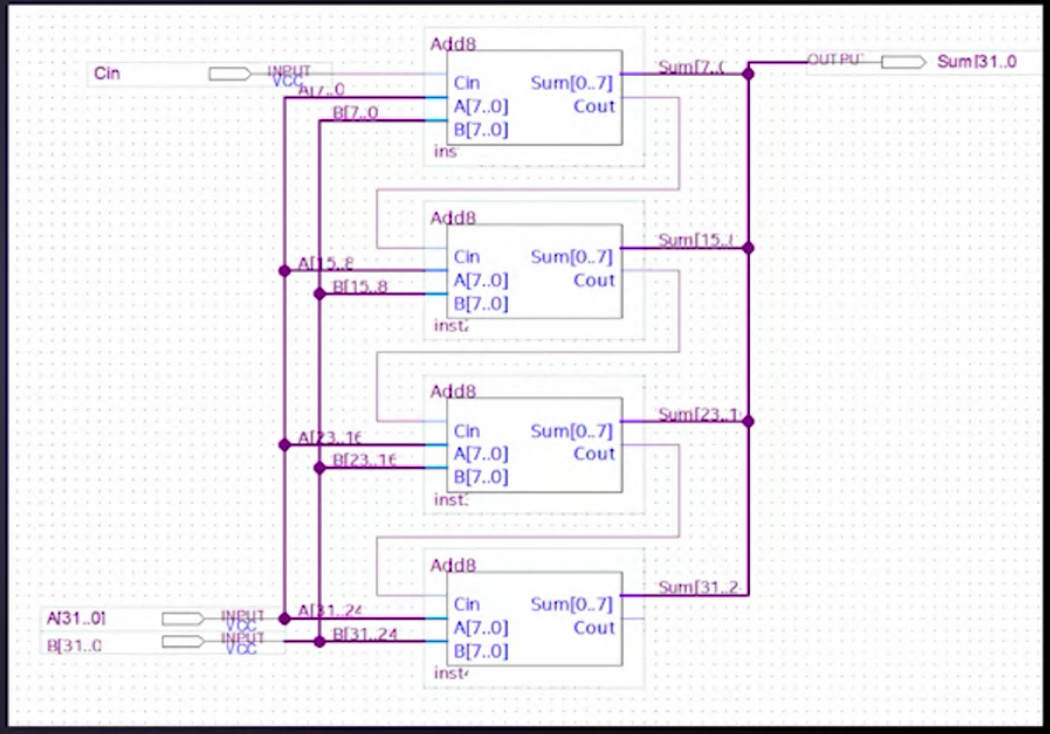
Designing with FPGAs

- Module 1: Introduced programmable logic devices and the FPGA.
- Module 2: Used Quartus Prime to work through a sample FPGA Design.
- Module 3: FPGA architectures and capabilities, adding a number of weapons to our arsenal.
- Module 4: Practice using those weapons, and extend our knowledge of FPGA design by using additional design tools.

Schematics Revisited

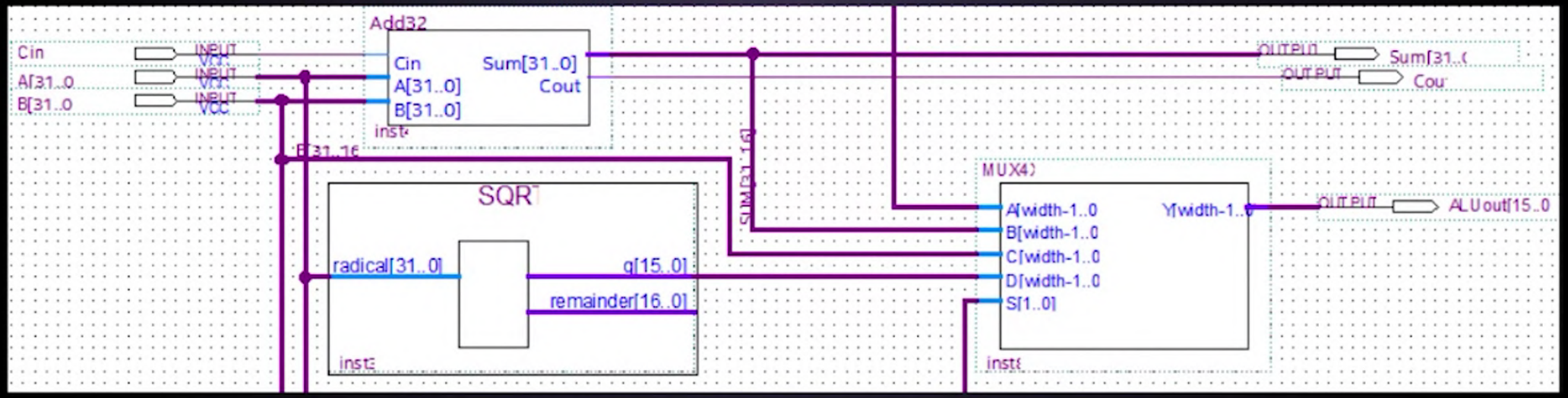
In the next video, you will further your knowledge of schematic entry for FPGAs by adding an adder circuit to the pipelined multiplier design. You will learn:

- How to add input and output ports to a schematic
- How to use the basic symbol library to create low level circuits
- How to create hierarchy in a schematic design



Improving Productivity with IP Blocks

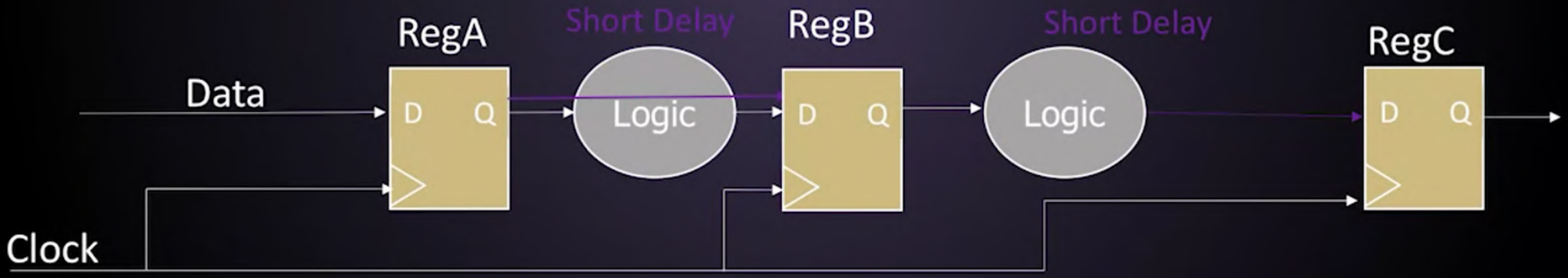
Now we will contrast your previous design entry experience with schematics to the use of IP blocks to more quickly create your design. You will add IP blocks to your design to create some functionality akin to a basic ALU.



Improving Timing with Pipelining

Large combinatorial delays between the input and output can dramatically lower the design maximum frequency. In some cases combinatorial delays can be broken into pieces by insertion of flip-flops known as retiming, or pipelining.

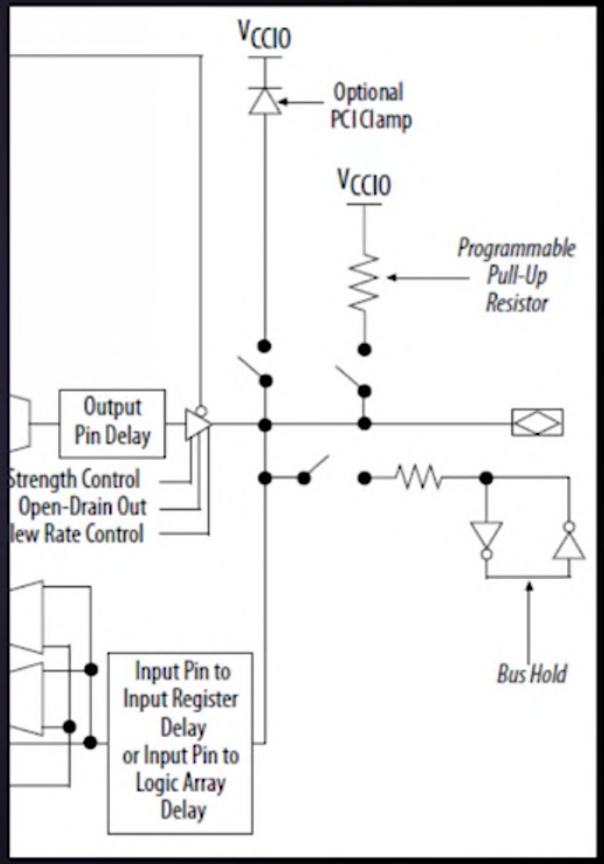
We will learn how pipelining in principle can be applied to improve timing in FPGA designs and how to use pipelining in IP blocks to improve timing.



FPGA IO: Getting In and Getting Out

FPGAs have extensive IO capabilities, including many IO standards and controllable circuit characteristics, but keeping track of the wealth of options can lead to errors. We can reduce IO pin assignment errors by the use of methods that help document and track IO usage.

Differential logic standards like LVDS and LVPECL are increasingly popular in FPGAs as they allow increased speed and better noise immunity at the same time. We will learn how to handle these interfaces.

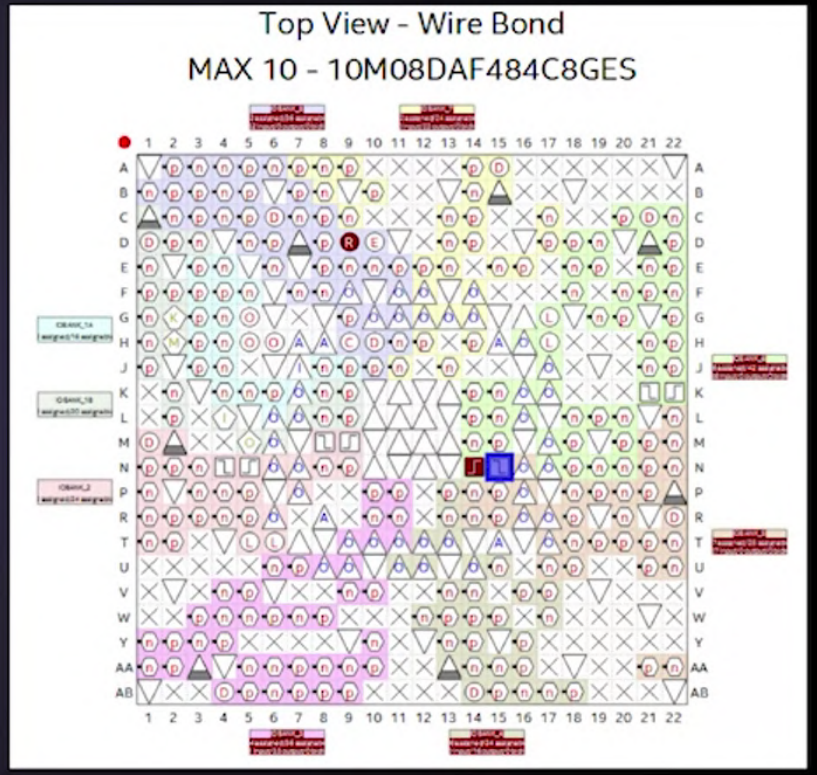


[1]

Making Pin Assignments Spot On

To do this, you will learn

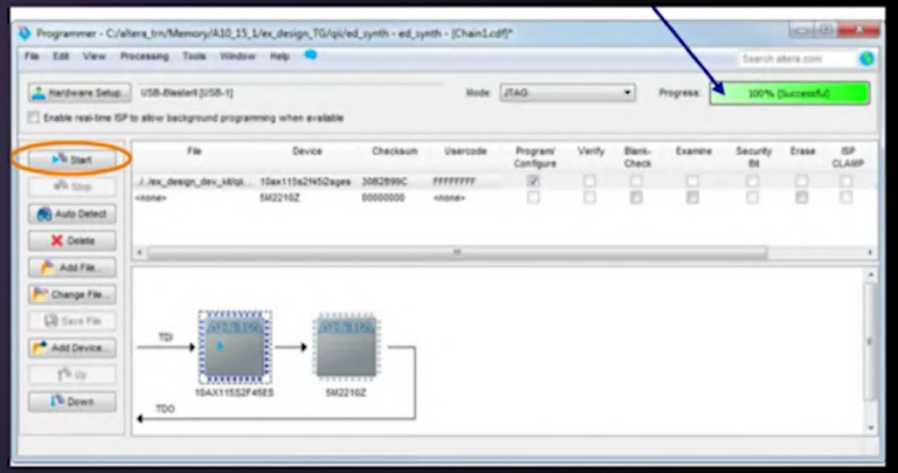
- How to do pin assignments using the Pin Planner and the Assignment Editor.
- How to meet IO standards and control IO properties using the Assignment Editor.
- How to check IO assignments using the IO Assignment Analyzer.



