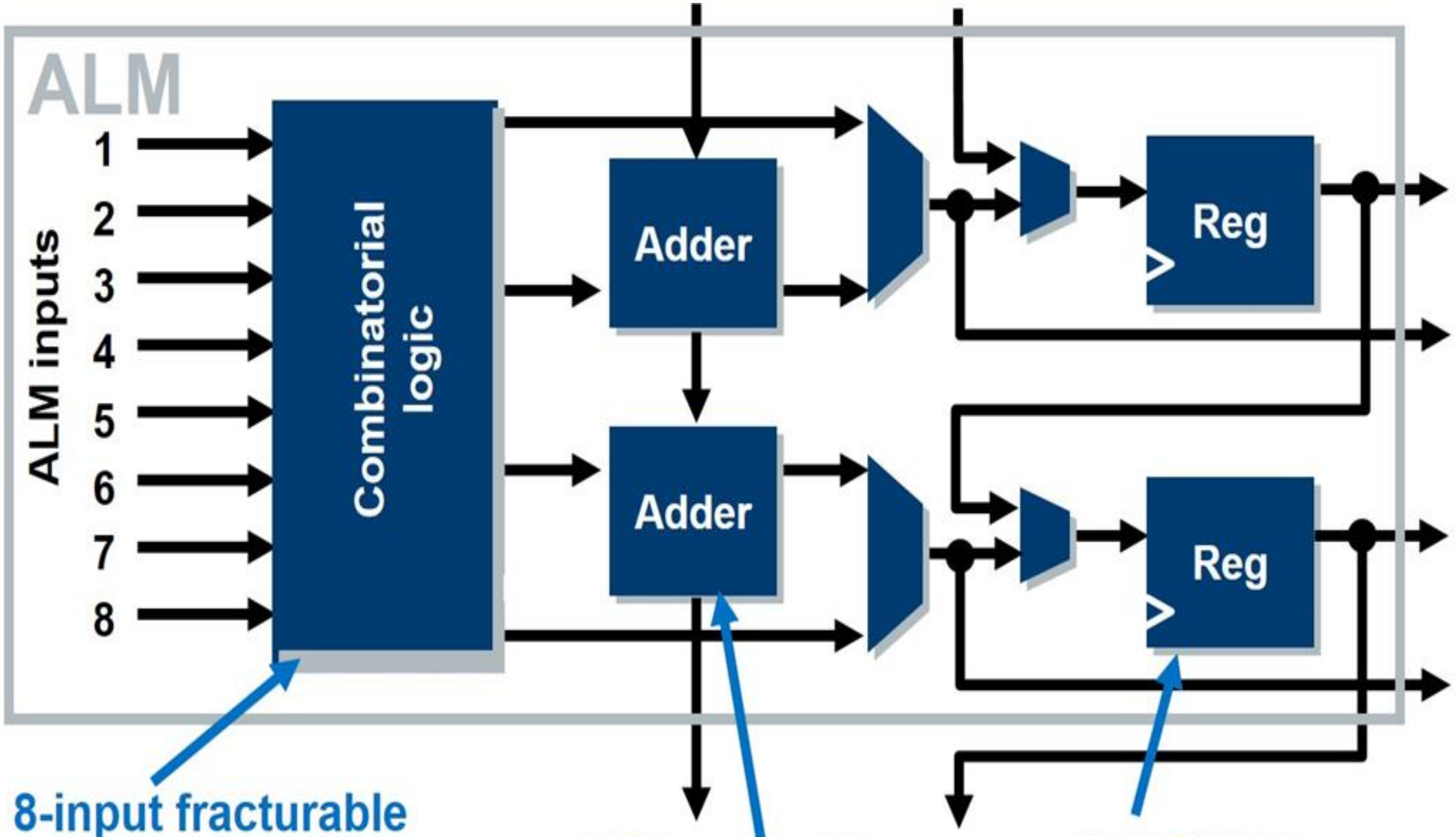
2:1 MUX
Z = (~S & A) | (S & B)

Lookup Tables

- A lookup table (LUT) is the foundation of an FPGA
- They come in various sizes – many FPGAs use a 4-bit lookup table (Intel FPGAs use 4- and 8-bit lookup tables)
- For 4-bit LUT, Z = f(A,B,C,D)
  - Where is f is a Boolean function of any four variables (16 possible combinations)



Logic Array Building Blocks



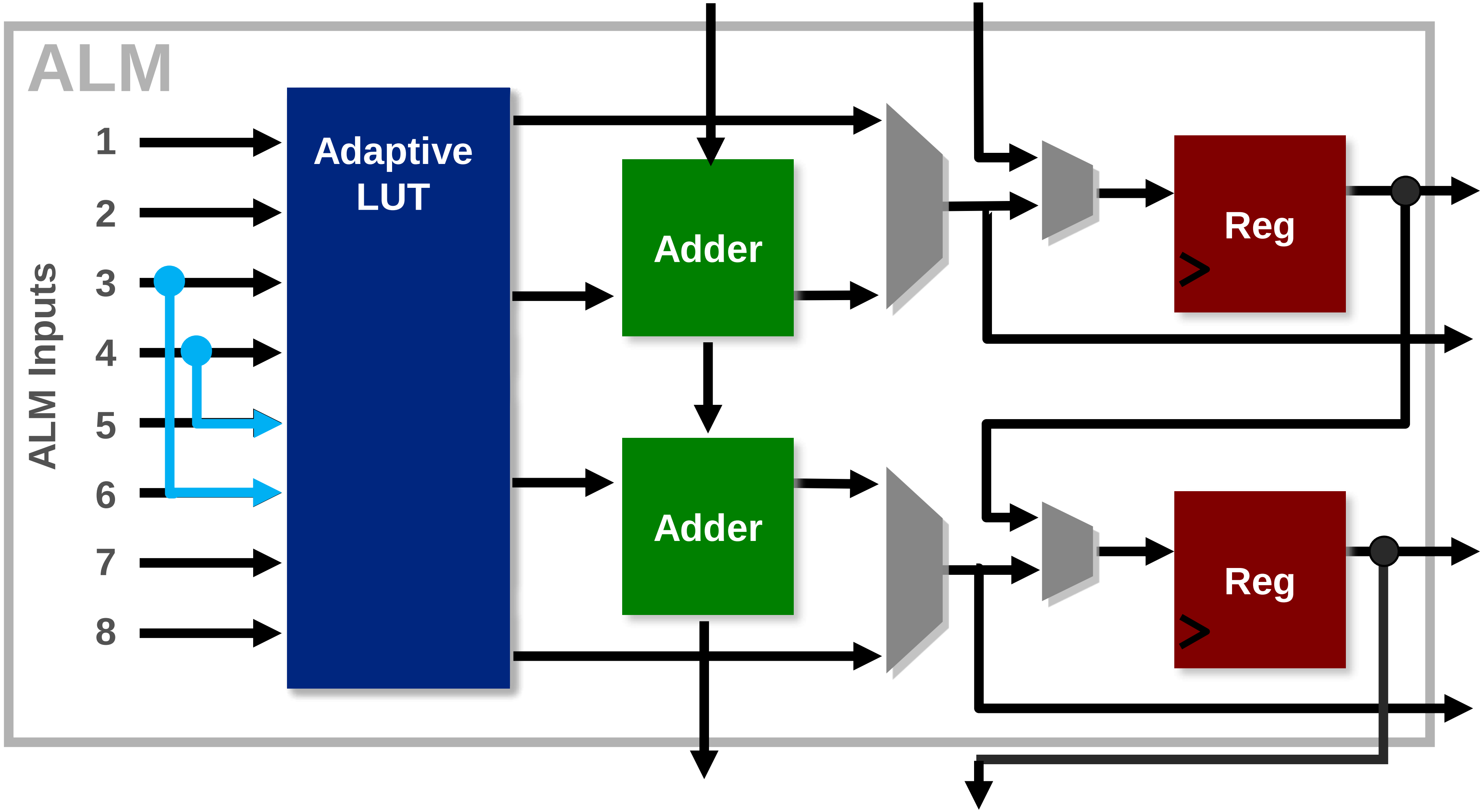
8-input fracturable look-up table (LUT)

2 3-input adders

2 registers

x 10

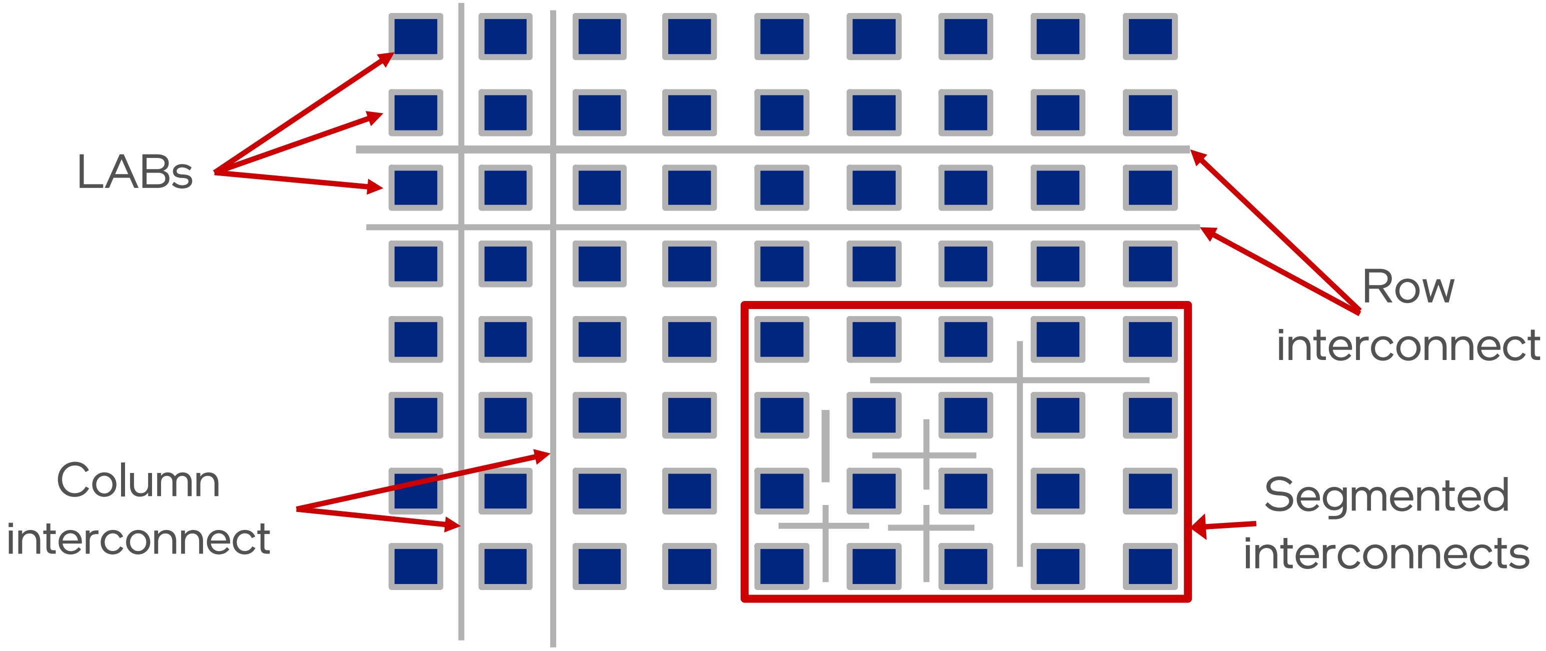
The Adaptive Logic Module (ALM)



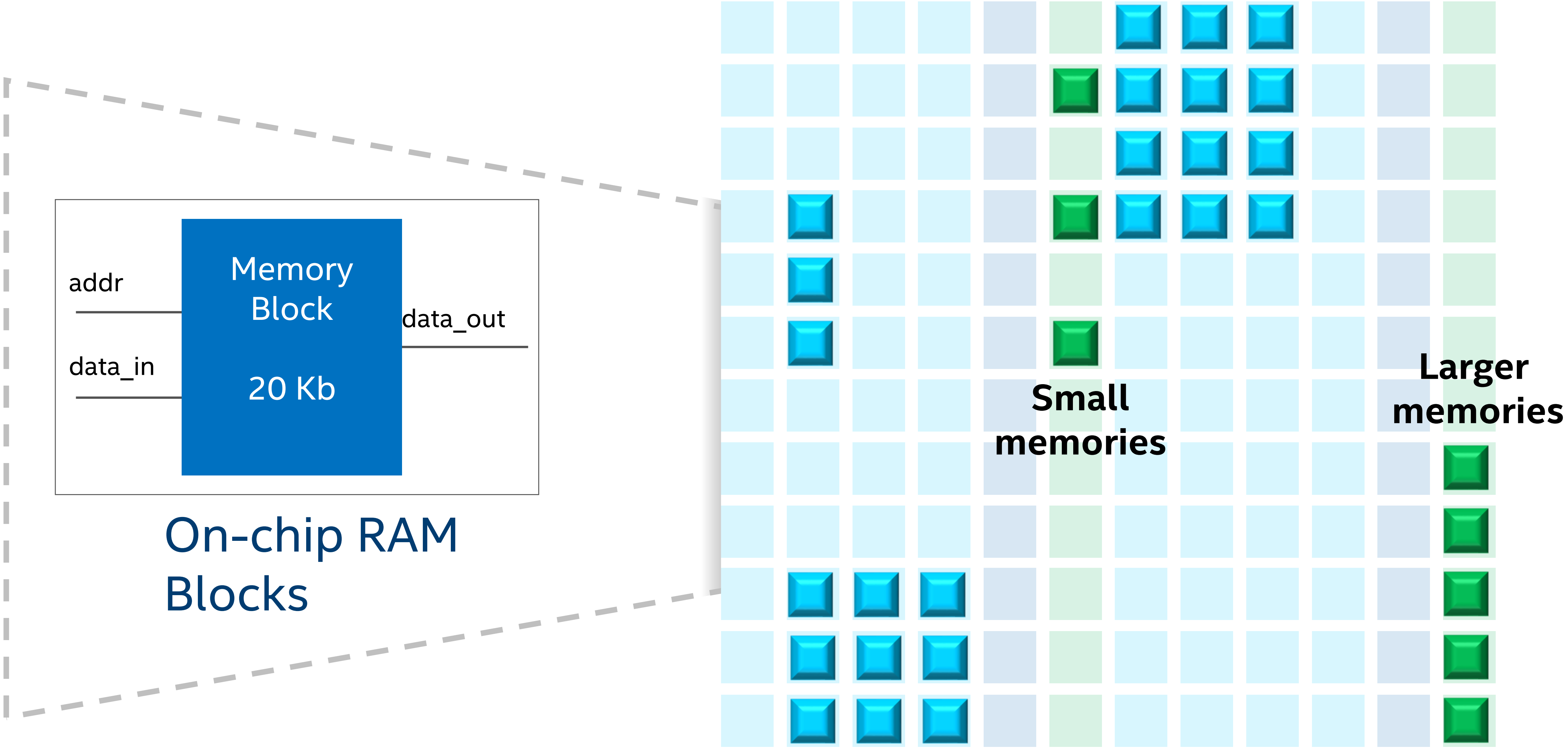
Logic Array Blocks (LABs)

