FPGA Development for Radar, Radio-Astronomy and Communications

THE RADAR MASTERS COURSE

Dept. Electrical Engineering, University of Cape Town
Private Bag, Rondebosch, 7701, South Africa

http://www.rrsg.uct.ac.za

Presented by John-Philip Taylor

Convened by Prof Daniel O'Hagan

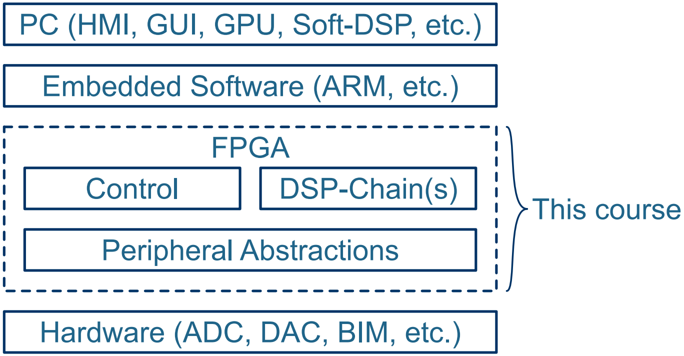
Tutored by Stephen Paine and Randy Cheng

Day 1 – 17 July 2017

Outline

Outline

Course Overview


PC (HMI, GUI, GPU, Soft-DSP, etc.)
Embedded Software (ARM, etc.)
FPGA
Control
DSP-Chain(s)
Peripheral Abstractions
This course
Hardware (ADC, DAC, BIM, etc.)

Course Overview

- Overall (Morning)
  - This course specifically targets low-level development
  - No high-level synthesis (HLS)
  - No soft-core or embedded processors / etc.
- Practicals (Afternoon)
  - The practicals are challenging to finish in the time provided, so keep the board and work on it after the course...
  - Primarily Verilog, but feel free to implement your practicals in VHDL
  - These are informal – feel free to structure your time as you see fit, or even do the practicals at home
- **Never be shy to ask questions**

Course Overview

- Overall (Morning)
  - This course specifically targets low-level development (when the high-level libraries don't provide the required functionality)
  - No high-level synthesis (HLS)
  - No soft-core or embedded processors / etc.
- Practicals (Afternoon)
  - The practicals are challenging to finish in the time provided, so keep the board and work on it after the course...
  - Primarily Verilog, but feel free to implement your practicals in VHDL
  - These are informal – feel free to structure your time as you see fit, or even do the practicals at home
- **Never be shy to ask questions**

Course Overview

- Overall (Morning)
  - This course specifically targets low-level development (when the high-level libraries don't provide the required functionality)
  - No high-level synthesis (HLS)
  - No soft-core or embedded processors / etc.
- Practicals (Afternoon)
  - The practicals are challenging to finish in the time provided, so keep the board and work on it after the course...
  - Primarily Verilog, but feel free to implement your practicals in VHDL
  - These are informal – feel free to structure your time as you see fit, or even do the practicals at home
- **Never be shy to ask questions**

Course Overview

- Overall (Morning)
  - This course specifically targets low-level development (when the high-level libraries don't provide the required functionality)
  - No high-level synthesis (HLS)
  - No soft-core or embedded processors / etc.
- Practicals (Afternoon)
  - The practicals are challenging to finish in the time provided, so keep the board and work on it after the course...
  - Primarily Verilog, but feel free to implement your practicals in VHDL
  - These are informal – feel free to structure your time as you see fit, or even do the practicals at home
- **Never be shy to ask questions**

Course Overview

- Overall (Morning)
  - This course specifically targets low-level development (when the high-level libraries don't provide the required functionality)
  - No high-level synthesis (HLS)
  - No soft-core or embedded processors / etc.
- Practicals (Afternoon)
  - The practicals are challenging to finish in the time provided, so keep the board and work on it after the course...
  - Primarily Verilog, but feel free to implement your practicals in VHDL
  - These are informal – feel free to structure your time as you see fit, or even do the practicals at home
- **Never be shy to ask questions**

THE RADAR MASTERS COURSE

Course Overview

- Overall (Morning)
  - This course specifically targets low-level development (when the high-level libraries don't provide the required functionality)
  - No high-level synthesis (HLS)
  - No soft-core or embedded processors / etc.
- Practicals (Afternoon)
  - The practicals are challenging to finish in the time provided, so keep the board and work on it after the course...
  - Primarily Verilog, but feel free to implement your practicals in VHDL
  - These are informal – feel free to structure your time as you see fit, or even do the practicals at home
- **Never be shy to ask questions**

Course Overview

- Overall (Morning)
  - This course specifically targets low-level development (when the high-level libraries don't provide the required functionality)
  - No high-level synthesis (HLS)
  - No soft-core or embedded processors / etc.
- Practicals (Afternoon)
  - The practicals are challenging to finish in the time provided, so keep the board and work on it after the course...
  - Primarily Verilog, but feel free to implement your practicals in VHDL
  - These are informal – feel free to structure your time as you see fit, or even do the practicals at home
- **Never be shy to ask questions**

Course Overview

- Overall (Morning)
  - This course specifically targets low-level development (when the high-level libraries don't provide the required functionality)
  - No high-level synthesis (HLS)
  - No soft-core or embedded processors / etc.
- Practicals (Afternoon)
  - The practicals are challenging to finish in the time provided, so keep the board and work on it after the course...
  - Primarily Verilog, but feel free to implement your practicals in VHDL (the presenter is well-versed in both)
  - These are informal – feel free to structure your time as you see fit, or even do the practicals at home
- **Never be shy to ask questions**

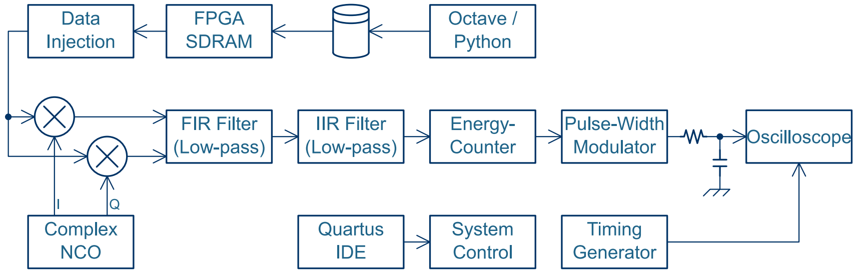
Course Overview

- Overall (Morning)
  - This course specifically targets low-level development (when the high-level libraries don’t provide the required functionality)
  - No high-level synthesis (HLS)
  - No soft-core or embedded processors / etc.
- Practicals (Afternoon)
  - The practicals are challenging to finish in the time provided, so keep the board and work on it after the course...
  - Primarily Verilog, but feel free to implement your practicals in VHDL (the presenter is well-versed in both)
  - These are informal – feel free to structure your time as you see fit, or even do the practicals at home
- **Never be shy to ask questions**

Course Overview

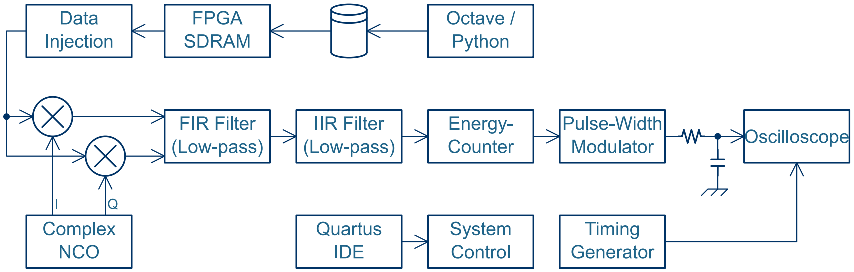
- Overall (Morning)
  - This course specifically targets low-level development (when the high-level libraries don't provide the required functionality)
  - No high-level synthesis (HLS)
  - No soft-core or embedded processors / etc.
- Practicals (Afternoon)
  - The practicals are challenging to finish in the time provided, so keep the board and work on it after the course...
  - Primarily Verilog, but feel free to implement your practicals in VHDL (the presenter is well-versed in both)
  - These are informal – feel free to structure your time as you see fit, or even do the practicals at home
- **Never be shy to ask questions**

Practicals



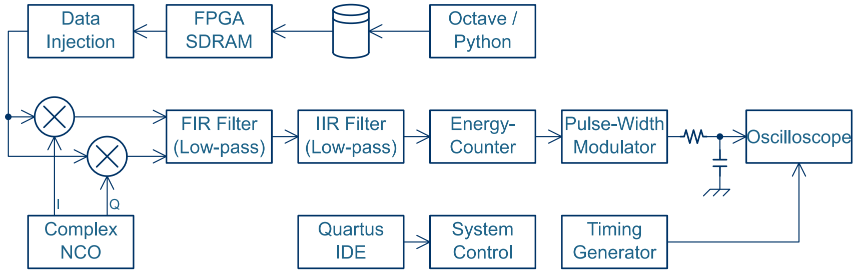
- JTAG interfacing to / from the computer (Source-and-Probes and TCL scripting)
- 8-bit, 100 MSps data injection
- 1024-point FIR-filter with 128× decimation

Practicals



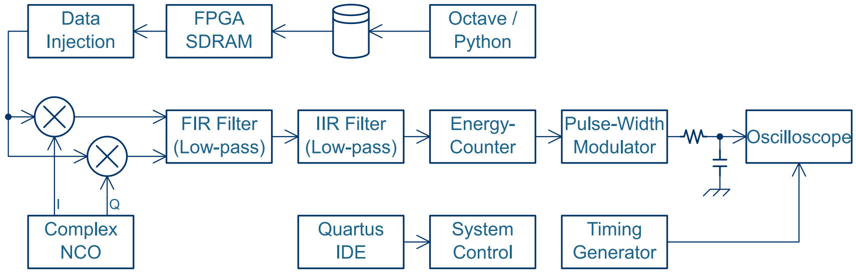
- JTAG interfacing to / from the computer (Source-and-Probes and TCL scripting)
- 8-bit, 100 MSps data injection
- 1024-point FIR-filter with 128× decimation

Practicals



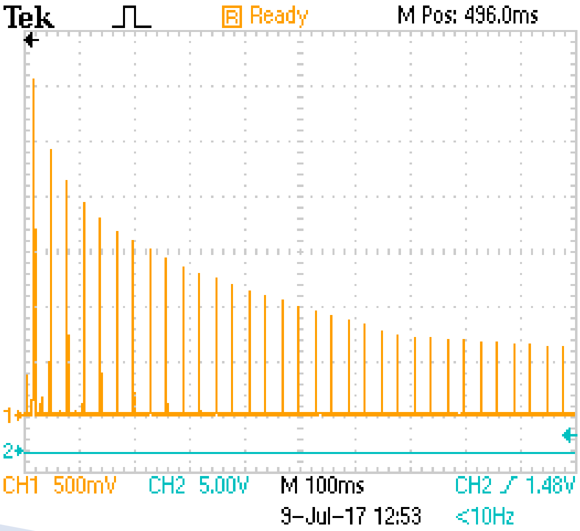
- JTAG interfacing to / from the computer (Source-and-Probes and TCL scripting)
- 8-bit, 100 MSps data injection
- 1024-point FIR-filter with 128× decimation

Practicals



- JTAG interfacing to / from the computer (Source-and-Probes and TCL scripting)
- 8-bit, 100 MSps data injection
- 1024-point FIR-filter with 128$\times$ decimation

Practicals


Tek
Ready
M Pos: 496.0ms
1
2
CH1 500mV
CH2 5.00V
M 100ms
CH2 ∕ 1.48V
9-Jul-17 12:53
<10Hz

The Audience

- Programming experience?
  - C / C++ / C#?
  - Embedded? PC?
  - GPU? OpenGL? OpenCL? DirectX? CUDA?
- FPGA Experience?
  - VHDL? Verilog? PyHDL? Migen? Graphical tools?
  - Synthesis? Simulation only?
  - What context / application?

The Audience

- Programming experience?
  - C / C++ / C#?
  - Embedded? PC?
  - GPU? OpenGL? OpenCL? DirectX? CUDA?
- FPGA Experience?
  - VHDL? Verilog? PyHDL? Migen? Graphical tools?
  - Synthesis? Simulation only?
  - What context / application?

THE RADAR MASTERS COURSE

The Audience

- Programming experience?
  - C / C++ / C#?
  - Embedded? PC?
  - GPU? OpenGL? OpenCL? DirectX? CUDA?
- FPGA Experience?
  - VHDL? Verilog? PyHDL? Migen? Graphical tools?
  - Synthesis? Simulation only?
  - What context / application?

The Audience

- Programming experience?
  - C / C++ / C#?
  - Embedded? PC?
  - GPU? OpenGL? OpenCL? DirectX? CUDA?
- FPGA Experience?
  - VHDL? Verilog? PyHDL? Migen? Graphical tools?
  - Synthesis? Simulation only?
  - What context / application?

The Audience

- Programming experience?
  - C / C++ / C#?
  - Embedded? PC?
  - GPU? OpenGL? OpenCL? DirectX? CUDA?
- FPGA Experience?
  - VHDL? Verilog? PyHDL? Migen? Graphical tools?
  - Synthesis? Simulation only?
  - What context / application?

The Audience

- Programming experience?
  - C / C++ / C#?
  - Embedded? PC?
  - GPU? OpenGL? OpenCL? DirectX? CUDA?
- FPGA Experience?
  - VHDL? Verilog? PyHDL? Migen? Graphical tools?
  - Synthesis? Simulation only?
  - What context / application?

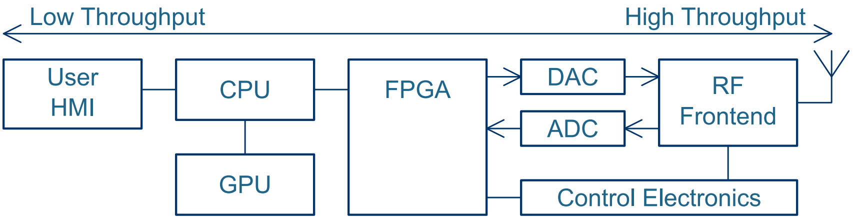
The Audience

- Programming experience?
  - C / C++ / C#?
  - Embedded? PC?
  - GPU? OpenGL? OpenCL? DirectX? CUDA?
- FPGA Experience?
  - VHDL? Verilog? PyHDL? Migen? Graphical tools?
  - Synthesis? Simulation only?
  - What context / application?

The Audience

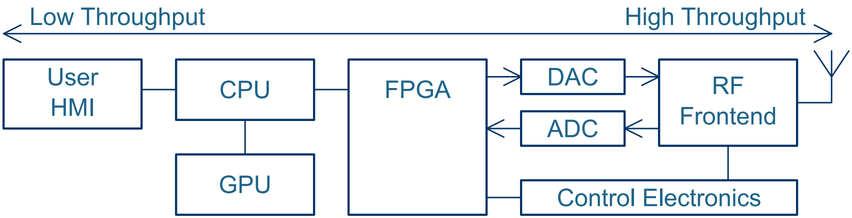
- Programming experience?
  - C / C++ / C#?
  - Embedded? PC?
  - GPU? OpenGL? OpenCL? DirectX? CUDA?
- FPGA Experience?
  - VHDL? Verilog? PyHDL? Migen? Graphical tools?
  - Synthesis? Simulation only?
  - What context / application?

Software-Defined Radio



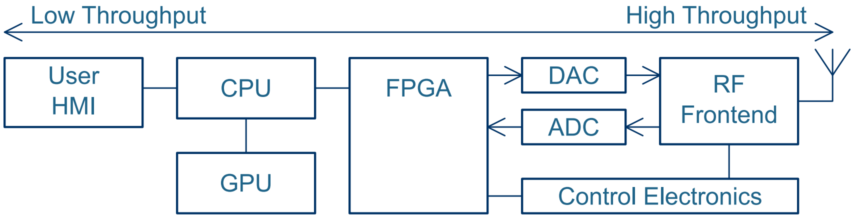
- In high-performance computing, the CPU, GPU and FPGA work together
- Each processing element is used for its strength

Software-Defined Radio



- In high-performance computing, the CPU, GPU and FPGA work together
- Each processing element is used for its strength

Software-Defined Radio



- In high-performance computing, the CPU, GPU and FPGA work together
- Each processing element is used for its strength

Processing Platforms

- CPU:
  - General-purpose; Easily modified; Short development cycle
  - Extensive history ⇒ Rich set of matured libraries and APIs
  - Does not handle parallel algorithms particularly well
- GPU:
- FPGA:

Processing Platforms

- CPU:
  - General-purpose; Easily modified; Short development cycle
  - Extensive history ⇒ Rich set of matured libraries and APIs
  - Does not handle parallel algorithms particularly well
- GPU:
- FPGA:

Processing Platforms

- CPU:
  - General-purpose; Easily modified; Short development cycle
  - Extensive history $\Rightarrow$ Rich set of matured libraries and APIs
  - Does not handle parallel algorithms particularly well
- GPU:
- FPGA:

Processing Platforms

- CPU:
  - General-purpose; Easily modified; Short development cycle
  - Extensive history $\Rightarrow$ Rich set of matured libraries and APIs
  - Does not handle parallel algorithms particularly well
- GPU:
  - Agile: can be reprogrammed in real-time (kernels are referenced by a handle, or memory address)
  - Extremely good at course-grained parallel algorithms
  - Medium development cycle
  - Memory bandwidth problems
- FPGA:

Processing Platforms

- CPU:
  - General-purpose; Easily modified; Short development cycle
  - Extensive history $\Rightarrow$ Rich set of matured libraries and APIs
  - Does not handle parallel algorithms particularly well
- GPU:
  - Agile: can be reprogrammed in real-time
  - Extremely good at course-grained parallel algorithms (little to none inter-worker communication: matrix multiplication; FFT; FIR filters; etc.)
  - Medium development cycle
  - Memory bandwidth problems
- FPGA:

Processing Platforms

- CPU:
  - General-purpose; Easily modified; Short development cycle
  - Extensive history $\Rightarrow$ Rich set of matured libraries and APIs
  - Does not handle parallel algorithms particularly well
- GPU:
  - Agile: can be reprogrammed in real-time
  - Extremely good at course-grained parallel algorithms
  - Medium development cycle – debugging and optimising is a challenging process
  - Memory bandwidth problems
- FPGA:

Processing Platforms

- CPU:
  - General-purpose; Easily modified; Short development cycle
  - Extensive history $\Rightarrow$ Rich set of matured libraries and APIs
  - Does not handle parallel algorithms particularly well
- GPU:
  - Agile: can be reprogrammed in real-time
  - Extremely good at course-grained parallel algorithms
  - Medium development cycle
  - Memory bandwidth problems (the bottle-neck is generally in data transfer, but less so in modern GPUs with advanced parallel copy-engines)
- FPGA:

Processing Platforms

- CPU:
  - General-purpose; Easily modified; Short development cycle
  - Extensive history $\Rightarrow$ Rich set of matured libraries and APIs
  - Does not handle parallel algorithms particularly well
- GPU:
  - Agile: can be reprogrammed in real-time
  - Extremely good at course-grained parallel algorithms
  - Medium development cycle
  - Memory bandwidth problems
- FPGA:
  - Highly flexible, but with a relatively slow clock
  - Provides ASIC functionality at a small fraction of the cost
  - Excellent at fine-grained and time-critical problems
  - Long development cycle; challenging to debug and optimise
  - Not agile: takes a few seconds to switch functionality

Processing Platforms

- CPU:
  - General-purpose; Easily modified; Short development cycle
  - Extensive history $\Rightarrow$ Rich set of matured libraries and APIs
  - Does not handle parallel algorithms particularly well
- GPU:
  - Agile: can be reprogrammed in real-time
  - Extremely good at course-grained parallel algorithms
  - Medium development cycle
  - Memory bandwidth problems
- FPGA:
  - Highly flexible, but with a relatively slow clock
  - Provides ASIC functionality at a small fraction of the cost
  - Excellent at fine-grained and time-critical problems
  - Long development cycle; challenging to debug and optimise
  - Not agile: takes a few seconds to switch functionality