Programming FPGAs

The Quartus Prime software contains an assembler, which creates the programming file, and a programmer, which downloads the file using a download cable like USB Blaster II, Byte Blaster II, or Ethernet Blaster II. You will learn

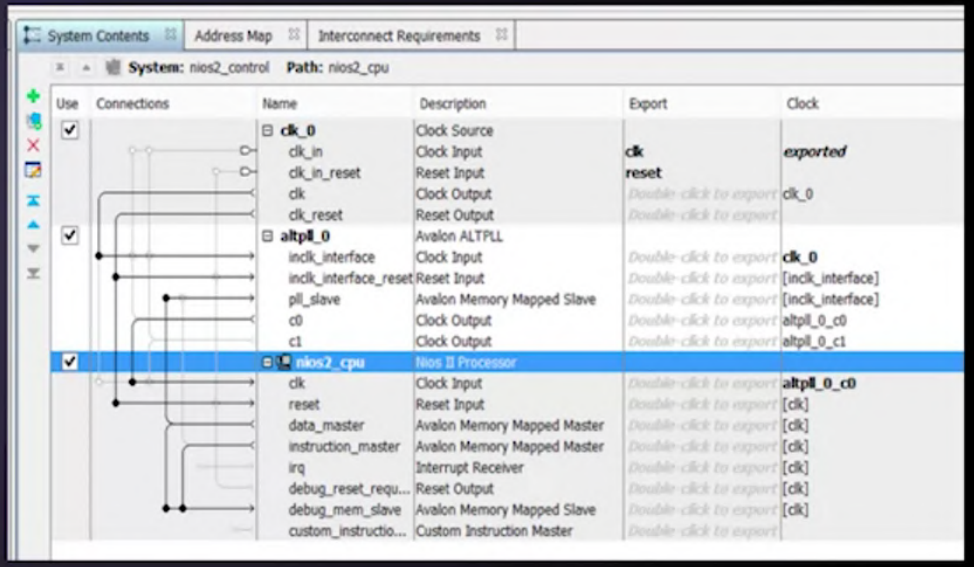
- How to create the right type of programming file for a given situation.
- How to setup your hardware chain for programming and which programming mode to use.
- How to download a program into the target device.



Becoming one with Q: Qsys System Design

Your training continues with an introduction to Qsys, a system design tool. You will learn:

- How to create a Qsys system.
- How to build a NIOS II soft-core processor.
- How to add memory and peripherals to a NIOS II Qsys design.
- How to build a bridge between the processor and FPGA fabric.
- How to bring the Qsys design into the top level to be compiled.



Videos in this module

1. FPGA Design Expertise
2. Advanced Schematic Entry for FPGA Design – Drawing and Hierarchy
3. Improving Productivity with IP Blocks
4. Improving Timing with Pipelining
5. FPGA IO: Getting In and Getting Out
6. Pin Assignments: Making them Spot On!
7. Programming the FPGA
8. Becoming one with Q: Qsys System Design
9. Becoming one with Q part II: Qsys System Design Finishing Touches

Schematic Design Entry in FPGAs

Objectives

In this presentation, you will further your knowledge of schematic entry for FPGAs by adding an adder circuit to the pipelined multiplier design. You will learn

- How to add input and output ports to a schematic
- How to use the basic symbol library to create low level circuits
- How to create hierarchy in a schematic design

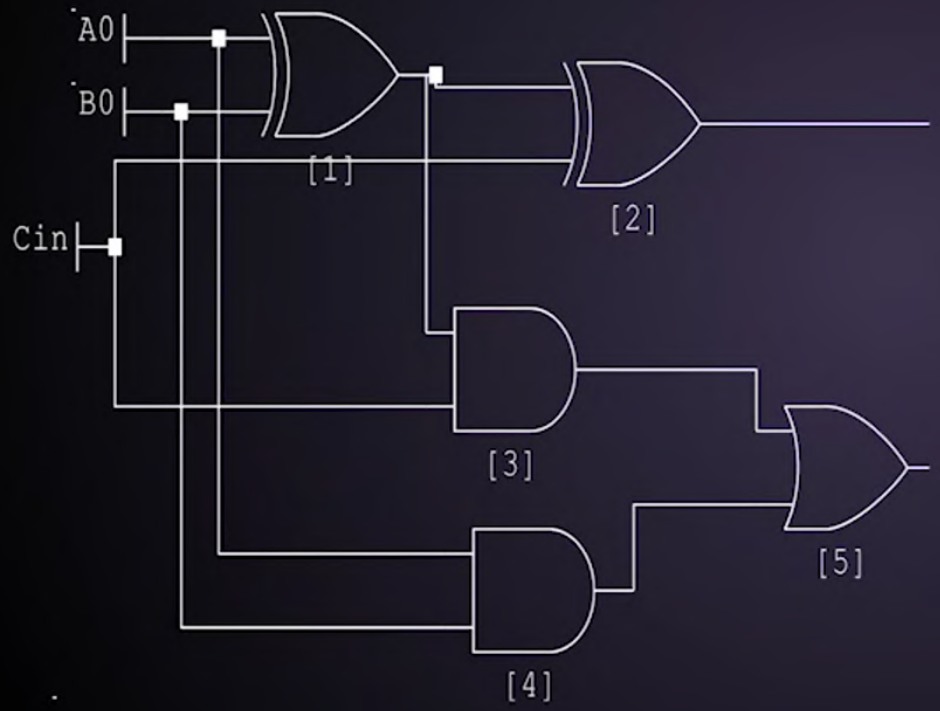
Agenda for this Video

1. Create a one-bit adder circuit,
2. Replicate it to make an 8-bit adder.
3. Replicate 8-bit adders to make a 32-bit adder.
4. Add the adder to the pipemult project.
5. Add output ports to the project.
6. View the results of the design in the RTL Viewer.
7. Compile the new design.

Preliminaries

1. Start Quartus Prime
2. Open the pipemult project we last worked on in Module 2
3. Download additional files:
   Add8.bsf, Add8.bdf, Add32.bsf, and Add32.bdf