- Groups of 10 ALMs
- Row and column programmable interconnect

- Arranged in an array
- Interconnect may span all or part of the array

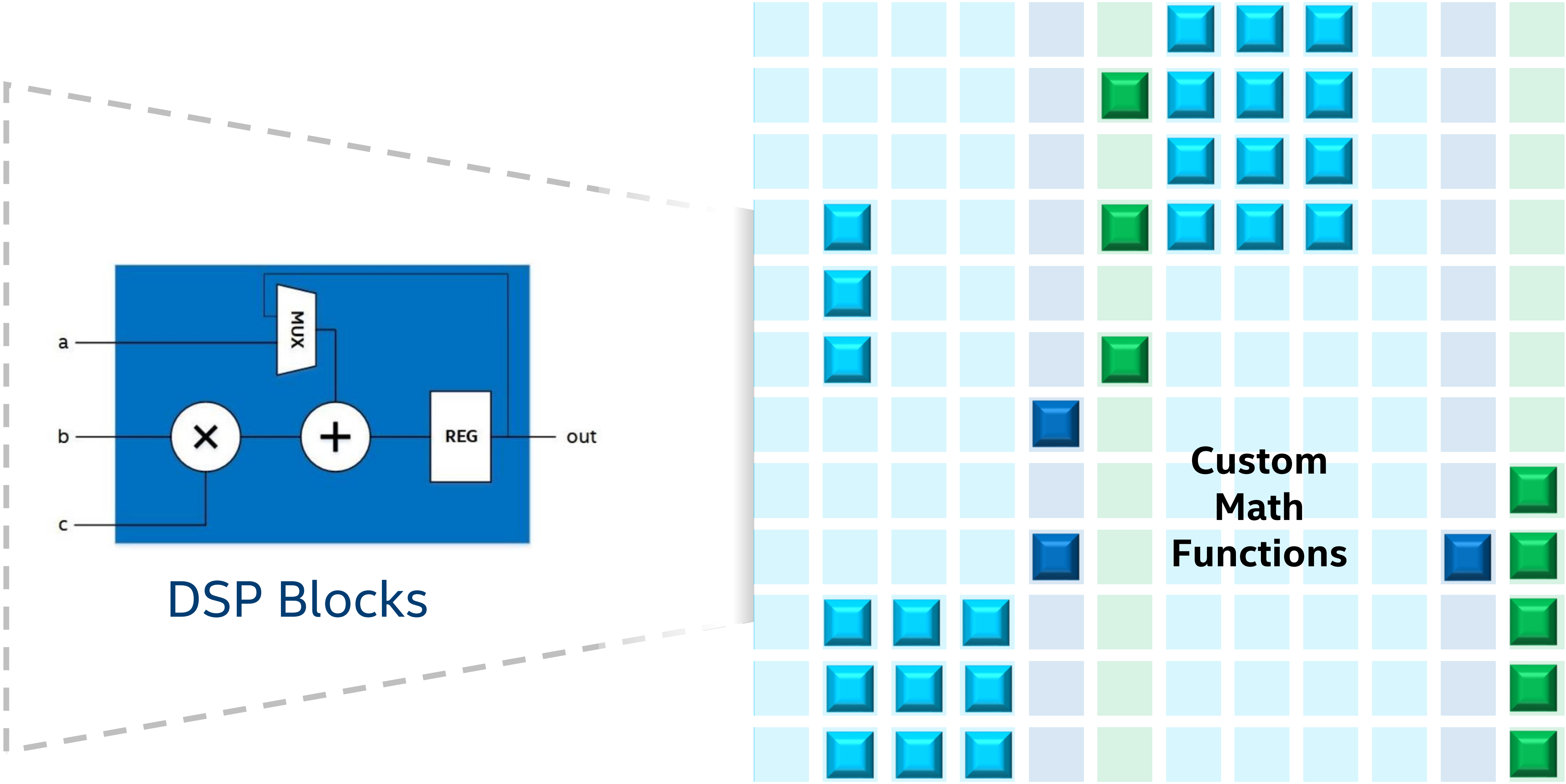


Blocks Used to Build What You've Coded



On-chip RAM Blocks

Blocks Used to Build What You've Coded

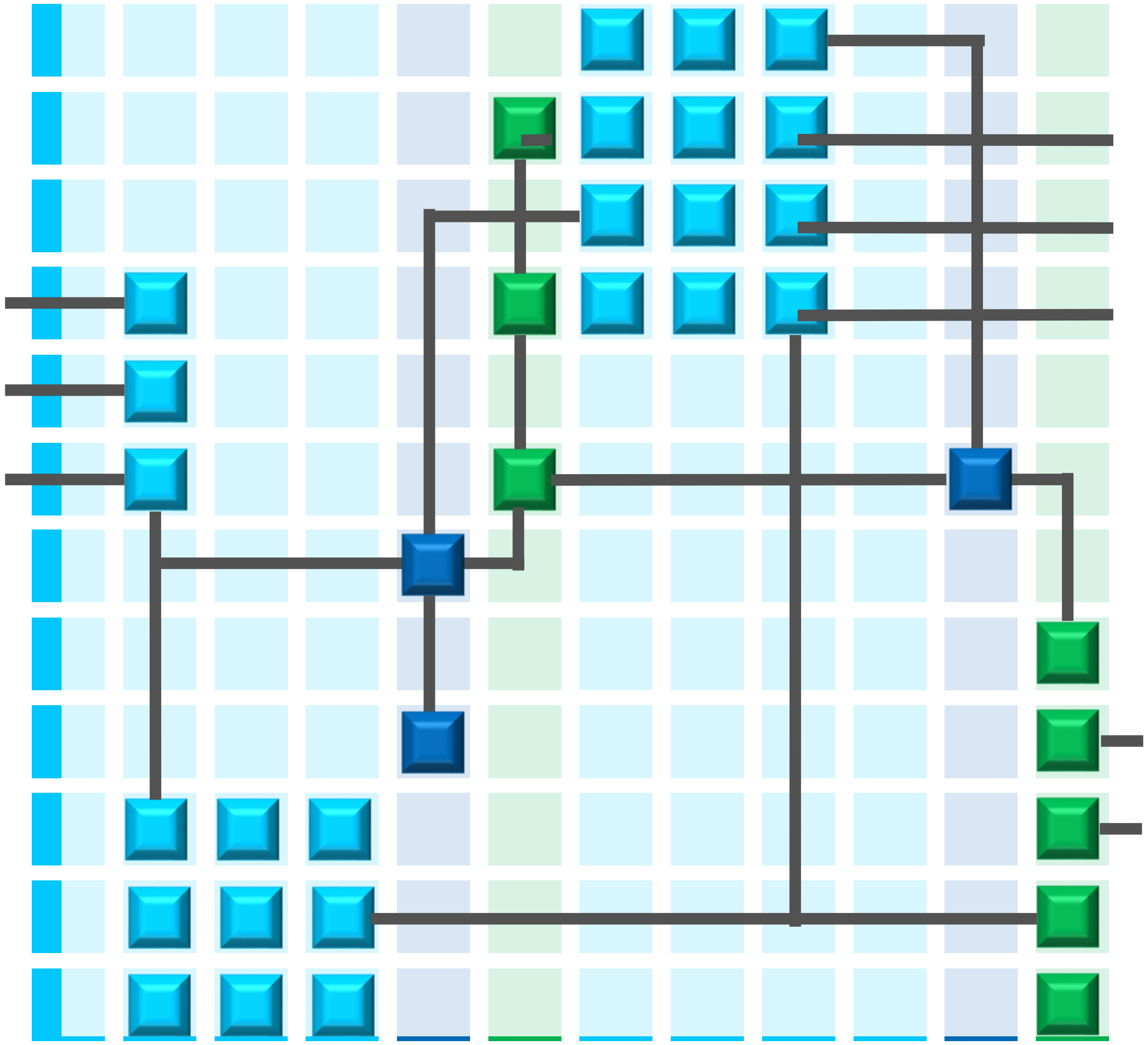


DSP Blocks

Custom Math Functions

Then, It's All Connected Together

Blocks are connected with **custom routing** determined by your code



Benefits of FPGA

REPROGRAMMABLE & FLEXIBLE

PRODUCT LONGEVITY

REDUCED TIME-TO-MARKET

MARKET-SIZE OPTIMIZED

FPGA Market…

Consumer Automotive

Entertainment
Broadband
Audio/video
Video display

Automotive
Navigation
Entertainment

Test, Measurement, & Medical

Instrumentation
Medical
Test equipment
Manufacturing

Communications Broadcast

Wireless
Cellular
Base stations
Wireless LAN

Networking
Switches
Routers

Wireline
Optical
Metro
Access

Broadcast
Studio
Satellite
Broadcasting

Military & Industrial

Military
Secure comm.
Radar
Guidance and control

Security & Energy Management
Card readers
Control systems
ATM

Computer & Storage

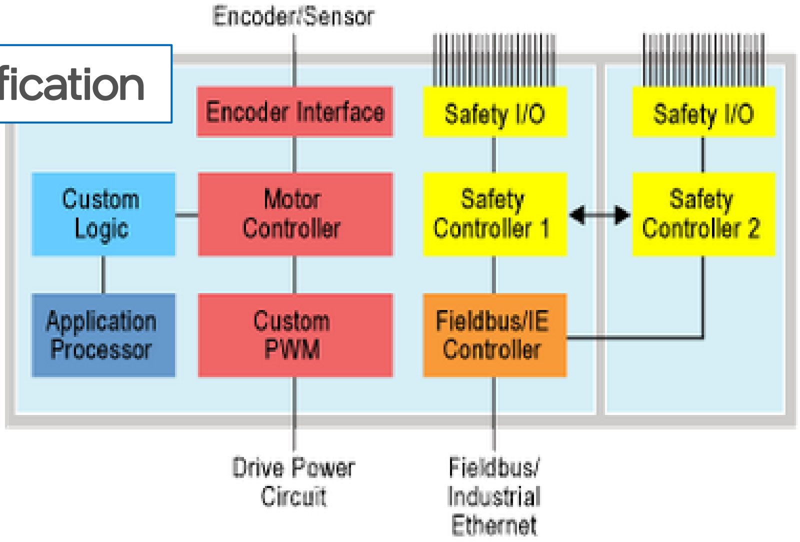
Computers
Servers
Mainframe

Storage
RAID
SAN

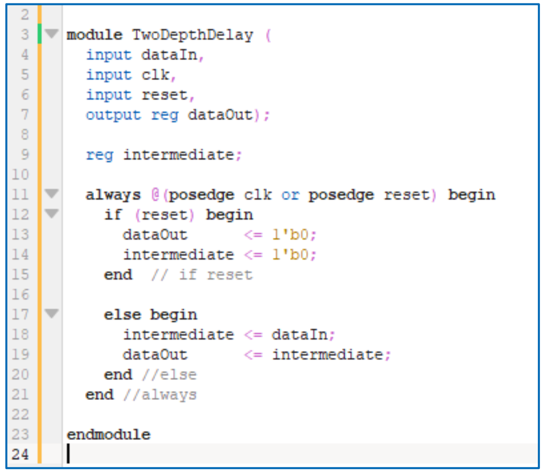
Office Automation
Copiers
Printers
MFP

Specification to HDL to Programmed FPGA

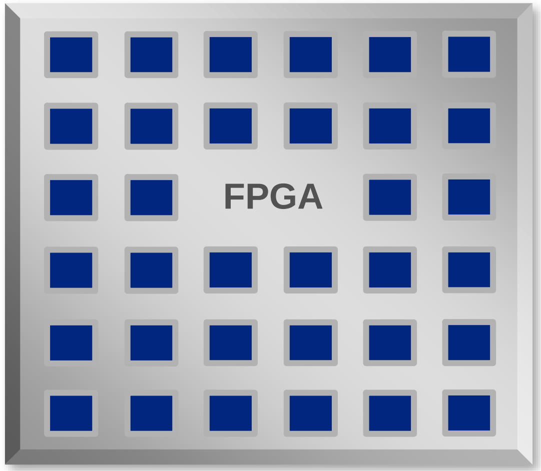
Specification



Intel® Quartus® Prime Software

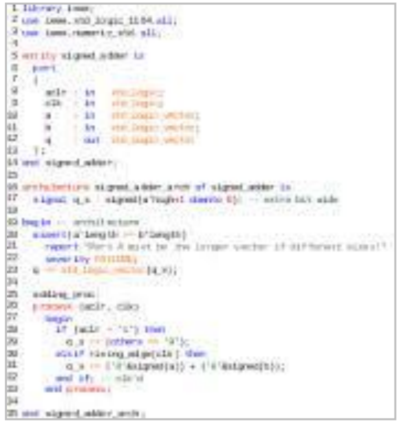


```
module TwoDepthDelay (
  input dataIn,
  input clk,
  input reset,
  output reg dataOut);

  reg intermediate;

  always @(posedge clk or posedge reset) begin
    if (reset) begin
      dataOut      <= 1'b0;
      intermediate <= 1'b0;
    end  // if reset

    else begin
      intermediate <= dataIn;
      dataOut      <= intermediate;
    end //else
  end //always

endmodule
```



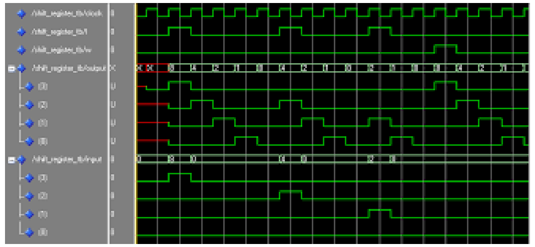
Traditional FPGA Design Process

HDL

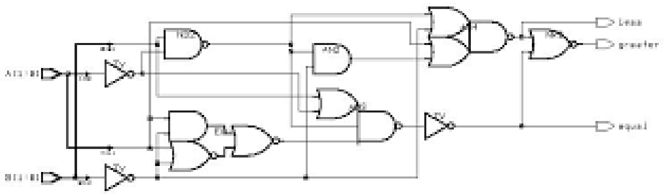


```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity signed_adder is
  port
  (
    aclr  : in   std_logic;
    clk   : in   std_logic;
    a     : in   std_logic_vector;
    b     : in   std_logic_vector;
    q     : out  std_logic_vector
  );
end signed_adder;

architecture signed_adder_arch of signed_adder is
  signal q_s : signed(a'high+1 downto 0); -- extra bit wide

begin -- architecture
  assert(a'length >= b'length)
    report "Port A must be the longer vector if different sizes!"
    severity FAILURE;
  q <= std_logic_vector(q_s);

  adding_proc:
  process (aclr, clk)
    begin
      if (aclr = '1') then
        q_s <= (others => '0');
      elsif rising_edge(clk) then
        q_s <= ('0'&signed(a)) + ('0'&signed(b));
      end if; -- clk'd
    end process;

end signed_adder_arch;
```

Behavioral Simulation



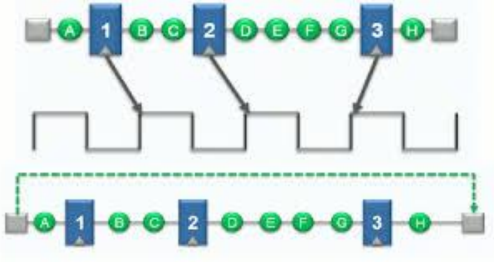
Synthesis



Place & Route / Timing Analysis / Timing Closure

Intel® Quartus® Prime

Design Software



Intel FPGAs

- Flexible, multi-functional reprogrammable silicon
- Custom hardware functionality
- Bare-metal speed and reliability
- Truly parallel in nature

INTEL® FPGA : APPLICATION SPECIFIC PERFORMANCE

The right performance and features for the right application

Management, Sensors and edge devices

intel MAX® 10 inside™

- Board Management
- Edge Compute
- I/O Expansion
- Automobile sensors, traffic sensors

Vision systems, and purpose-built, application-specific hardware

intel CYCLONE® 10 inside™

- Machine Vision
- Embedded Vision
- Robotics
- Infotainment

Scalable and efficient computing performance

intel ARRIA® 10 inside™

- Datacenter
- Networking
- Military / Defense
- ADAS

Cloud, datacenter, and HPC

intel STRATIX® 10 inside™

- Datacenter / CSP Acceleration
- 5G Wireless
- Network Communications
- Military / Defense

Cloud, datacenter, and HPC

intel AGILEX™ inside™

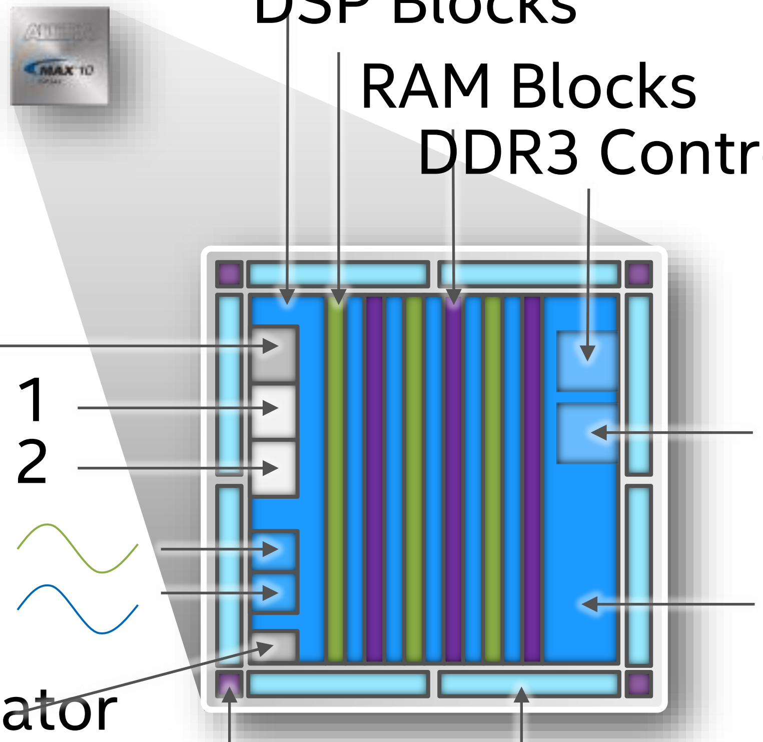
- Datacenter / CSP Acceleration
- 5G Wireless Infrastructure
- Network Communications
- Military / Defense

MAX 10 FPGA

Packages as

small as

3 x 3 mm



Up to 50K Logic Elements

DSP Blocks

RAM Blocks

DDR3 Controller

Flash Memory (NOR)

User Flash

Configuration Flash 1

Configuration Flash 2

12-bit SAR ADCs

Power Regulator

4 PLLs

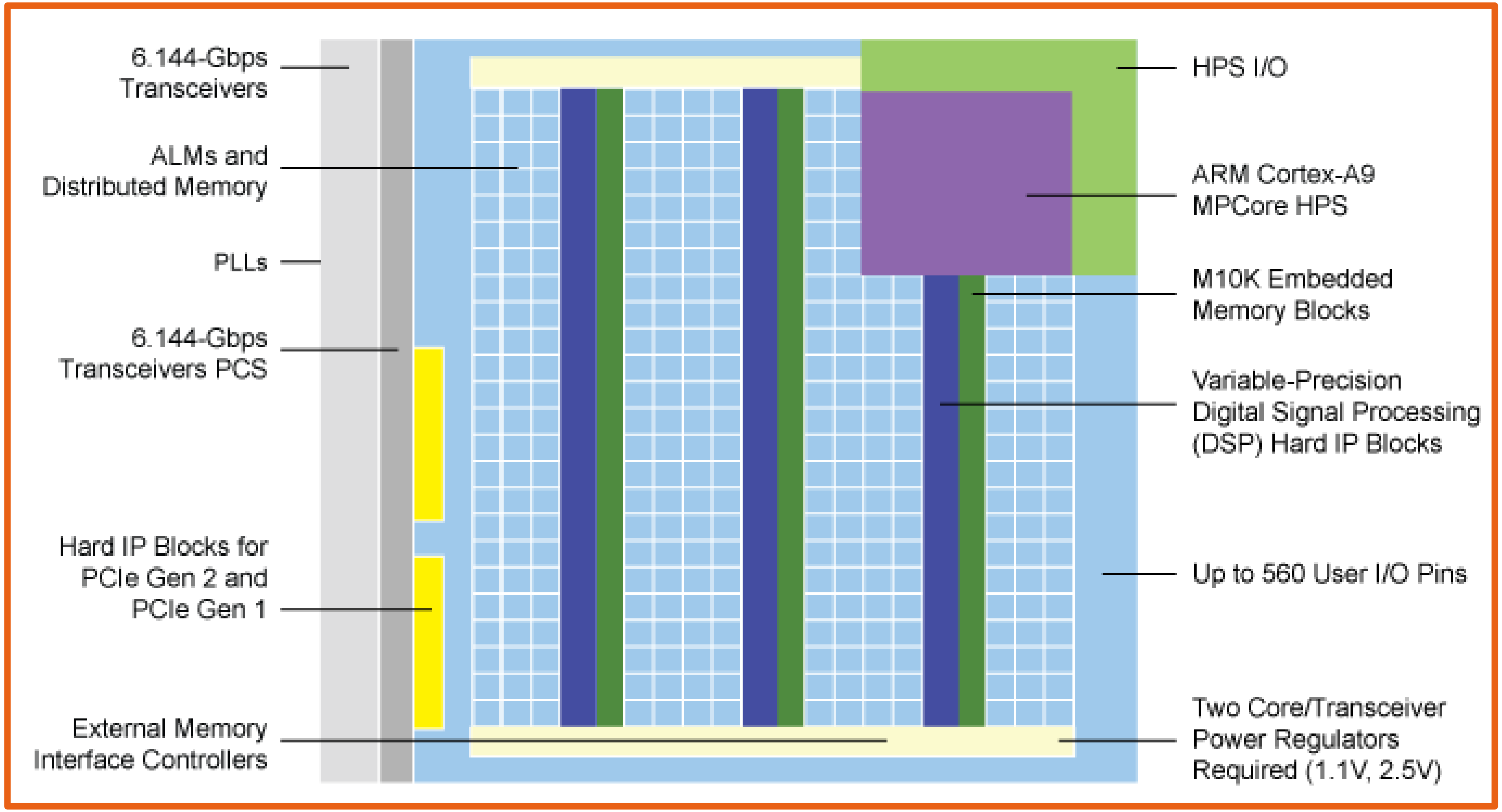
8 I/O Banks

Nios® II

32-bit Processor

On-chip Oscillator

FPGAs "Hardened" features (Cyclone V)