Processing Platforms

- CPU:
  - General-purpose; Easily modified; Short development cycle
  - Extensive history $\Rightarrow$ Rich set of matured libraries and APIs
  - Does not handle parallel algorithms particularly well
- GPU:
  - Agile: can be reprogrammed in real-time
  - Extremely good at course-grained parallel algorithms
  - Medium development cycle
  - Memory bandwidth problems
- FPGA:
  - Highly flexible, but with a relatively slow clock
  - Provides ASIC functionality at a small fraction of the cost
  - Excellent at fine-grained and time-critical problems
  - Long development cycle; challenging to debug and optimise
  - Not agile: takes a few seconds to switch functionality

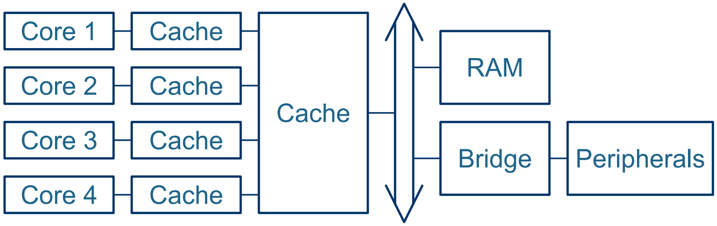
Processing Platforms

- CPU:
  - General-purpose; Easily modified; Short development cycle
  - Extensive history $\Rightarrow$ Rich set of matured libraries and APIs
  - Does not handle parallel algorithms particularly well
- GPU:
  - Agile: can be reprogrammed in real-time
  - Extremely good at course-grained parallel algorithms
  - Medium development cycle
  - Memory bandwidth problems
- FPGA:
  - Highly flexible, but with a relatively slow clock
  - Provides ASIC functionality at a small fraction of the cost
  - Excellent at fine-grained and time-critical problems
  - Long development cycle; challenging to debug and optimise
  - Not agile: takes a few seconds to switch functionality

Processing Platforms

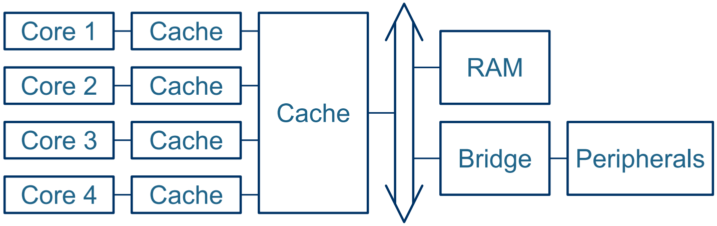
- CPU:
  - General-purpose; Easily modified; Short development cycle
  - Extensive history $\Rightarrow$ Rich set of matured libraries and APIs
  - Does not handle parallel algorithms particularly well
- GPU:
  - Agile: can be reprogrammed in real-time
  - Extremely good at course-grained parallel algorithms
  - Medium development cycle
  - Memory bandwidth problems
- FPGA:
  - Highly flexible, but with a relatively slow clock
  - Provides ASIC functionality at a small fraction of the cost
  - Excellent at fine-grained and time-critical problems
  - Long development cycle; challenging to debug and optimise
  - Not agile: takes a few seconds to switch functionality

Programming Model – CPU



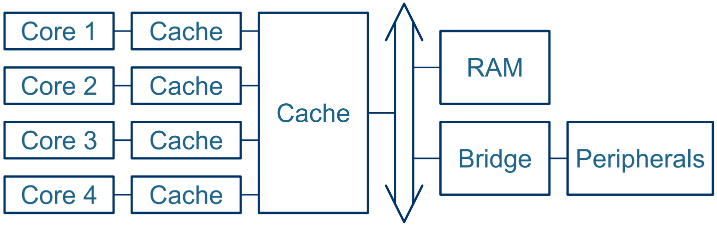
- Fetch-decode-execute cycle – serialised execution
- The RAM is divided into program, stack and heap areas
- All CPU cores share the same RAM, but is cache-assisted to reduce contention

Programming Model – CPU



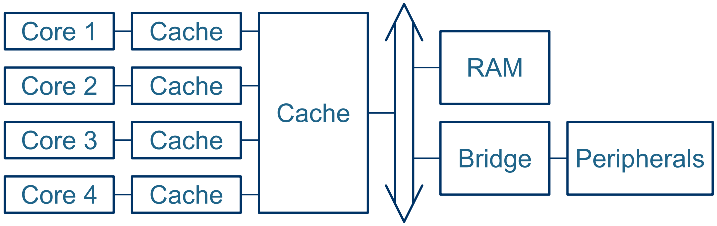
- Fetch-decode-execute cycle – serialised execution
- The RAM is divided into program, stack and heap areas
- All CPU cores share the same RAM, but is cache-assisted to reduce contention

Programming Model – CPU



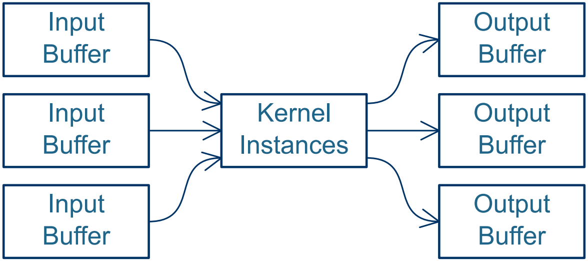
- Fetch-decode-execute cycle – serialised execution
- The RAM is divided into program, stack and heap areas
- All CPU cores share the same RAM, but is cache-assisted to reduce contention

Programming Model – CPU



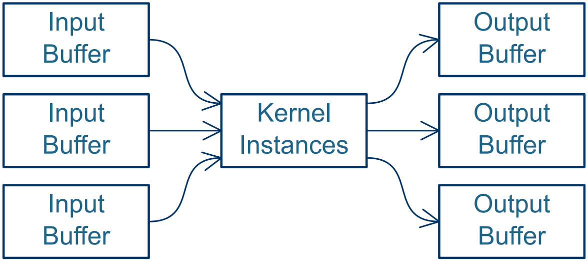
- Fetch-decode-execute cycle – serialised execution
- The RAM is divided into program, stack and heap areas
- All CPU cores share the same RAM, but is cache-assisted to reduce contention

Programming Model – GPU



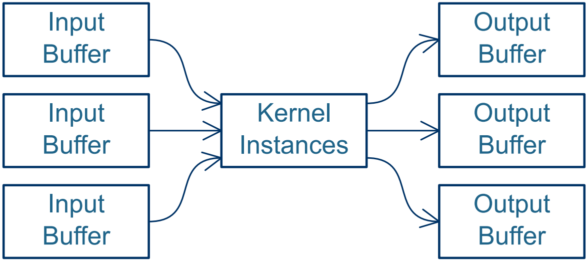
- Client-server interface with the CPU
- Runs a pipeline of kernels, in SIMD operation

Programming Model – GPU



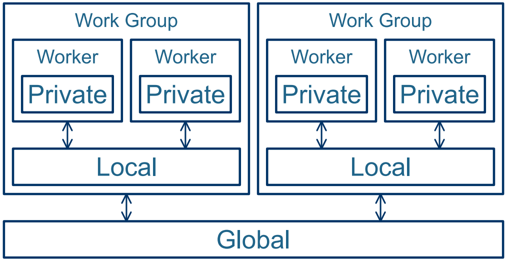
- Client-server interface with the CPU
- Runs a pipeline of kernels, in SIMD operation

Programming Model – GPU



- Client-server interface with the CPU
- Runs a pipeline of kernels, in SIMD operation

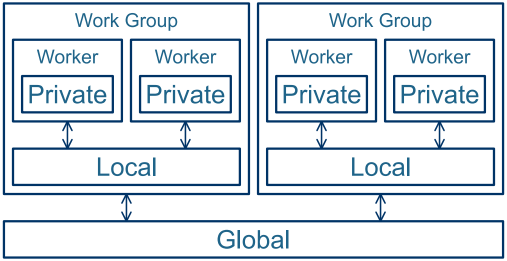
Programming Model – GPU



- The hardware is organised into groups of processors
- Within a group, execution is in lock-step
- Execution within one group is independent of other groups
- Three levels of memory

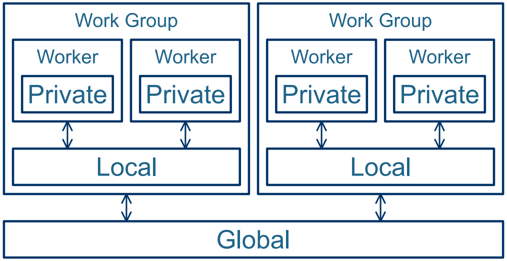
THE RADAR MASTERS COURSE

Programming Model – GPU



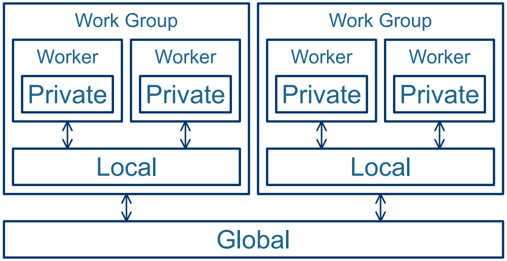
- The hardware is organised into groups of processors
- Within a group, execution is in lock-step
- Execution within one group is independent of other groups
- Three levels of memory

Programming Model – GPU



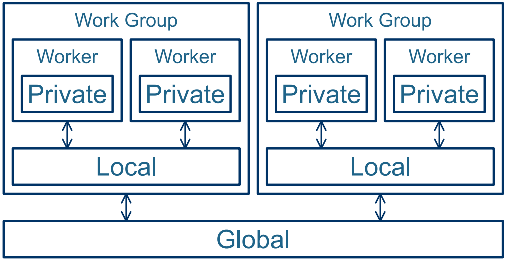
- The hardware is organised into groups of processors
- Within a group, execution is in lock-step
- Execution within one group is independent of other groups
- Three levels of memory

Programming Model – GPU



- The hardware is organised into groups of processors
- Within a group, execution is in lock-step
- Execution within one group is independent of other groups
- Three levels of memory

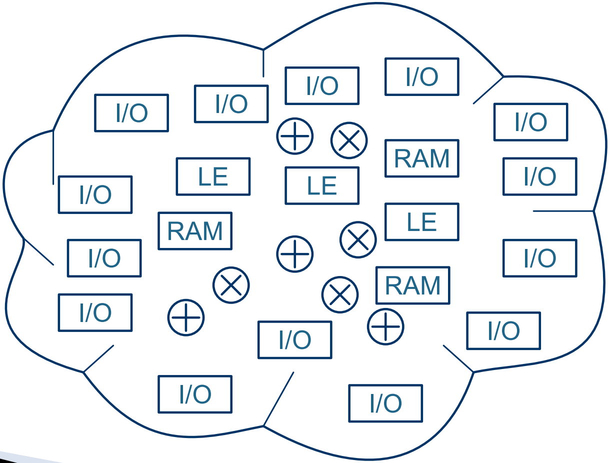
Programming Model – GPU



- The hardware is organised into groups of processors
- Within a group, execution is in lock-step
- Execution within one group is independent of other groups
- Three levels of memory

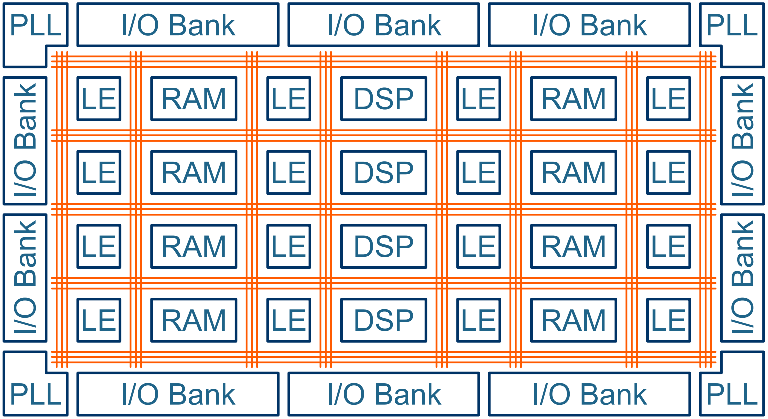
Programming Model – FPGA

- Any architecture you like...

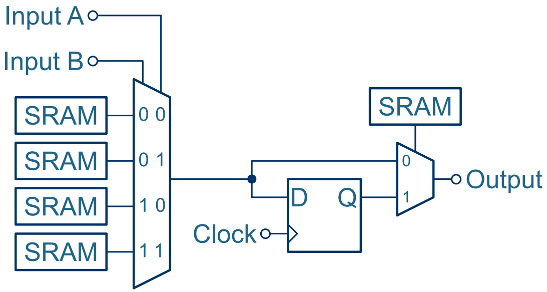


Outline

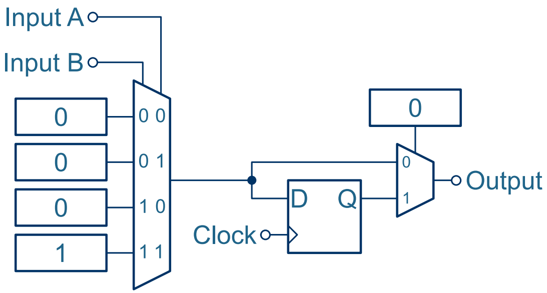
FPGA Architecture Overview



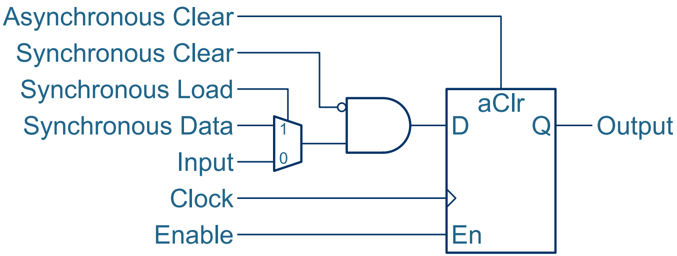
Logic Elements



Logic Elements



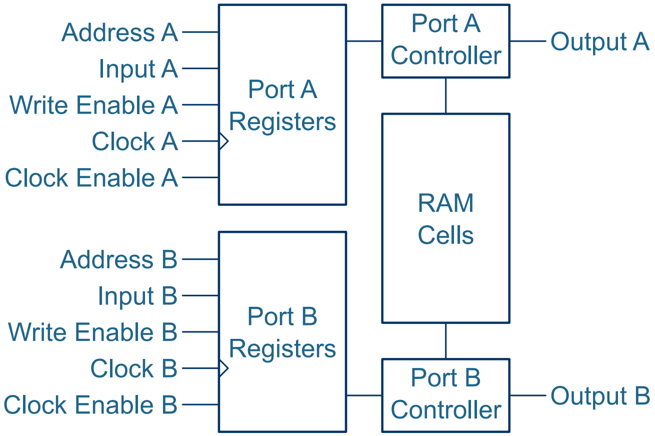
Register Detail



Register Detail

A-Clr	En	S-Clr	S-Ld	S-Dat	In	Clk	Output
1	$\times$	$\times$	$\times$	$\times$	$\times$	$\times$	0
0	1	1	$\times$	$\times$	$\times$	$\uparrow$	0
0	1	0	1	0	$\times$	$\uparrow$	0
0	1	0	1	1	$\times$	$\uparrow$	1
0	1	0	0	$\times$	0	$\uparrow$	0
0	1	0	0	$\times$	1	$\uparrow$	1
0	0	$\times$	$\times$	$\times$	$\times$	$\times$	no-change

Internal RAM Blocks



Internal RAM Blocks

- MAX-10 M9K ports can be configured as:

$$\begin{array}{rcr} 8192 & \times & 1 \\ 4096 & \times & 2 \\ 2048 & \times & 4 \\ 1024 & \times & 8 \\ 1024 & \times & 9 \\ 512 & \times & 16 \\ 512 & \times & 18 \\ 256 & \times & 32 \\ 256 & \times & 36 \end{array}$$

- The two ports can have different configurations
- The two ports can have independent clocks

Internal RAM Blocks

- MAX-10 M9K ports can be configured as:

$$\begin{aligned} 8192 &\times 1 \\ 4096 &\times 2 \\ 2048 &\times 4 \\ 1024 &\times 8 \\ 1024 &\times 9 \\ 512 &\times 16 \\ 512 &\times 18 \\ 256 &\times 32 \\ 256 &\times 36 \end{aligned}$$

- The two ports can have different configurations
- The two ports can have independent clocks

Internal RAM Blocks

- MAX-10 M9K ports can be configured as:

$$\begin{array}{rcr} 8192 & \times & 1 \\ 4096 & \times & 2 \\ 2048 & \times & 4 \\ 1024 & \times & 8 \\ 1024 & \times & 9 \\ 512 & \times & 16 \\ 512 & \times & 18 \\ 256 & \times & 32 \\ 256 & \times & 36 \end{array}$$

- The two ports can have different configurations
- The two ports can have independent clocks

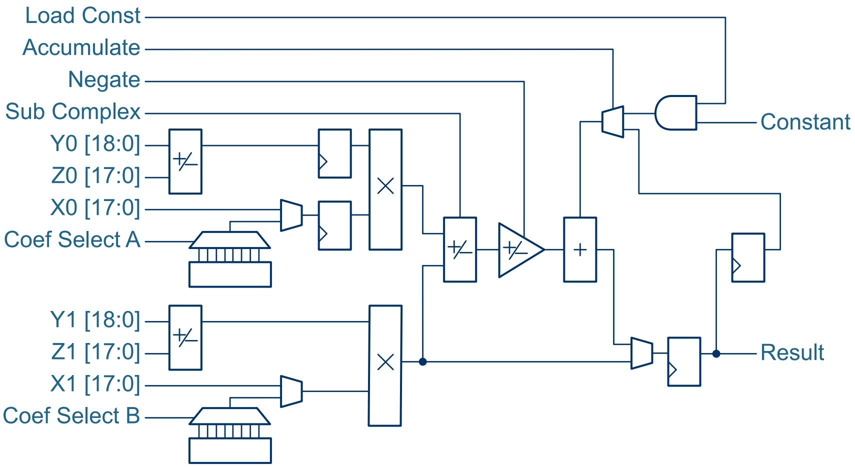
Internal RAM Blocks

- Internal RAM can be initialised by means of a "memory initialisation file" (MIF)

```
WIDTH =    8;
DEPTH = 4096;

ADDRESS_RADIX = HEX;
DATA_RADIX    = HEX;

CONTENT BEGIN
 [000..011]: 00;
  012      : 7E;
  013      : 81;
  014      : A5;
  ...
  FFD      : 00;
  FFE      : FF;
  FFF      : 00;
END;
```

THE RADAR MASTERS COURSE

DSP Blocks


Load Const
Accumulate
Negate
Sub Complex
Y0 [18:0]
Z0 [17:0]
X0 [17:0]
Coef Select A
Y1 [18:0]
Z1 [17:0]
X1 [17:0]
Coef Select B
Constant
Result
+
×

Register Detail

Each Cyclone V DSP block can be configured as:

- Three 9$\times$9-bit multipliers
- Two 18$\times$18-bit multipliers (unsigned)
- Two 18$\times$19-bit multipliers (signed)
- One 18$\times$25-bit multiplier
- One 20$\times$24-bit multiplier
- One 27$\times$27-bit multiplier
- One 18$\times$19-bit multiply-accumulate
- One 18$\times$18-bit multiply-accumulate with adder
- Half of a 18$\times$19-bit complex multiplier

MAX-10 Multipliers

- The MAX-10 does not have DSP blocks – it only has embedded multipliers (no accumulators or adders as part of the block)
- Each multiplier block can be configured as two 9×9-bit multipliers or one 18×18-bit multiplier
- Each input can be configured as signed or unsigned
- The inputs and outputs can optionally be registered inside the multiplier block, which improves timing

MAX-10 Multipliers

- The MAX-10 does not have DSP blocks – it only has embedded multipliers (no accumulators or adders as part of the block)
- Each multiplier block can be configured as
  two 9×9-bit multipliers or
  one 18×18-bit multiplier
- Each input can be configured as signed or unsigned
- The inputs and outputs can optionally be registered inside the multiplier block, which improves timing

MAX-10 Multipliers

- The MAX-10 does not have DSP blocks – it only has embedded multipliers (no accumulators or adders as part of the block)
- Each multiplier block can be configured as
  two 9×9-bit multipliers or
  one 18×18-bit multiplier
- Each input can be configured as signed or unsigned
- The inputs and outputs can optionally be registered inside the multiplier block, which improves timing