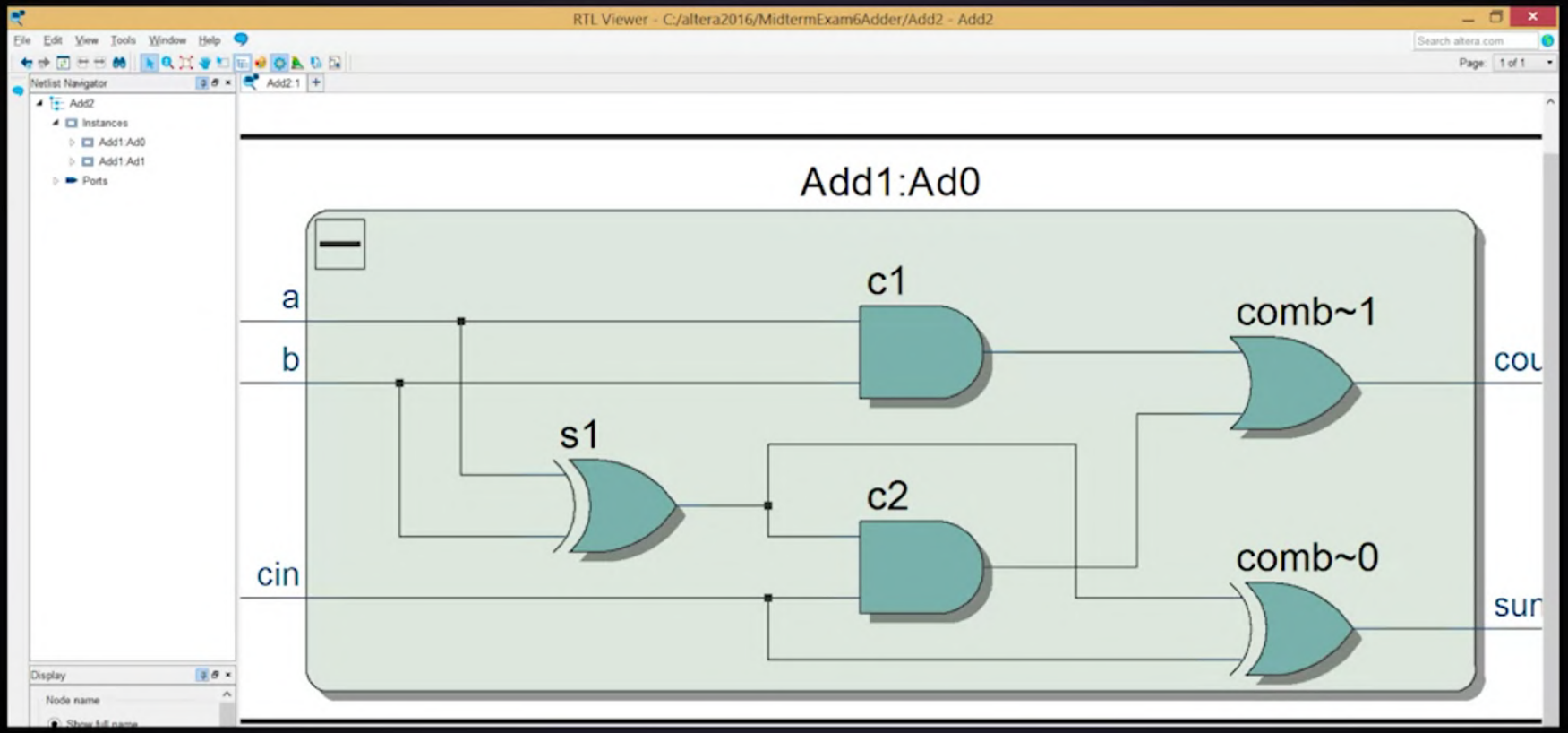
Full 1-bit Adder in Gates



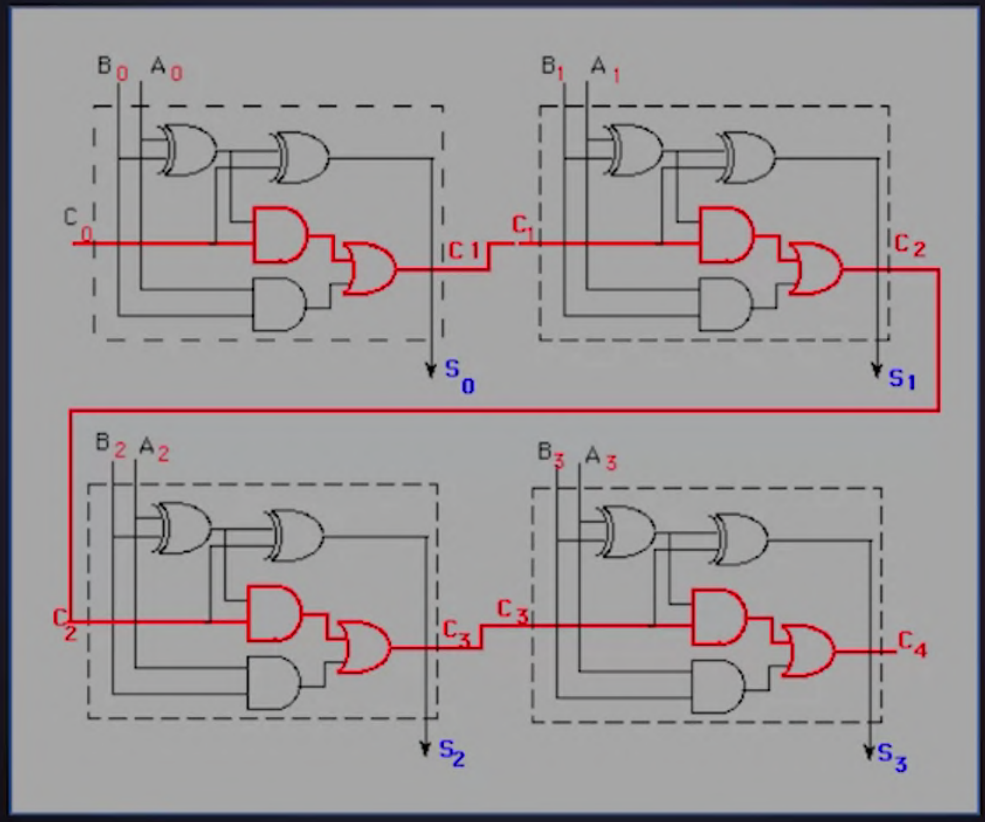
Truth Table

A0	B0	Cin	So	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

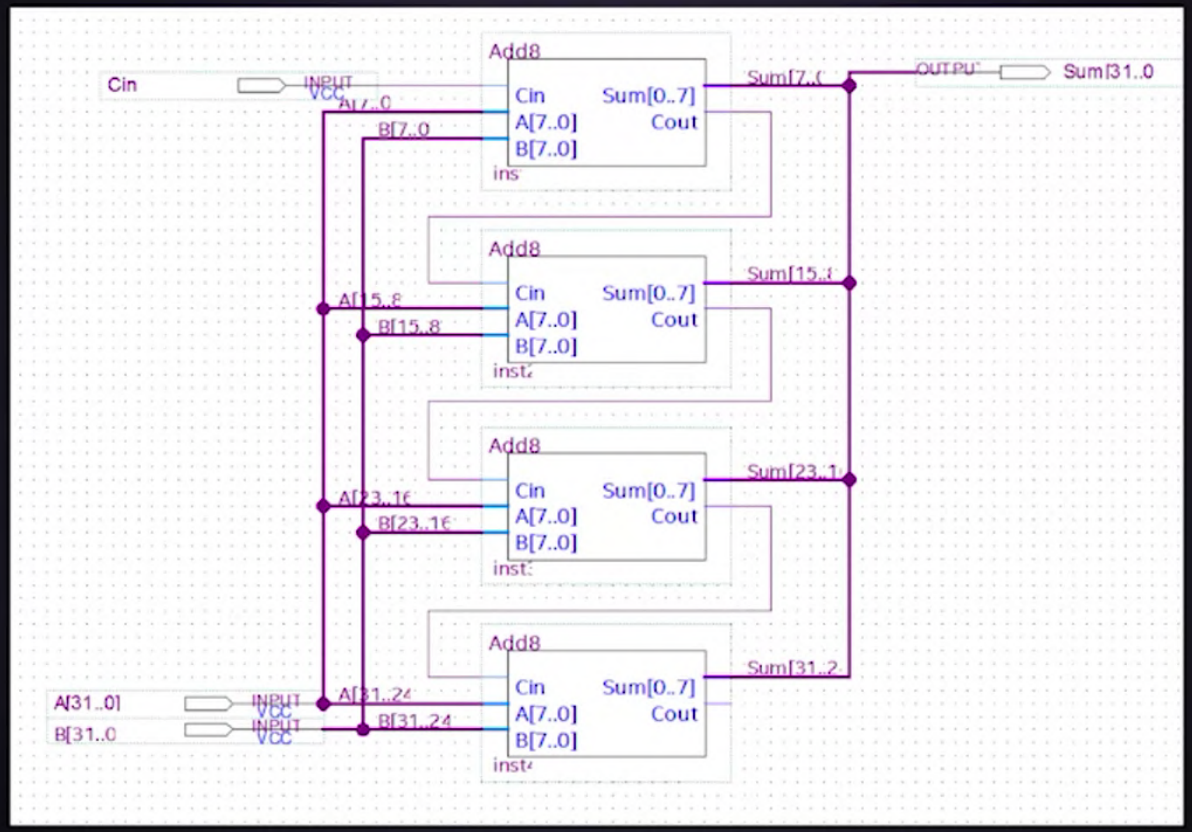
Full 1-bit Adder implemented in FGPA , RTL View.



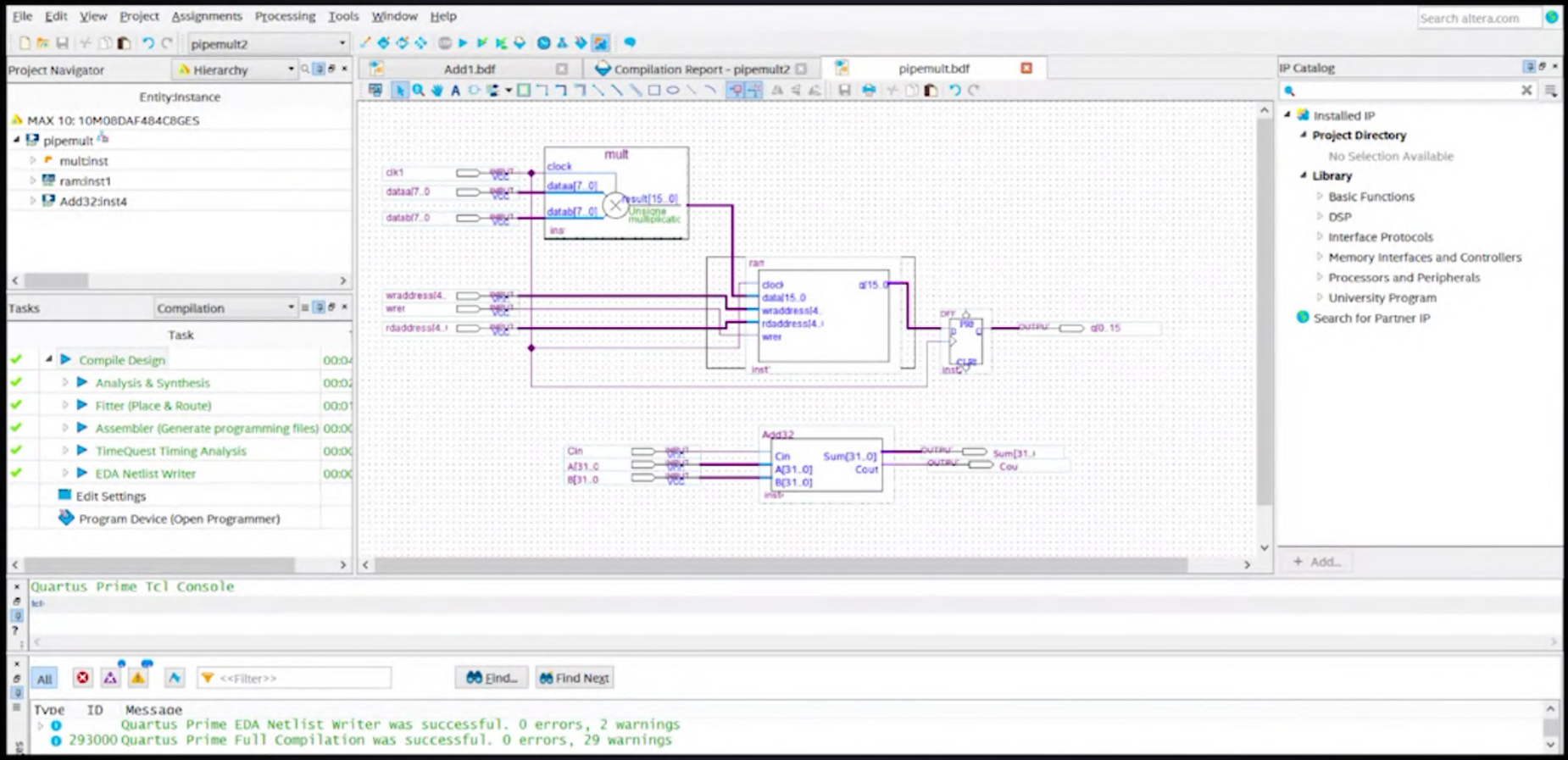
Full 4-bit Adder made of 4 1-bit adders
Note use of component hierarchy



Full 32-bit Adder made of 4 8-bit adders
Note use of component hierarchy



The Completed Circuit



Schematic Entry in FPGAs Summary

- Schematic entry can be a useful design entry method, allowing investigation of low level gate diagrams as well as larger blocks.
- The use of hierarchy can simplify designs, and make the functionality easier to understand.
- Implementation of the logic may be different from the RTL model as it is applied to the underlying FPGA technology to improve performance.

In this video, you have learned

- How to add input and output ports to a schematic.
- How to use the basic symbol library to create low level circuits.
- How to create hierarchy in a schematic design.
- How to verify logic design by use of the technology viewers

Block Diagram Design with IP blocks

Objectives

In this presentation, you will contrast your previous design entry experience with schematics to use of IP to more quickly create your design. You will add IP blocks to your design to create some functionality akin to a basic ALU.

You will learn

- How to add and configure library IP blocks.
- How to make your own IP blocks from HDL code.
- How to concatenate and split buses in a schematic design.