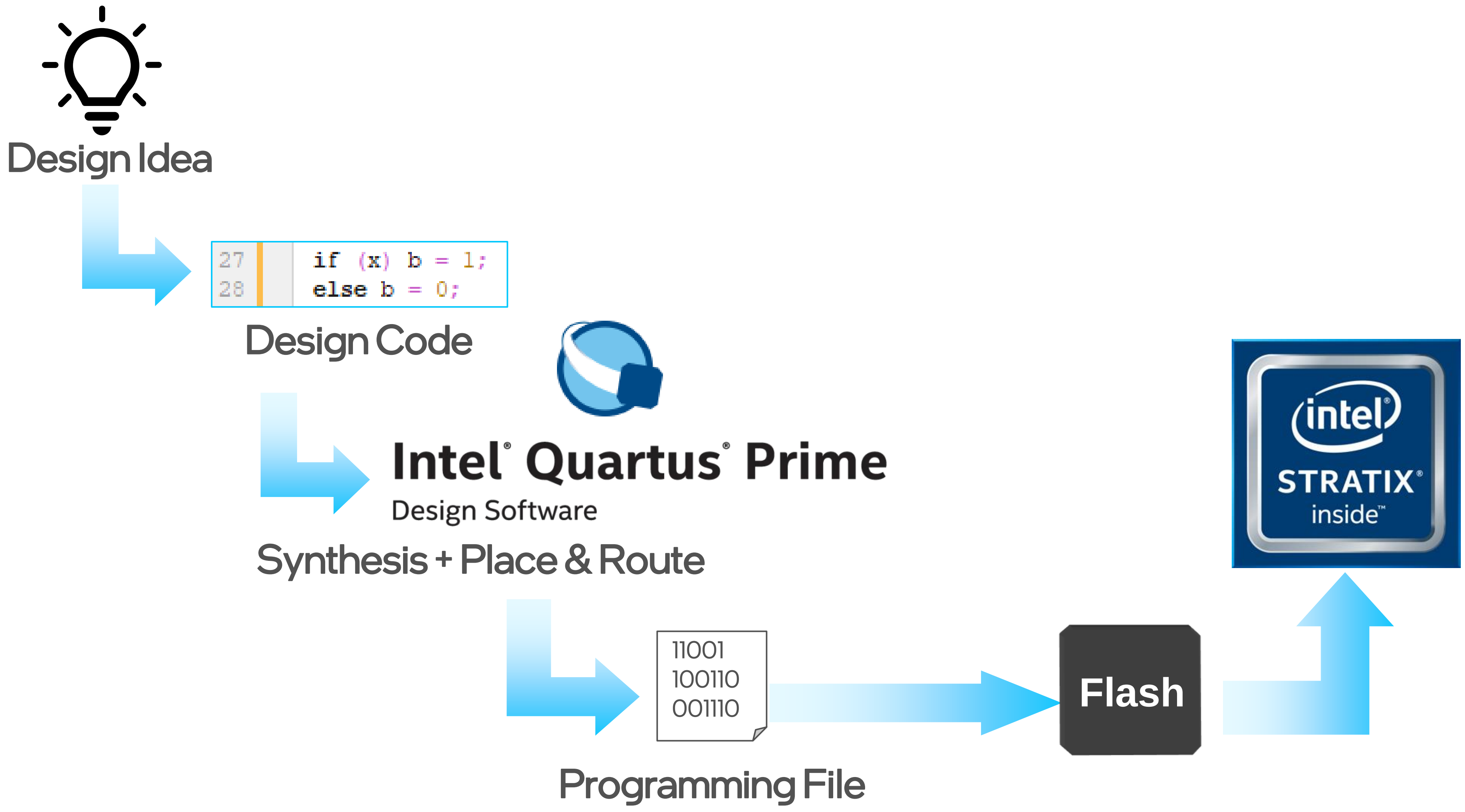


6.144-Gbps Transceivers
ALMs and Distributed Memory
PLLs
6.144-Gbps Transceivers PCS
Hard IP Blocks for PCIe Gen 2 and PCIe Gen 1
External Memory Interface Controllers
HPS I/O
ARM Cortex-A9 MPCore HPS
M10K Embedded Memory Blocks
Variable-Precision Digital Signal Processing (DSP) Hard IP Blocks
Up to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

Section Two

High Level Description of FPGA Design

How do you design for an FPGA?



Design Idea

```
if (x) b = 1;
else b = 0;
```

Design Code

Intel® Quartus® Prime

Design Software

Synthesis + Place & Route

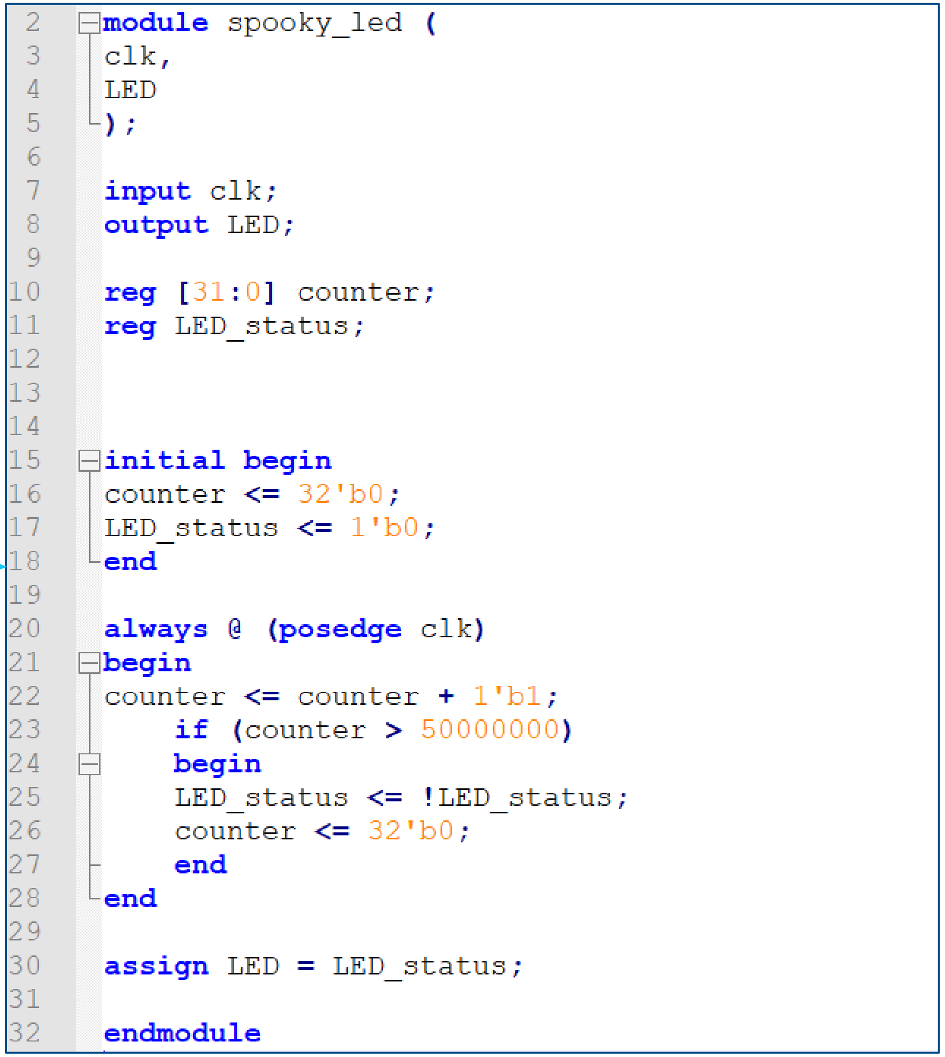
Programming File

Design Entry

Various forms of design entry

- Verilog
- VHDL
- State machine
- Schematic

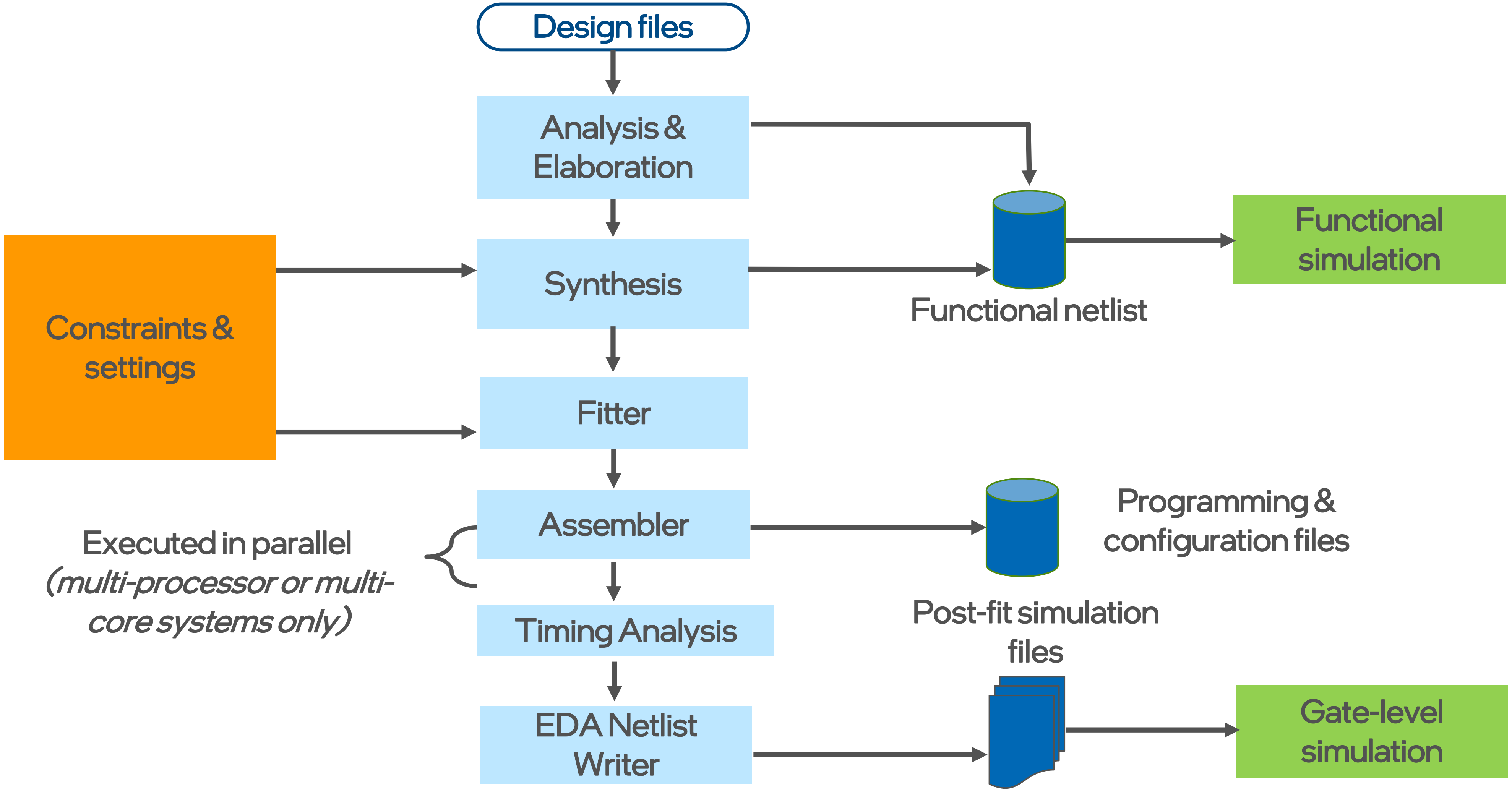
Design Idea



```
module spooky_led (
clk,
LED
);

input clk;
output LED;

reg [31:0] counter;
reg LED_status;



initial begin
counter <= 32'b0;
LED_status <= 1'b0;
end

always @ (posedge clk)
begin
counter <= counter + 1'b1;
    if (counter > 50000000)
    begin
    LED_status <= !LED_status;
    counter <= 32'b0;
    end
end

assign LED = LED_status;

endmodule
```

Design code (HDL)

Intel® Quartus® Prime Design Software Full Compilation Flow

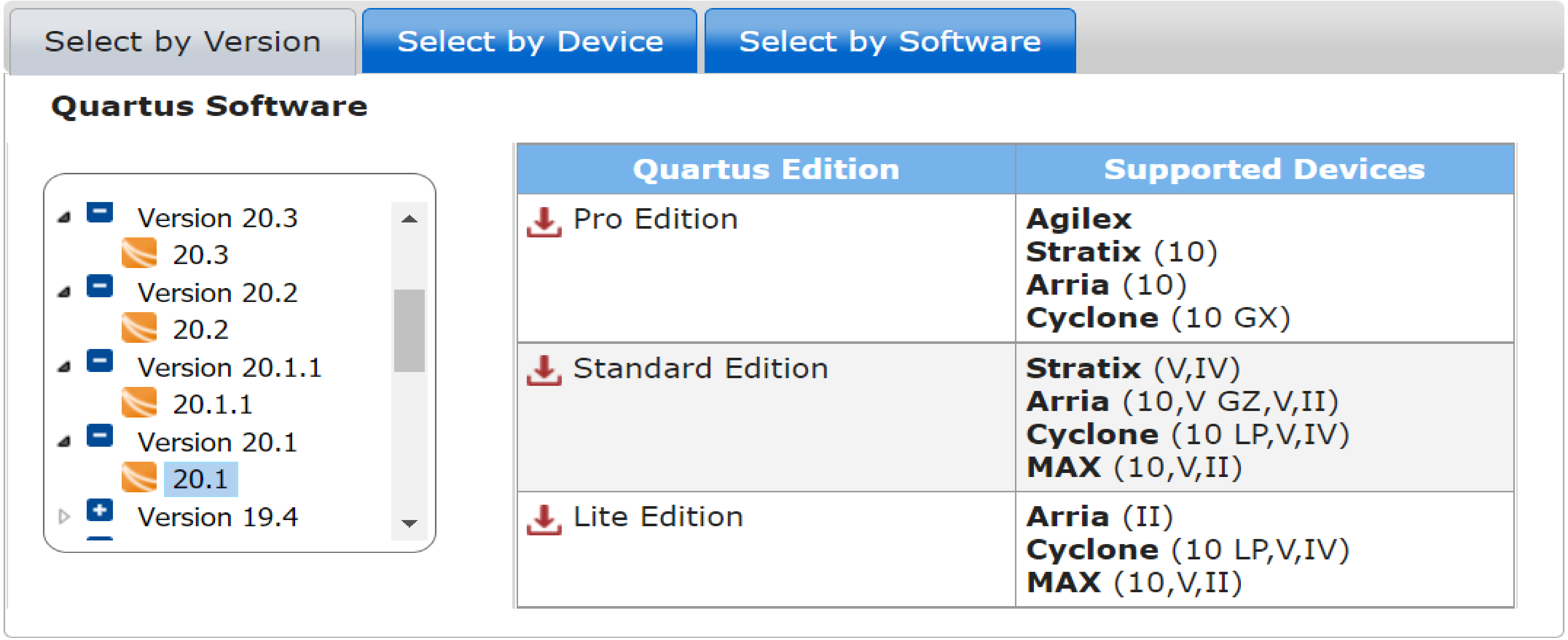


Executed in parallel *(multi-processor or multi-core systems only)*

Intel® Quartus® Prime Design Software

- Fully-integrated development tool
    - Multiple design entry methods
    - Logic synthesis
    - Place & route
    - Device programming
- Simulation
    - Supports standard HDL simulation tools
    - Includes Questa*-Intel FPGA Starter Edition tool
        - Optional upgrade to Questa-Intel FPGA Edition tool
    - See comparison
        - https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/questa-edition.html

Software Selector



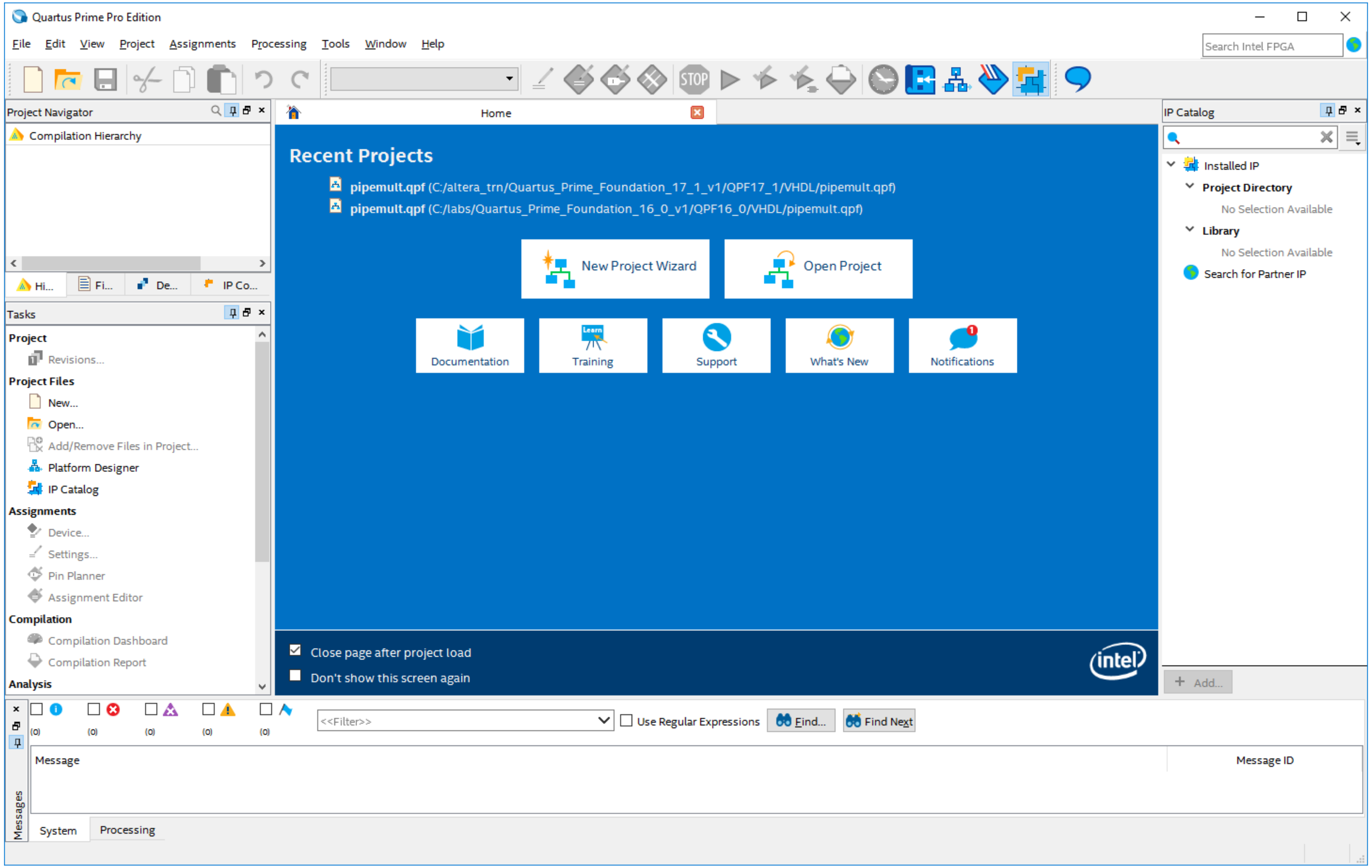
Quartus Software

Quartus Edition	Supported Devices
Pro Edition	**Agilex** **Stratix** (10) **Arria** (10) **Cyclone** (10 GX)
Standard Edition	**Stratix** (V,IV) **Arria** (10,V GZ,V,II) **Cyclone** (10 LP,V,IV) **MAX** (10,V,II)
Lite Edition	**Arria** (II) **Cyclone** (10 LP,V,IV) **MAX** (10,V,II)

Section Three

Designing with Intel® Quartus® Prime Software

Welcome to the Intel® Quartus® Prime Design Software!