MAX-10 Multipliers

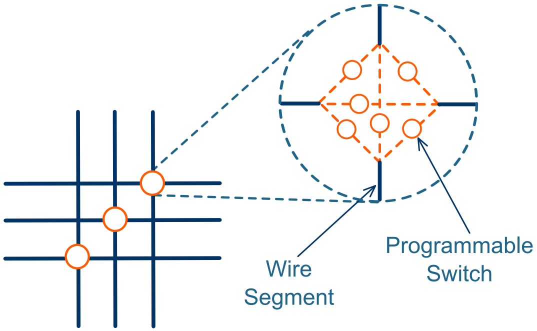
- The MAX-10 does not have DSP blocks – it only has embedded multipliers (no accumulators or adders as part of the block)
- Each multiplier block can be configured as two 9×9-bit multipliers or one 18×18-bit multiplier
- Each input can be configured as signed or unsigned
- The inputs and outputs can optionally be registered inside the multiplier block, which improves timing

MAX-10 Multipliers

- The MAX-10 does not have DSP blocks – it only has embedded multipliers (no accumulators or adders as part of the block)
- Each multiplier block can be configured as
  two 9×9-bit multipliers or
  one 18×18-bit multiplier
- Each input can be configured as signed or unsigned
- The inputs and outputs can optionally be registered inside the multiplier block, which improves timing

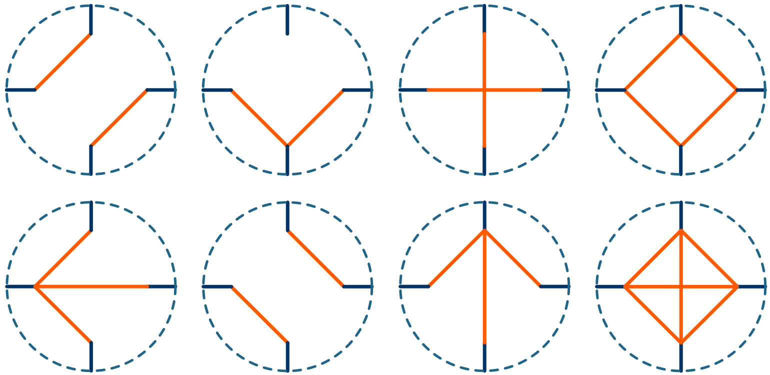
Interconnect

- Each interconnect crossing contains 6 switches
- These switches can be configured in various ways



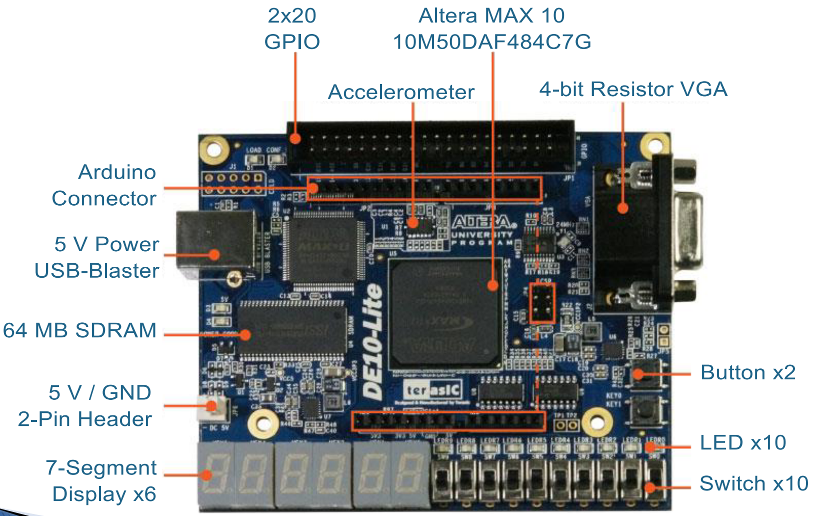
Interconnect

- Each interconnect crossing contains 6 switches
- These switches can be configured in various ways



Outline

Board Outline



FPGA Details

- MAX-10 series (10M50DAF484C7G)
- 2 ADCs (each 12-bit, 1 MSps shared over 9 channels)
- 49 760 logic elements
- 182 M9K memory blocks
- 736 kiB user flash memory
- 144 multiplier blocks
- 4 phase-locked loop blocks
- 360 I/O Pins

FPGA Details

- MAX-10 series (10M50DAF484C7G)
- 2 ADCs (each 12-bit, 1 MSps shared over 9 channels)
- 49 760 logic elements
- 182 M9K memory blocks
- 736 kiB user flash memory
- 144 multiplier blocks
- 4 phase-locked loop blocks
- 360 I/O Pins

FPGA Details

- MAX-10 series (10M50DAF484C7G)
- 2 ADCs (each 12-bit, 1 MSps shared over 9 channels)
- 49 760 logic elements
- 182 M9K memory blocks
- 736 kiB user flash memory
- 144 multiplier blocks
- 4 phase-locked loop blocks
- 360 I/O Pins

FPGA Details

- MAX-10 series (10M50DAF484C7G)
- 2 ADCs (each 12-bit, 1 MSps shared over 9 channels)
- 49 760 logic elements
- 182 M9K memory blocks
- 736 kiB user flash memory
- 144 multiplier blocks
- 4 phase-locked loop blocks
- 360 I/O Pins

FPGA Details

- MAX-10 series (10M50DAF484C7G)
- 2 ADCs (each 12-bit, 1 MSps shared over 9 channels)
- 49 760 logic elements
- 182 M9K memory blocks
- 736 kiB user flash memory
- 144 multiplier blocks
- 4 phase-locked loop blocks
- 360 I/O Pins

FPGA Details

- MAX-10 series (10M50DAF484C7G)
- 2 ADCs (each 12-bit, 1 MSps shared over 9 channels)
- 49 760 logic elements
- 182 M9K memory blocks
- 736 kiB user flash memory
- 144 multiplier blocks
- 4 phase-locked loop blocks
- 360 I/O Pins

FPGA Details

- MAX-10 series (10M50DAF484C7G)
- 2 ADCs (each 12-bit, 1 MSps shared over 9 channels)
- 49 760 logic elements
- 182 M9K memory blocks
- 736 kiB user flash memory
- 144 multiplier blocks
- 4 phase-locked loop blocks
- 360 I/O Pins

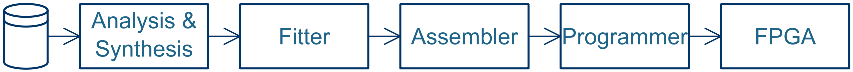
FPGA Details

- MAX-10 series (10M50DAF484C7G)
- 2 ADCs (each 12-bit, 1 MSps shared over 9 channels)
- 49 760 logic elements
- 182 M9K memory blocks
- 736 kiB user flash memory
- 144 multiplier blocks
- 4 phase-locked loop blocks
- 360 I/O Pins

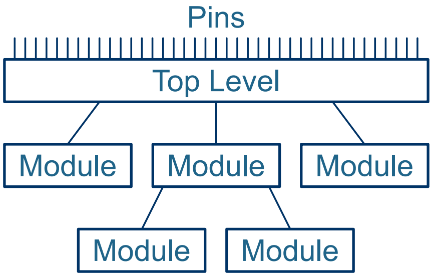
Coffee Break...

Outline

Development Cycle



Design Entry – Modular Design

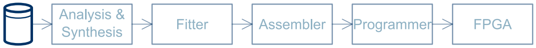


Analysis & Synthesis
Fitter
Assembler
Programmer
FPGA

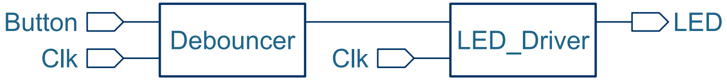
Design Entry – HDL

- Verilog / VHDL / PyHDL / Migen / etc.

```
module Top_Level(input Clk, input Button, output LED);
  wire Debounced_Button;

  Debouncer Debouncer_Inst(
    Clk, Button, Debounced_Button
  );

  LED_Driver LED_Driver_Inst(
    Clk, Debounced_Button, LED
  );
endmodule
```

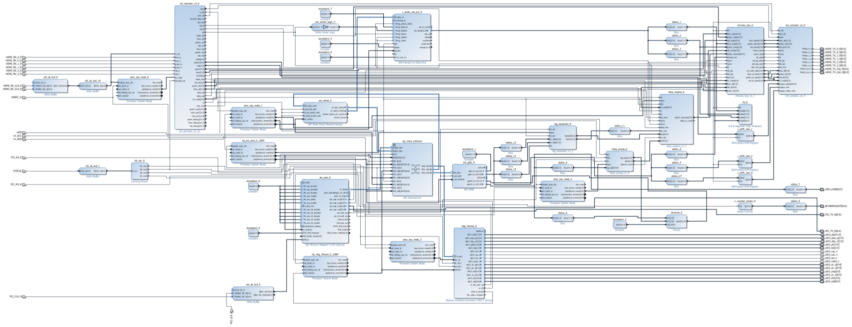


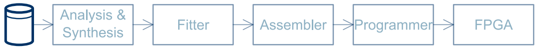
Design Entry – Schematic



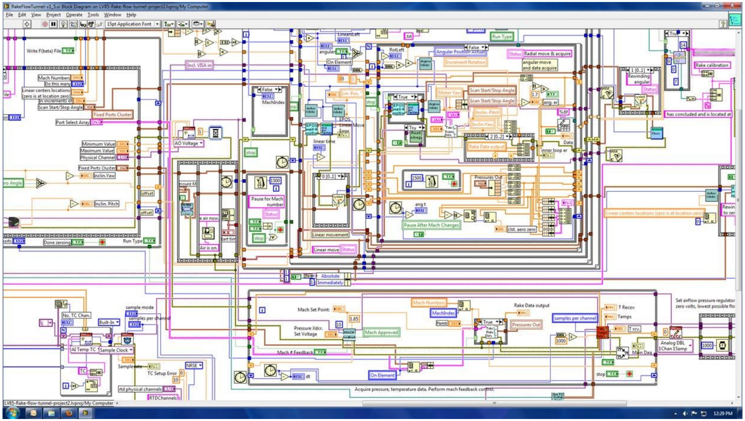
Analysis & Synthesis
Fitter
Assembler
Programmer
FPGA

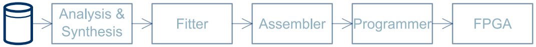
Design Entry – Schematic



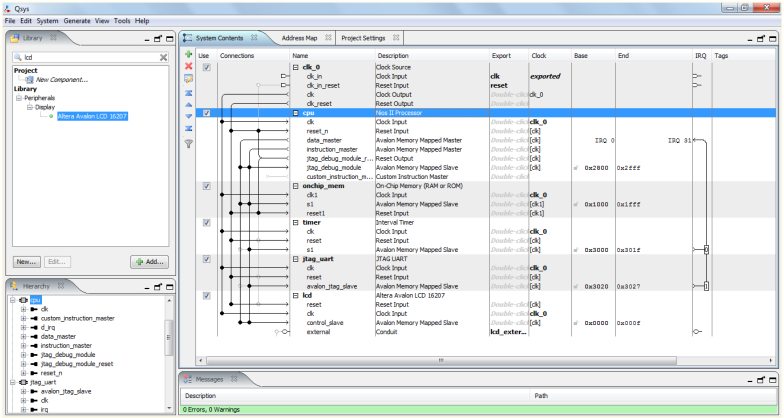


Design Entry – Schematic


RakeFlowTunnel-v1_5.vi Block Diagram on LV85-Rake-flow-tunnel-project2.lvproj/My Computer
File Edit View Project Operate Tools Window Help
Acquire pressure, temperature data. Perform mach feedback control.



Design Entry – Qsys

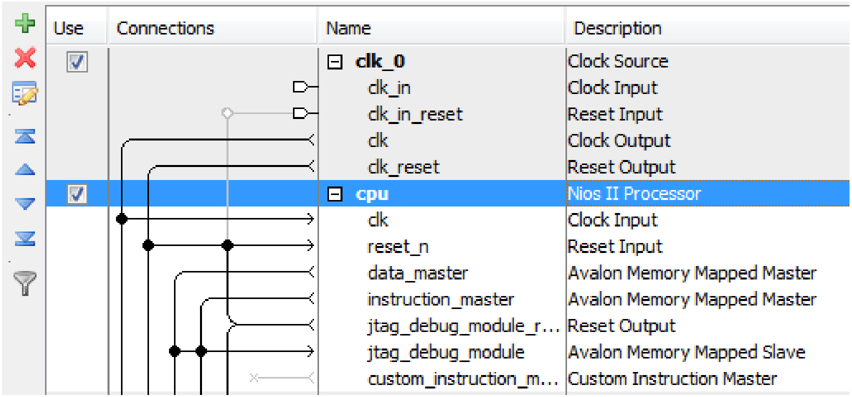


Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags
✓		**clk_0**	Clock Source						
		clk_in	Clock Input	**clk**	***exported***				
		clk_in_reset	Reset Input	**reset**					
		clk	Clock Output	*Double-click*	clk_0				
		clk_reset	Reset Output	*Double-click*					
✓		**cpu**	Nios II Processor						
		clk	Clock Input	*Double-click*	**clk_0**				
		reset_n	Reset Input	*Double-click*	[clk]				
		data_master	Avalon Memory Mapped Master	*Double-click*	[clk]	IRQ 0	IRQ 31		
		instruction_master	Avalon Memory Mapped Master	*Double-click*	[clk]				
		jtag_debug_module_r…	Reset Output	*Double-click*	[clk]				
		jtag_debug_module	Avalon Memory Mapped Slave	*Double-click*	[clk]	0x2800	0x2fff		
		custom_instruction_m…	Custom Instruction Master	*Double-click*					
✓		**onchip_mem**	On-Chip Memory (RAM or ROM)						
		clk1	Clock Input	*Double-click*	**clk_0**				
		s1	Avalon Memory Mapped Slave	*Double-click*	[clk1]	0x1000	0x1fff		
		reset1	Reset Input	*Double-click*	[clk1]				
✓		**timer**	Interval Timer						
		clk	Clock Input	*Double-click*	**clk_0**				
		reset	Reset Input	*Double-click*	[clk]				
		s1	Avalon Memory Mapped Slave	*Double-click*	[clk]	0x3000	0x301f	0	
✓		**jtag_uart**	JTAG UART						
		clk	Clock Input	*Double-click*	**clk_0**				
		reset	Reset Input	*Double-click*	[clk]				
		avalon_jtag_slave	Avalon Memory Mapped Slave	*Double-click*	[clk]	0x3020	0x3027	1	
✓		**lcd**	Altera Avalon LCD 16207						
		reset	Reset Input	*Double-click*	[clk]				
		clk	Clock Input	*Double-click*	**clk_0**				
		control_slave	Avalon Memory Mapped Slave	*Double-click*	[clk]	0x0000	0x000f		
		external	Conduit	**lcd_exter...**					

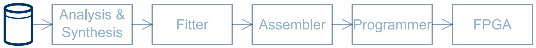
Description	Path
0 Errors, 0 Warnings	

Analysis & Synthesis
Fitter
Assembler
Programmer
FPGA

Design Entry – Qsys

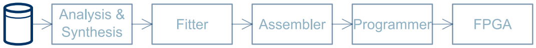


Use	Connections	Name	Description
✓		⊟ clk_0	Clock Source
		clk_in	Clock Input
		clk_in_reset	Reset Input
		clk	Clock Output
		clk_reset	Reset Output
✓		⊟ cpu	Nios II Processor
		clk	Clock Input
		reset_n	Reset Input
		data_master	Avalon Memory Mapped Master
		instruction_master	Avalon Memory Mapped Master
		jtag_debug_module_r...	Reset Output
		jtag_debug_module	Avalon Memory Mapped Slave
		custom_instruction_m...	Custom Instruction Master



Design Entry – HLS

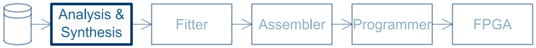
```
void paralleltest(
  bool _doWrite, int _writeAddr, int  _writeData,
  bool _doRead,  int _readAddr,  int* _readData
){
  #pragma HLS INTERFACE ap_ctrl_none port=return
  #pragma HLS PIPELINE II=1
  #pragma HLS DEPENDENCE variable=buffer inter WAR false
  #pragma HLS RESOURCE   variable=buffer core=RAM_2P_BRAM

  static const int BufferSize = 1024;
  static       int buffer[BufferSize];

  if (_doWrite){buffer[_writeAddr % BufferSize] = _writeData;}
  if (_doRead) {*_readData = buffer[_readAddr % BufferSize];}
}
```



Analysis and Synthesis

- Analyse code and perform optimisations (trim dead paths, check connectivity, etc.)
- Synthesise logic tables from expressions
- Match the logic tables and data flow graphs to the target hardware architecture
- Synthesise connection graphs

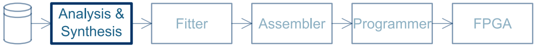
Note: Verilog automatically generates a 1-bit wire for any net that is used without definition, which often happens on typo's and spelling errors. Use the connectivity warnings in the analysis report to find these problems.



Analysis and Synthesis

- Analyse code and perform optimisations (trim dead paths, check connectivity, etc.)
- Synthesise logic tables from expressions
- Match the logic tables and data flow graphs to the target hardware architecture
- Synthesise connection graphs

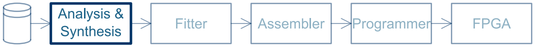
Note: Verilog automatically generates a 1-bit wire for any net that is used without definition, which often happens on typo's and spelling errors. Use the connectivity warnings in the analysis report to find these problems.



Analysis and Synthesis

- Analyse code and perform optimisations (trim dead paths, check connectivity, etc.)
- Synthesise logic tables from expressions
- Match the logic tables and data flow graphs to the target hardware architecture
- Synthesise connection graphs

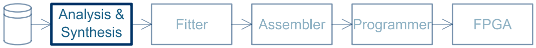
Note: Verilog automatically generates a 1-bit wire for any net that is used without definition, which often happens on typo's and spelling errors. Use the connectivity warnings in the analysis report to find these problems.



Analysis and Synthesis

- Analyse code and perform optimisations (trim dead paths, check connectivity, etc.)
- Synthesise logic tables from expressions
- Match the logic tables and data flow graphs to the target hardware architecture
- Synthesise connection graphs

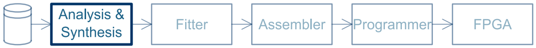
Note: Verilog automatically generates a 1-bit wire for any net that is used without definition, which often happens on typo's and spelling errors. Use the connectivity warnings in the analysis report to find these problems.



Analysis and Synthesis

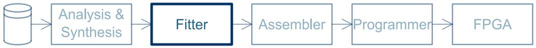
- Analyse code and perform optimisations (trim dead paths, check connectivity, etc.)
- Synthesise logic tables from expressions
- Match the logic tables and data flow graphs to the target hardware architecture
- Synthesise connection graphs

Note: Verilog automatically generates a 1-bit wire for any net that is used without definition, which often happens on typo's and spelling errors. Use the connectivity warnings in the analysis report to find these problems.



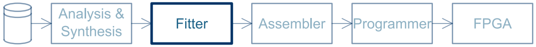
Fitter

- The fitter performs a place and route function (similar to a PCB auto-router)
- This is a computationally-intensive process, so be patient
- Uses design constraints:
  - Clock rate
  - Combinational logic time delay
  - Multi-cycle timing requirements
  - False-path specifications
  - Routing delay
  - External timing requirements
  - User-defined placement
  - etc.



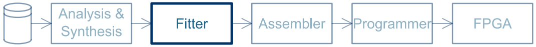
Fitter

- The fitter performs a place and route function (similar to a PCB auto-router)
- This is a computationally-intensive process, so be patient
- Uses design constraints:
  - Clock rate
  - Combinational logic time delay
  - Multi-cycle timing requirements
  - False-path specifications
  - Routing delay
  - External timing requirements
  - User-defined placement
  - etc.



Fitter

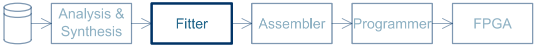
- The fitter performs a place and route function (similar to a PCB auto-router)
- This is a computationally-intensive process, so be patient ...
- Uses design constraints:
  - Clock rate
  - Combinational logic time delay
  - Multi-cycle timing requirements
  - False-path specifications
  - Routing delay
  - External timing requirements
  - User-defined placement
  - etc.