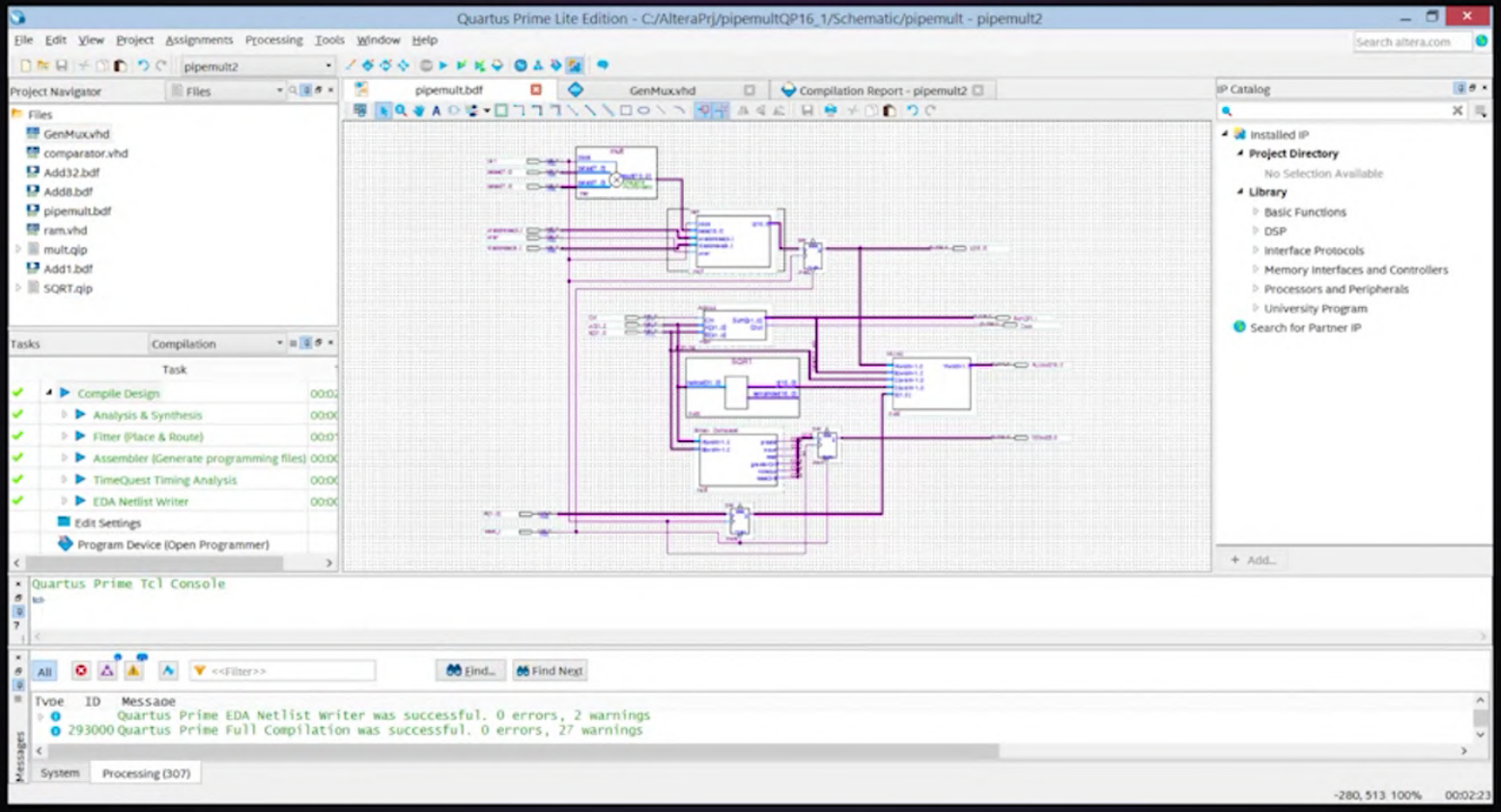
Agenda for this Video

1. Finish our proto ALU by adding
    a comparator,
    a square root function,
    and a Mux driven by an instruction code.
2. View the results of the design in the RTL Viewer
3. Compile the new design
4. Update TimeQuest timing analysis

Preliminaries

1. Start Quartus Prime
2. Open the pipemult project we last worked on in Module 4 Video 2.
3. Download additional files into the project directory:
   GenMux.vhd, and comparator.vhd.

The Completed Circuit



Design with IP Blocks Summary

- Use of IP blocks can vastly improve productivity by simplifying design entry.
- Altera provides a wealth of freely available IP blocks for use in your designs. You can also create your own blocks from HDL code.

In this video, you have learned

- How to add and configure library IP blocks.
- How to make your own IP blocks from HDL code.
- How to concatenate and split buses in a schematic design.

Improving timing with pipelining

Objectives

In this presentation, you will use timing constraints to constrain your design, analyze the result, and then apply pipelining to get improved performance.

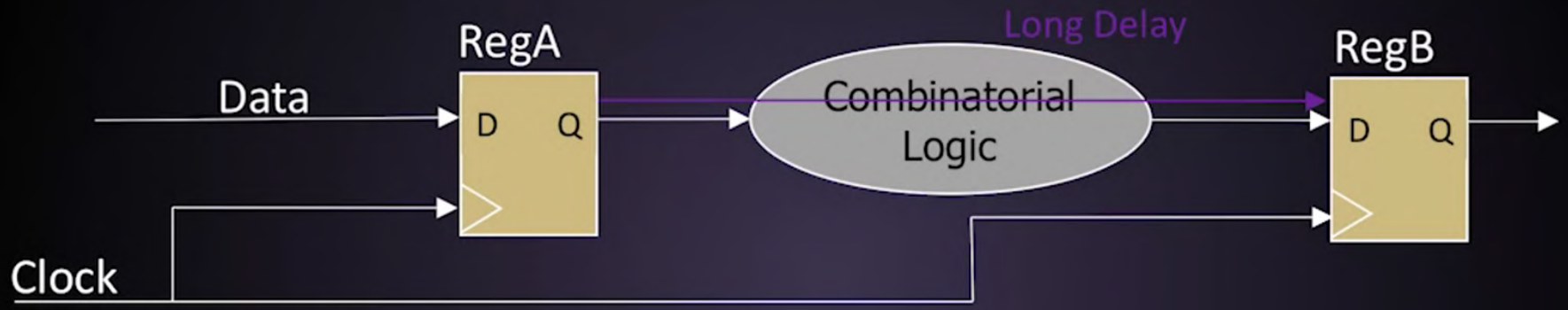
You will learn

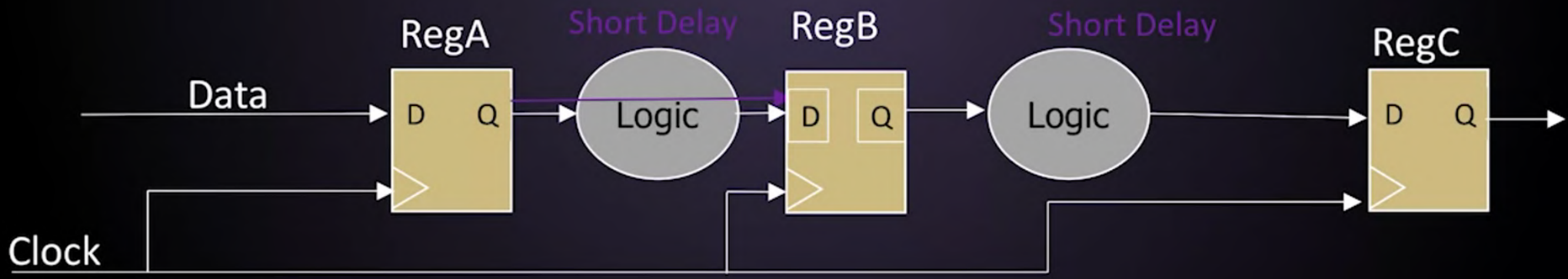
- How pipelining in principle can be applied to improve timing in FPGA designs
- How to use pipelining in IP blocks to improve timing.

Agenda for this Video

1. Add timing constraints to our proto ALU
2. View the results in TimeQuest
3. Change the design to improve performance
4. Update the TimeQuest timing analysis

Pipelining in FPGAs





Preparation

1. Start Quartus Prime
2. Open the pipemult project we last worked on in the previous video.

Improving Timing with Pipelining Summary

- Large combinatorial delays between the input and output can dramatically lower the design maximum frequency.
- In some cases combinatorial delays can be broken into pieces by insertion of flip-flops known as retiming, or pipelining depending on how it is done.
- IP blocks can be configured to use pipelining in some cases, which will add latency but improve overall speed of the design.

In this video, you have learned

- How pipelining in principle can be applied to improve timing in FPGA designs
- How to use pipelining in IP blocks to improve timing.
- How to analyze timing analyzer results to identify critical paths that may benefit from pipelining to improve timing.

FPGA IO: Connections to the External World

Objectives

In this presentation, you will learn how IO standards drive the assignment of internal signals to the external pins of the FPGA.

When you assign a signal from your design to a particular pin on the device, this is known as a pin assignment.

IO Pin assignment is how your design interfaces with the outside world, so it is important to get it right.

FPGA IO: Connections to the External World

You will learn

Details of commonly used IO standards in FPGAs and how they are important to overall performance and system requirements.

How to determine the type of IO that needs to be used for various applications.

The 5 methods for making pin assignments and tracking IO attributes.

Agenda for this Video

1. Review IO standards and structures.

2. Learn the 5 methods for IO pin assignment and attribute control in FPGAs.

I/O Standards Effect Pin Assignments

Most common sources of FPGA failures or delays:

#1 Timing Closure

#2 High Speed Interfaces (PCIe, DDR3)

#3 Pin assignments and I/O errors

I/O Standards and Pin Assignments

I/O Standards vary greatly, even within one FPGA

I/O Organized in banks with one voltage reference. +3.3, +2.5, +1.8 are common; one bank may be different from the next.

Single ended I/O are LVTTL or LVCMOS, with many variations including HTL, SSTL, GTL, etc.

Differential I/O typically LVDS but also LVPECL, CML, etc.

MOST IMPORTANT: Global nets are usually assigned to specific pins that can also be I/O. Board layout must use these pins as clocks or reset if global nets are to be used.

I/O Standards and Pin Assignments

	LVTTL		LVCMOS	
	Min	**Max**	**Min**	**Max**
V_{CCI}/V_{DD}	3.0v	3.6v	3.0v	3.6v
V_{IH}	2.0v		2.0v	
V_{IL}		0.8v		0.8v
V_{OH}	2.4v		$V_{CCI} - 0.2v$	
V_{OL}		0.4v		0.2v

[1]

I/O Standards and Pin Assignments

Some parts have PCI-compliant I/O with clamp diode

	5V PCI (like TTL)		3.3V PCI (like CMOS)	
	Min	**Max**	**Min**	**Max**
V_{CCI}/V_{DD}	4.75v	5.25v	3.0v	3.6v
V_{IH}	2.0v		$0.5V_{CCI}$	
V_{IL}		0.8v		$0.3V_{CCI}$
V_{OH}	2.4v		$0.9V_{CCI}$	
V_{OL}		0.55v		$0.1V_{CCI}$

[2]

I/O Standards and Pin Assignments

LVCMOS has extensions to lower voltages, less power

	LVCMOS2 (2.5V LVCMOS)		LVCMOS3 (1.8V LVCMOS)		1.5V LVCMOS	
	Min	Max	Min	Max	Min	Max
V_{DD}	2.375v	2.625v	1.71v	1.89v	1.425v	1.575v
V_{IH}	1.7v		$0.65V_{CCI}$		$0.65V_{CCI}$	
V_{IL}		0.7v		$0.35V_{CCI}$		$0.35V_{CCI}$
V_{OH}	2.1		V_{CCI} – 0.45v		V_{CCI} – 0.4v	
V_{OL}		0.2v		0.45v		0.4v

[3]

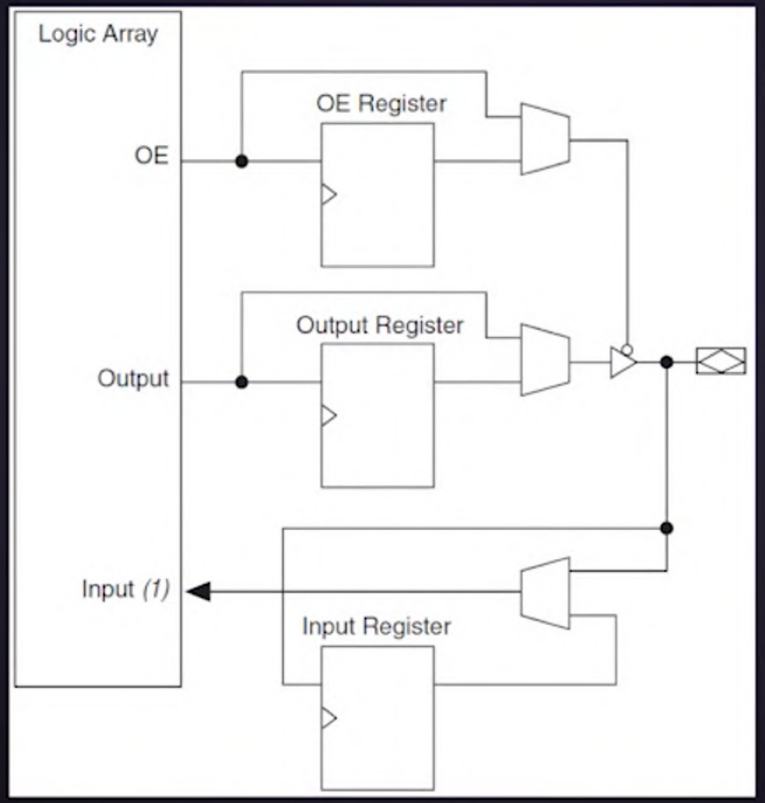
Single-Ended Referenced Standards

SSTL and HSTL
- Stub Series Terminated Logic, High-Speed Transceiver Logic
- I/O Referenced to Common Reference Voltage (approximately Mid-rail)
- Smaller Voltage Swing than LVTTL or LVCMOS