Quartus Prime Pro Edition

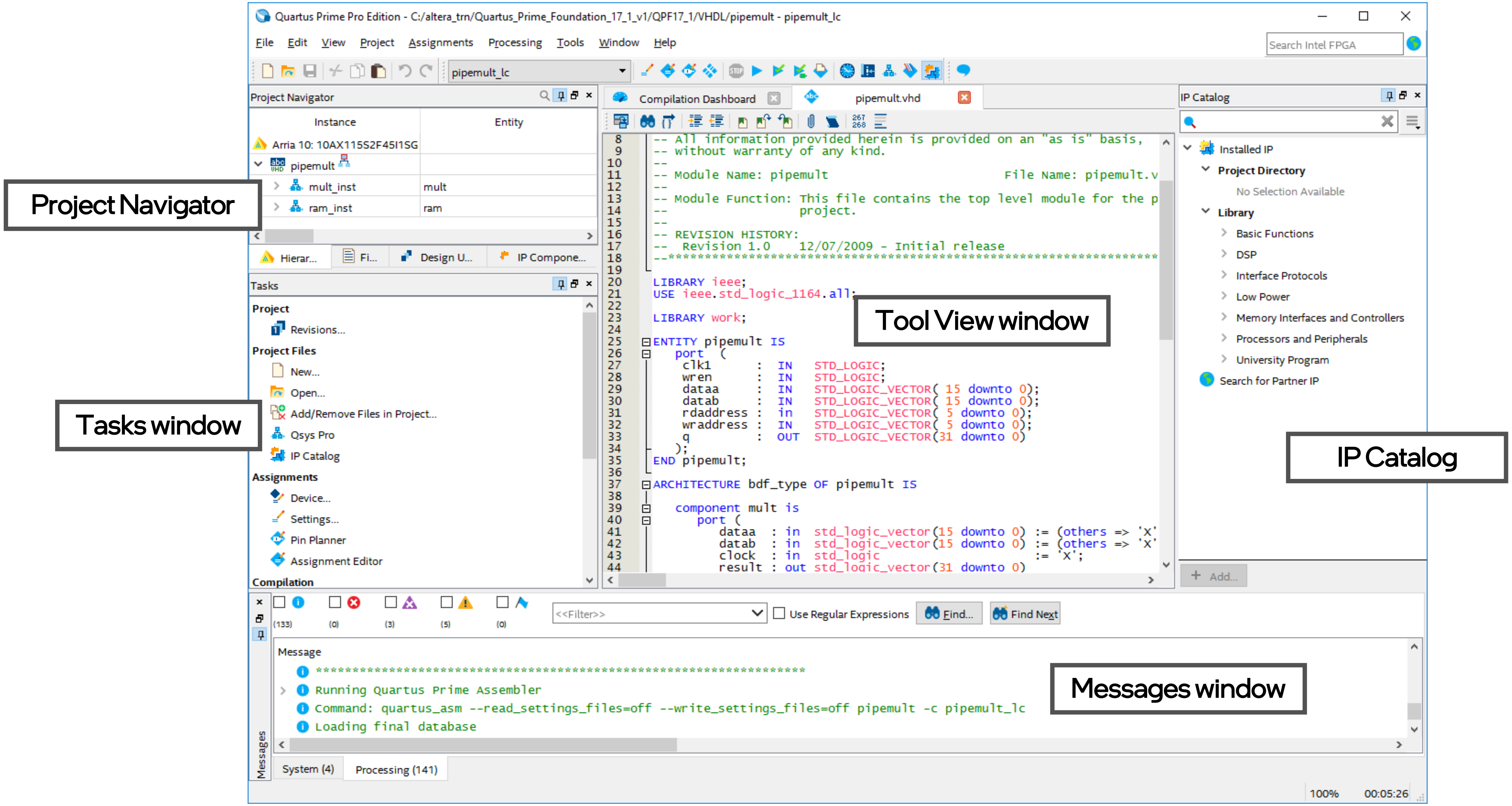
Recent Projects

- pipemult.qpf (C:/altera_trn/Quartus_Prime_Foundation_17_1_v1/QPF17_1/VHDL/pipemult.qpf)
- pipemult.qpf (C:/labs/Quartus_Prime_Foundation_16_0_v1/QPF16_0/VHDL/pipemult.qpf)

☑ Close page after project load

□ Don't show this screen again

Default Operating Environment



Quartus Prime Pro Edition - C:/altera_trn/Quartus_Prime_Foundation_17_1_v1/QPF17_1/VHDL/pipemult - pipemult_lc

Instance	Entity
Arria 10: 10AX115S2F45I1SG	
pipemult	
mult_inst	mult
ram_inst	ram

```
-- All information provided herein is provided on an "as is" basis,
-- without warranty of any kind.
--
-- Module Name: pipemult                          File Name: pipemult.v
--
-- Module Function: This file contains the top level module for the p
--                  project.
--
-- REVISION HISTORY:
--  Revision 1.0    12/07/2009 - Initial release
--*****************************************************************

LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY work;

ENTITY pipemult IS
   port  (
    clk1      :  IN   STD_LOGIC;
    wren      :  IN   STD_LOGIC;
    dataa     :  IN   STD_LOGIC_VECTOR( 15 downto 0);
    datab     :  IN   STD_LOGIC_VECTOR( 15 downto 0);
    rdaddress :  in   STD_LOGIC_VECTOR( 5 downto 0);
    wraddress :  IN   STD_LOGIC_VECTOR( 5 downto 0);
    q         :  OUT  STD_LOGIC_VECTOR(31 downto 0)
   );
END pipemult;

ARCHITECTURE bdf_type OF pipemult IS

   component mult is
      port (
         dataa  : in  std_logic_vector(15 downto 0) := (others => 'X'
         datab  : in  std_logic_vector(15 downto 0) := (others => 'X'
         clock  : in  std_logic                     := 'X';
         result : out std_logic_vector(31 downto 0)
```

Message

```
*******************************************************************
Running Quartus Prime Assembler
Command: quartus_asm --read_settings_files=off --write_settings_files=off pipemult -c pipemult_lc
Loading final database
```

Project Navigator

Tasks window

Tool View window

IP Catalog

Messages window

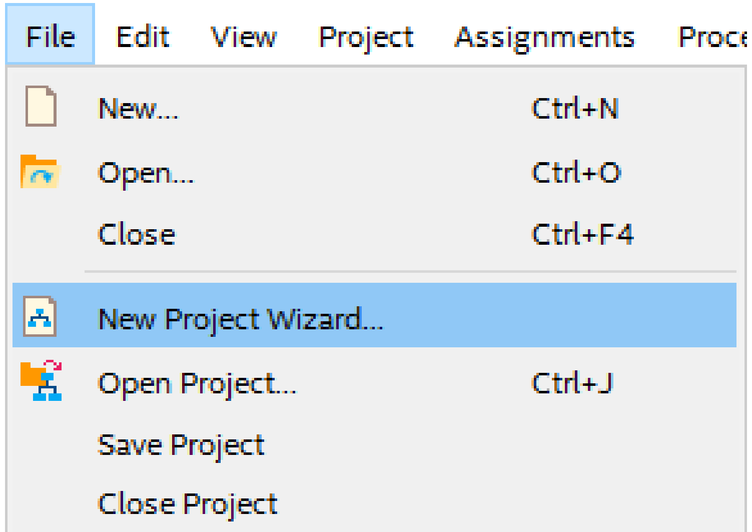
Section Three

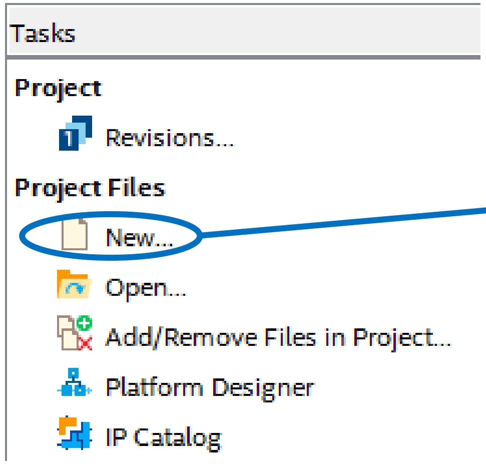
Creating Projects

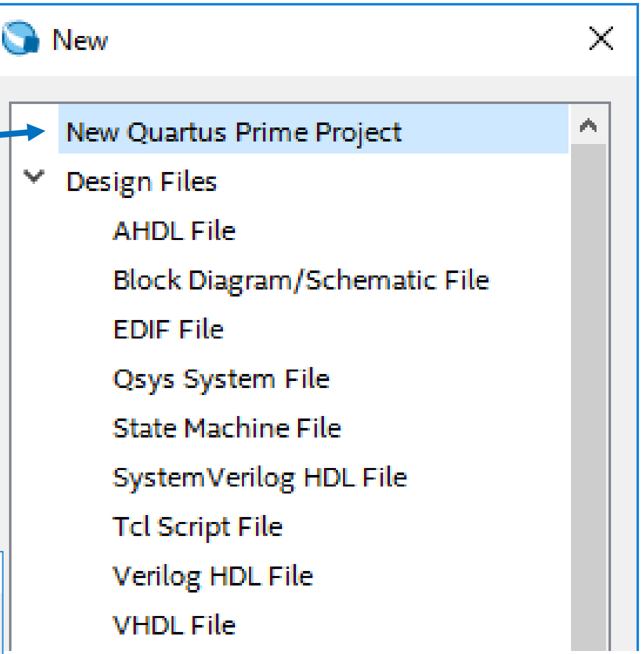
Intel® Quartus® Prime Design Software Projects

- Description
    - Collection of related design files & libraries
    - Must have a designated top-level entity
    - Target a single device
    - Store settings in the software settings file **(.qsf)**
    - Compiled netlist information stored in the **qdb** folder in the project directory
- Create new projects with the **New Project Wizard**
    - Can be created using Tcl scripts

New Project Wizard



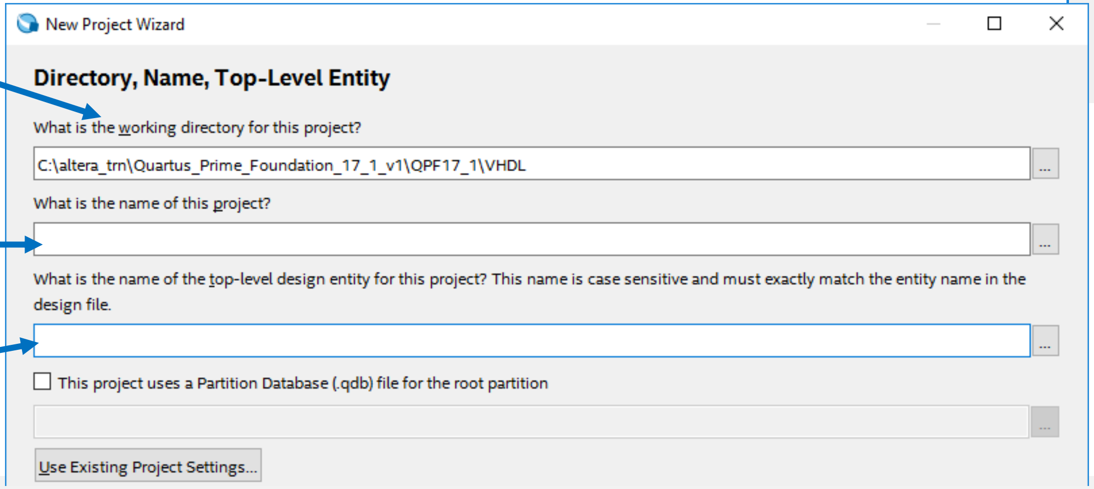




Select working directory

Name of project can be any name; recommend using top-level file name

Top-level entity does not need to be the same name as top-level file name


New Project Wizard

Directory, Name, Top-Level Entity

What is the working directory for this project?

C:\altera_trn\Quartus_Prime_Foundation_17_1_v1\QPF17_1\VHDL

What is the name of this project?

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

☐ This project uses a Partition Database (.qdb) file for the root partition

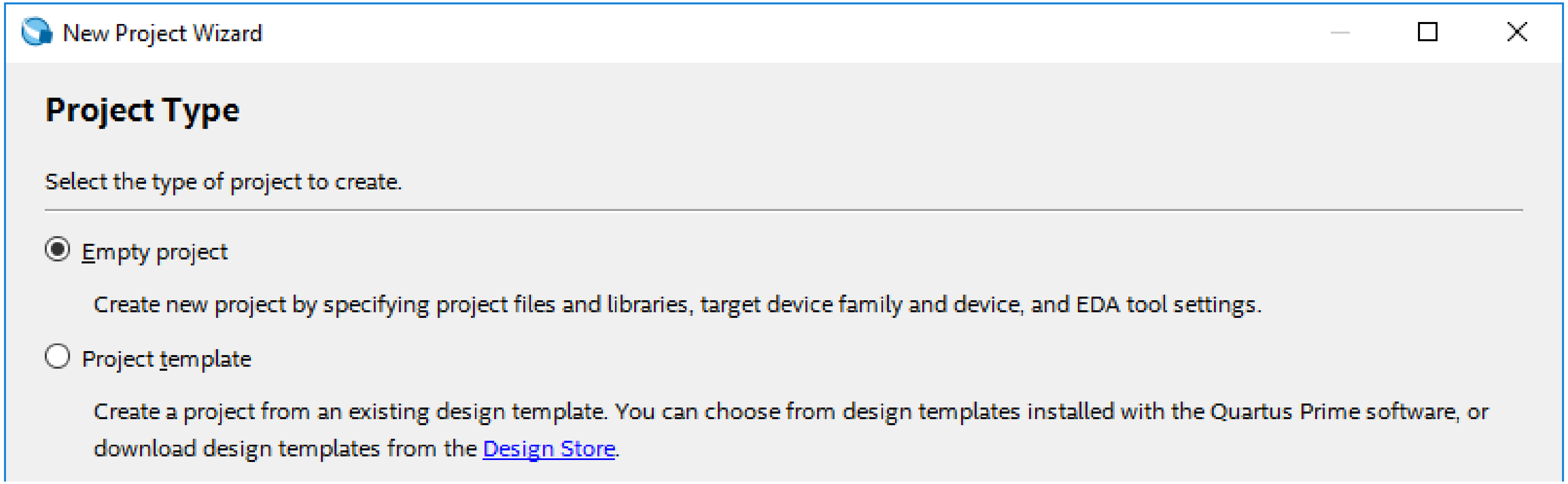
Use Existing Project Settings...

Tcl: project_new <project_name>

Project Type

- Create a blank project or use templates from Intel® FPGA Design Store

  https://fpgacloud.intel.com/devstore/



Project Type

Select the type of project to create.

● Empty project

Create new project by specifying project files and libraries, target device family and device, and EDA tool settings.

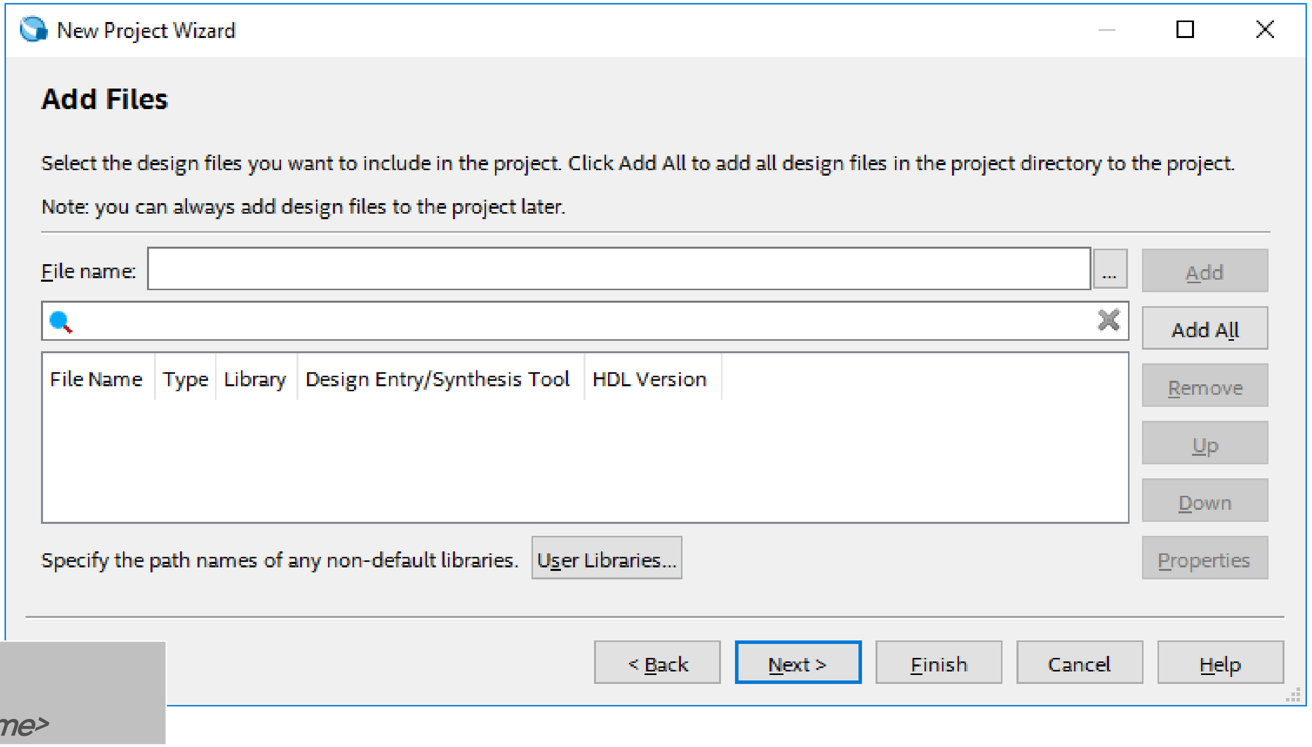
○ Project template

Create a project from an existing design template. You can choose from design templates installed with the Quartus Prime software, or download design templates from the Design Store.