Fitter

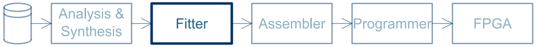
- The fitter performs a place and route function (similar to a PCB auto-router)
- This is a computationally-intensive process, so be patient

  ... ...
- Uses design constraints:
  - Clock rate
  - Combinational logic time delay
  - Multi-cycle timing requirements
  - False-path specifications
  - Routing delay
  - External timing requirements
  - User-defined placement
  - etc.



Fitter

- The fitter performs a place and route function (similar to a PCB auto-router)
- This is a computationally-intensive process, so be patient
  ... ...
- Uses design constraints:
  - Clock rate
  - Combinational logic time delay
  - Multi-cycle timing requirements
  - False-path specifications
  - Routing delay
  - External timing requirements
  - User-defined placement
  - etc.

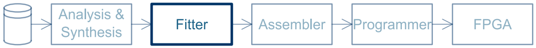


Fitter

- The fitter performs a place and route function (similar to a PCB auto-router)
- This is a computationally-intensive process, so be patient

  ... ...

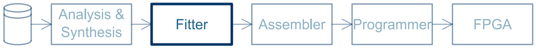
- Uses design constraints:
  - Clock rate
  - Combinational logic time delay
  - Multi-cycle timing requirements
  - False-path specifications
  - Routing delay
  - External timing requirements
  - User-defined placement
  - etc.



Fitter

- The fitter performs a place and route function (similar to a PCB auto-router)
- This is a computationally-intensive process, so be patient

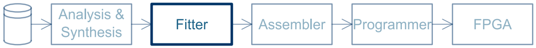
  ... ...
- Uses design constraints:
  - Clock rate
  - Combinational logic time delay
  - Multi-cycle timing requirements
  - False-path specifications
  - Routing delay
  - External timing requirements
  - User-defined placement
  - etc.



Fitter

- The fitter performs a place and route function (similar to a PCB auto-router)
- This is a computationally-intensive process, so be patient

  ... ...
- Uses design constraints:
  - Clock rate
  - Combinational logic time delay
  - Multi-cycle timing requirements
  - False-path specifications
  - Routing delay
  - External timing requirements
  - User-defined placement
  - etc.

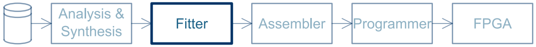


THE RADAR MASTERS COURSE

Fitter

- The fitter performs a place and route function (similar to a PCB auto-router)
- This is a computationally-intensive process, so be patient

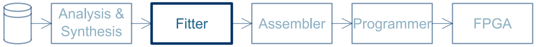
  ... ...
- Uses design constraints:
  - Clock rate
  - Combinational logic time delay
  - Multi-cycle timing requirements
  - False-path specifications
  - Routing delay
  - External timing requirements
  - User-defined placement
  - etc.



Fitter

- The fitter performs a place and route function (similar to a PCB auto-router)
- This is a computationally-intensive process, so be patient

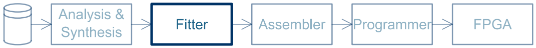
  ... ...
- Uses design constraints:
  - Clock rate
  - Combinational logic time delay
  - Multi-cycle timing requirements
  - False-path specifications
  - Routing delay
  - External timing requirements
  - User-defined placement
  - etc.



Fitter

- The fitter performs a place and route function (similar to a PCB auto-router)
- This is a computationally-intensive process, so be patient

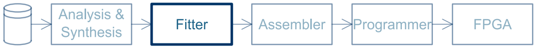
  ... ...
- Uses design constraints:
  - Clock rate
  - Combinational logic time delay
  - Multi-cycle timing requirements
  - False-path specifications
  - Routing delay
  - External timing requirements
  - User-defined placement
  - etc.



Fitter

- The fitter performs a place and route function (similar to a PCB auto-router)
- This is a computationally-intensive process, so be patient

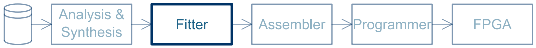
  ... ...
- Uses design constraints:
  - Clock rate
  - Combinational logic time delay
  - Multi-cycle timing requirements
  - False-path specifications
  - Routing delay
  - External timing requirements
  - User-defined placement
  - etc.



Fitter

- The fitter performs a place and route function (similar to a PCB auto-router)
- This is a computationally-intensive process, so be patient

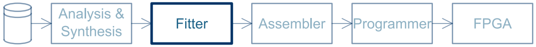
  ... ...
- Uses design constraints:
  - Clock rate
  - Combinational logic time delay
  - Multi-cycle timing requirements
  - False-path specifications
  - Routing delay
  - External timing requirements
  - User-defined placement
  - etc.



Fitter

- The fitter performs a place and route function (similar to a PCB auto-router)
- This is a computationally-intensive process, so be patient

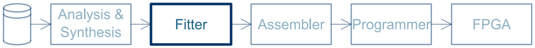
  ... ...
- Uses design constraints:
  - Clock rate
  - Combinational logic time delay
  - Multi-cycle timing requirements
  - False-path specifications
  - Routing delay
  - External timing requirements
  - User-defined placement
  - etc.



Fitter

- The fitter performs a place and route function (similar to a PCB auto-router)
- This is a computationally-intensive process, so be patient

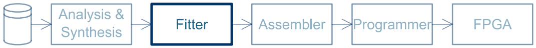
  … … … … … …
- Uses design constraints:
  - Clock rate
  - Combinational logic time delay
  - Multi-cycle timing requirements
  - False-path specifications
  - Routing delay
  - External timing requirements
  - User-defined placement
  - etc.



Fitter

- The fitter performs a place and route function (similar to a PCB auto-router)
- This is a computationally-intensive process, so be patient

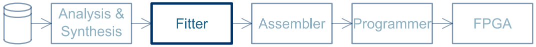
  ... ...
- Uses design constraints:
  - Clock rate
  - Combinational logic time delay
  - Multi-cycle timing requirements
  - False-path specifications
  - Routing delay
  - External timing requirements
  - User-defined placement
  - etc.



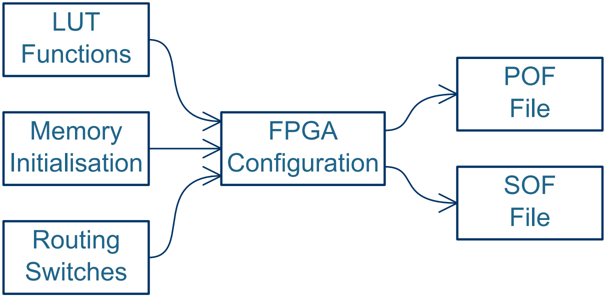
Fitter

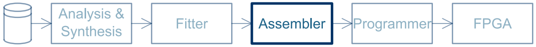
- The fitter performs a place and route function (similar to a PCB auto-router)
- This is a computationally-intensive process, so be patient

  ... ...
- Uses design constraints:
  - Clock rate
  - Combinational logic time delay
  - Multi-cycle timing requirements
  - False-path specifications
  - Routing delay
  - External timing requirements
  - User-defined placement
  - etc.

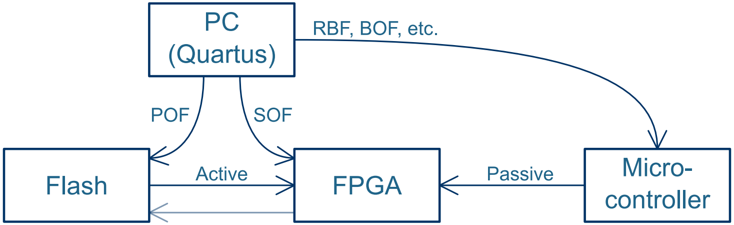


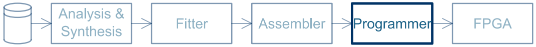
Assembler


LUT Functions
Memory Initialisation
Routing Switches
FPGA Configuration
POF File
SOF File

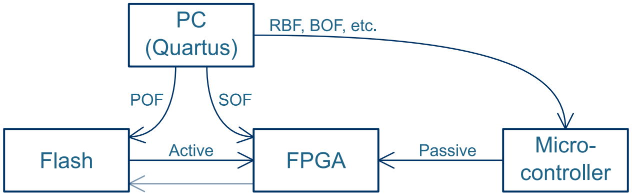

Analysis & Synthesis
Fitter
Assembler
Programmer
FPGA

Programming Architectures





Programming Architectures

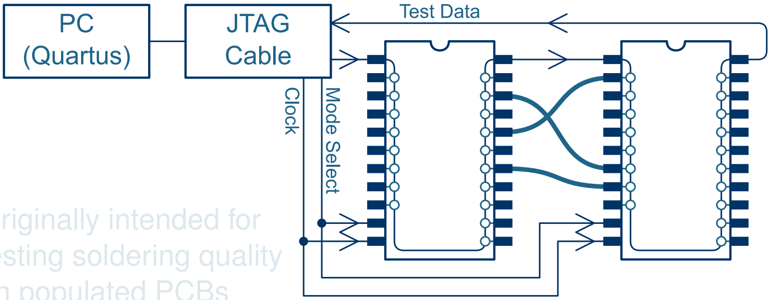


- SOF ⇒ SRAM Object File
- POF ⇒ Programmer Object File
- RBF ⇒ Raw Binary File
- BOF ⇒ Borph Object File

Analysis &
Synthesis
Fitter
Assembler
Programmer
FPGA

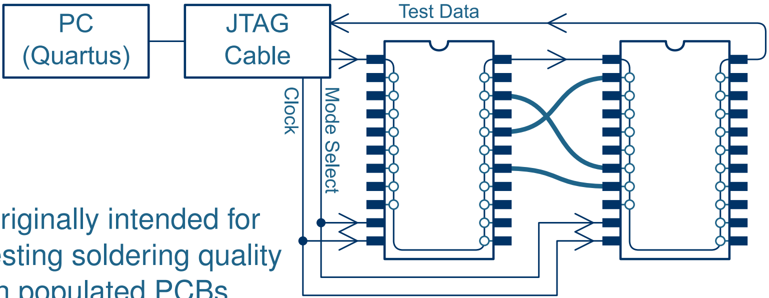
Outline

JTAG – Joint Test Action Group



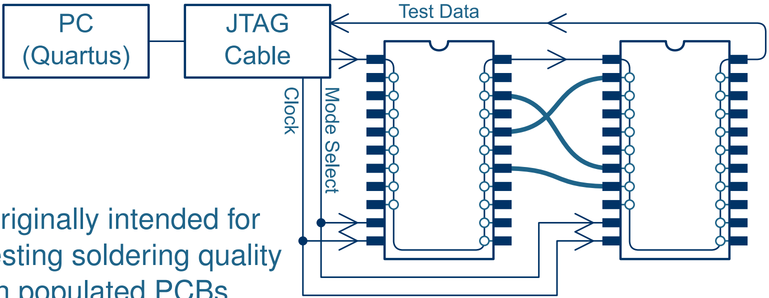
- Originally intended for testing soldering quality on populated PCBs
- Connects to the PC over USB / Ethernet / etc.
- Connected devices form a long chain of shift-registers

JTAG – Joint Test Action Group



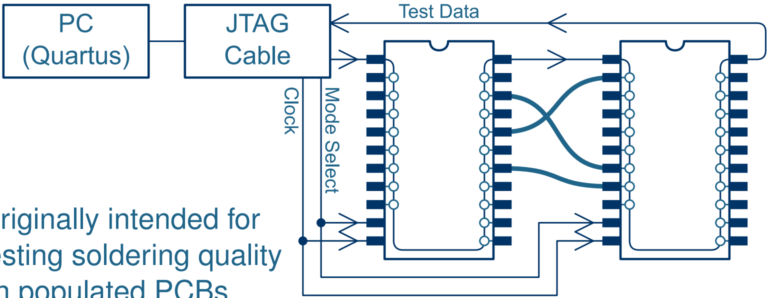
- Originally intended for testing soldering quality on populated PCBs
- Connects to the PC over USB / Ethernet / etc.
- Connected devices form a long chain of shift-registers

JTAG – Joint Test Action Group



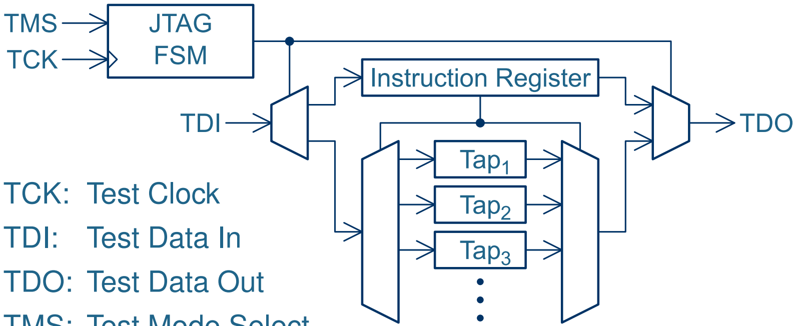
- Originally intended for testing soldering quality on populated PCBs
- Connects to the PC over USB / Ethernet / etc.
- Connected devices form a long chain of shift-registers

JTAG – Joint Test Action Group



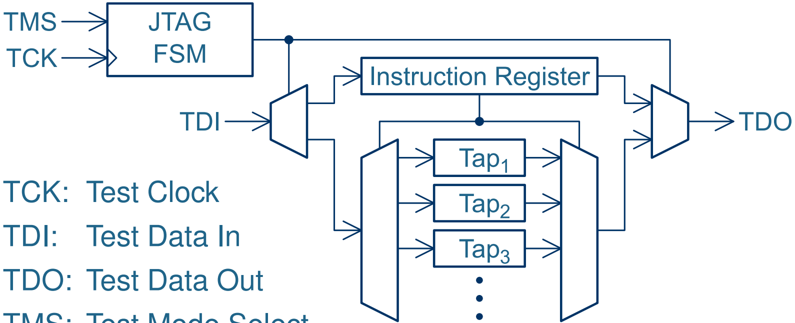
- Originally intended for testing soldering quality on populated PCBs
- Connects to the PC over USB / Ethernet / etc.
- Connected devices form a long chain of shift-registers

JTAG – Joint Test Action Group



- TCK: Test Clock
- TDI: Test Data In
- TDO: Test Data Out
- TMS: Test Mode Select
- Taps could include: Device pins; Internal flash; Status registers; Debug registers; Virtual JTAG interface; etc.

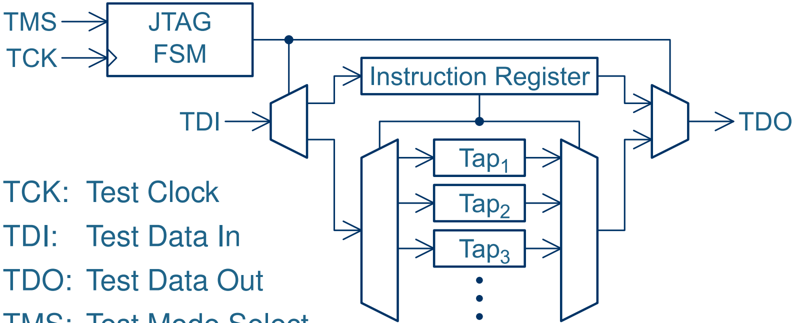
JTAG – Joint Test Action Group



- TCK: Test Clock
- TDI: Test Data In
- TDO: Test Data Out
- TMS: Test Mode Select
- Taps could include: Device pins; Internal flash; Status registers; Debug registers; Virtual JTAG interface; etc.

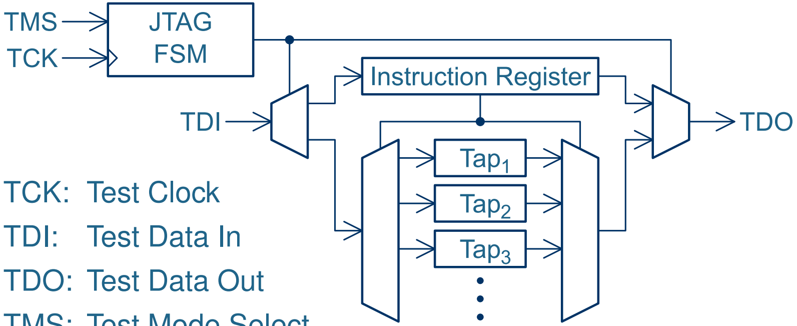
THE RADAR MASTERS COURSE

JTAG – Joint Test Action Group



- TCK: Test Clock
- TDI: Test Data In
- TDO: Test Data Out
- TMS: Test Mode Select
- Taps could include: Device pins; Internal flash; Status registers; Debug registers; Virtual JTAG interface; etc.

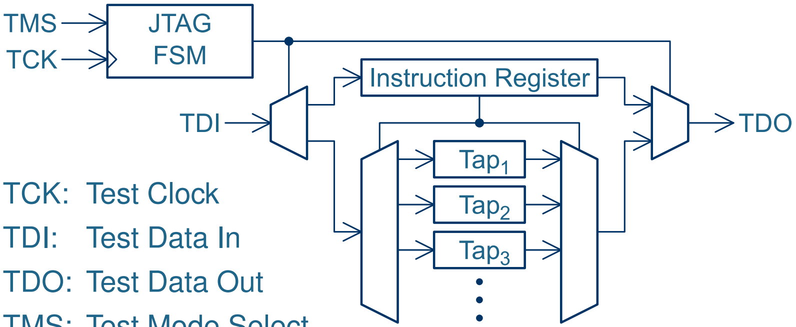
JTAG – Joint Test Action Group



- TCK: Test Clock
- TDI: Test Data In
- TDO: Test Data Out
- TMS: Test Mode Select
- Taps could include: Device pins; Internal flash; Status registers; Debug registers; Virtual JTAG interface; etc.

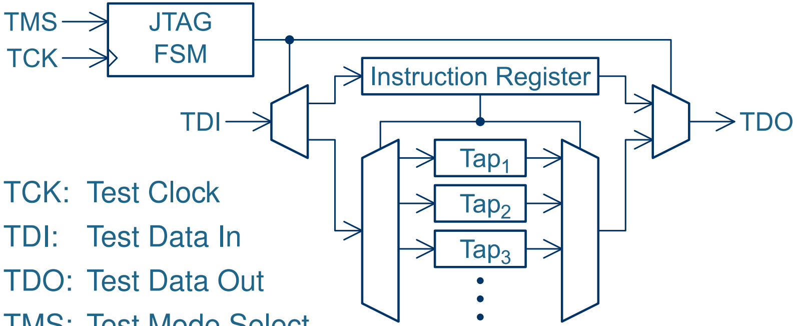
THE RADAR MASTERS COURSE

JTAG – Joint Test Action Group



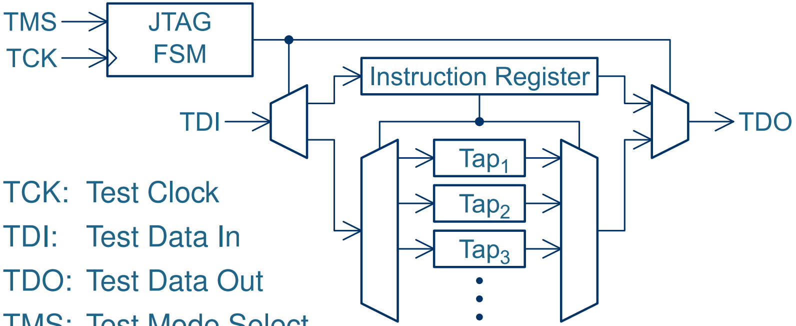
- TCK: Test Clock
- TDI: Test Data In
- TDO: Test Data Out
- TMS: Test Mode Select
- Taps could include: Device pins; Internal flash; Status registers; Debug registers; Virtual JTAG interface; etc.

JTAG – Joint Test Action Group



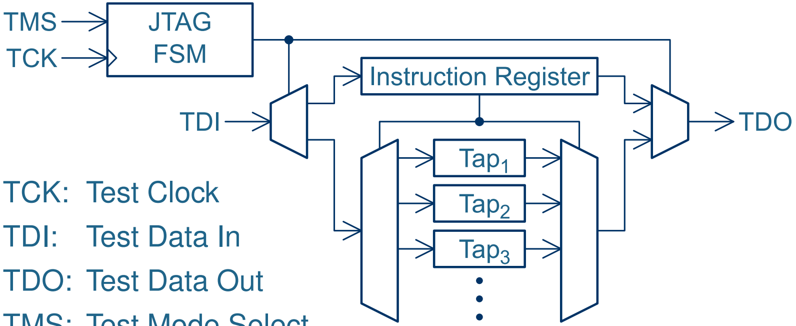
- TCK: Test Clock
- TDI: Test Data In
- TDO: Test Data Out
- TMS: Test Mode Select
- Taps could include: Device pins; Internal flash; Status registers; Debug registers; Virtual JTAG interface; etc.