Differential Standards

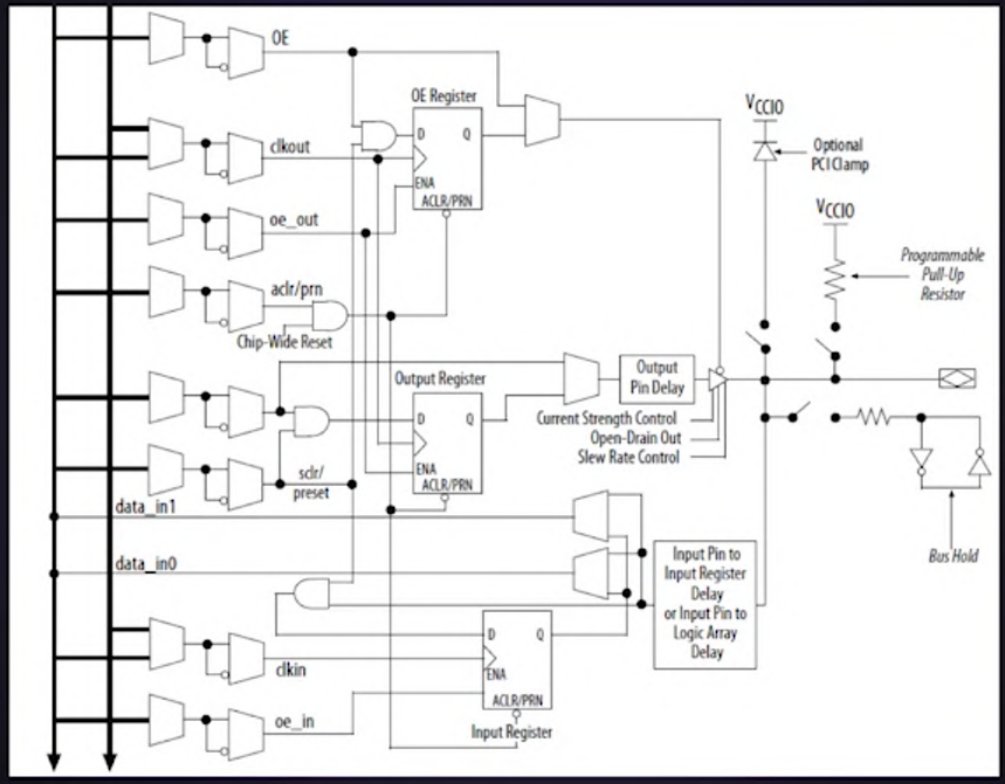
LVDS and LVPECL
- Low-voltage Differential Signaling, Current Mode Logic
- Uses Two Conductors per Signal (Called “Signal Pair”)
- Smaller Swing than HSTL or SSTL, as Small as GTL
- Much Better Noise Immunity than Single-Ended Standards

Typical FPGA IO Structure



[4]

Typical FPGA IO Structure



[5]

I/O Standards and Pin Assignments

Other characteristics also vary from bank to bank

Drive Strength – 2, 4, 6, 8, 12, 16, 24 mA or more

Slew rates – usually high or low

Hot Swappable

Some are input or output only

DDR compatible

PCI compatible

Some Tri-stateable, some are not.

How do we keep track of all this?

IGLOO

I/O Standard	Output Drive (mA)
PCI[1]	PCI
PCIX[1]	PCI
LVDS[2]	24
LVPECL[2]	24
HSTL[3] (Class I)	8
HSTL[3] (Class II)	15
SSTL2[3] (Class I)	17
SSTL2[3] (Class II)	21
SSTL3[3] (Class I)	16
SSTL3[3] (Class II)	24
GTL[3] 33	25
GTL[3] 25	25
GTLP[3] 33	51
GTLP[3] 25	40

[6]

Pin Assignment Methods

In Altera FPGAs, you can do pin assignments in one of 5 ways:

Using Quartus GUI tools, either the Pin planner or Assignment Editor.
Importing from an sdc file or Excel spreadsheet file.
Writing directives in an HDL (Verilog or VHDL) file.
Automating by use of a TCL script.
Importing from a PCB layout CAD tool.

I/O Standards and Pin Assignments

The CAD layout tool can also govern pin assignments:

You can integrate PCB design tools into your work flow to help correctly map pin assignments to the symbols in your system circuit schematics and board layout.

The Quartus II software integrates with board layout tools by allowing import and export of pin assignment information in

Quartus II Settings Files (.qsf),

Pin-Out File (.pin), and

FPGA Xchange-Format File (.fx) files.

I/O Standards and Pin Assignments

Scripts can also be used to define pin locations

Here are some typical tcl commands

```
set_location_assignment PIN_N2 -to reset
set_location_assignment PIN_P1 -to clk1
set_location_assignment PIN_M3 -to ticket[1]
set_location_assignment PIN_AE24 -to ~LVDS150p/nCEO~
set_location_assignment PIN_C2 -to accel
set_location_assignment PIN_K4 -to ticket[3]
```

I/O Standards and Pin Assignments

HDL code can be the source of pin assignments

Example Verilog HDL Synthesis Attribute

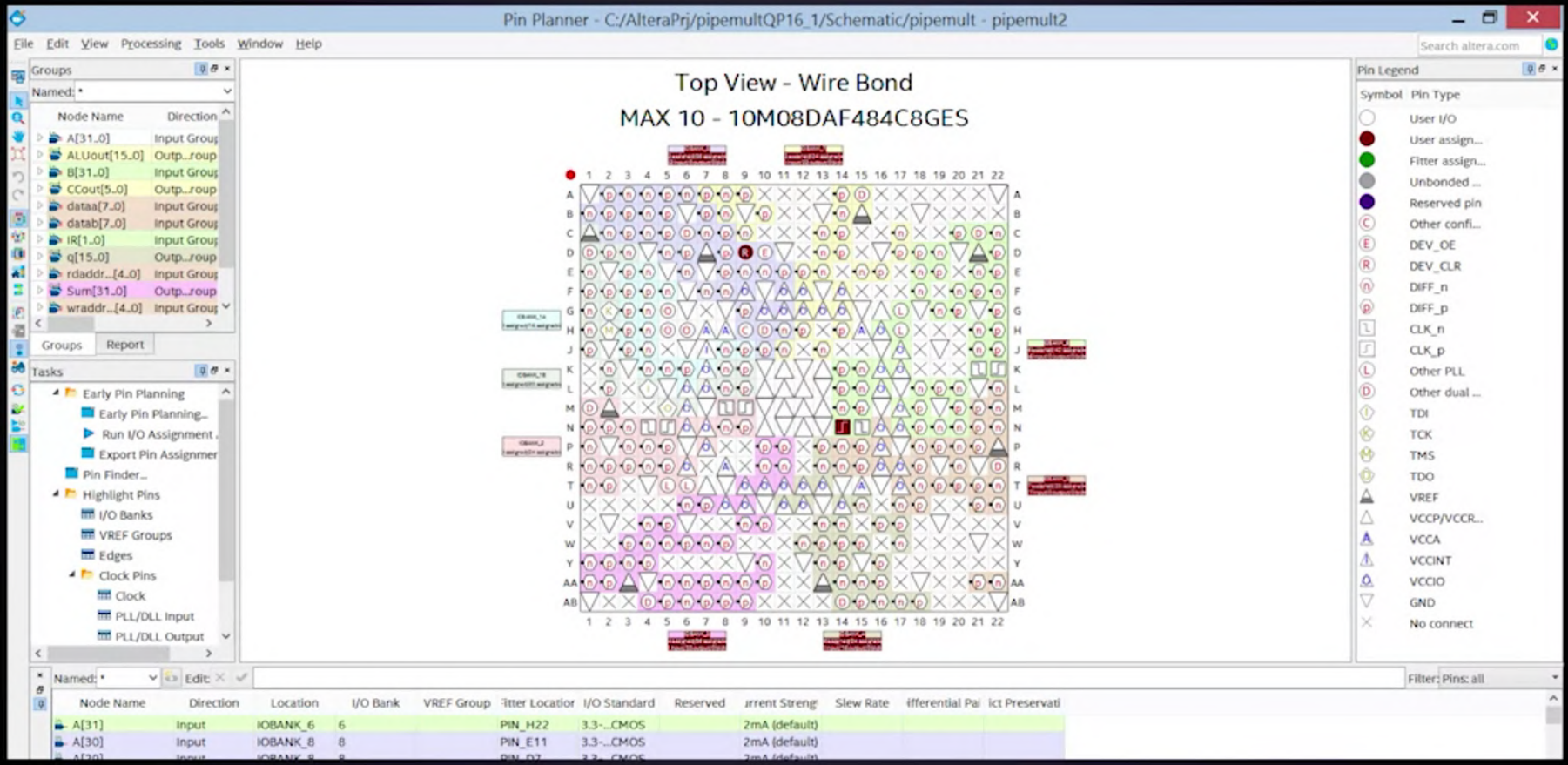
```
input my_pin1 /* synthesis altera_attribute = "-name FAST_INPUT_REGISTER ON;
-name IO_STANDARD \"2.5 V\" " */ ;
```

Example VHDL Synthesis Attribute

```
entity my_entity is
port(
my_pin1: in std_logic
);
end my_entity;

architecture rtl of my_entity is
attribute useioff : boolean;
attribute useioff of my_pin1 : signal is true;
attribute chip_pin : string;
attribute chip_pin of my_pin1 : signal is "C1";
begin -- The architecture body
end rtl;
```

FPGA Vendor GUI Tools may be best for Pin Assignment



FPGA IO Summary

- FPGAs have extensive IO capabilities, including many IO standards and controllable circuit characteristics, but keeping track of the wealth of options leads to frequent errors. We can reduce IO pin assignment errors by the use of methods that help document and track IO usage.
- Differential logic standards like LVDS and LVPECL are increasingly popular in FPGAs as they allow increased speed and better noise immunity at the same time. However, these standards require conductor pairs, further complicating the pin assignment process.

In this video, you have learned

- Details of commonly used IO standards in FPGAs and how they are important to overall performance and system requirements.
- How to determine the type of IO that needs to be used for various applications.
- The 5 methods for making pin assignments and tracking IO attributes.

Pin Assignments

Objectives

In this presentation, you will learn how to assign internal signals to the external pins of the FPGA.

IO Pin assignment is how your design interfaces with the outside world, so it is important to get it right.

You will learn

- How to do pin assignments using the pin planner and the assignment editor.
- How to meet IO standards and control IO properties using the assignment editor.
- How to check IO assignments using the IO Assignment Analyzer.