Add Files

- Add design files
  - Graphic
  - VHDL
  - Verilog
  - SystemVerilog
  - EDIF
  - VQM
  - Intel® FPGA IP
  - Platform Designer

- Add library paths
  - User libraries
  - Pre-compiled VHLD packages



Add Files

Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.

Note: you can always add design files to the project later.

File name:

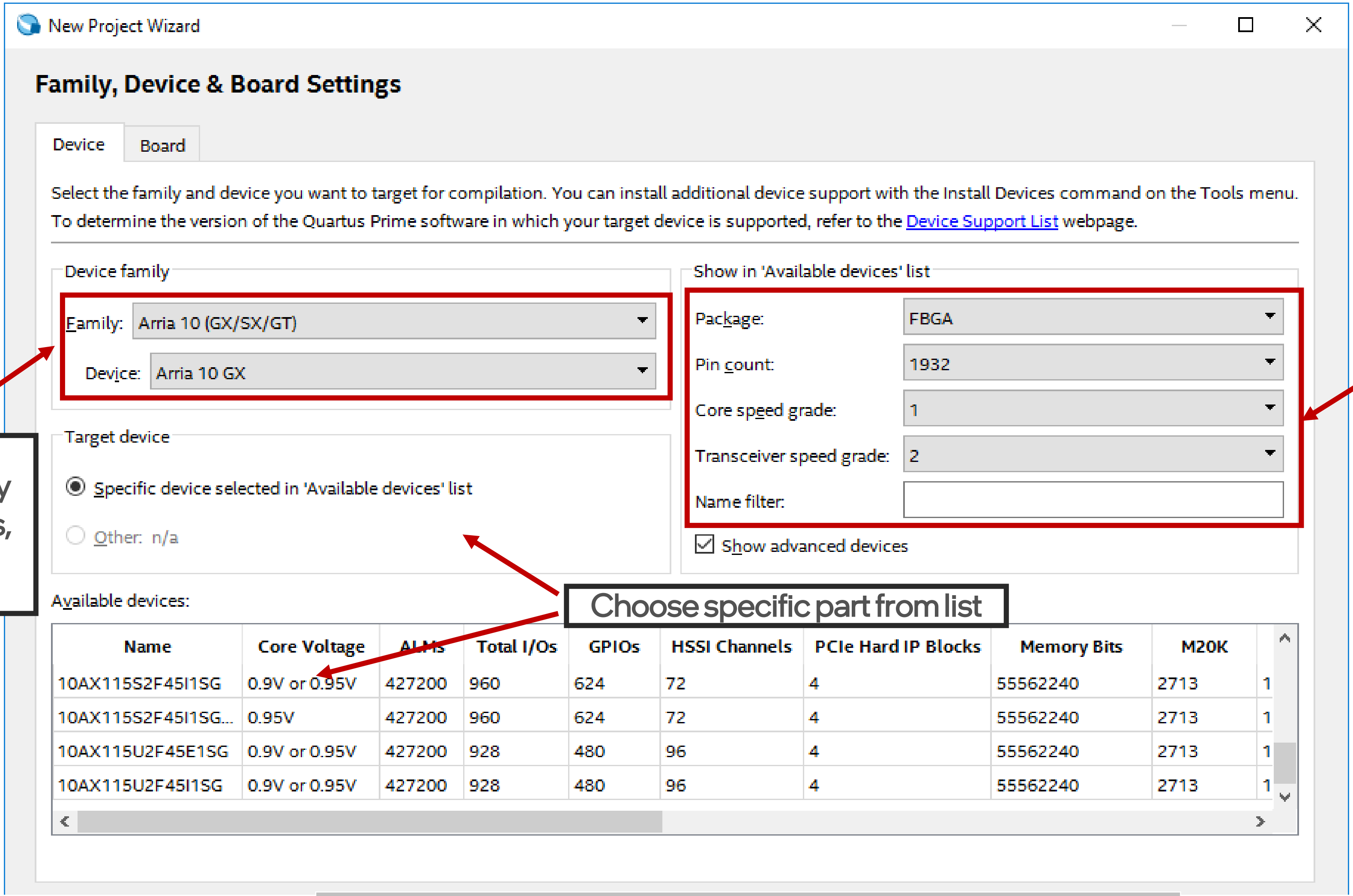
File Name	Type	Library	Design Entry/Synthesis Tool	HDL Version

Specify the path names of any non-default libraries. User Libraries...

Tcl: set_global_assignment –name VHDL_FILE <filename.vhd>

Tcl: set_global_assignment –name USER_LIBRARIES <library_path_name>

Device Selection



New Project Wizard

Family, Device & Board Settings

Device | Board

Select the family and device you want to target for compilation. You can install additional device support with the Install Devices command on the Tools menu. To determine the version of the Quartus Prime software in which your target device is supported, refer to the Device Support List webpage.

Device family

Family: Arria 10 (GX/SX/GT)

Device: Arria 10 GX

Target device

● Specific device selected in 'Available devices' list

○ Other: n/a

Show in 'Available devices' list

Package: FBGA

Pin count: 1932

Core speed grade: 1

Transceiver speed grade: 2

Name filter:

☑ Show advanced devices

Available devices:

Name	Core Voltage	ALMs	Total I/Os	GPIOs	HSSI Channels	PCIe Hard IP Blocks	Memory Bits	M20K	
10AX115S2F45I1SG	0.9V or 0.95V	427200	960	624	72	4	55562240	2713	1
10AX115S2F45I1SG...	0.95V	427200	960	624	72	4	55562240	2713	1
10AX115U2F45E1SG	0.9V or 0.95V	427200	928	480	96	4	55562240	2713	1
10AX115U2F45I1SG	0.9V or 0.95V	427200	928	480	96	4	55562240	2713	1

Choose device family & family category (transceiver options, SoC options, etc.)

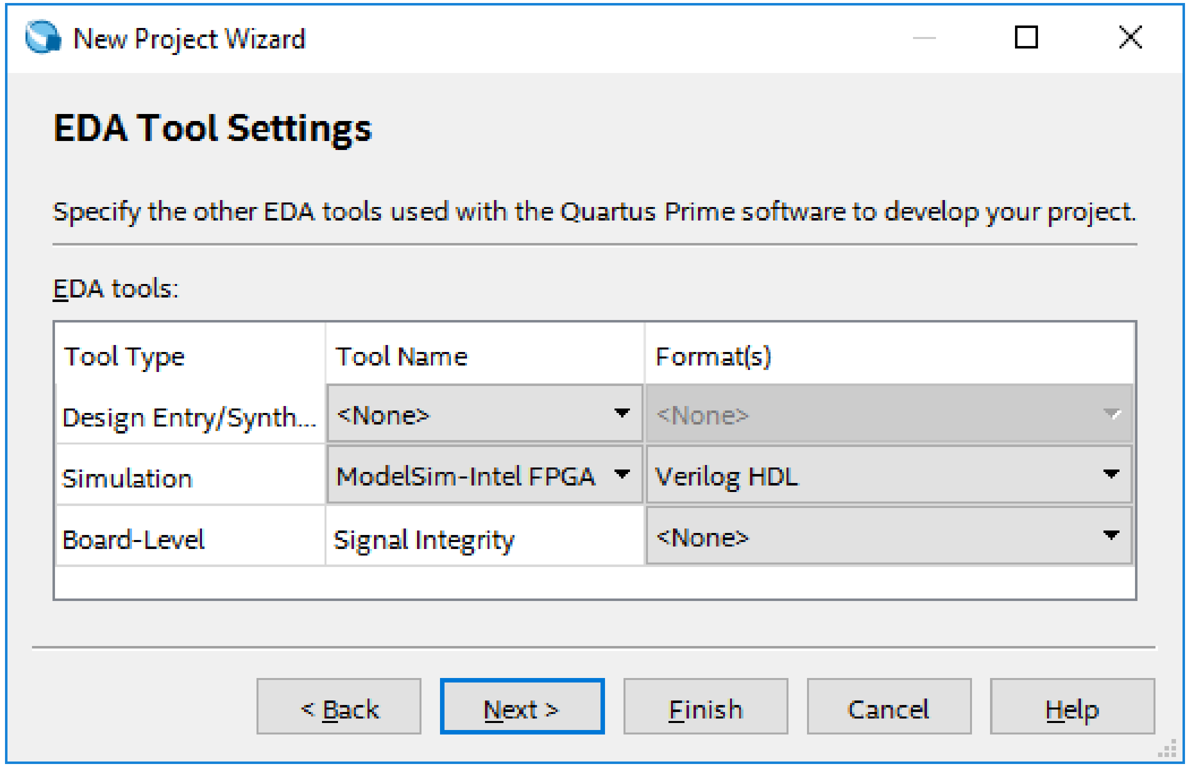
Filter device list

Choose specific part from list

Tcl: set_global_assignment –name FAMILY "device family name"
Tcl: set_global_assignment –name DEVICE <part_number>

EDA Tools Settings

- Choose EDA tools and file formats
- Settings can be changed or added later



New Project Wizard

EDA Tool Settings

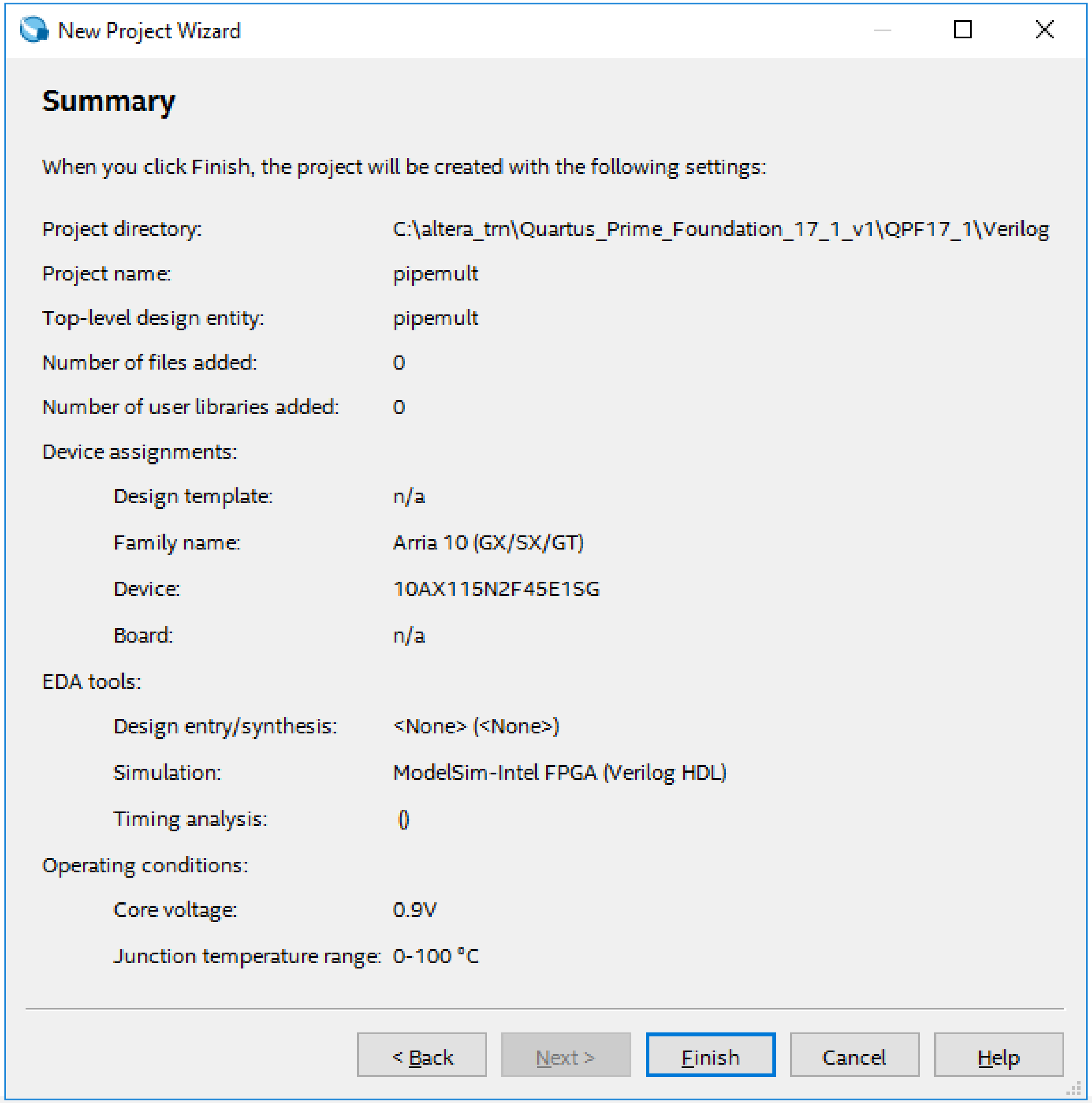
Specify the other EDA tools used with the Quartus Prime software to develop your project.

EDA tools:

Tool Type	Tool Name	Format(s)
Design Entry/Synth...	<None>	<None>
Simulation	ModelSim-Intel FPGA	Verilog HDL
Board-Level	Signal Integrity	<None>

Done!

- Review results & click **Finish** when done

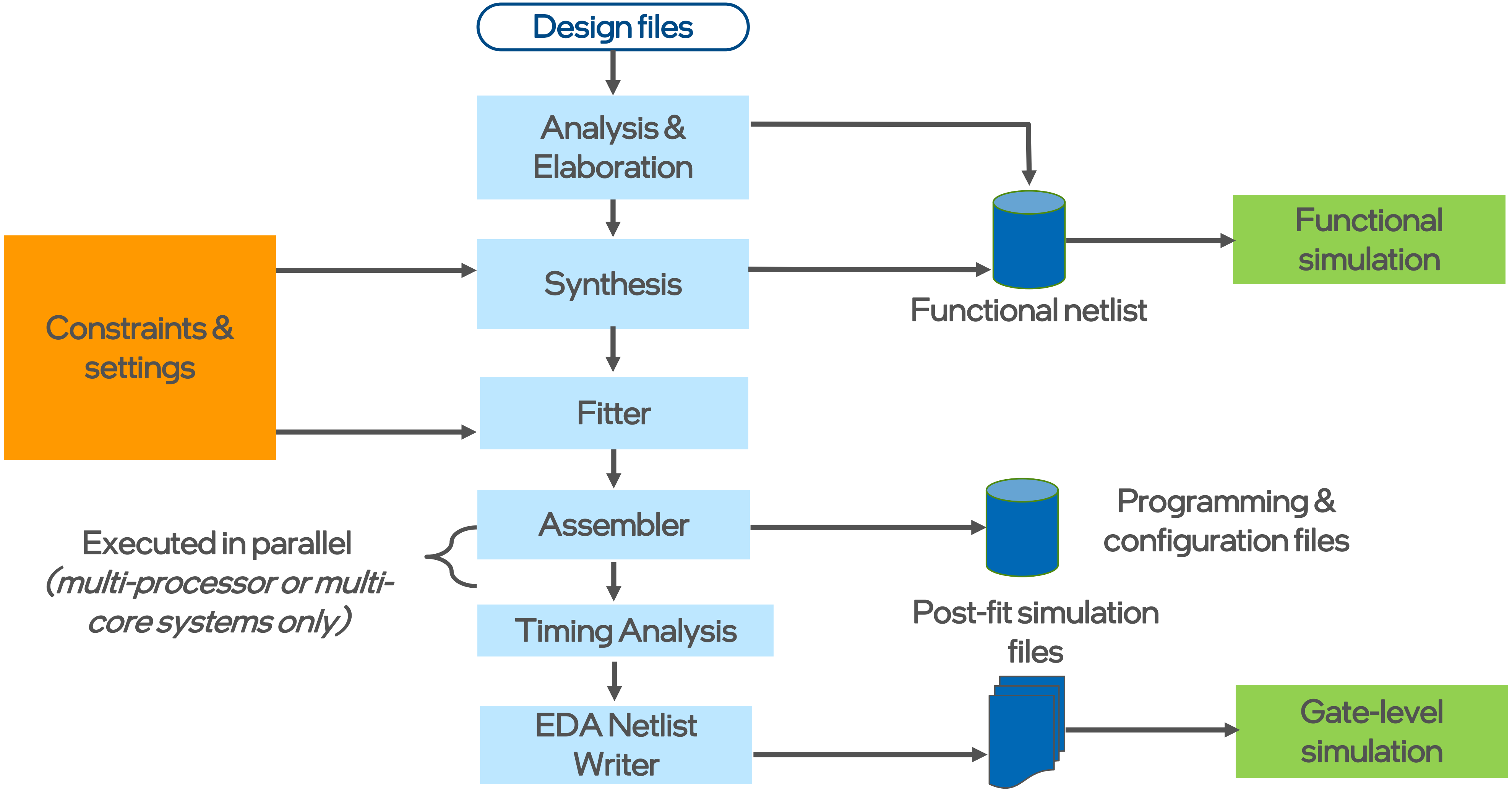


Summary

When you click Finish, the project will be created with the following settings:

Project directory:	C:\altera_trn\Quartus_Prime_Foundation_17_1_v1\QPF17_1\Verilog
Project name:	pipemult
Top-level design entity:	pipemult
Number of files added:	0
Number of user libraries added:	0
Device assignments:	
Design template:	n/a
Family name:	Arria 10 (GX/SX/GT)
Device:	10AX115N2F45E1SG
Board:	n/a
EDA tools:	
Design entry/synthesis:	<None> (<None>)
Simulation:	ModelSim-Intel FPGA (Verilog HDL)
Timing analysis:	()
Operating conditions:	
Core voltage:	0.9V
Junction temperature range:	0-100 °C