JTAG – Joint Test Action Group



- TCK: Test Clock
- TDI: Test Data In
- TDO: Test Data Out
- TMS: Test Mode Select
- Taps could include: Device pins; Internal flash; Status registers; Debug registers; Virtual JTAG interface; etc.

JTAG – Joint Test Action Group



- TCK: Test Clock
- TDI: Test Data In
- TDO: Test Data Out
- TMS: Test Mode Select
- Taps could include: Device pins; Internal flash; Status registers; Debug registers; Virtual JTAG interface; etc.

JTAG Debugging

Quartus includes powerful JTAG-based debugging tools – find online tutorials and play around with:

- Sources & Probes (also in today's practical)
  `https://www.youtube.com/watch?v=Mftgi318Nrc`
- In-system memory editor
  `https://www.youtube.com/watch?v=_VcVtFvJnuY`
- Signal-tap logic analyser
  `https://www.youtube.com/watch?v=vhkzxCEXuaA`

JTAG Debugging

Quartus includes powerful JTAG-based debugging tools – find online tutorials and play around with:

- Sources & Probes (also in today's practical)
  https://www.youtube.com/watch?v=Mftgi318Nrc
- In-system memory editor
  https://www.youtube.com/watch?v=_VcVtFvJnuY
- Signal-tap logic analyser
  https://www.youtube.com/watch?v=vhkzxCEXuaA

JTAG Debugging

Quartus includes powerful JTAG-based debugging tools – find online tutorials and play around with:

- Sources & Probes (also in today's practical)
  `https://www.youtube.com/watch?v=Mftgi318Nrc`
- In-system memory editor
  `https://www.youtube.com/watch?v=_VcVtFvJnuY`
- Signal-tap logic analyser
  `https://www.youtube.com/watch?v=vhkzxCEXuaA`

JTAG Debugging

Quartus includes powerful JTAG-based debugging tools – find online tutorials and play around with:

- Sources & Probes (also in today's practical)
  `https://www.youtube.com/watch?v=Mftgi318Nrc`
- In-system memory editor
  `https://www.youtube.com/watch?v=_VcVtFvJnuY`
- Signal-tap logic analyser
  `https://www.youtube.com/watch?v=vhkzxCEXuaA`

Coffee Break...

Outline

Module Definition

```
/* Use comments to describe the function...
   Multiple lines are possible */

module MyModule(
  input         Clock, // Try to be consistent
  input         Reset, // on port placement

  input  [ 7:0]Input,  // Note that Verilog is
  output [ 9:0]Output, // case sensitive
  inout  [12:0]Bidirectional
);

// The module body goes here...

endmodule
```

Wires

```
wire      A;       // A single-bit wire
wire [7:0]B;       // An 8-bit wire
wire [7:0]C[5:0]; // A 6-element array of 8-bit wires

wire [7:0]X = B + C[3]; // Wire definition and
                        // assignment in one
```

- Wires are internal connections
- See them as wires on a bread-board

Wires

```
wire      A;        // A single-bit wire
wire [7:0]B;        // An 8-bit wire
wire [7:0]C[5:0];   // A 6-element array of 8-bit wires

wire [7:0]X = B + C[3]; // Wire definition and
                        // assignment in one
```

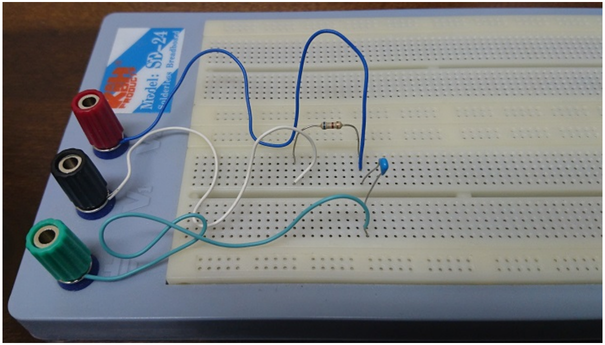
- Wires are internal connections
- See them as wires on a bread-board

Wires

```
wire      A;        // A single-bit wire
wire [7:0]B;        // An 8-bit wire
wire [7:0]C[5:0]; // A 6-element array of 8-bit wires

wire [7:0]X = B + C[3]; // Wire definition and
                        // assignment in one
```

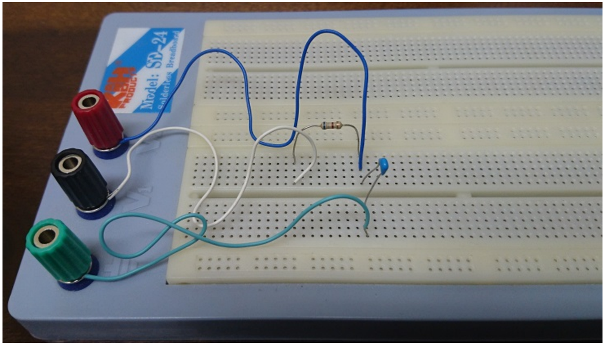
- Wires are internal connections
- See them as wires on a bread-board

Wires



```
module PWM_Filter(input V1, output V2, input GND);
 wire Blue; wire White = V1; wire Cyan = GND;
 Resistor  #(  680) R1(White, Blue);
 Capacitor #(10000) C1(Blue , Cyan);
 assign V2 = Blue;
endmodule
```

Wires



```
module PWM_Filter(input V1, output V2, input GND);
 Resistor  #(  680) R1(V1, V2 );
 Capacitor #(10000) C1(V2, GND);
endmodule
```

Operators – Reduction

```
wire [7:0] X, Y, Z; // Defines 3x 8-bit wires
wire       A, B, C; // Defines 3x 1-bit wires

assign A = &X; // AND-reduce X and assign to A
assign B = |Y; //  OR-reduce Y and assign to B
assign C = ^Z; // XOR-reduce Z and assign to C
assign B = !Y; // Logical NOT: equivalent to B = ~|Y
```

Operators – Bitwise

```
wire [7:0] X, Y, Z; // Defines 3x 8-bit wires

assign Z = -X;    // 2's complement X and assign to Z
assign Z = ~Y;    // Bitwise NOT Y and assign to Z
assign Z = X | Y; // Bitwise OR  X with Y and assign to Z
assign Z = X & Y; // Bitwise AND X with Y and assign to Z
assign Z = X ^ Y; // Bitwise XOR X with Y and assign to Z
```

Operators – Concatenation

```
wire [7:0] A;
wire [3:0] B, C;

assign A = {B, C}; // Concatenates B--C and assign to A
assign {B, C} = A; // Assign A to the concatenation B--C
assign A = {2{B}}; // Replicate B 2 times and assign to A
```

Operators – Arithmetic

```
wire [ 7:0] A, B, C;
wire [15:0] X;

assign A = B + C; // Add C to B and assign to A
assign A = B - C; // Subtract C from B and assign to A
assign X = B * C; // Multiply B by C and assign to X

assign A = B << 5; // Left-shift B and assign to A
assign A = B >> 6; // Right-shift B and assign to A
```

Operators – Logical

```
wire [ 7:0]A, B, C;
wire       X;

assign X = A >  B; // A greater than B?          Result to X
assign X = A <  B; // A less than B?             Result to X
assign X = A >= B; // A greater or equal to B? Result to X
assign X = A <= B; // A less or equal to B?     Result to X
assign X = A == B; // A equal to B?  A          Result to X
assign X = A != B; // A not equal to B?          Result to X

assign X = A && B; // Equivalent to X = (|A) & (|B)
assign X = A || B; // Equivalent to X = (|A) | (|B)

assign C = X ? A : B // If X is 1, assign A to C,
                     // otherwise assign B to C
```

Operators – Signed Operations

- All operations are unsigned, unless specified otherwise
- The following always yield unsigned results:
  - Any operation with at least one unsigned operand
  - A literal without the ‘s’ modifier
  - Bit-select (eg. `A[5]`)
  - Part-select (eg. `A[5:3]`)
  - Concatenations

```
wire        [ 7:0]A; // Unsigned vector
wire signed [ 7:0]B; //   Signed vector
wire signed [15:0]X; //   Signed vector

assign X = $signed(A) * B; // A must be cast
```

Operators – Signed Operations

- All operations are unsigned, unless specified otherwise
- The following always yield unsigned results:
  - Any operation with at least one unsigned operand
  - A literal without the 's' modifier
  - Bit-select (eg. `A[5]`)
  - Part-select (eg. `A[5:3]`)
  - Concatenations

```
wire        [ 7:0]A; // Unsigned vector
wire signed [ 7:0]B; //   Signed vector
wire signed [15:0]X; //   Signed vector

assign X = $signed(A) * B; // A must be cast
```

Operators – Signed Operations

- All operations are unsigned, unless specified otherwise
- The following always yield unsigned results:
  - Any operation with at least one unsigned operand
  - A literal without the ‘s’ modifier
  - Bit-select (eg. A[5])
  - Part-select (eg. A[5:3])
  - Concatenations

```
wire        [ 7:0]A; // Unsigned vector
wire signed [ 7:0]B; //   Signed vector
wire signed [15:0]X; //   Signed vector

assign X = $signed(A) * B; // A must be cast
```

Operators – Signed Operations

- All operations are unsigned, unless specified otherwise
- The following always yield unsigned results:
  - Any operation with at least one unsigned operand
  - A literal without the ‘s’ modifier
  - Bit-select (eg. `A[5]`)
  - Part-select (eg. `A[5:3]`)
  - Concatenations

```
wire        [ 7:0]A; // Unsigned vector
wire signed [ 7:0]B; //   Signed vector
wire signed [15:0]X; //   Signed vector

assign X = $signed(A) * B; // A must be cast
```

Operators – Signed Operations

- All operations are unsigned, unless specified otherwise
- The following always yield unsigned results:
  - Any operation with at least one unsigned operand
  - A literal without the ‘s’ modifier
  - Bit-select (eg. `A[5]`)
  - Part-select (eg. `A[5:3]`)
  - Concatenations

```
wire        [ 7:0]A; // Unsigned vector
wire signed [ 7:0]B; //   Signed vector
wire signed [15:0]X; //   Signed vector

assign X = $signed(A) * B; // A must be cast
```

Operators – Signed Operations

- All operations are unsigned, unless specified otherwise
- The following always yield unsigned results:
  - Any operation with at least one unsigned operand
  - A literal without the 's' modifier
  - Bit-select (eg. `A[5]`)
  - Part-select (eg. `A[5:3]`)
  - Concatenations

```
wire        [ 7:0]A; // Unsigned vector
wire signed [ 7:0]B; //   Signed vector
wire signed [15:0]X; //   Signed vector

assign X = $signed(A) * B; // A must be cast
```

Operators – Signed Operations

- All operations are unsigned, unless specified otherwise
- The following always yield unsigned results:
  - Any operation with at least one unsigned operand
  - A literal without the ‘s’ modifier
  - Bit-select (eg. `A[5]`)
  - Part-select (eg. `A[5:3]`)
  - Concatenations

```
wire        [ 7:0]A; // Unsigned vector
wire signed [ 7:0]B; //   Signed vector
wire signed [15:0]X; //   Signed vector

assign X = $signed(A) * B; // A must be cast
```

Operators – Signed Operations

- In general, most operations should be done using standard unsigned arithmetic
- In the rare cases where the sign makes a difference (multiplication and relational statements), cast explicitly

```
wire [ 7:0]A;
wire [ 7:0]B;
wire [15:0]X;

assign X = $signed(A) * $signed(B);

PWM PWM1(Clk, {~X[15], X[14:0]}, PWM_Output);
```

Operators – Signed Operations

- In general, most operations should be done using standard unsigned arithmetic
- In the rare cases where the sign makes a difference (multiplication and relational statements), cast explicitly

```
wire [ 7:0]A;
wire [ 7:0]B;
wire [15:0]X;

assign X = $signed(A) * $signed(B);

PWM PWM1(Clk, {~X[15], X[14:0]}, PWM_Output);
```

Vector Lengths

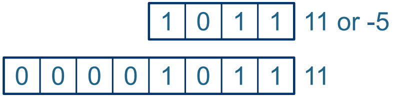
Note: Verilog does not enforce vector length matching:

- If the left-hand-side is longer than the right-hand-side, the most-significant side is padded with zeros
- If the left-hand-side is shorter than the right-hand-side, the most-significant side is truncated
- **This is done without warning!**
- Consult the “Connectivity Checks” report of the Analysis and Synthesis compilation stage to check for unintended size-mismatches

Vector Lengths

Note: Verilog does not enforce vector length matching:

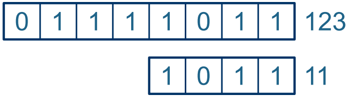
- If the left-hand-side is longer than the right-hand-side, the most-significant side is padded with zeros
- If the left-hand-side is shorter than the right-hand-side, the most-significant side is truncated
- **This is done without warning!**
- Consult the “Connectivity Checks” report of the Analysis and Synthesis compilation stage to check for unintended size-mismatches



Vector Lengths

Note: Verilog does not enforce vector length matching:

- If the left-hand-side is longer than the right-hand-side, the most-significant side is padded with zeros
- If the left-hand-side is shorter than the right-hand-side, the most-significant side is truncated
- **This is done without warning!**
- Consult the “Connectivity Checks” report of the Analysis and Synthesis compilation stage to check for unintended size-mismatches



Vector Lengths

Note: Verilog does not enforce vector length matching:

- If the left-hand-side is longer than the right-hand-side, the most-significant side is padded with zeros
- If the left-hand-side is shorter than the right-hand-side, the most-significant side is truncated
- **This is done without warning!**
- Consult the "Connectivity Checks" report of the Analysis and Synthesis compilation stage to check for unintended size-mismatches

Vector Lengths

Note: Verilog does not enforce vector length matching:

- If the left-hand-side is longer than the right-hand-side, the most-significant side is padded with zeros
- If the left-hand-side is shorter than the right-hand-side, the most-significant side is truncated
- **This is done without warning!**
- Consult the “Connectivity Checks” report of the Analysis and Synthesis compilation stage to check for unintended size-mismatches

Literals

```
// All underscores "_" are ignored
 5'b_1_0110  //  5-bit binary
11'h_5_CE    // 11-bit hexadecimal
13'o_1_7642  // 13-bit octal
17'd_123_456 // 17-bit decimal

"H" // 8-bit ASCII constant

// Left bits (most significant) are padded with zeros,
// unless the most significant specified is 'Z' or 'X'
78'bZ // 78-bit high-impedance

// Use the 's' specifier for "signed" literals
-8'sd125 // 2's complement of the positive
         // signed decimal constant 125
```

Module Instances

```
module My_Submodule(input Clk, input A, input B, output X);
 // Module body...
endmodule

module Top_Level(
 input  Clock_50MHz,
 input  Input1, Input2,
 output Output
);

// Positional port mapping:
My_Submodule Instance_1(
 Clock_50MHz,
 Input1, Input2,
 Output
);
```

Module Instances

```
module My_Submodule(input Clk, input A, input B, output X);
 // Module body...
endmodule

module Top_Level(
 input  Clock_50MHz,
 input  Input1, Input2,
 output Output
);

// Named port mapping:
My_Submodule Instance_2(
 .X  (Output     ),
 .A  (Input1     ),
 .B  (Input2     ),
 .Clk(Clock_50MHz)
);
```

Outline

Overview

- Creating and setting up a new project (from scratch)
- Compiling the design and programming the device
- JTAG-based debugging tools
- **Note:** The White Lab is re-imaged daily, so remember to save your project somewhere else at the end of the day

Overview

- Creating and setting up a new project (from scratch)
- Compiling the design and programming the device
- JTAG-based debugging tools
- **Note:** The White Lab is re-imaged daily, so remember to save your project somewhere else at the end of the day

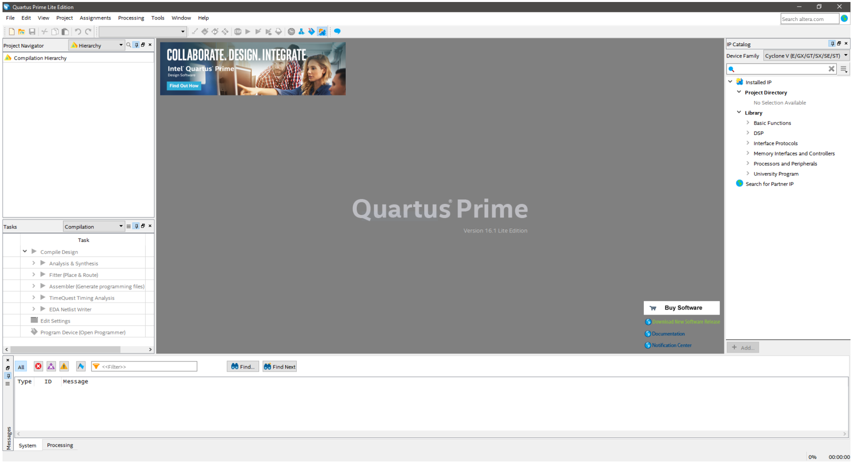
Overview

- Creating and setting up a new project (from scratch)
- Compiling the design and programming the device
- JTAG-based debugging tools
- **Note:** The White Lab is re-imaged daily, so remember to save your project somewhere else at the end of the day

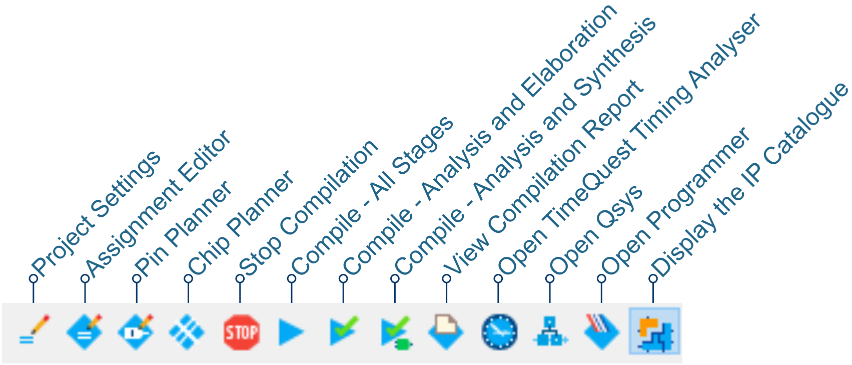
Overview

- Creating and setting up a new project (from scratch)
- Compiling the design and programming the device
- JTAG-based debugging tools
- **Note:** The White Lab is re-imaged daily, so remember to save your project somewhere else at the end of the day

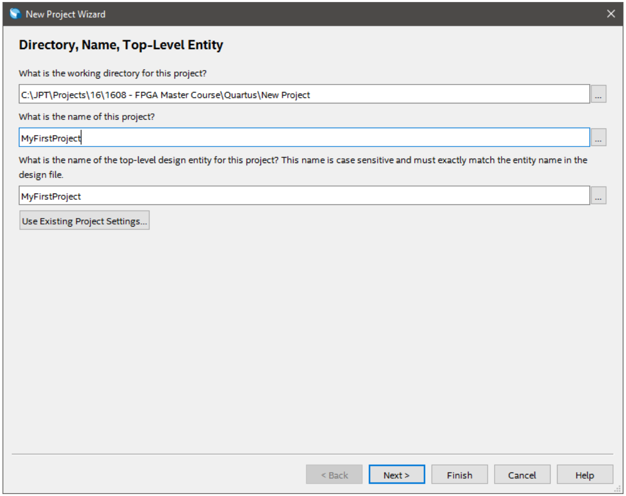
IDE Layout


Quartus Prime Lite Edition
File Edit View Project Assignments Processing Tools Window Help
Search altera.com
Project Navigator
Hierarchy
Compilation Hierarchy
COLLABORATE. DESIGN. INTEGRATE
Intel Quartus Prime
Design Software
Find Out How
Quartus Prime
Version 16.1 Lite Edition
IP Catalog
Device Family
Cyclone V (E/GX/GT/SX/SE/ST)
Installed IP
Project Directory
No Selection Available
Library
Basic Functions
DSP
Interface Protocols
Memory Interfaces and Controllers
Processors and Peripherals
University Program
Search for Partner IP
Tasks
Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files)
TimeQuest Timing Analysis
EDA Netlist Writer
Edit Settings
Program Device (Open Programmer)
Buy Software
Download New Software Release
Documentation
Notification Center
Add...
All
<<Filter>>
Find...
Find Next
Type
ID
Message
Messages
System
Processing
0%
00:00:00

Toolbar


Project Settings
Assignment Editor
Pin Planner
Chip Planner
Stop Compilation
Compile - All Stages
Compile - Analysis and Elaboration
Compile - Analysis and Synthesis
View Compilation Report
Open TimeQuest Timing Analyser
Open Qsys
Open Programmer
Display the IP Catalogue
STOP

New Project Wizard



Directory, Name, Top-Level Entity

What is the working directory for this project?