Agenda for this Video

1. Assign pins to your design using the pin planner.
2. Assign pins to your design using the assignment editor.
3. Check your pin assignments using the IO assignment analyzer.

Pin Assignment Methods

In Altera FPGAs, you can do pin assignments in one of 5 ways:

1. Using Quartus GUI tools, either the Pin planner or Assignment Editor.
2. Importing from an sdc file or Excel spreadsheet file.
3. Writing directives in an HDL (Verilog or VHDL) file.
4. Automating by use of a TCL script.
5. Importing from a PCB layout CAD tool.

Pin Assignment Summary

- Pin assignments can easily be done in Quartus Prime using either the Pin Planner graphical tool or the Assignment Editor spreadsheet.
- It is important to assign global signals like clocks and reset to global network pins.
- The IO Assignment Analyzer can check the pin assignments to make sure they are complete and correct.

In this video, you have learned

- How to do pin assignments using the Pin Planner and the Assignment Editor.
- How to meet IO standards and control IO properties using the assignment editor.
- How to check IO assignments using the IO Assignment Analyzer.

Programming the FPGA

In this presentation, you will learn how to program and configure your FPGA for use. You will learn:

- How FPGAs are configured using a variety of configuration schemes, including the hardware design and software required for each.
- What programming file types to use in each configuration situation.
- How to use the Quartus Prime programmer to examine and program FGPAs and PLDs in a chain.

Programming and Configuration

Programming = the action of placing into memory contents of a file that will be used to define the behavior of the device, whether it be a microcontroller or a programmable logic device.

Configuration = the process of loading a controlling memory image into the device at powerup to establish the characteristic operating behavior of the device.

Altera Configuration Modes

Configuration modes include:

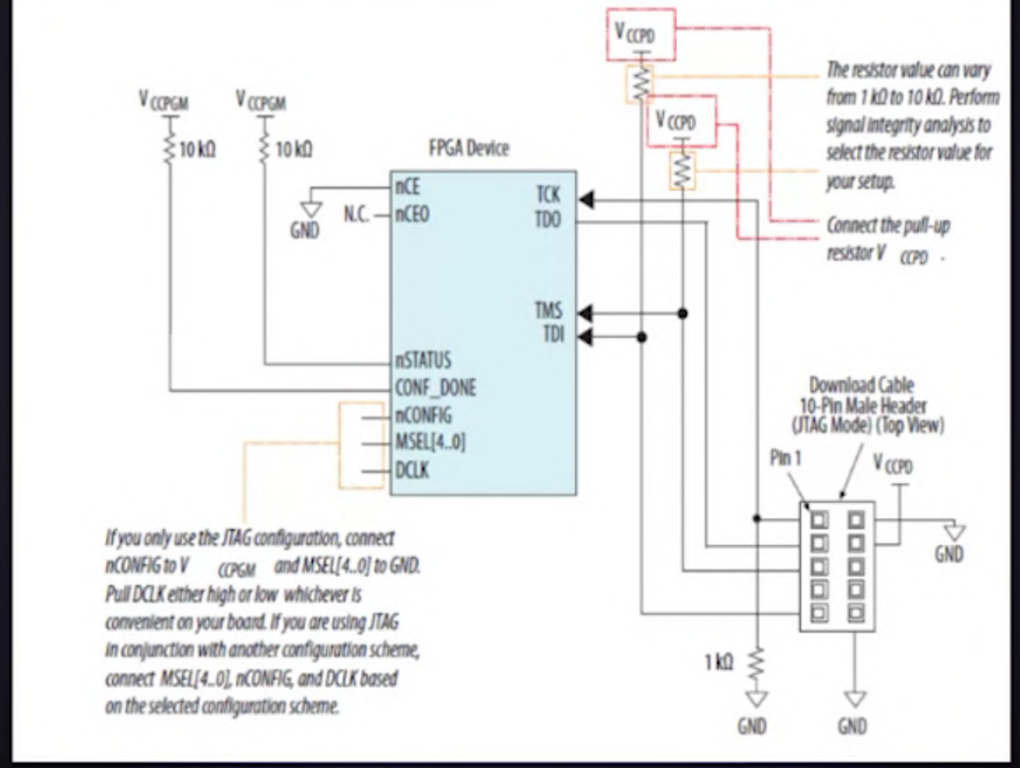
- JTAG
- Active Serial
- Passive Serial
- Fast Passive Parallel
- Configuration via Protocol (CVP)

Altera Configuration Modes

Configuration Mode	Data width	Max Clock Rate (MHz)	Decompression	Design Security	Partial Reconfiguration	Remote System Update
JTAG	1	33	-	-	-	-
Active Serial (AS) through EPCS or EPCQ serial FLASH memory configuration device	1 or 4	100	Yes	Yes	-	Yes
Passive Serial (PS) through CPLD or External Microcontroller	1	125	Yes	Yes	-	-
Fast Passive Parallel	8	125	Yes	Yes	-	-
Fast Passive Parallel	16	125	Yes	Yes	Yes	Parallel Loader
Configuration Via Protocol (CVP) via PCIe	x1, x2, and x4 lanes	-	Yes	Yes	Yes	

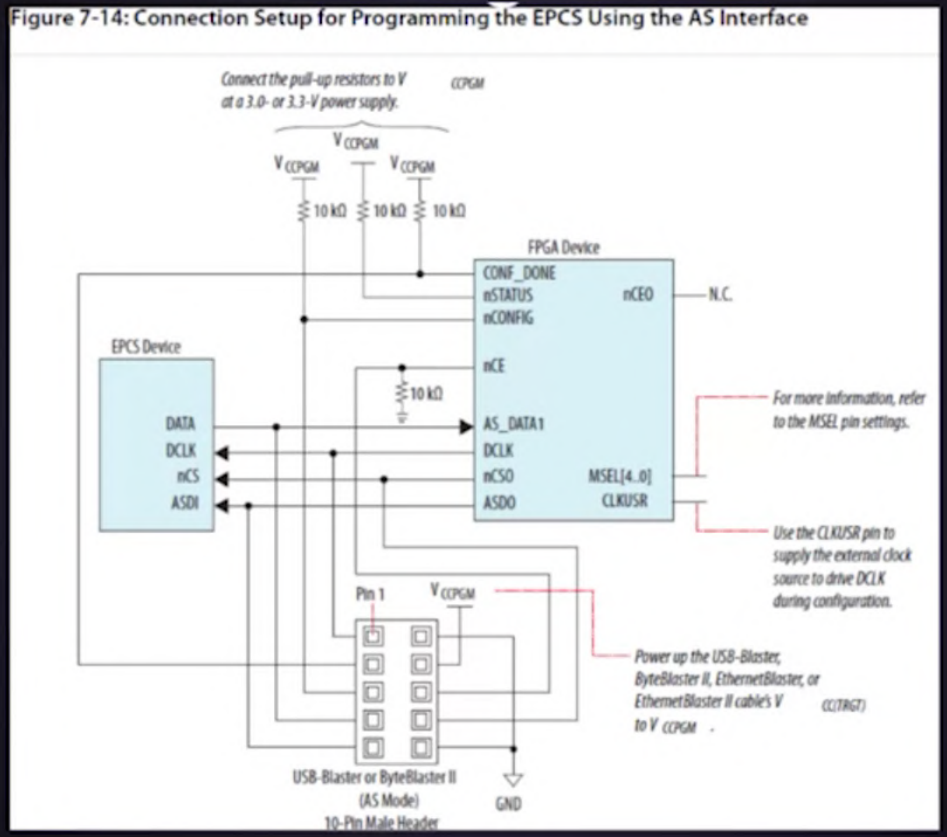
JTAG Configuration Mode

Figure 7-21: JTAG Configuration of a Single Device Using a Download Cable



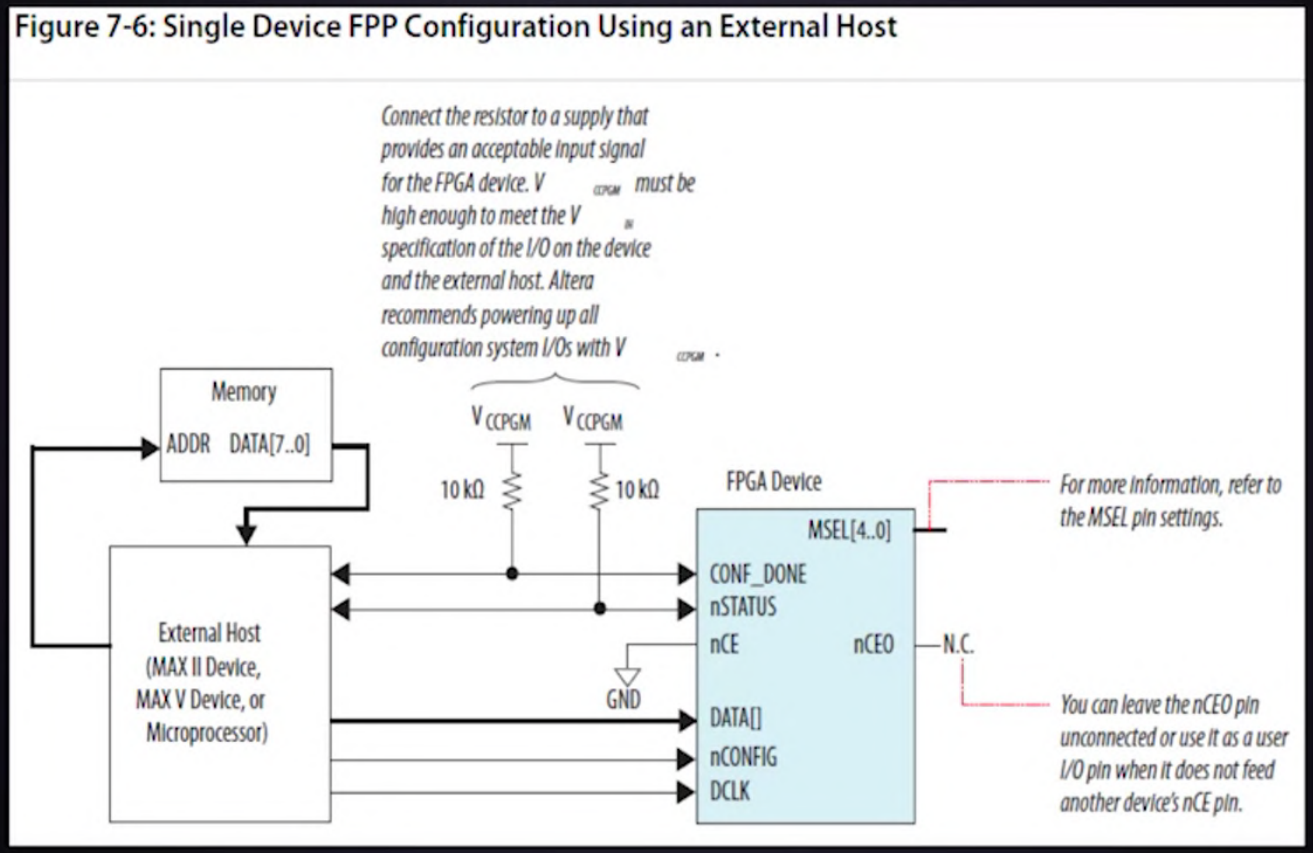
[1]