Intel® Quartus® Prime Design Software Full Compilation Flow

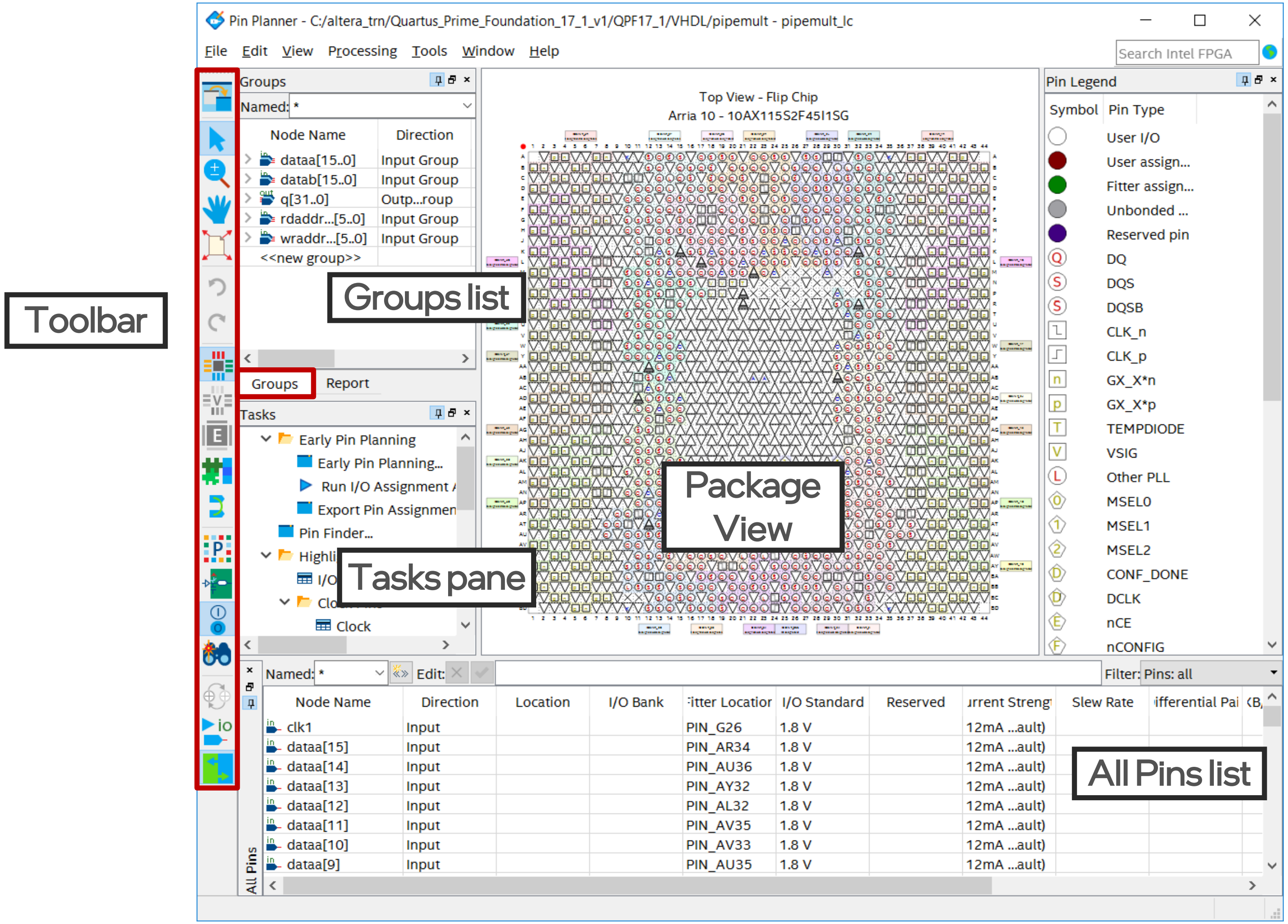


Pin Planner

- Interactive graphical tool for assigning pins
  - Drag & drop pin assignments
  - Set pin I/O standards
  - Reserve future I/O locations
- Default window panes
  - Package View
  - All Pins list
  - Groups list
  - Tasks window
  - Report window

Assignments menu → **Pin Planner**, toolbar, or **Tasks** window

Pin Planner Window



Pin Planner - C:/altera_trn/Quartus_Prime_Foundation_17_1_v1/QPF17_1/VHDL/pipemult - pipemult_lc

File Edit View Processing Tools Window Help

Search Intel FPGA

Groups

Named: *

Node Name	Direction
dataa[15..0]	Input Group
datab[15..0]	Input Group
q[31..0]	Outp...roup
rdaddr...[5..0]	Input Group
wraddr...[5..0]	Input Group
<<new group>>	

Groups Report

Tasks

- Early Pin Planning
  - Early Pin Planning...
  - Run I/O Assignment A
  - Export Pin Assignmen
- Pin Finder...
- Highli
  - I/O
  - Clo
    - Clock

Top View - Flip Chip
Arria 10 - 10AX115S2F45I1SG

Pin Legend

Symbol	Pin Type
○	User I/O
●	User assign...
●	Fitter assign...
●	Unbonded ...
●	Reserved pin
Q	DQ
S	DQS
S	DQSB
⅂	CLK_n
⅃	CLK_p
n	GX_X*n
p	GX_X*p
T	TEMPDIODE
V	VSIG
L	Other PLL
0	MSEL0
1	MSEL1
2	MSEL2
D	CONF_DONE
D	DCLK
E	nCE
F	nCONFIG

Named: * Edit: Filter: Pins: all

Node Name	Direction	Location	I/O Bank	itter Locatior	I/O Standard	Reserved	urrent Streng	Slew Rate	ifferential Pai	(B,
clk1	Input			PIN_G26	1.8 V		12mA ...ault)			
dataa[15]	Input			PIN_AR34	1.8 V		12mA ...ault)			
dataa[14]	Input			PIN_AU36	1.8 V		12mA ...ault)			
dataa[13]	Input			PIN_AY32	1.8 V		12mA ...ault)			
dataa[12]	Input			PIN_AL32	1.8 V		12mA ...ault)			
dataa[11]	Input			PIN_AV35	1.8 V		12mA ...ault)			
dataa[10]	Input			PIN_AV33	1.8 V		12mA ...ault)			
dataa[9]	Input			PIN_AU35	1.8 V		12mA ...ault)			

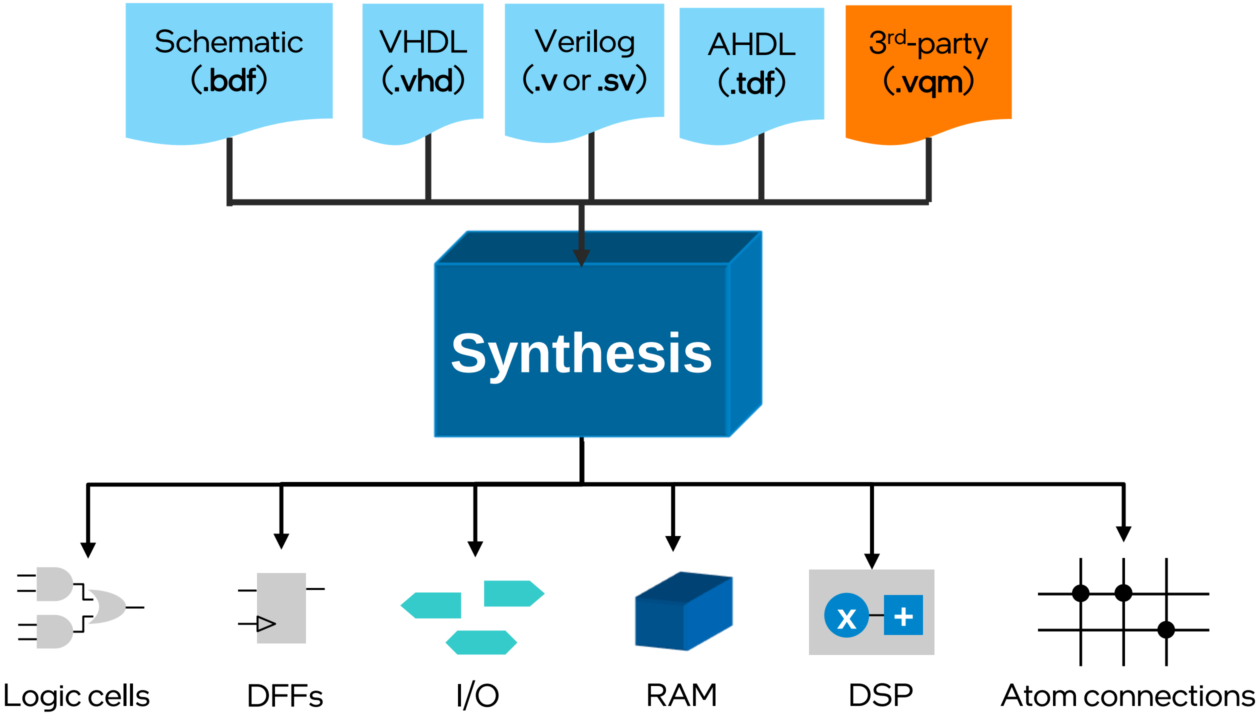
All Pins

Section Three

Software Compilation

What is Synthesis?

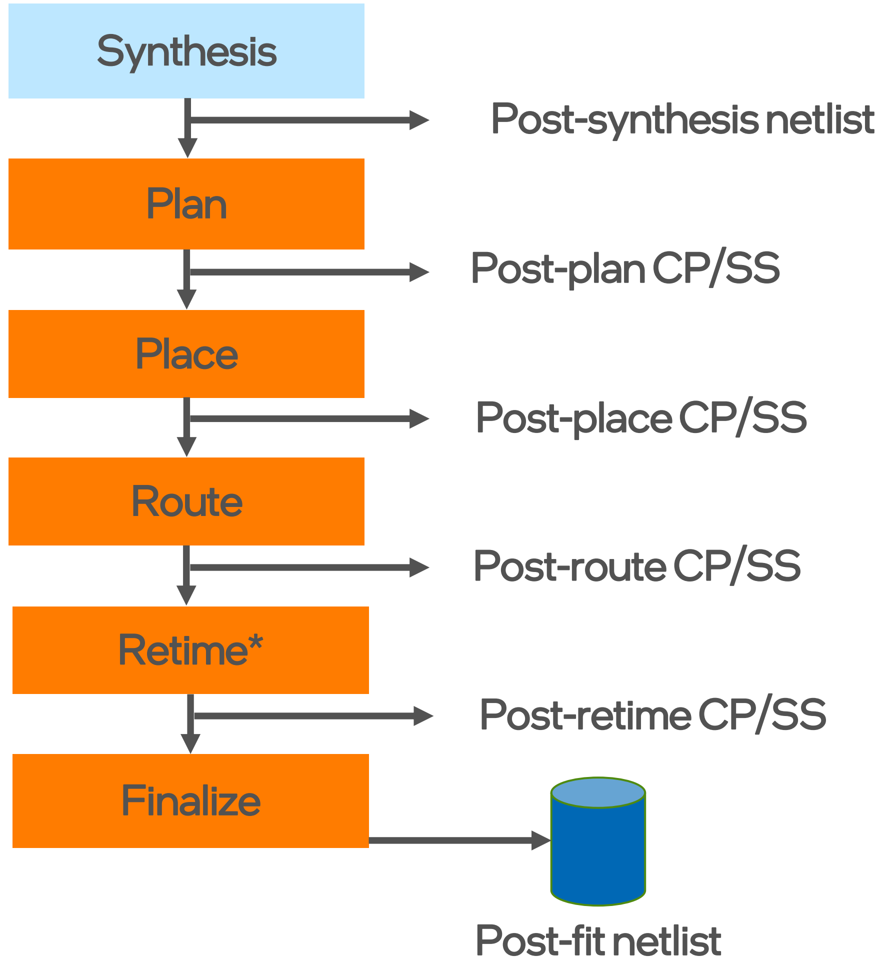
Translates HDL source files into an **atom netlist**



What is the Fitter? (1)

Place & route engine for finding a solution in a “reasonable” amount of time

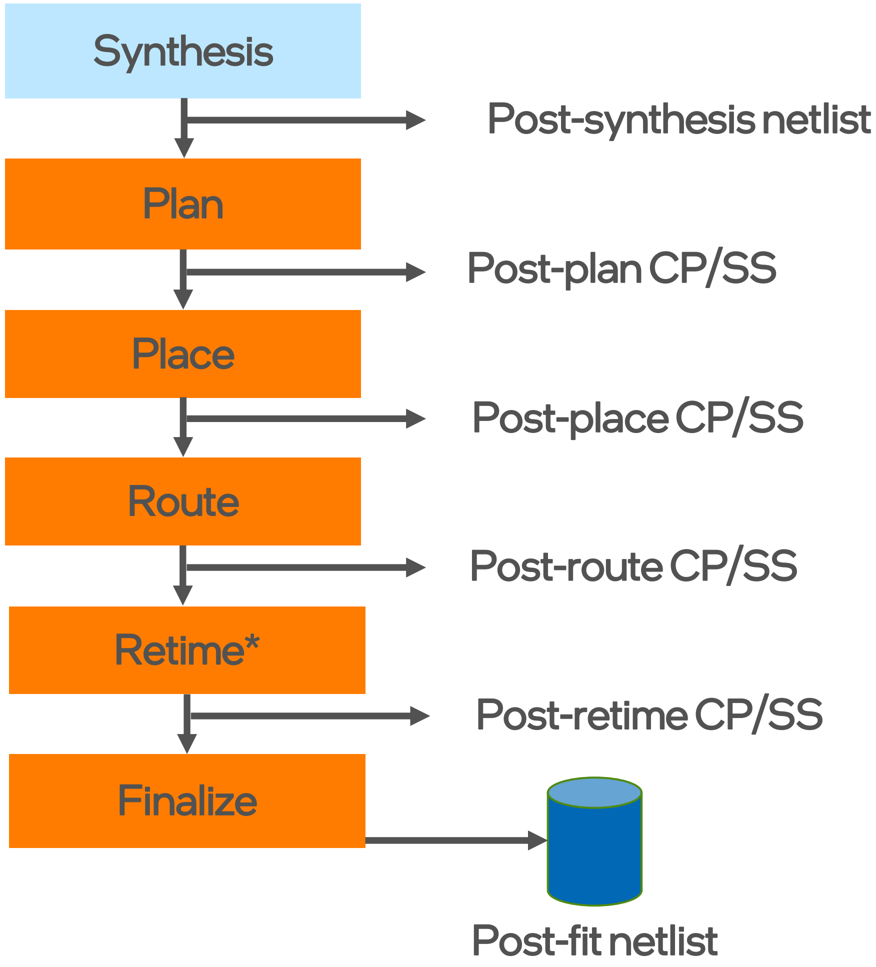
- Consists of multiple stages
  - End of each stage referred to as a **checkpoint** (CP)
- Prior stages must be complete before running later stages



* Retime for Intel® Hyperflex™ FPGA architectures

What is the Fitter? (2)

- Output of each stage is a database referred to as a **snapshot (SS)**
- Useful for incremental optimization
  - Optimize design at each checkpoint
  - Save compile time by only running affected stage(s)



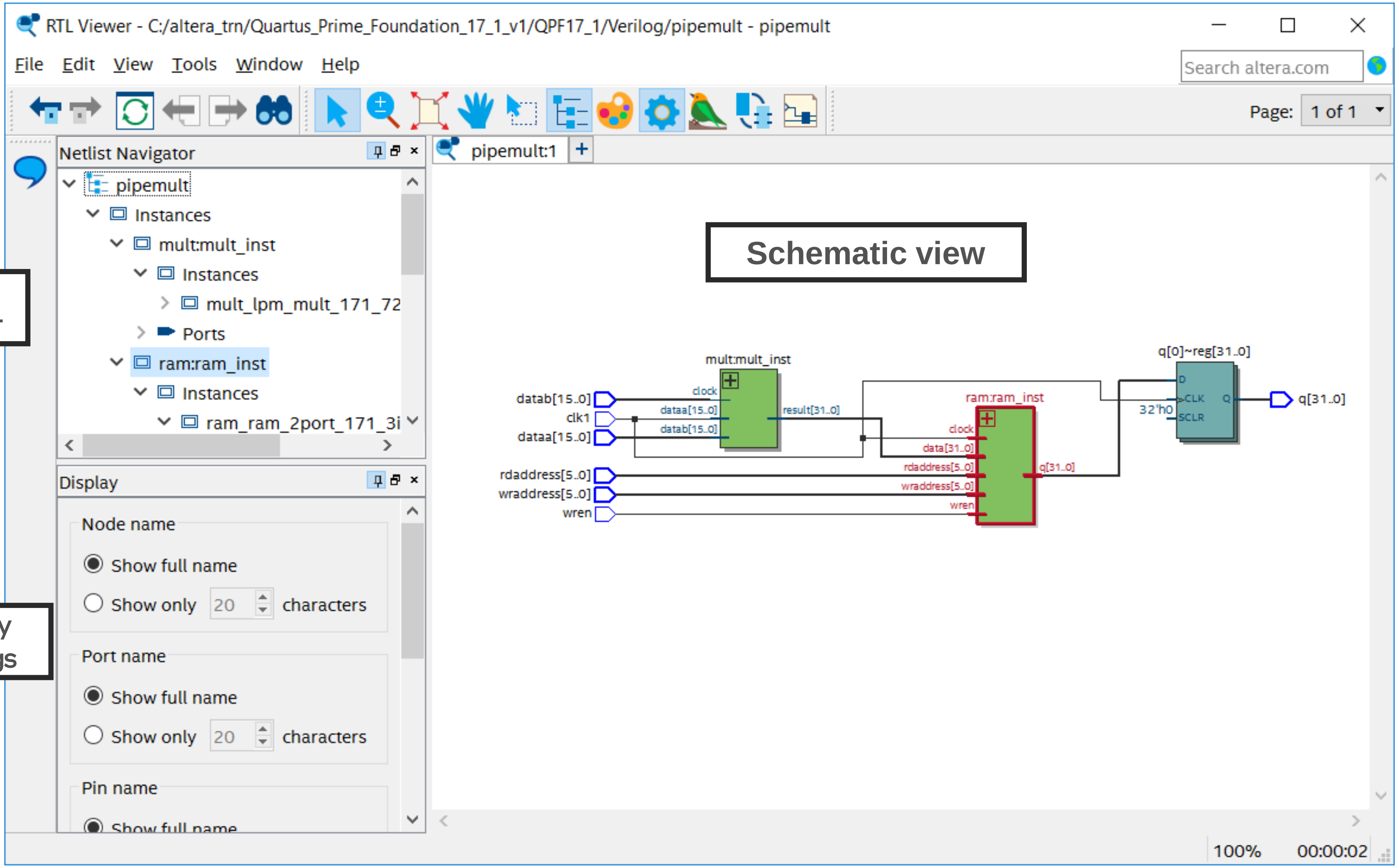
* Retime for Intel® Hyperflex™ FPGA architectures

Netlist Viewers

Tools menu → **Netlist Viewers**

Netlist Navigator

Display Settings



RTL Viewer - C:/altera_trn/Quartus_Prime_Foundation_17_1_v1/QPF17_1/Verilog/pipemult - pipemult

File Edit View Tools Window Help

Page: 1 of 1

Netlist Navigator

- pipemult
  - Instances
    - mult:mult_inst
      - Instances
        - mult_lpm_mult_171_72
      - Ports
    - ram:ram_inst
      - Instances
        - ram_ram_2port_171_3i