C:\JPT\Projects\16\1608 - FPGA Master Course\Quartus\New Project

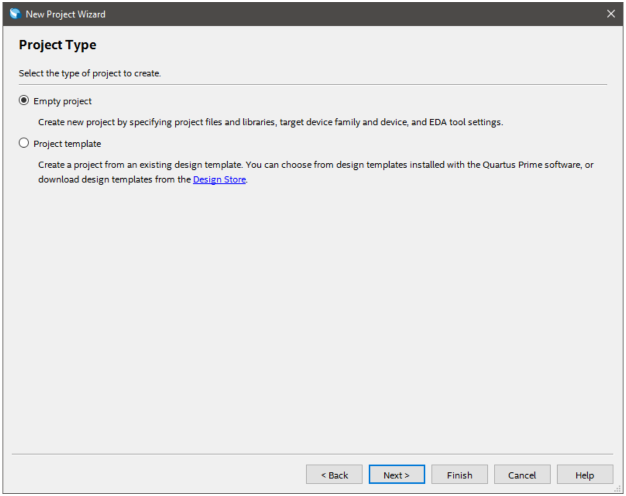
What is the name of this project?

MyFirstProject

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.

MyFirstProject

Project Type



Project Type

Select the type of project to create.

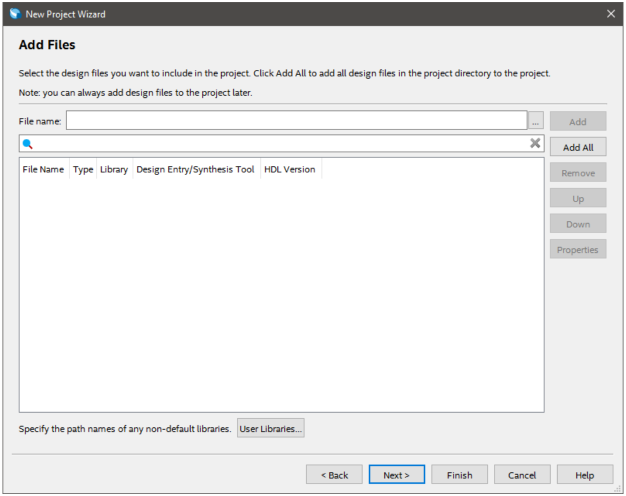
[✓ Empty project]

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 Existing Files



New Project Wizard

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.

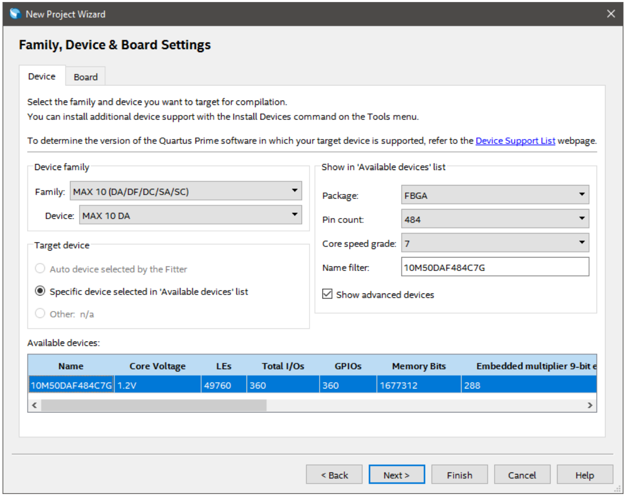
Note: you can always add design files to the project later.

File name:

File Name	Type	Library	Design Entry/Synthesis Tool	HDL Version

Specify the path names of any non-default libraries.

Choose a Device



Family, Device & Board Settings

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: MAX 10 (DA/DF/DC/SA/SC)

Device: MAX 10 DA

Target device

( ) Auto device selected by the Fitter
(●) Specific device selected in 'Available devices' list
( ) Other: n/a

Show in 'Available devices' list

Package: FBGA

Pin count: 484

Core speed grade: 7

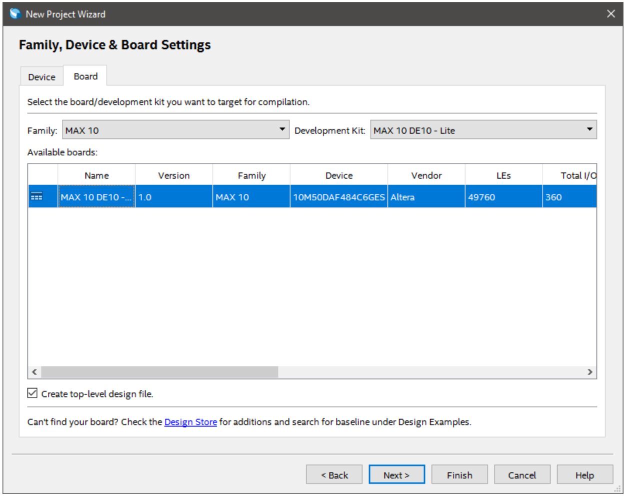
Name filter: 10M50DAF484C7G

☑ Show advanced devices

Available devices:

Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-bit e
10M50DAF484C7G	1.2V	49760	360	360	1677312	288

Or Choose a Board



New Project Wizard

Family, Device & Board Settings

Select the board/development kit you want to target for compilation.

Family: MAX 10 Development Kit: MAX 10 DE10 - Lite

Available boards:

	Name	Version	Family	Device	Vendor	LEs	Total I/O
	MAX 10 DE10 -...	1.0	MAX 10	10M50DAF484C6GES	Altera	49760	360

☑ Create top-level design file.

Can't find your board? Check the Design Store for additions and search for baseline under Design Examples.

Or Choose a Board

If you cannot find the DE10-Lite in the list, you'll need to install the Baseline Pinout project from the Altera website.

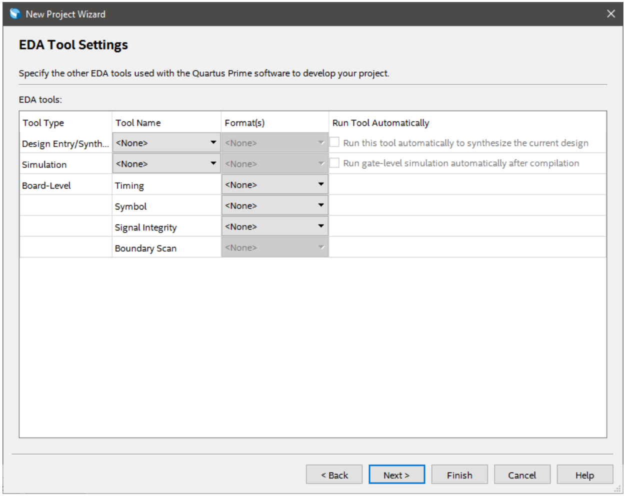
For today – use the "Choose a Device" option.

Or Choose a Board

If you cannot find the DE10-Lite in the list, you'll need to install the Baseline Pinout project from the Altera website.

For today – use the "Choose a Device" option.

3rd Party Tools



EDA Tool Settings

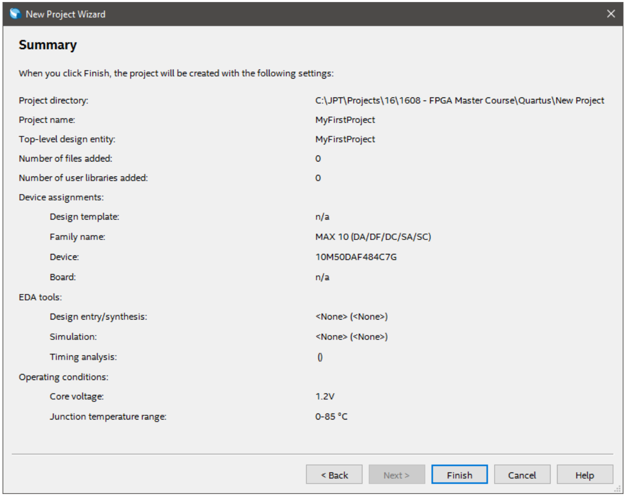
Specify the other EDA tools used with the Quartus Prime software to develop your project.

EDA tools:

Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entry/Synth...	<None>	<None>	☐ Run this tool automatically to synthesize the current design
Simulation	<None>	<None>	☐ Run gate-level simulation automatically after compilation
Board-Level	Timing	<None>	
	Symbol	<None>	
	Signal Integrity	<None>	
	Boundary Scan	<None>	

THE RADAR MASTERS COURSE

Summary

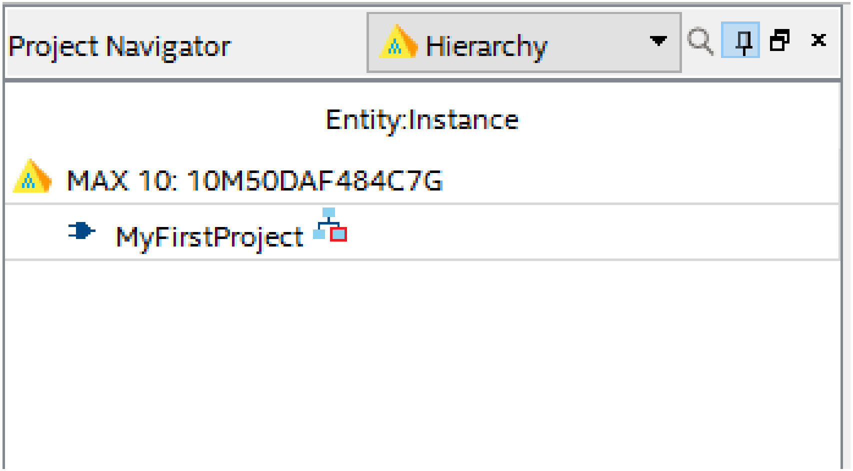


Summary

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

Project directory:	C:\JPT\Projects\16\1608 - FPGA Master Course\Quartus\New Project
Project name:	MyFirstProject
Top-level design entity:	MyFirstProject
Number of files added:	0
Number of user libraries added:	0
Device assignments:	
Design template:	n/a
Family name:	MAX 10 (DA/DF/DC/SA/SC)
Device:	10M50DAF484C7G
Board:	n/a
EDA tools:	
Design entry/synthesis:	<None> (<None>)
Simulation:	<None> (<None>)
Timing analysis:	()
Operating conditions:	
Core voltage:	1.2V
Junction temperature range:	0-85 °C

Project Tree

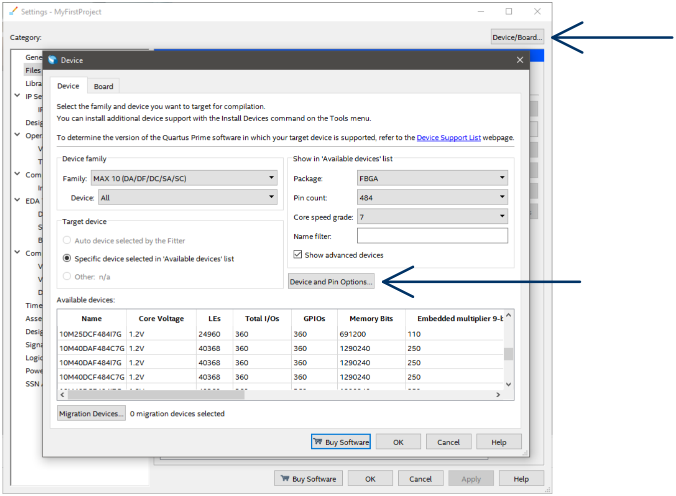


Entity:Instance

MAX 10: 10M50DAF484C7G

MyFirstProject

Device Settings



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: MAX 10 (DA/DF/DC/SA/SC)
Device: All

Show in 'Available devices' list
Package: FBGA
Pin count: 484
Core speed grade: 7
Name filter:
[✓] Show advanced devices

Target device
Auto device selected by the Fitter
(●) Specific device selected in 'Available devices' list
Other: n/a

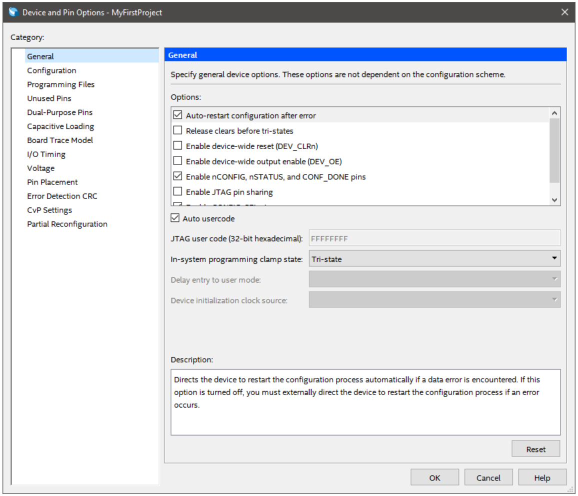
Device and Pin Options...

Available devices:

Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-b
10M25DCF484I7G	1.2V	24960	360	360	691200	110
10M40DAF484C7G	1.2V	40368	360	360	1290240	250
10M40DAF484I7G	1.2V	40368	360	360	1290240	250
10M40DCF484C7G	1.2V	40368	360	360	1290240	250

Migration Devices... 0 migration devices selected

Device and Pin Options



Device and Pin Options - MyFirstProject

Category:

- General
- Configuration
- Programming Files
- Unused Pins
- Dual-Purpose Pins
- Capacitive Loading
- Board Trace Model
- I/O Timing
- Voltage
- Pin Placement
- Error Detection CRC
- CvP Settings
- Partial Reconfiguration

General

Specify general device options. These options are not dependent on the configuration scheme.

Options:

- [✓] Auto-restart configuration after error
- [ ] Release clears before tri-states
- [ ] Enable device-wide reset (DEV_CLRn)
- [ ] Enable device-wide output enable (DEV_OE)
- [✓] Enable nCONFIG, nSTATUS, and CONF_DONE pins
- [ ] Enable JTAG pin sharing

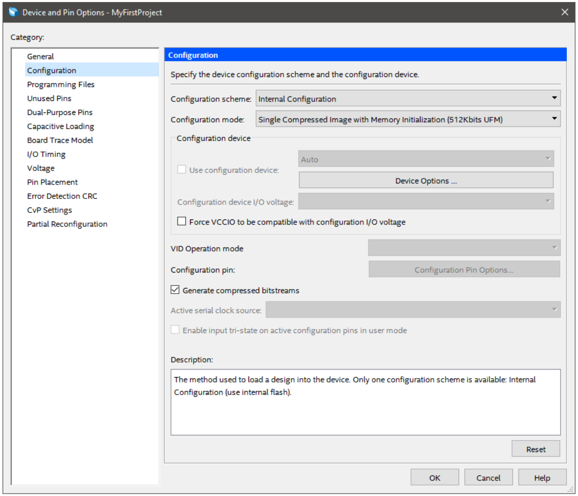
[✓] Auto usercode

JTAG user code (32-bit hexadecimal):	FFFFFFFF
In-system programming clamp state:	Tri-state
Delay entry to user mode:	
Device initialization clock source:	

Description:

Directs the device to restart the configuration process automatically if a data error is encountered. If this option is turned off, you must externally direct the device to restart the configuration process if an error occurs.

Enable Memory Initialisation



Device and Pin Options - MyFirstProject

Category:

- General
- Configuration
- Programming Files
- Unused Pins
- Dual-Purpose Pins
- Capacitive Loading
- Board Trace Model
- I/O Timing
- Voltage
- Pin Placement
- Error Detection CRC
- CvP Settings
- Partial Reconfiguration

Configuration

Specify the device configuration scheme and the configuration device.

Configuration scheme: Internal Configuration

Configuration mode: Single Compressed Image with Memory Initialization (512Kbits UFM)

Configuration device

☐ Use configuration device: Auto

Device Options ...

Configuration device I/O voltage:

☐ Force VCCIO to be compatible with configuration I/O voltage

VID Operation mode

Configuration pin: Configuration Pin Options...

☑ Generate compressed bitstreams

Active serial clock source:

☐ Enable input tri-state on active configuration pins in user mode

Description:

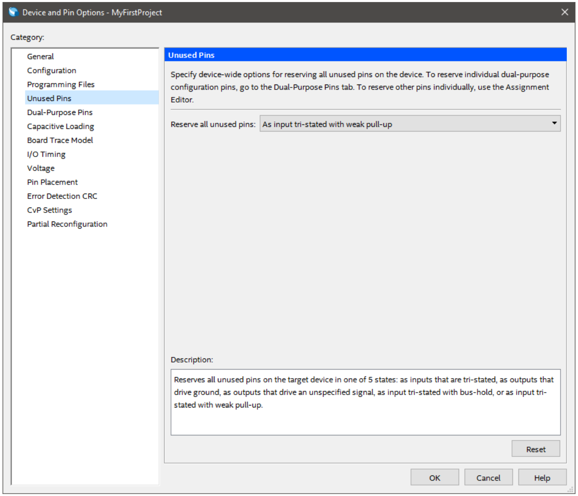
The method used to load a design into the device. Only one configuration scheme is available: Internal Configuration (use internal flash).

Reset

OK Cancel Help

THE RADAR MASTERS COURSE

Unused Pins to Tri-state



Unused Pins

Specify device-wide options for reserving all unused pins on the device. To reserve individual dual-purpose configuration pins, go to the Dual-Purpose Pins tab. To reserve other pins individually, use the Assignment Editor.

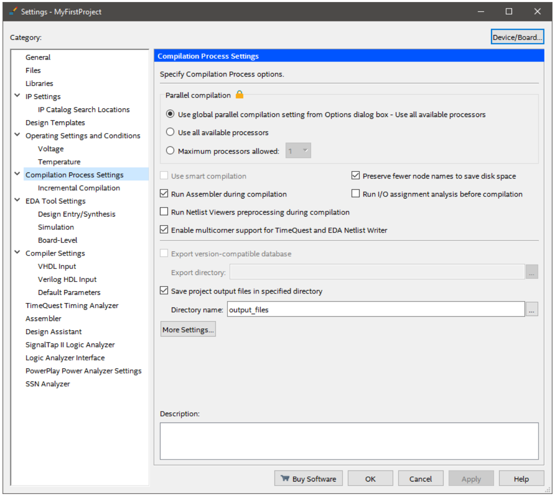
Reserve all unused pins: As input tri-stated with weak pull-up

Description:

Reserves all unused pins on the target device in one of 5 states: as inputs that are tri-stated, as outputs that drive ground, as outputs that drive an unspecified signal, as input tri-stated with bus-hold, or as input tri-stated with weak pull-up.

THE RADAR MASTERS COURSE

Output Files Folder



Compilation Process Settings

Specify Compilation Process options.

Parallel compilation

● Use global parallel compilation setting from Options dialog box - Use all available processors

○ Use all available processors

○ Maximum processors allowed: 1

☐ Use smart compilation

☑ Preserve fewer node names to save disk space

☑ Run Assembler during compilation

☐ Run I/O assignment analysis before compilation

☐ Run Netlist Viewers preprocessing during compilation

☑ Enable multicorner support for TimeQuest and EDA Netlist Writer

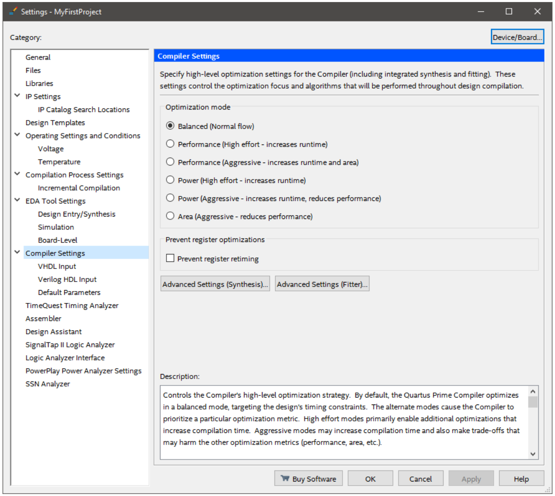
☐ Export version-compatible database

Export directory:

☑ Save project output files in specified directory

Directory name: output_files

Compiler Optimisation Mode



Compiler Settings

Specify high-level optimization settings for the Compiler (including integrated synthesis and fitting). These settings control the optimization focus and algorithms that will be performed throughout design compilation.