Active Serial Configuration Mode



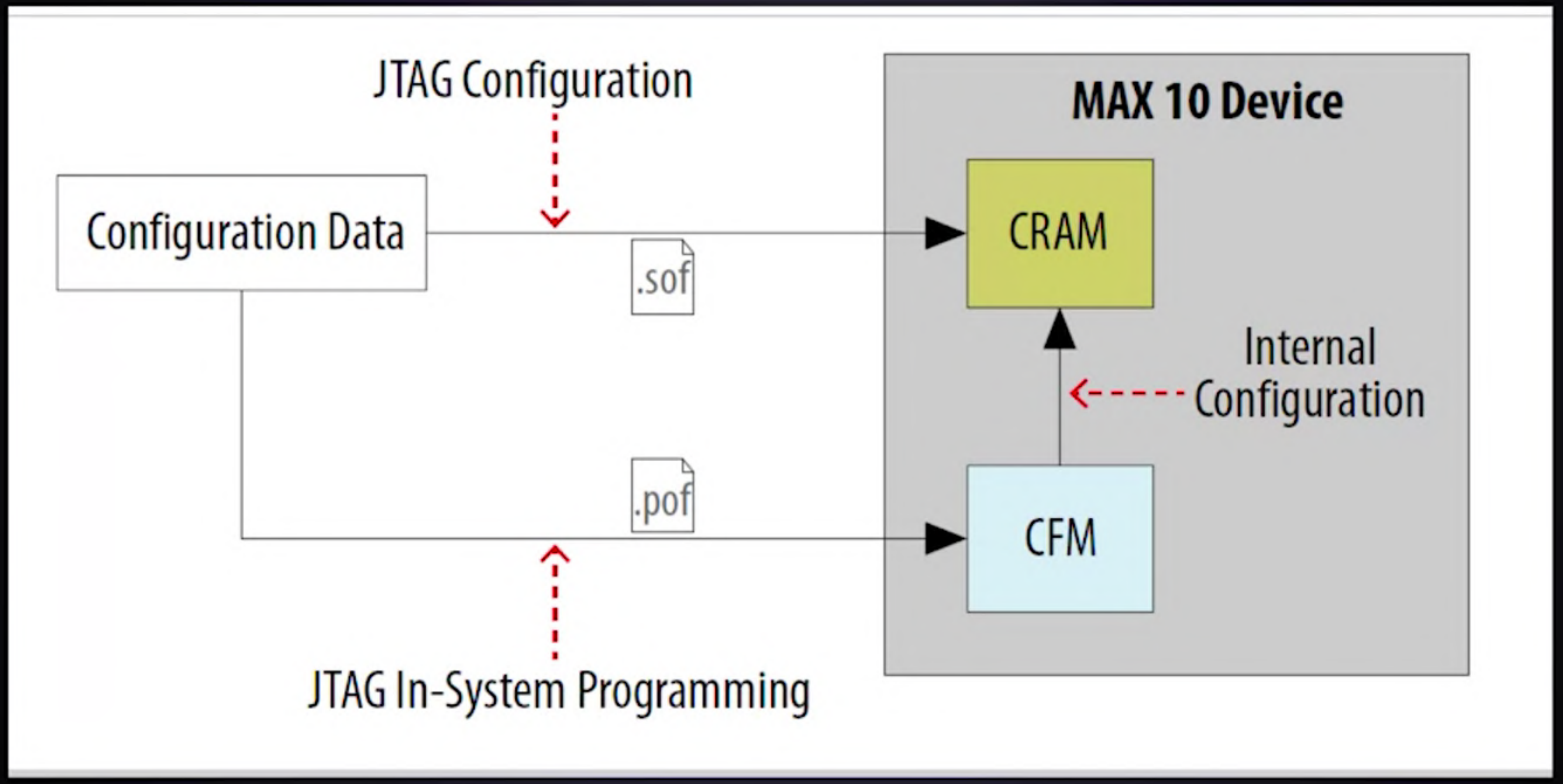
[2]

Fast Passive Parallel Configuration Mode



[3]

Altera MAX10 Programming



[4]

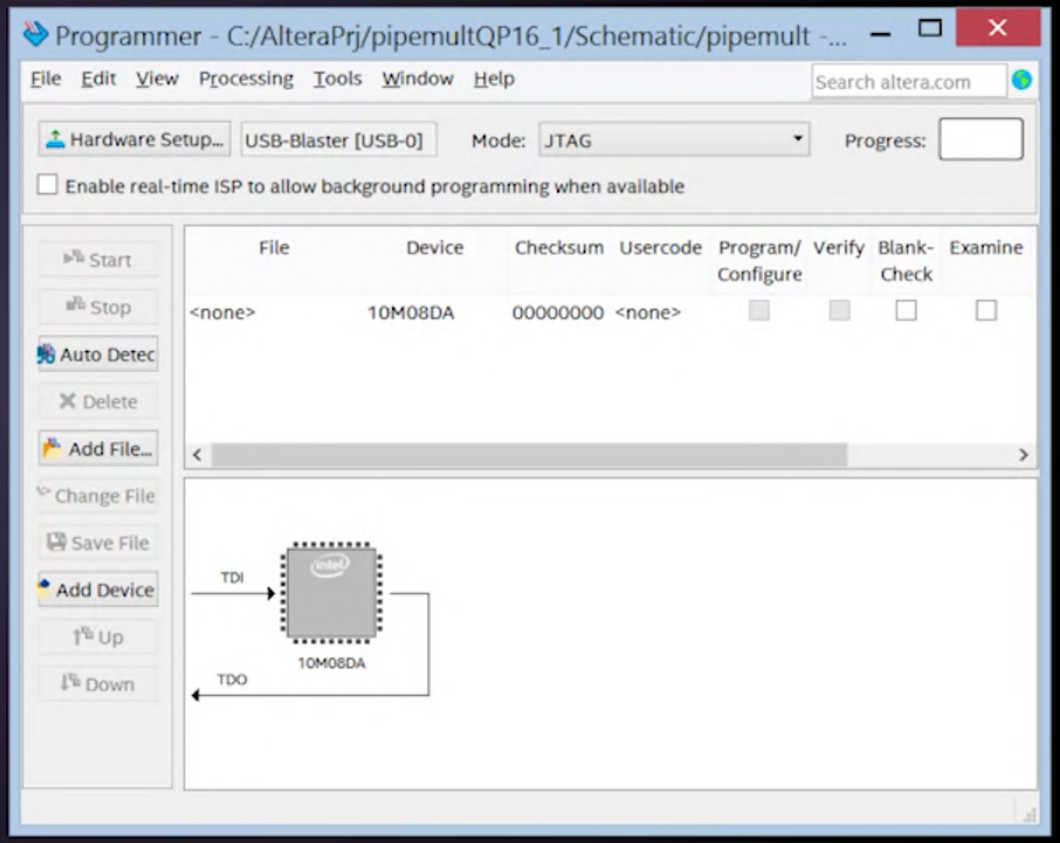
Programming File Types

There are a variety of programming file types for various configuration situations, including:

- .sof (SRAM Object File) - which is used to configure FPGAs directly from Quartus Prime software through a download cable.
- .pof (Programming Object File) – Used to Program CPLDs, FLASH FPGAs, and configuration FLASH memories.
- .jam/.jbc – ASCII file used by processors and test equipment to program devices via JTAG
- .jic(JTAG Indirect Configuration File) – is used to program EPCS (Altera serial configuration) devices

Quartus Prime Programmer

- Controls programming of the FPGA
- Creates a chain description file (.cdf)



Programming the FPGA Summary

You have learned

- How FPGAs are configured using a variety of configuration schemes from JTAG to Active Serial to Fast Passive Parallel. Each scheme has unique hardware design and software required.
- What programming file types to use in each configuration situation.
- How to use the Quartus Prime programmer to examine and program FGPAs and PLDs in a chain.

Becoming one with Q

Objectives

- In the previous sessions, we have learned how to design some logic to create a proto-ALU. However, we can leverage the power of Quartus to make a true ALU very quickly, and can in fact make an entire softcore CPU in the FPGA using a design tool known as Qsys.
- In this presentation, you will learn how to design a system within the FPGA using the system design tool Qsys. By the time you are done, people may mistake you for Q! You will learn:
  - How to use a project template to accelerate the beginning of a design effort.
  - How to create a Qsys system.
  - How to add clocks.
  - How to add a NIOS II processor