Display

Node name

● Show full name

○ Show only 20 characters

Port name

● Show full name

○ Show only 20 characters

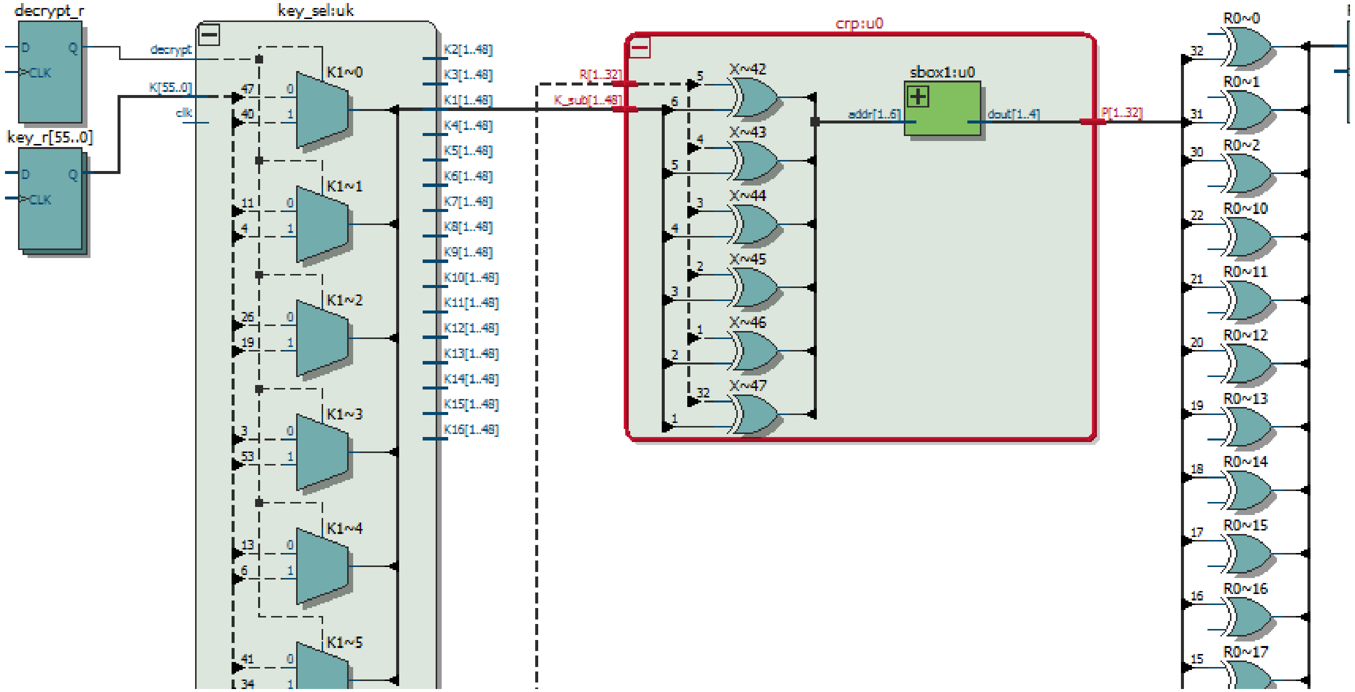
Pin name

● Show full name

Note: Must perform elaboration first (e.g. Analysis & Elaboration OR Analysis & Synthesis)

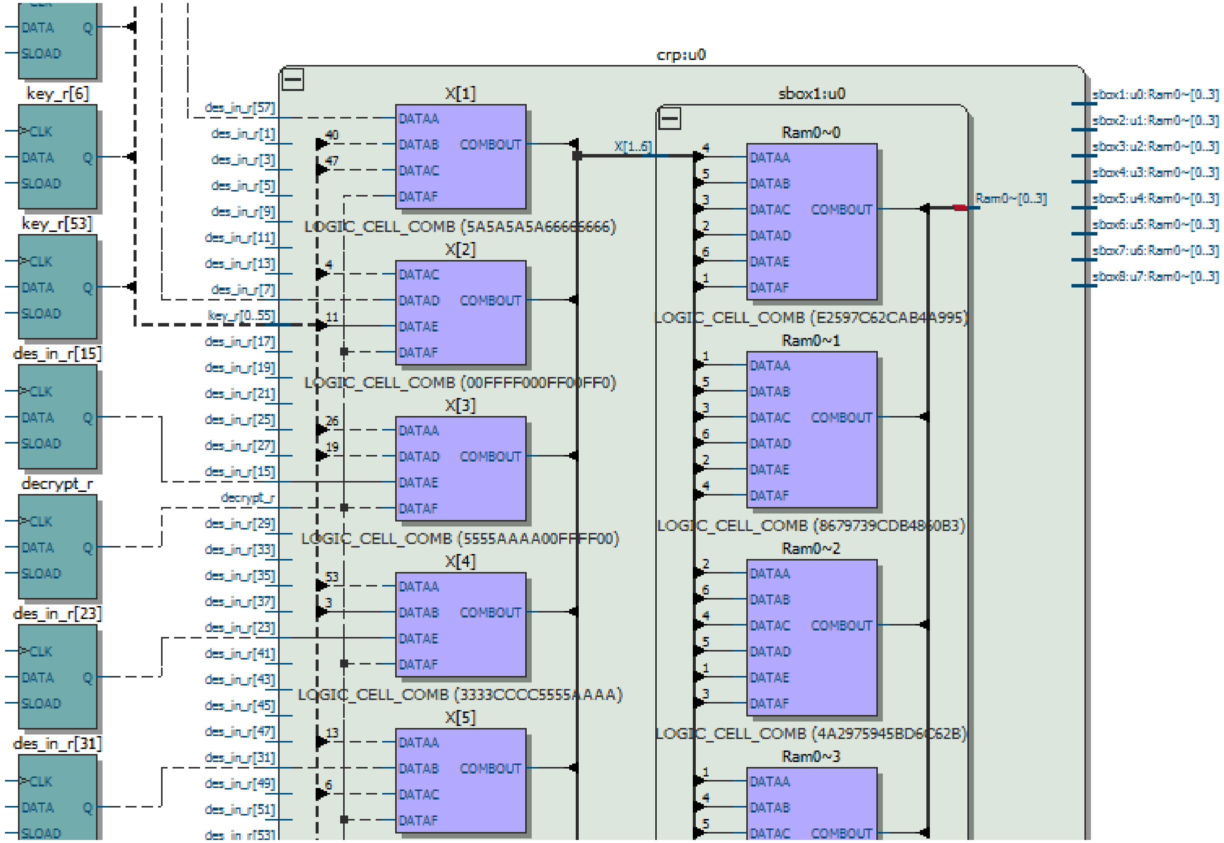
Schematic View (RTL Viewer)

- Represents design using logic blocks & nets
  - I/O pins
  - Registers
  - Muxes
  - Gates (AND, OR, etc.)
  - Operators (adders, multipliers

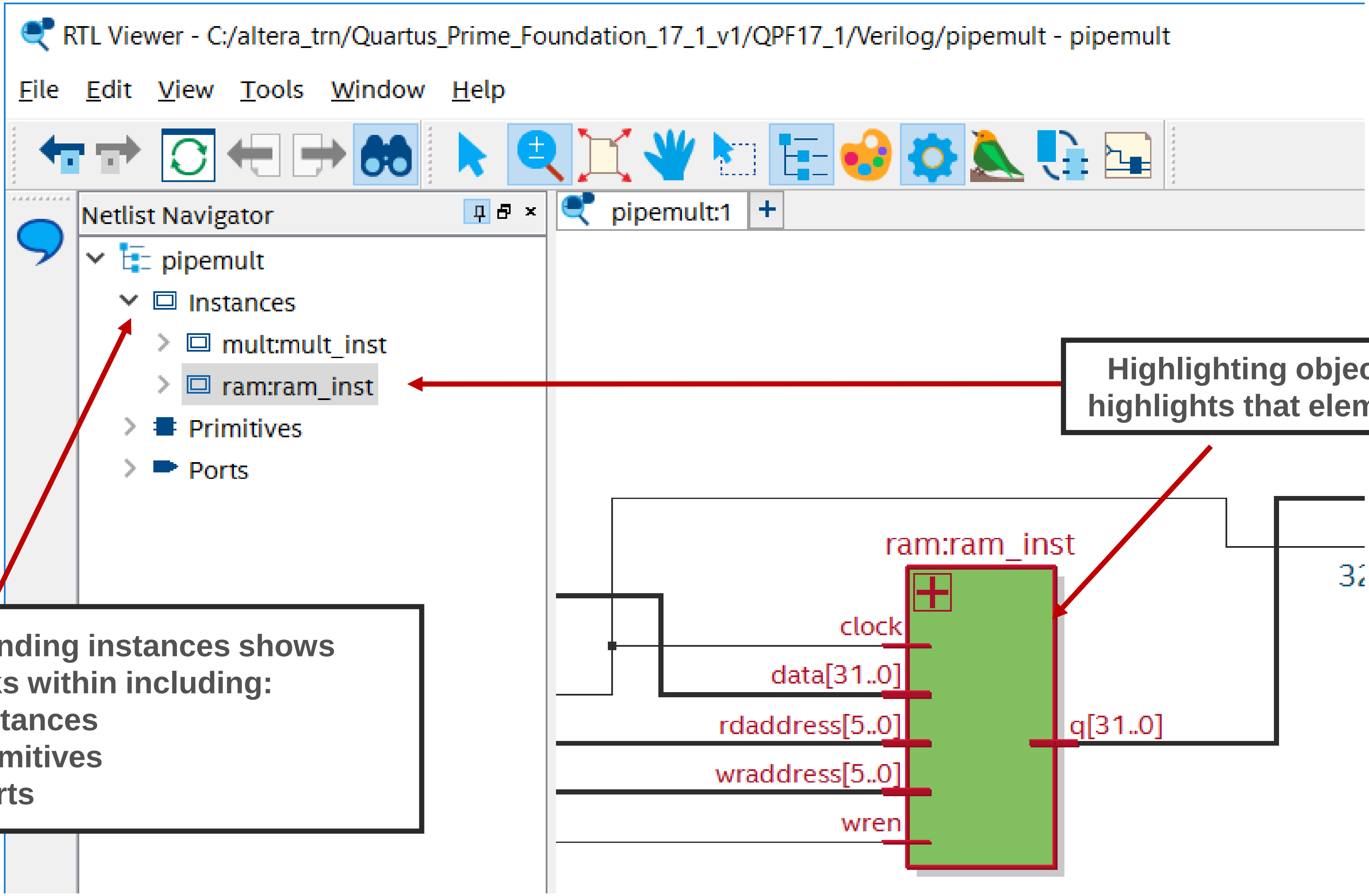


Schematic View (Technology Map Viewer)

- Represents design using atoms
  - I/O pins & cells
  - Logic cells (Lcells)
  - Memory blocks
  - MAC (DSP blocks)



Using the Netlist Navigator



RTL Viewer - C:/altera_trn/Quartus_Prime_Foundation_17_1_v1/QPF17_1/Verilog/pipemult - pipemult

Highlighting object in Netlist Navigator highlights that element in schematic view

Expanding instances shows blocks within including:

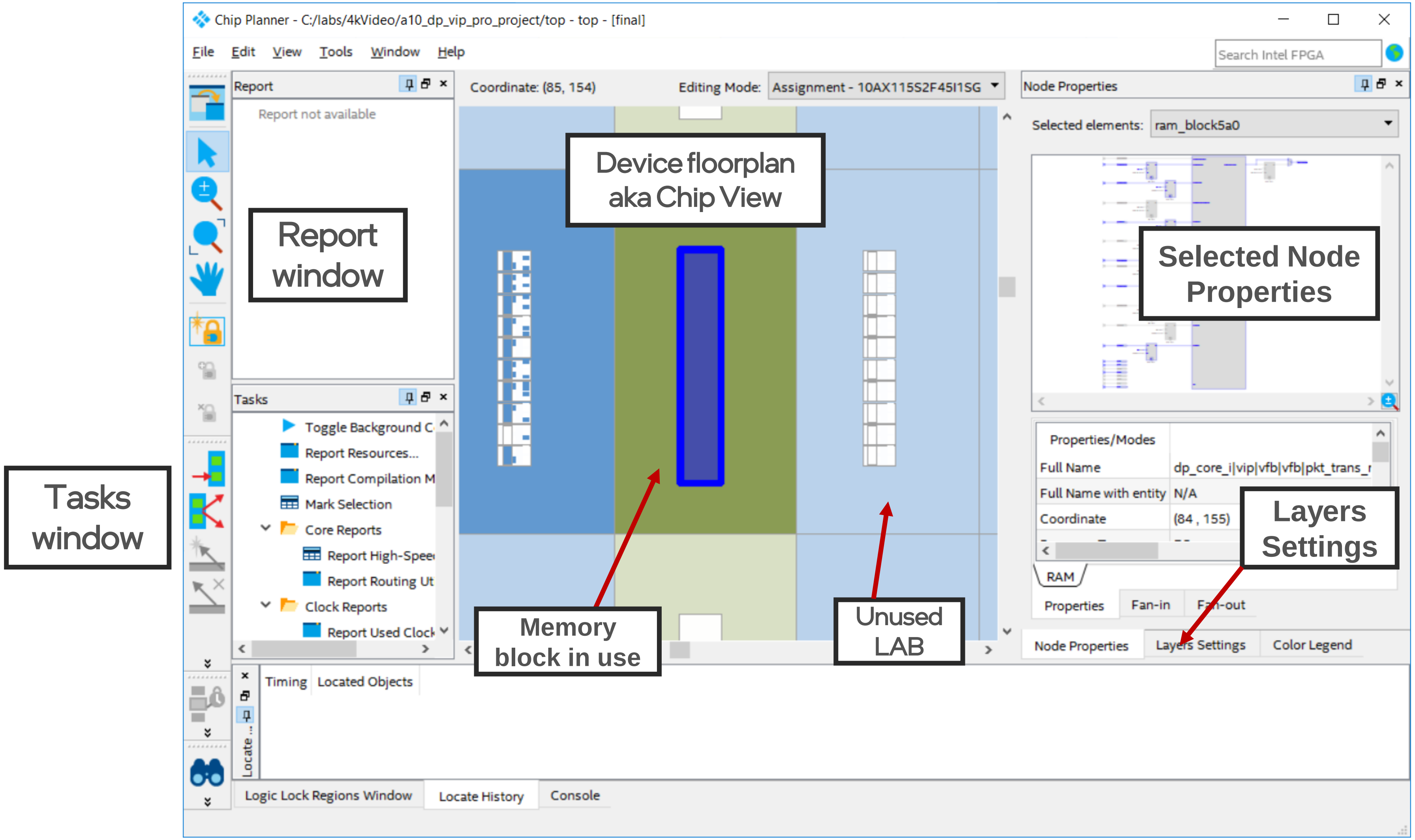
- Instances
- Primitives
- Ports

Chip Planner

- Graphical view of design resource usage in target device
- Displays
  - Graphical layout of device resources
  - Routing channels between device resources
  - Global clock regions
- Uses
  - View placement of design logic
  - View connectivity between resources used in design
  - Make placement assignments
  - Debugging placement-related issues

Chip Planner

Tools menu or toolbar



Report window

Tasks window

Device floorplan aka Chip View

Memory block in use

Unused LAB

Selected Node Properties

Layers Settings

Chip Planner - C:/labs/4kVideo/a10_dp_vip_pro_project/top - top - [final]

Coordinate: (85, 154)

Editing Mode: Assignment - 10AX115S2F45I1SG

Selected elements: ram_block5a0

Properties/Modes	
Full Name	dp_core_i|vip|vfb|vfb|pkt_trans_r
Full Name with entity	N/A
Coordinate	(84 , 155)

Why Did I Receive Undesired Results?

- Incorrect coding
- Non-recommended coding style
- Suboptimal synthesis & fitting settings/constraints
- Use tools described to find problems and help fix them
- For more details on optimizing designs based on undesired results, please attend the course Intel® FPGA Timing Closure
- To take advantage of the Intel Arria® 10 and Cyclone® 10 architecture, register for the free, online class Creating High-Performance Designs in 20 nm Intel FPGAs

Compilation Summary

- Compilation includes synthesis & fitting
- Compilation Report contains detailed information on compilation results
- Use Intel® Quartus® Prime Design Software tools to understand how design was processed
  - RTL Viewer
  - Technology Map Viewers
  - Chip Planner

Section Three-F

A Taste of Programming

Programming Files

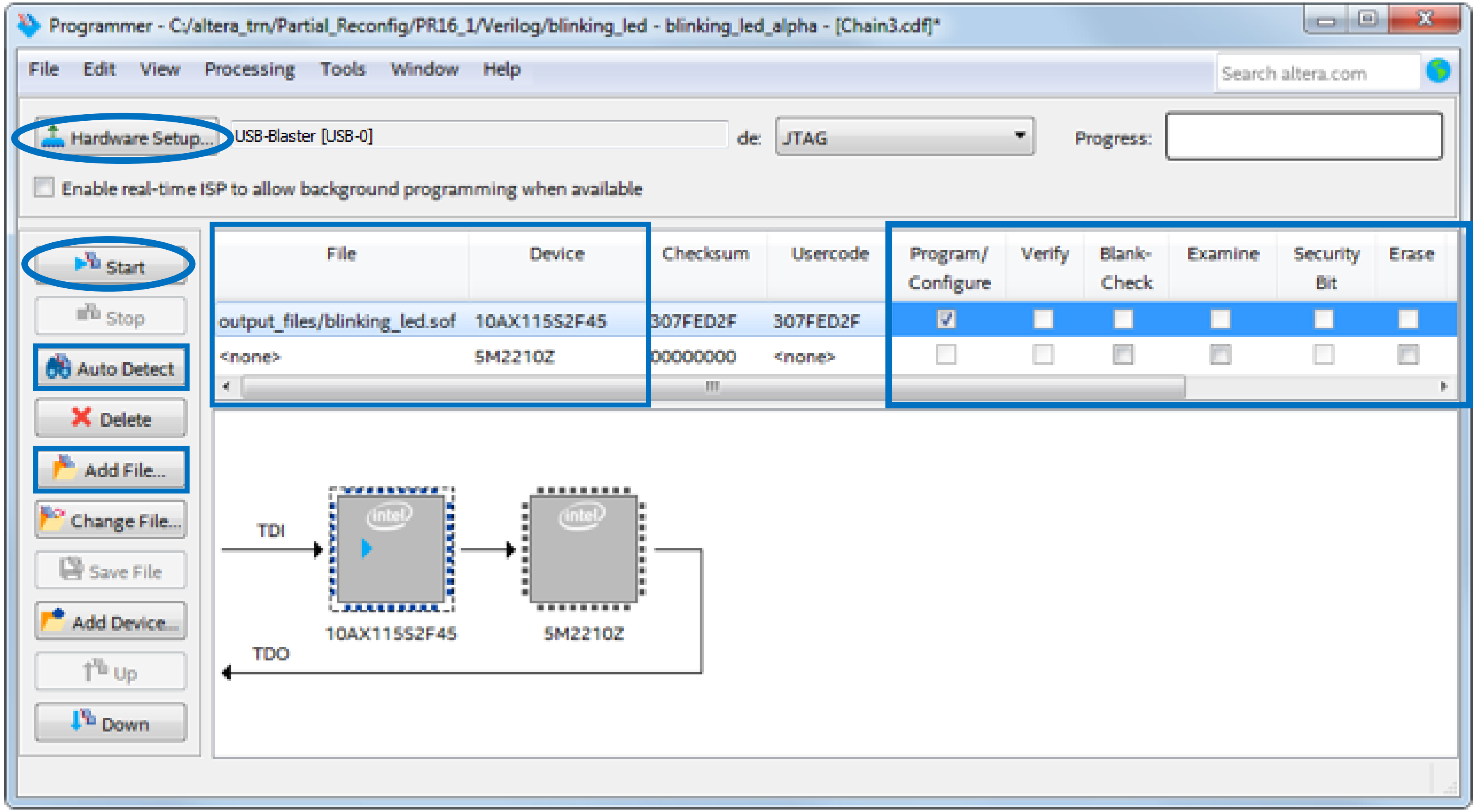
- **.sof** (SRAM Object File)
  - Used to configure FPGAs directly from Intel® Quartus® Prime Design Software through download cable
  - Always generated by default during a full compilation by the Assembler