Optimization mode

- ● Balanced (Normal flow)
- ○ Performance (High effort - increases runtime)
- ○ Performance (Aggressive - increases runtime and area)
- ○ Power (High effort - increases runtime)
- ○ Power (Aggressive - increases runtime, reduces performance)
- ○ Area (Aggressive - reduces performance)

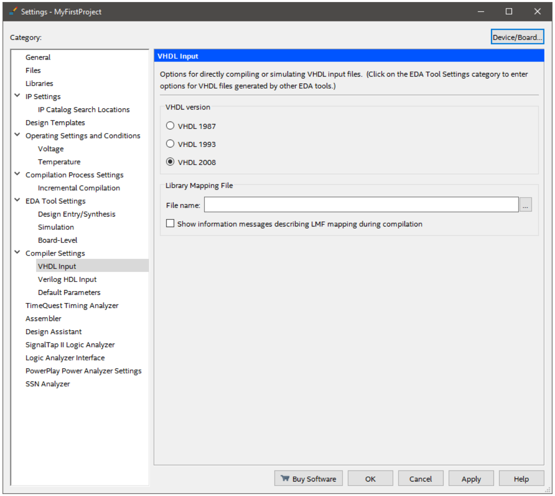
Prevent register optimizations

- ☐ Prevent register retiming

Description:

Controls the Compiler's high-level optimization strategy. By default, the Quartus Prime Compiler optimizes in a balanced mode, targeting the design's timing constraints. The alternate modes cause the Compiler to prioritize a particular optimization metric. High effort modes primarily enable additional optimizations that increase compilation time. Aggressive modes may increase compilation time and also make trade-offs that may harm the other optimization metrics (performance, area, etc.).

VHDL Version



VHDL Input

Options for directly compiling or simulating VHDL input files. (Click on the EDA Tool Settings category to enter options for VHDL files generated by other EDA tools.)

VHDL version

○ VHDL 1987
○ VHDL 1993
● VHDL 2008

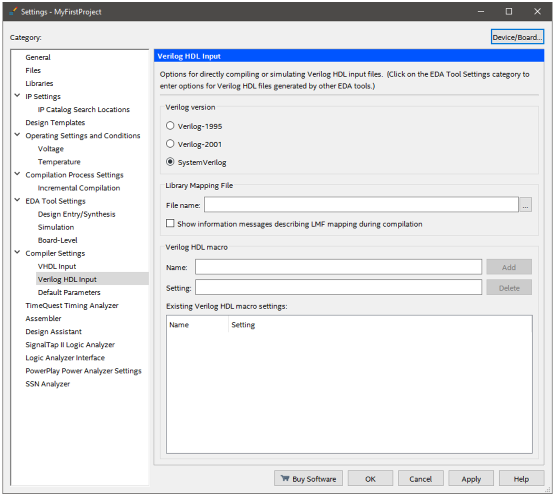
Library Mapping File

File name:

☐ Show information messages describing LMF mapping during compilation

THE RADAR MASTERS COURSE

Verilog Version



Settings - MyFirstProject

Category:

Device/Board...

- General
- Files
- Libraries
- IP Settings
  - IP Catalog Search Locations
- Design Templates
- Operating Settings and Conditions
  - Voltage
  - Temperature
- Compilation Process Settings
  - Incremental Compilation
- EDA Tool Settings
  - Design Entry/Synthesis
  - Simulation
  - Board-Level
- Compiler Settings
  - VHDL Input
  - Verilog HDL Input
  - Default Parameters
- TimeQuest Timing Analyzer
- Assembler
- Design Assistant
- SignalTap II Logic Analyzer
- Logic Analyzer Interface
- PowerPlay Power Analyzer Settings
- SSN Analyzer

Verilog HDL Input

Options for directly compiling or simulating Verilog HDL input files. (Click on the EDA Tool Settings category to enter options for Verilog HDL files generated by other EDA tools.)

Verilog version

- ○ Verilog-1995
- ○ Verilog-2001
- ● SystemVerilog

Library Mapping File

File name:

☐ Show information messages describing LMF mapping during compilation

Verilog HDL macro

Name:

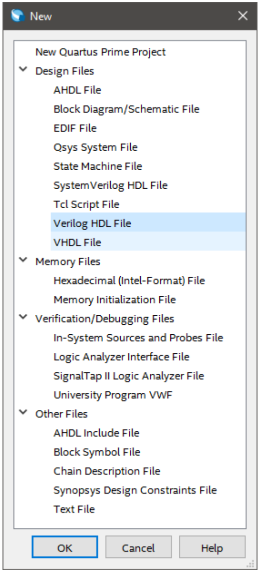
Setting:

Existing Verilog HDL macro settings:

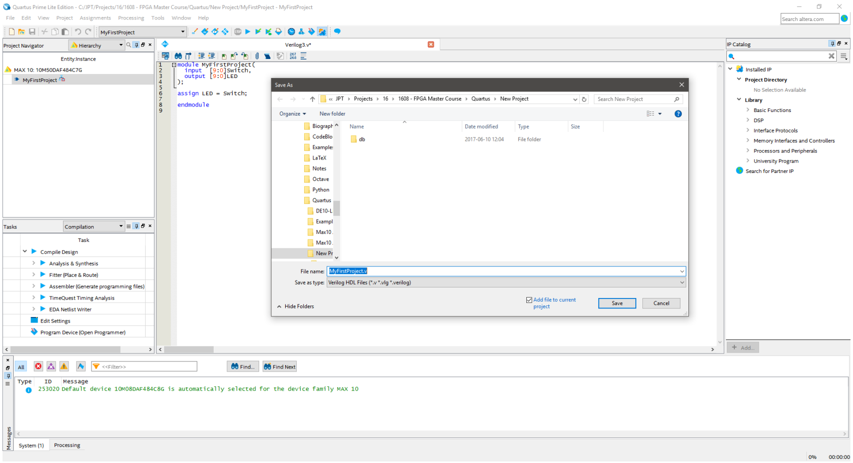
Name	Setting

Creating the Top-level Module

```
module MyFirstProject(
  input  [9:0]Switch,
  output [9:0]LED
);

assign LED = Switch;

endmodule
```


New
New Quartus Prime Project
Design Files
AHDL File
Block Diagram/Schematic File
EDIF File
Qsys System File
State Machine File
SystemVerilog HDL File
Tcl Script File
Verilog HDL File
VHDL File
Memory Files
Hexadecimal (Intel-Format) File
Memory Initialization File
Verification/Debugging Files
In-System Sources and Probes File
Logic Analyzer Interface File
SignalTap II Logic Analyzer File
University Program VWF
Other Files
AHDL Include File
Block Symbol File
Chain Description File
Synopsys Design Constraints File
Text File
OK
Cancel
Help

Save the Top-level Module



Verilog3.v*

```
module MyFirstProject(
  input  [9:0]Switch,
  output [9:0]LED
);

assign LED = Switch;

endmodule
```

MAX 10: 10M50DAF484C7G
MyFirstProject

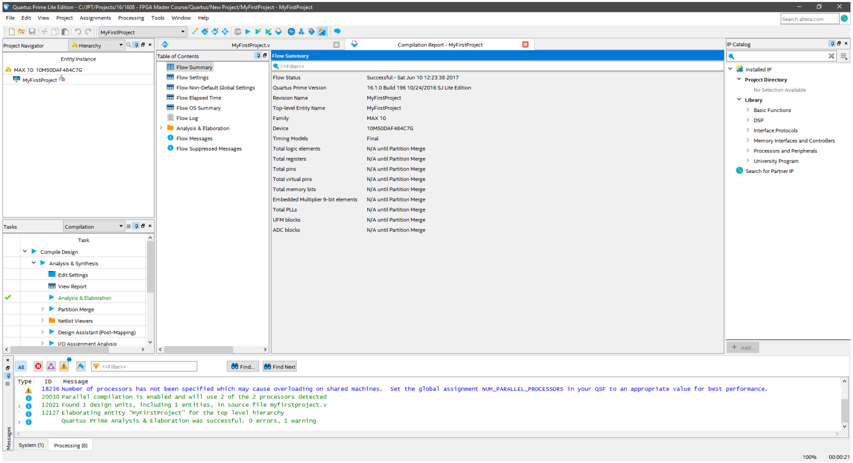
Name	Date modified	Type	Size
db	2017-06-10 12:04	File folder	

File name: MyFirstProject.v
Save as type: Verilog HDL Files (*.v *.vlg *.verilog)
☑ Add file to current project

```
Type  ID     Message
      253020 Default device 10M08DAF484C8G is automatically selected for the device family MAX 10
```

THE RADAR MASTERS COURSE

Run Analysis and Elaboration

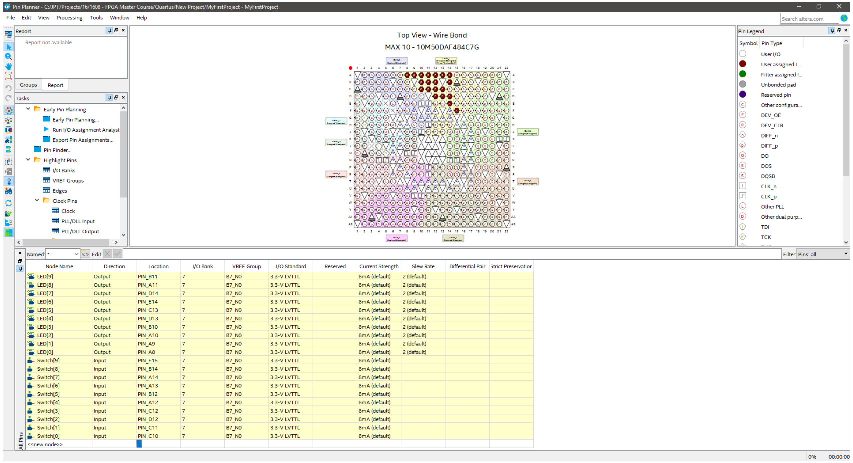


Flow Summary

Flow Status	Successful - Sat Jun 10 12:23:38 2017
Quartus Prime Version	16.1.0 Build 196 10/24/2016 SJ Lite Edition
Revision Name	MyFirstProject
Top-level Entity Name	MyFirstProject
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	N/A until Partition Merge
Total registers	N/A until Partition Merge
Total pins	N/A until Partition Merge
Total virtual pins	N/A until Partition Merge
Total memory bits	N/A until Partition Merge
Embedded Multiplier 9-bit elements	N/A until Partition Merge
Total PLLs	N/A until Partition Merge
UFM blocks	N/A until Partition Merge
ADC blocks	N/A until Partition Merge

```
Type  ID    Message
      18236 Number of processors has not been specified which may cause overloading on shared machines.  Set the global assignment NUM_PARALLEL_PROCESSORS in your QSF to an appropriate value for best performance.
      20030 Parallel compilation is enabled and will use 2 of the 2 processors detected
      12021 Found 1 design units, including 1 entities, in source file myfirstproject.v
      12127 Elaborating entity "MyFirstProject" for the top level hierarchy
            Quartus Prime Analysis & Elaboration was successful. 0 errors, 1 warning
```

Open the Pin Planner

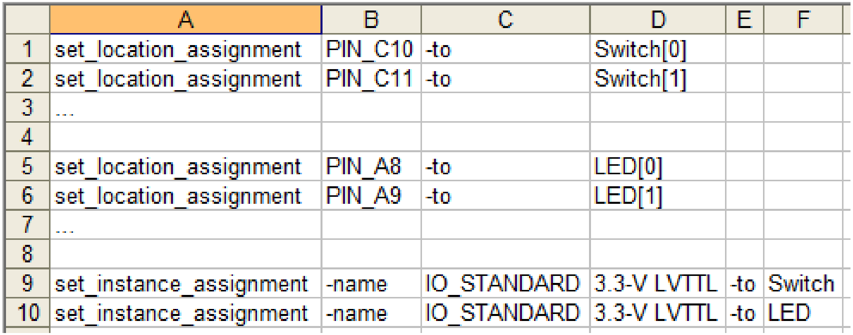


Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
LED[9]	Output	PIN_B11	7	B7_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[8]	Output	PIN_A11	7	B7_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[7]	Output	PIN_D14	7	B7_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[6]	Output	PIN_E14	7	B7_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[5]	Output	PIN_C13	7	B7_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[4]	Output	PIN_D13	7	B7_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[3]	Output	PIN_B10	7	B7_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[2]	Output	PIN_A10	7	B7_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[1]	Output	PIN_A9	7	B7_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[0]	Output	PIN_A8	7	B7_N0	3.3-V LVTTL		8mA (default)	2 (default)		
Switch[9]	Input	PIN_F15	7	B7_N0	3.3-V LVTTL		8mA (default)			
Switch[8]	Input	PIN_B14	7	B7_N0	3.3-V LVTTL		8mA (default)			
Switch[7]	Input	PIN_A14	7	B7_N0	3.3-V LVTTL		8mA (default)			
Switch[6]	Input	PIN_A13	7	B7_N0	3.3-V LVTTL		8mA (default)			
Switch[5]	Input	PIN_B12	7	B7_N0	3.3-V LVTTL		8mA (default)			
Switch[4]	Input	PIN_A12	7	B7_N0	3.3-V LVTTL		8mA (default)			
Switch[3]	Input	PIN_C12	7	B7_N0	3.3-V LVTTL		8mA (default)			
Switch[2]	Input	PIN_D12	7	B7_N0	3.3-V LVTTL		8mA (default)			
Switch[1]	Input	PIN_C11	7	B7_N0	3.3-V LVTTL		8mA (default)			
Switch[0]	Input	PIN_C10	7	B7_N0	3.3-V LVTTL		8mA (default)			
<<new node>>										

THE RADAR MASTERS COURSE

Easier Pin-assignments...

- Copy from PDF manual to Excel
- Copy from Excel to Quartus Settings File (QSF)



	A	B	C	D	E	F
1	set_location_assignment	PIN_C10	-to	Switch[0]		
2	set_location_assignment	PIN_C11	-to	Switch[1]		
3	...					
4						
5	set_location_assignment	PIN_A8	-to	LED[0]		
6	set_location_assignment	PIN_A9	-to	LED[1]		
7	...					
8						
9	set_instance_assignment	-name	IO_STANDARD	3.3-V LVTTL	-to	Switch
10	set_instance_assignment	-name	IO_STANDARD	3.3-V LVTTL	-to	LED

Easier Pin-assignments...

- Copy from PDF manual to Excel
- Copy from Excel to Quartus Settings File (QSF)

```
set_location_assignment PIN_C10 -to Switch[0]
set_location_assignment PIN_C11 -to Switch[1]
...

set_location_assignment PIN_A8 -to LED[0]
set_location_assignment PIN_A9 -to LED[1]
...

set_instance_assignment \
  -name IO_STANDARD "3.3-V LVTTL" -to Switch
set_instance_assignment \
  -name IO_STANDARD "3.3-V LVTTL" -to LED
```

Other Pin Properties

- Use a slow slew-rate to limit bandwidth on long lines
- Use a fast slew-rate for time-critical signals
- Use a high drive strength for LEDs
- Use a high drive strength for pins that are higher speed and / or capacitively loaded
- Use a low drive strength when saving power is required (also limits the bandwidth)
- The MAX-10 supports Schmitt-triggered inputs

Other Pin Properties

- Use a slow slew-rate to limit bandwidth on long lines
- Use a fast slew-rate for time-critical signals
- Use a high drive strength for LEDs
- Use a high drive strength for pins that are higher speed and / or capacitively loaded
- Use a low drive strength when saving power is required (also limits the bandwidth)
- The MAX-10 supports Schmitt-triggered inputs

Other Pin Properties

- Use a slow slew-rate to limit bandwidth on long lines
- Use a fast slew-rate for time-critical signals
- Use a high drive strength for LEDs
- Use a high drive strength for pins that are higher speed and / or capacitively loaded
- Use a low drive strength when saving power is required (also limits the bandwidth)
- The MAX-10 supports Schmitt-triggered inputs

Other Pin Properties

- Use a slow slew-rate to limit bandwidth on long lines
- Use a fast slew-rate for time-critical signals
- Use a high drive strength for LEDs
- Use a high drive strength for pins that are higher speed and / or capacitively loaded
- Use a low drive strength when saving power is required (also limits the bandwidth)
- The MAX-10 supports Schmitt-triggered inputs

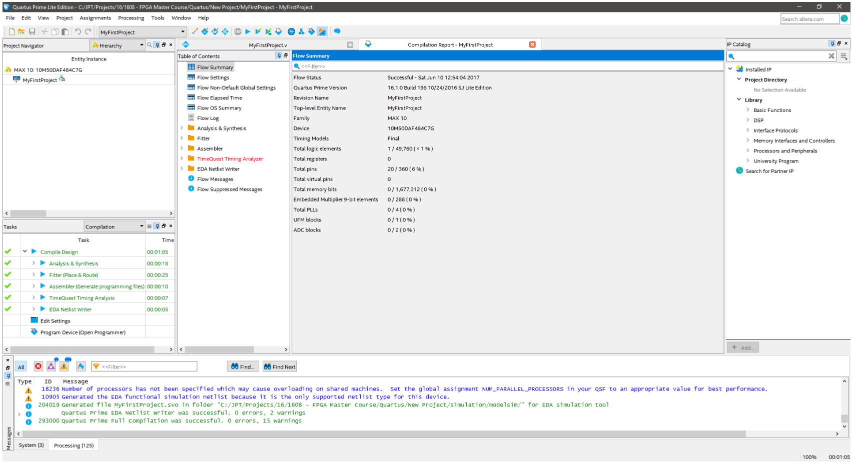
Other Pin Properties

- Use a slow slew-rate to limit bandwidth on long lines
- Use a fast slew-rate for time-critical signals
- Use a high drive strength for LEDs
- Use a high drive strength for pins that are higher speed and / or capacitively loaded
- Use a low drive strength when saving power is required (also limits the bandwidth)
- The MAX-10 supports Schmitt-triggered inputs

Other Pin Properties

- Use a slow slew-rate to limit bandwidth on long lines
- Use a fast slew-rate for time-critical signals
- Use a high drive strength for LEDs
- Use a high drive strength for pins that are higher speed and / or capacitively loaded
- Use a low drive strength when saving power is required (also limits the bandwidth)
- The MAX-10 supports Schmitt-triggered inputs

Run Full Compilation



Flow Summary

Flow Status	Successful - Sat Jun 10 12:54:04 2017
Quartus Prime Version	16.1.0 Build 196 10/24/2016 SJ Lite Edition
Revision Name	MyFirstProject
Top-level Entity Name	MyFirstProject
Family	MAX 10
Device	10M50DAF484C7G
Timing Models	Final
Total logic elements	1 / 49,760 (< 1 %)
Total registers	0
Total pins	20 / 360 (6 %)
Total virtual pins	0
Total memory bits	0 / 1,677,312 (0 %)
Embedded Multiplier 9-bit elements	0 / 288 (0 %)
Total PLLs	0 / 4 (0 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	0 / 2 (0 %)

Tasks: Compilation

Task	Time
✓ Compile Design	00:01:05
✓ Analysis & Synthesis	00:00:18
✓ Fitter (Place & Route)	00:00:25
✓ Assembler (Generate programming files)	00:00:10
✓ TimeQuest Timing Analysis	00:00:07
✓ EDA Netlist Writer	00:00:05
Edit Settings	
Program Device (Open Programmer)	

```
Type   ID   Message
     18236 Number of processors has not been specified which may cause overloading on shared machines.  Set the global assignment NUM_PARALLEL_PROCESSORS in your QSF to an appropriate value for best performance.
     10905 Generated the EDA functional simulation netlist because it is the only supported netlist type for this device.
    204019 Generated file MyFirstProject.svo in folder "C:/JPT/Projects/16/1608 - FPGA Master Course/Quartus/New Project/simulation/modelsim/" for EDA simulation tool
           Quartus Prime EDA Netlist Writer was successful. 0 errors, 2 warnings
    293000 Quartus Prime Full Compilation was successful. 0 errors, 15 warnings
```

System (3)   Processing (125)   100%   00:01:05

Run Full Compilation

Note the red “Timing Analyser” report...