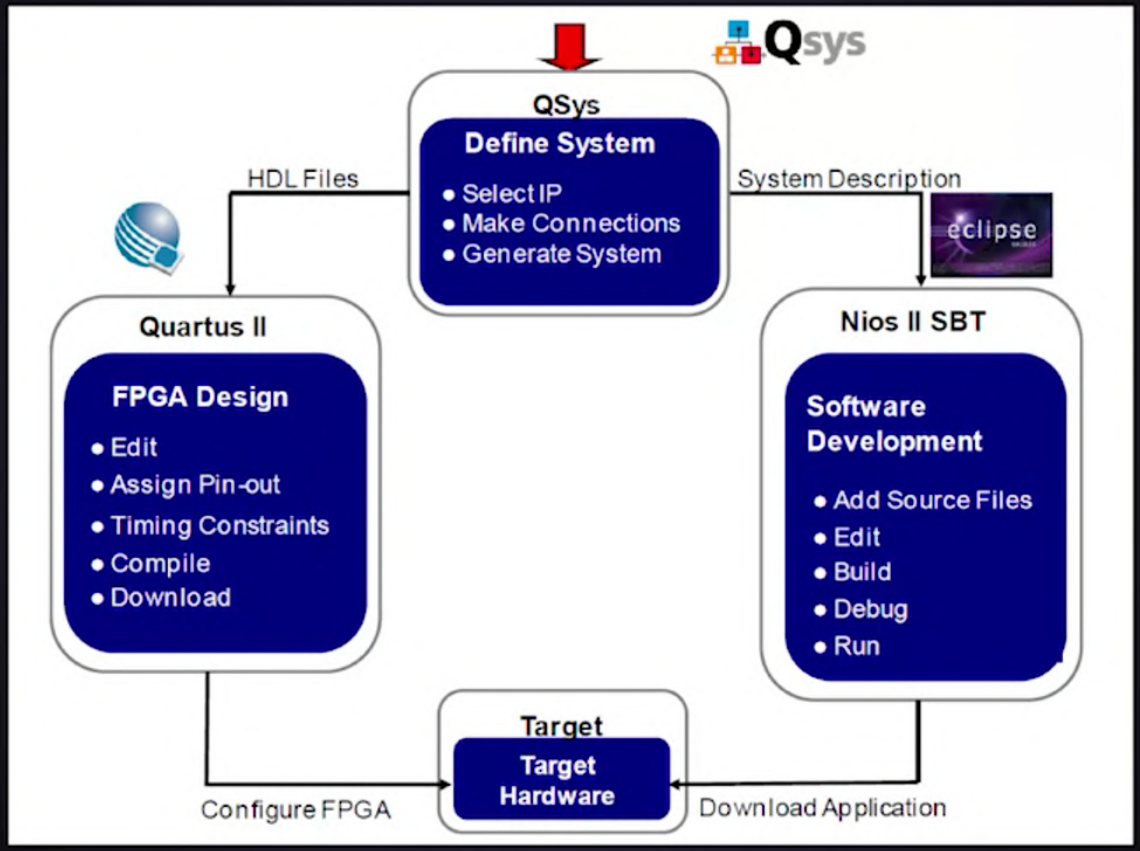
Agenda for this Video

1. Create a new project using a board specific project template.

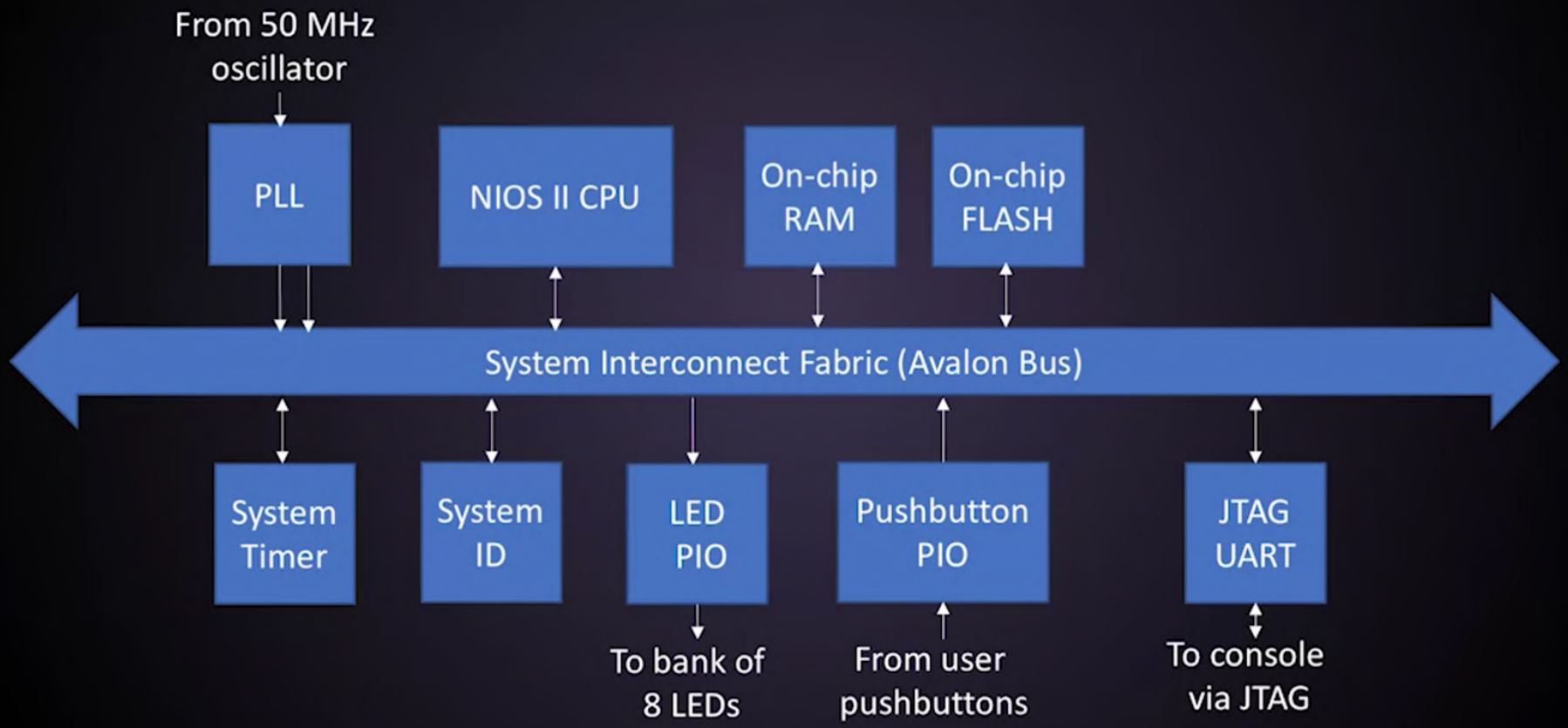
2. Create a Qsys system by adding first a PLL as a clock generator.

3. Add a softcore processor to the system.

Qsys Design Flow



System Block Diagram



Video Summary

- Qsys is a powerful and easy to use tool to create complete systems using block diagram concepts with detailed connections.
- Rather than design a CPU in logic from scratch, a better alternative is to use existing CPU designs like Altera's NIOS II. The design can be done very quickly, and is optimized for performance and resource usage.

In this video, you have learned

- How to use a project template to accelerate the beginning of a design effort.
- How to create a Qsys system.
- How to add Phase-Locked Loops as clock sources.
- How to create a NIOS II softcore processor.

Becoming one with Q

Objectives

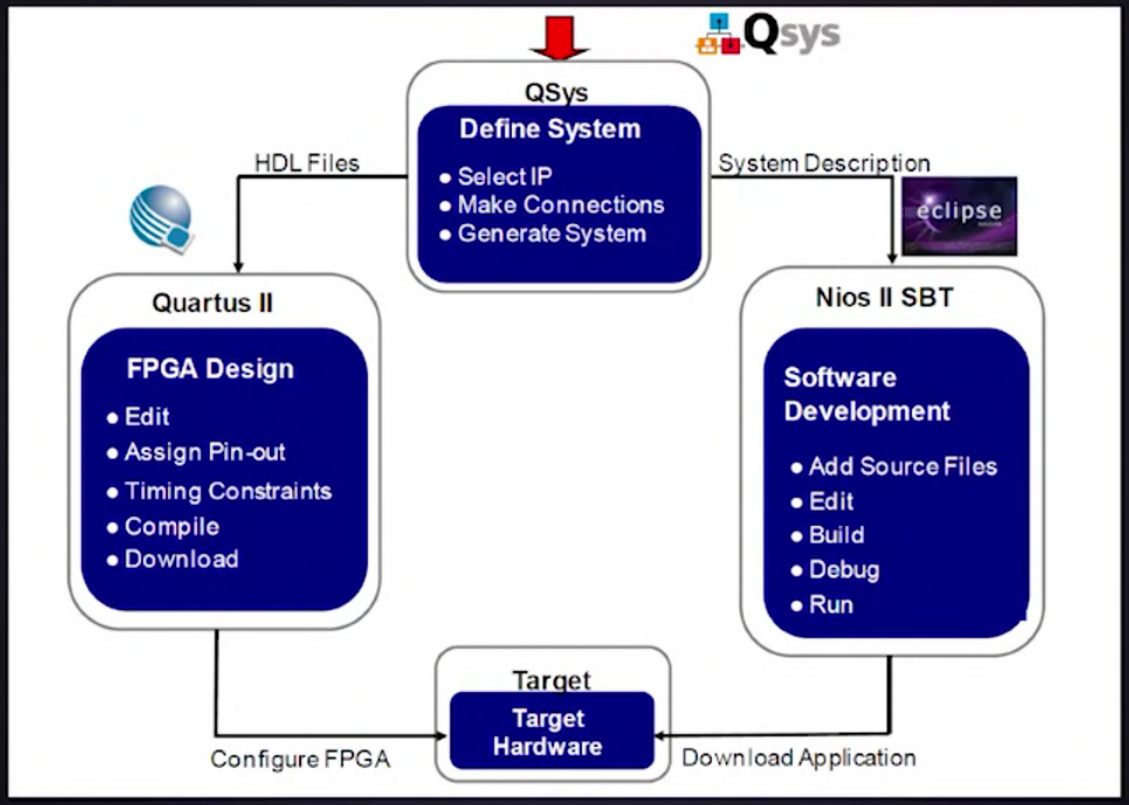
- In the previous session, we introduced the Qsys system design tool and the NIOS II soft processor. We started a Qsys design including the processor.
- In this presentation, we will complete the design by adding memory and peripherals to the embedded system, and do some analysis of the result. You will learn:

How to add memory and peripherals to a NIOS II Qsys design.
The purpose of a bus bridge and how to implement it in a Qsys design.
How to bring the Qsys design into the top level to be compiled.

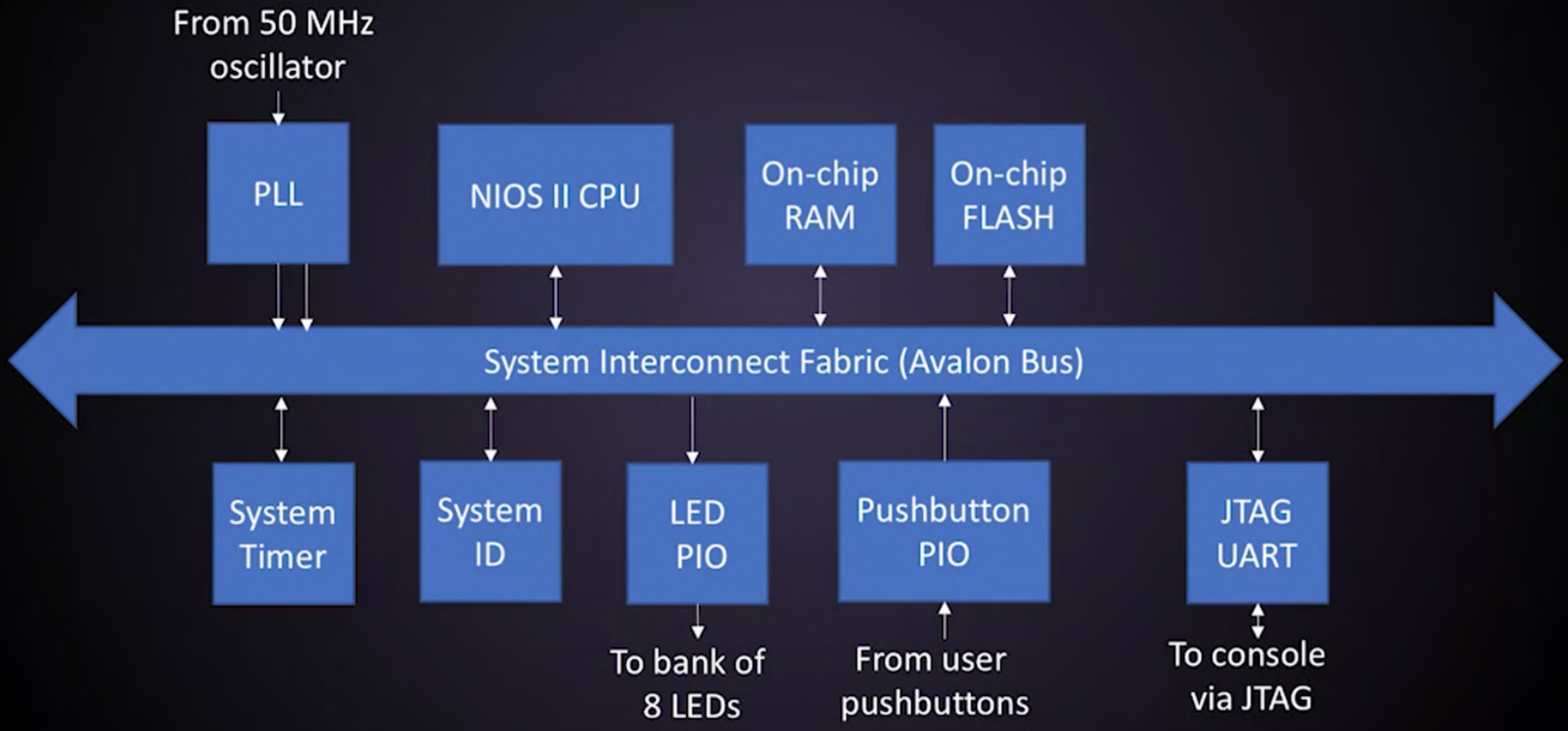
Agenda for this Video

1. Complete the Qsys project by adding memory and several simple peripherals, including a UART and a timer.
2. Instantiate the Qsys design in the top level by making it into a schematic symbol.
3. Compile the Qsys design.
4. Analyze the design using the RTL viewer.

Qsys Design Flow



System Block Diagram



Becoming one with Qsys
Part A Complete

Becoming one with Qsys
Part B

Video Summary

- The highly functional NIOS2 soft processor hardware can be created and customized quickly in Altera FPGAs by using the Qsys system design tool.
- Qsys empowers designers by leveraging IP, incorporating a wide variety of processor, peripheral, and memory components into a system connected together by buses and bridges.

In this video, you have learned

- How to add memory and peripherals to a NIOS II Qsys design.
- The purpose of a bus bridge and how to implement it in a Qsys design.
- How to bring the Qsys design into the top level to be compiled either by creating a block symbol schematic or by instantiation in a top-level Verilog file.