Programming File Conversion

- The assembler, by default, automatically generates **.sof** files for any FPGA
  - May generate additional single-device programming file types from Device settings dialog box (**Device and Pin Options → Programming Files**)
- **.sof** files can only be used by the programmer to directly configure FPGA
- Other configuration solutions may require converting **.sof** file(s) into other file formats

The Programmer

Tools menu → **Programmer**



Programmer - C:/altera_trn/Partial_Reconfig/PR16_1/Verilog/blinking_led - blinking_led_alpha - [Chain3.cdf]*

Hardware Setup... USB-Blaster [USB-0]

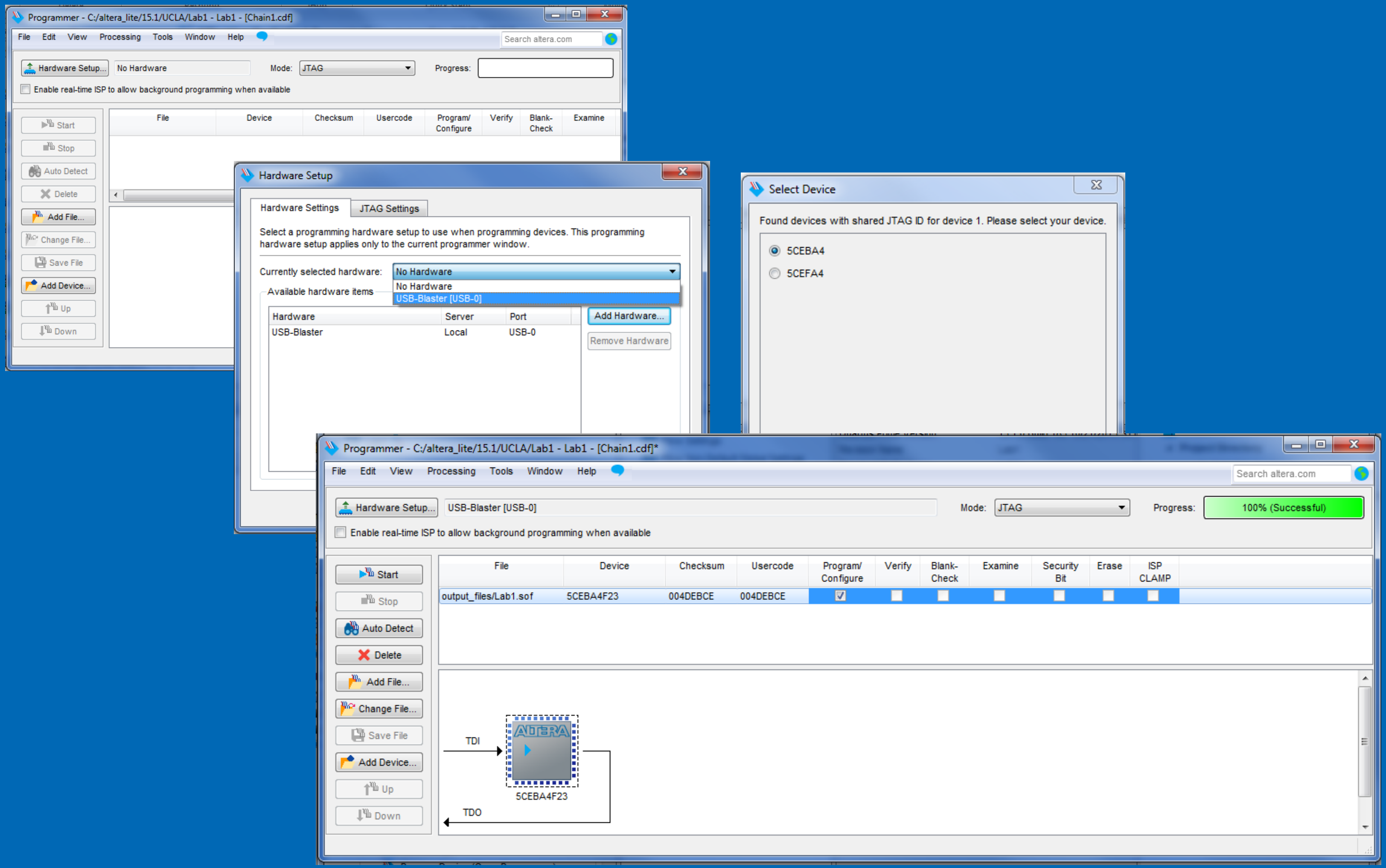
de: JTAG

Progress:

□ Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase
output_files/blinking_led.sof	10AX115S2F45	307FED2F	307FED2F	☑	□	□	□	□	□
<none>	5M2210Z	00000000	<none>	□	□	□	□	□	□

Program your FPGA



Programmer - C:/altera_lite/15.1/UCLA/Lab1 - Lab1 - [Chain1.cdf]

Hardware Setup... No Hardware — Mode: JTAG — Progress:

☐ Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine

Hardware Setup

Hardware Settings | JTAG Settings

Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.

Currently selected hardware: No Hardware

- No Hardware
- USB-Blaster [USB-0]

Available hardware items

Hardware	Server	Port
USB-Blaster	Local	USB-0

Select Device

Found devices with shared JTAG ID for device 1. Please select your device.

● 5CEBA4

○ 5CEFA4

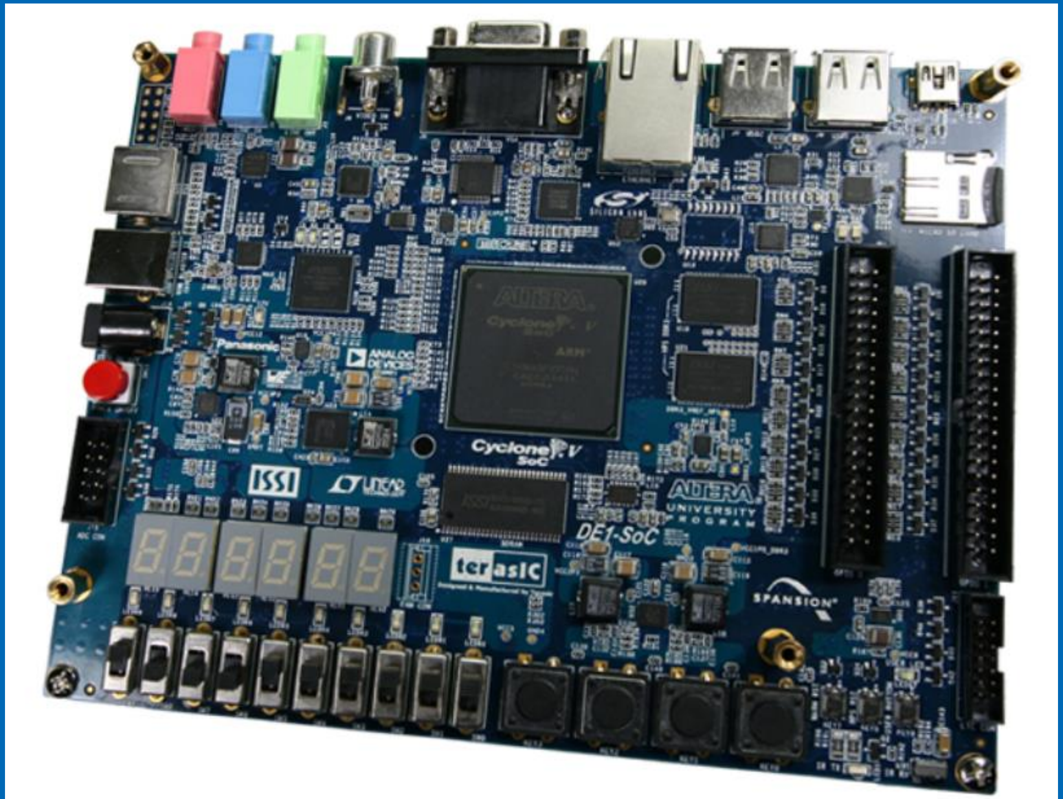
Programmer - C:/altera_lite/15.1/UCLA/Lab1 - Lab1 - [Chain1.cdf]*

Hardware Setup... USB-Blaster [USB-0] — Mode: JTAG — Progress: 100% (Successful)

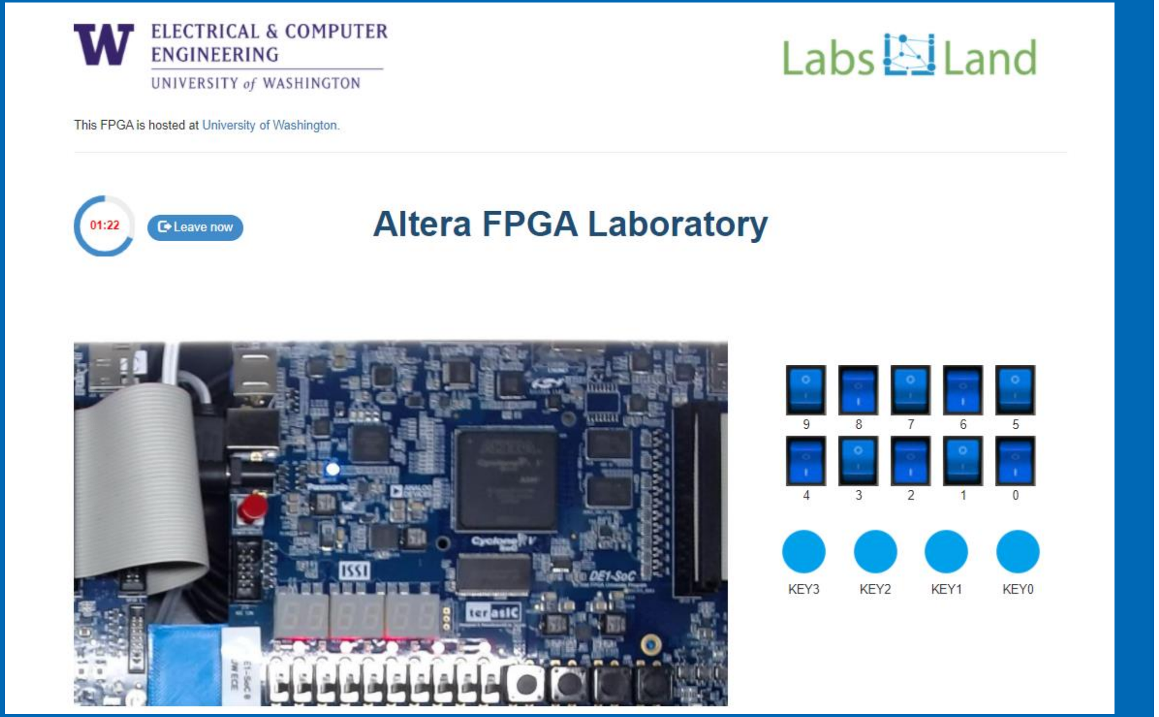
☐ Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	ISP CLAMP
output_files/Lab1.sof	5CEBA4F23	004DEBCE	004DEBCE	☑	☐	☐	☐	☐	☐	☐

Test your design



Live Hardware



This FPGA is hosted at University of Washington.

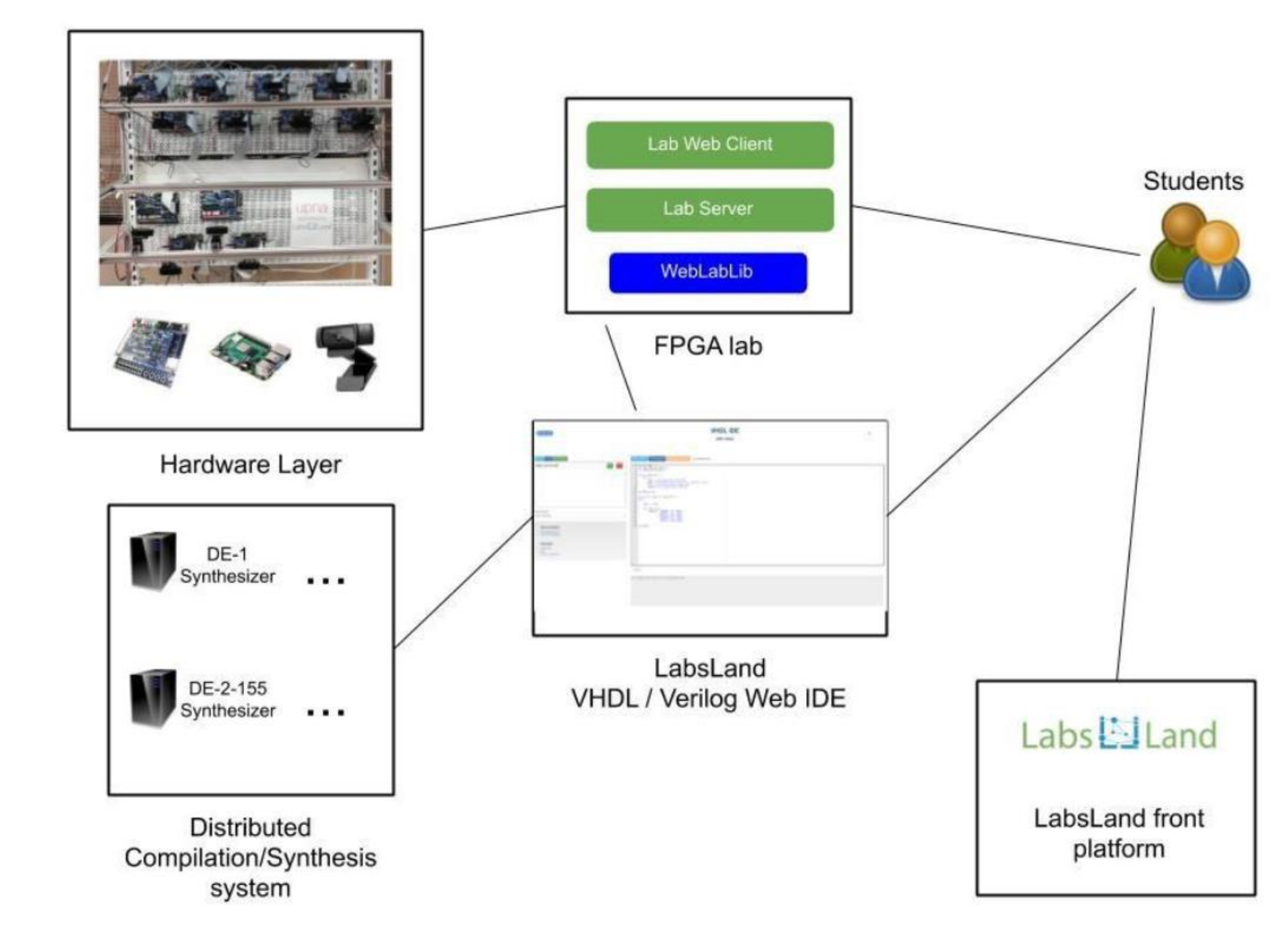
Altera FPGA Laboratory

Labsland

Labs Land

Real laboratories, on the Internet

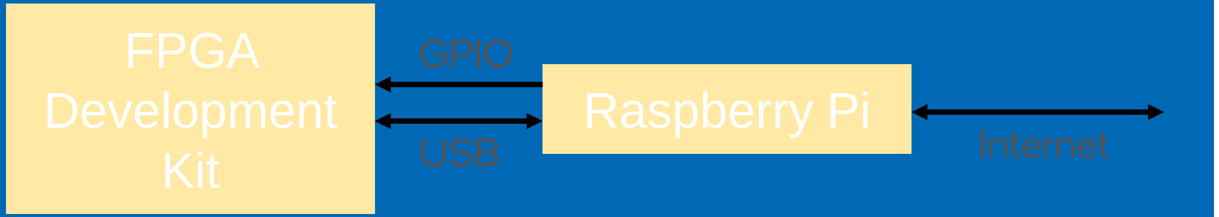
- LabsLand connects schools and universities with **real laboratories** available somewhere else on the Internet. A real laboratory can be a small Arduino powered robotrin Spain, a kinematics setup in Brazil or a radioactivity testing lab in Australia. They are real laboratories, not simulations: the laboratories are physically there, and students from these schools and universities access them.
- Fee based model to access labs (no charge today!)



Labs Land

Real laboratories, on the Internet

- Two modes:
- IDE mode is an overlay on Quartus. No software or hardware install required by the user. Compiles are on the cloud. Some limitations on IP usage and debug capabilities
- No-IDE mode requires local software installation and utilizes Labsland remote FPGA hardware installation – this is what you will use in today's lab


FPGA Development Kit

GPIO

USB

Raspberry Pi

Internet

Compilation Support Resources

- Intel® Quartus® Prime Software User Guide chapters
  - Design Compilation User Guide: Design Compilation
  - Design Optimization User Guide: Optimizing the Design Netlist
  - Design Optimization User Guide: Analyzing and Optimizing the Design Floorplan
- Training courses
  - Incremental Optimization with the Intel Quartus Prime Pro Edition Software (online)
  - Using the Intel Quartus Prime Pro Software: Chip Planner (online)
  - Intel FPGA Timing Closure (instructor-led)

intel®