Run Full Compilation

Note the red “Timing Analyser” report...
We’ll worry about that tomorrow...

Program the FPGA

- Plug in the DE10-Lite and open the Programmer Window
- Make sure the “Hardware Setup” says “USB-Blaster”
- If you cannot select the USB-Blaster from the list of hardware devices, you’ll need to update the driver – point your Device-Manager’s “Update Driver” wizard to `C:/intel/16.1/quartus/drivers`
- Click on “Add File...” and choose `output_files/MyFirstProject.sof`
- Click on “Start” to program the FPGA
- Play with the switches to make sure your design is working
- The SOF is volatile. If you’re happy with the design and want to make it non-volatile, choose `output_files/MyFirstProject.pof` instead.

Program the FPGA

- Plug in the DE10-Lite and open the Programmer Window
- Make sure the “Hardware Setup” says “USB-Blaster”
- If you cannot select the USB-Blaster from the list of hardware devices, you’ll need to update the driver – point your Device-Manager’s “Update Driver” wizard to `C:/intel/16.1/quartus/drivers`
- Click on “Add File...” and choose `output_files/MyFirstProject.sof`
- Click on “Start” to program the FPGA
- Play with the switches to make sure your design is working
- The SOF is volatile. If you’re happy with the design and want to make it non-volatile, choose `output_files/MyFirstProject.pof` instead.

Program the FPGA

- Plug in the DE10-Lite and open the Programmer Window
- Make sure the “Hardware Setup” says “USB-Blaster”
- If you cannot select the USB-Blaster from the list of hardware devices, you’ll need to update the driver – point your Device-Manager’s “Update Driver” wizard to `C:/intel/16.1/quartus/drivers`
- Click on “Add File...” and choose `output_files/MyFirstProject.sof`
- Click on “Start” to program the FPGA
- Play with the switches to make sure your design is working
- The SOF is volatile. If you’re happy with the design and want to make it non-volatile, choose `output_files/MyFirstProject.pof` instead.

Program the FPGA

- Plug in the DE10-Lite and open the Programmer Window
- Make sure the “Hardware Setup” says “USB-Blaster”
- If you cannot select the USB-Blaster from the list of hardware devices, you’ll need to update the driver – point your Device-Manager’s “Update Driver” wizard to `C:/intel/16.1/quartus/drivers`
- Click on “Add File...” and choose `output_files/MyFirstProject.sof`
- Click on “Start” to program the FPGA
- Play with the switches to make sure your design is working
- The SOF is volatile. If you’re happy with the design and want to make it non-volatile, choose `output_files/MyFirstProject.pof` instead.

Program the FPGA

- Plug in the DE10-Lite and open the Programmer Window
- Make sure the "Hardware Setup" says "USB-Blaster"
- If you cannot select the USB-Blaster from the list of hardware devices, you'll need to update the driver – point your Device-Manager's "Update Driver" wizard to `C:/intel/16.1/quartus/drivers`
- Click on "Add File..." and choose `output_files/MyFirstProject.sof`
- Click on "Start" to program the FPGA
- Play with the switches to make sure your design is working
- The SOF is volatile. If you're happy with the design and want to make it non-volatile, choose `output_files/MyFirstProject.pof` instead.

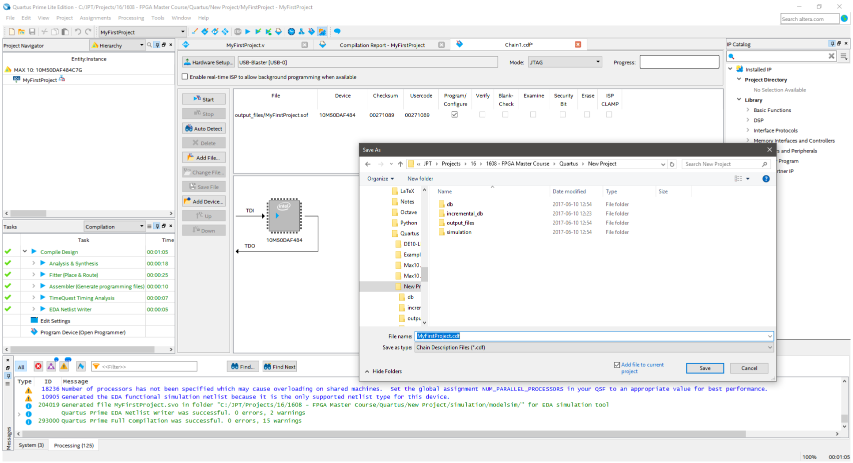
Program the FPGA

- Plug in the DE10-Lite and open the Programmer Window
- Make sure the “Hardware Setup” says “USB-Blaster”
- If you cannot select the USB-Blaster from the list of hardware devices, you’ll need to update the driver – point your Device-Manager’s “Update Driver” wizard to `C:/intel/16.1/quartus/drivers`
- Click on “Add File...” and choose `output_files/MyFirstProject.sof`
- Click on “Start” to program the FPGA
- Play with the switches to make sure your design is working
- The SOF is volatile. If you’re happy with the design and want to make it non-volatile, choose `output_files/MyFirstProject.pof` instead.

Program the FPGA

- Plug in the DE10-Lite and open the Programmer Window
- Make sure the “Hardware Setup” says “USB-Blaster”
- If you cannot select the USB-Blaster from the list of hardware devices, you’ll need to update the driver – point your Device-Manager’s “Update Driver” wizard to `C:/intel/16.1/quartus/drivers`
- Click on “Add File...” and choose `output_files/MyFirstProject.sof`
- Click on “Start” to program the FPGA
- Play with the switches to make sure your design is working
- The SOF is volatile. If you’re happy with the design and want to make it non-volatile, choose `output_files/MyFirstProject.pof` instead.

Save the Configuration


Quartus Prime Lite Edition - C:/JPT/Projects/16/1608 - FPGA Master Course/Quartus/New Project/MyFirstProject - MyFirstProject

Hardware Setup... USB-Blaster [USB-0] Mode: 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	ISP CLAMP
output_files/MyFirstProject.sof	10M50DAF484	00271089	00271089	✓						

Task	Time
✓ Compile Design	00:01:05
✓ Analysis & Synthesis	00:00:18
✓ Fitter (Place & Route)	00:00:25
✓ Assembler (Generate programming files)	00:00:10
✓ TimeQuest Timing Analysis	00:00:07
✓ EDA Netlist Writer	00:00:05
Edit Settings	
Program Device (Open Programmer)	

Save As

« JPT › Projects › 16 › 1608 - FPGA Master Course › Quartus › New Project

Name	Date modified	Type	Size
db	2017-06-10 12:54	File folder	
incremental_db	2017-06-10 12:23	File folder	
output_files	2017-06-10 12:54	File folder	
simulation	2017-06-10 12:54	File folder	

File name: MyFirstProject.cdf

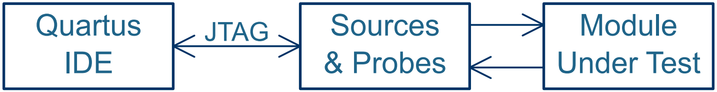
Save as type: Chain Description Files (*.cdf)

Add file to current project ✓

```
Type  ID     Message
      18236 Number of processors has not been specified which may cause overloading on shared machines.  Set the global assignment NUM_PARALLEL_PROCESSORS in your QSF to an appropriate value for best performance.
      10905 Generated the EDA functional simulation netlist because it is the only supported netlist type for this device.
     204019 Generated file MyFirstProject.svo in folder "C:/JPT/Projects/16/1608 - FPGA Master Course/Quartus/New Project/simulation/modelsim/" for EDA simulation tool
            Quartus Prime EDA Netlist Writer was successful. 0 errors, 2 warnings
     293000 Quartus Prime Full Compilation was successful. 0 errors, 15 warnings
```

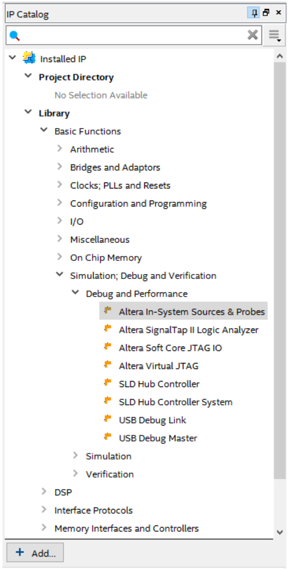
THE RADAR MASTERS COURSE

JTAG: Source and Probes



IP Catalogue

- The IP Catalogue is used to create wrapper modules for:
  - Internal RAM,
  - Phase-locked loops
  - DDR controllers
  - PCIe controllers
  - DSP modules
  - ADC modules
  - etc.
- More about that tomorrow. For now, choose Sources and Probes...

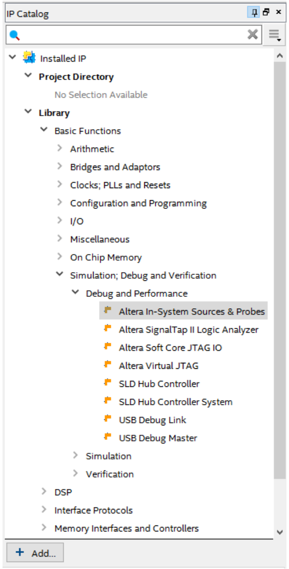

IP Catalog

- Installed IP
  - Project Directory
    - No Selection Available
  - Library
    - Basic Functions
      - Arithmetic
      - Bridges and Adaptors
      - Clocks; PLLs and Resets
      - Configuration and Programming
      - I/O
      - Miscellaneous
      - On Chip Memory
      - Simulation; Debug and Verification
        - Debug and Performance
          - Altera In-System Sources & Probes
          - Altera SignalTap II Logic Analyzer
          - Altera Soft Core JTAG IO
          - Altera Virtual JTAG
          - SLD Hub Controller
          - SLD Hub Controller System
          - USB Debug Link
          - USB Debug Master
        - Simulation
        - Verification
    - DSP
    - Interface Protocols
    - Memory Interfaces and Controllers

+ Add...

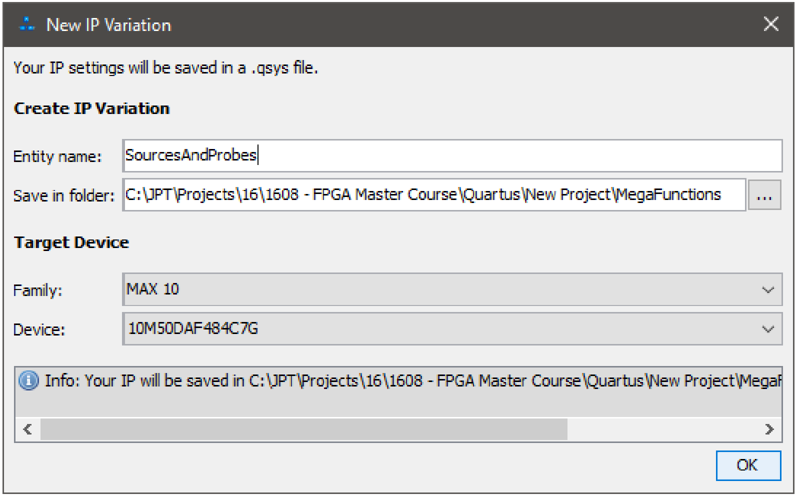
IP Catalogue

- The IP Catalogue is used to create wrapper modules for:
  - Internal RAM,
  - Phase-locked loops
  - DDR controllers
  - PCIe controllers
  - DSP modules
  - ADC modules
  - etc.
- More about that tomorrow. For now, choose Sources and Probes...



THE RADAR MASTERS COURSE

Sources and Probes



Your IP settings will be saved in a .qsys file.

Create IP Variation

Entity name: SourcesAndProbes

Save in folder: C:\JPT\Projects\16\1608 - FPGA Master Course\Quartus\New Project\MegaFunctions

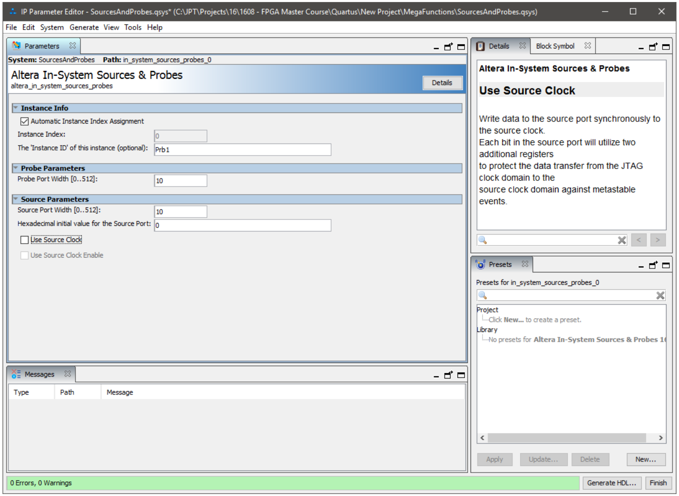
Target Device

Family: MAX 10

Device: 10M50DAF484C7G

Info: Your IP will be saved in C:\JPT\Projects\16\1608 - FPGA Master Course\Quartus\New Project\MegaF

Set the Parameters



IP Parameter Editor - SourcesAndProbes.qsys* (C:\JPT\Projects\16\1608 - FPGA Master Course\Quartus\New Project\MegaFunctions\SourcesAndProbes.qsys)

File Edit System Generate View Tools Help

System: SourcesAndProbes **Path:** in_system_sources_probes_0

Altera In-System Sources & Probes
altera_in_system_sources_probes

Instance Info

[✓] Automatic Instance Index Assignment

Instance Index: 0

The 'Instance ID' of this instance (optional): Prb1

Probe Parameters

Probe Port Width [0..512]: 10

Source Parameters

Source Port Width [0..512]: 10

Hexadecimal initial value for the Source Port: 0

[ ] Use Source Clock

[ ] Use Source Clock Enable

Altera In-System Sources & Probes

Use Source Clock

Write data to the source port synchronously to the source clock.
Each bit in the source port will utilize two additional registers
to protect the data transfer from the JTAG clock domain to the
source clock domain against metastable events.

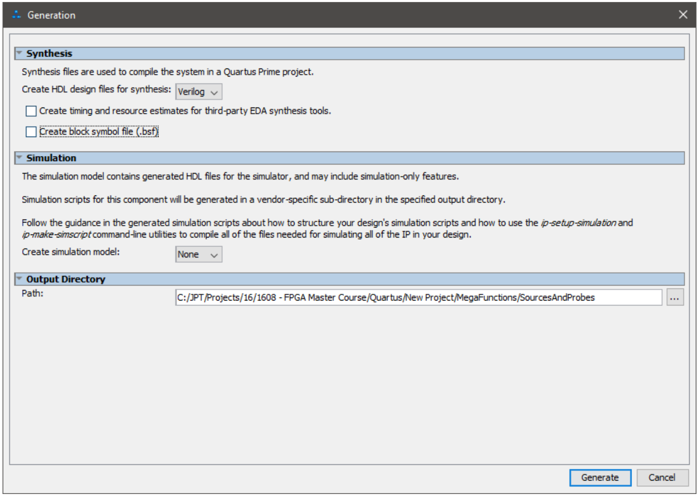
Presets for in_system_sources_probes_0

Project
Click **New...** to create a preset.
Library
No presets for **Altera In-System Sources & Probes 1**

Type	Path	Message

0 Errors, 0 Warnings

Generate the HDL



Synthesis

Synthesis files are used to compile the system in a Quartus Prime project.

Create HDL design files for synthesis: Verilog

☐ Create timing and resource estimates for third-party EDA synthesis tools.

☐ Create block symbol file (.bsf)

Simulation

The simulation model contains generated HDL files for the simulator, and may include simulation-only features.

Simulation scripts for this component will be generated in a vendor-specific sub-directory in the specified output directory.

Follow the guidance in the generated simulation scripts about how to structure your design's simulation scripts and how to use the *ip-setup-simulation* and *ip-make-simscript* command-line utilities to compile all of the files needed for simulating all of the IP in your design.

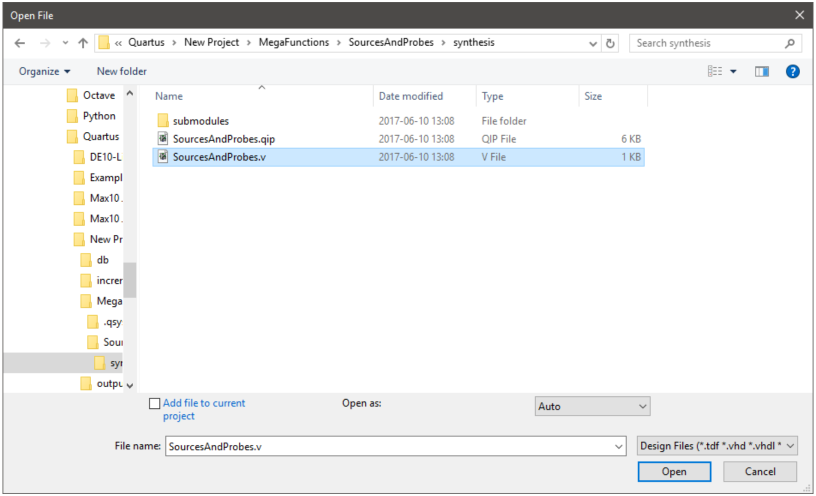
Create simulation model: None

Output Directory

Path: C:/JPT/Projects/16/1608 - FPGA Master Course/Quartus/New Project/MegaFunctions/SourcesAndProbes

THE RADAR MASTERS COURSE

Open the Wrapper


Open File

« Quartus › New Project › MegaFunctions › SourcesAndProbes › synthesis

Name	Date modified	Type	Size
submodules	2017-06-10 13:08	File folder	
SourcesAndProbes.qip	2017-06-10 13:08	QIP File	6 KB
SourcesAndProbes.v	2017-06-10 13:08	V File	1 KB

Add file to current project

Open as: Auto

File name: SourcesAndProbes.v

Design Files (*.tdf *.vhd *.vhdl *

THE RADAR MASTERS COURSE

Instantiate the Module

```
// In the wizard-generated wrapper...
module SourcesAndProbes (
  input  wire [9:0] probe,  // probes.probe
  output wire [9:0] source  // sources.source
);

// In your module-of-interest...
wire [9:0]Source;
SourcesAndProbes SourcesAndProbes_inst(
  .source(Source),
  .probe (Switch)
);
assign LED = Switch ^ Source;
```

Note: Remember to add the `.qip` or `.qsys` file to the project

Direct Option

Or you could use the MegaFunction directly (without the wizard):

```
wire [9:0]Source;

altsource_probe #(
  .instance_id              ("Prb1"),
  .sld_auto_instance_index ("YES"),
  .probe_width              (10),
  .source_width             (10)
)SourcesAndProbes_inst(
  .source_ena(1'b1),
  .source     (Source),
  .probe      (Switch)
);

assign LED = Switch ^ Source;
```

Using the Sources and Probes

- Compile and program the FPGA
- Open the “In-System Sources and Probes Editor” (see next slide)
- Set up the hardware (USB-Blaster)
- Select the instance you want (“Prb1” in this case)
- Click the “Continuously Read Probe Data” toolbar icon:
- Click to change the source values
- Play around with it to discover new features

THE RADAR MASTERS COURSE

Using the Sources and Probes

- Compile and program the FPGA
- Open the "In-System Sources and Probes Editor" (see next slide)
- Set up the hardware (USB-Blaster)
- Select the instance you want ("Prb1" in this case)
- Click the "Continuously Read Probe Data" toolbar icon:
- Click to change the source values
- Play around with it to discover new features

Using the Sources and Probes

- Compile and program the FPGA
- Open the “In-System Sources and Probes Editor” (see next slide)
- Set up the hardware (USB-Blaster)
- Select the instance you want (“Prb1” in this case)
- Click the “Continuously Read Probe Data” toolbar icon:
- Click to change the source values
- Play around with it to discover new features

Using the Sources and Probes

- Compile and program the FPGA
- Open the “In-System Sources and Probes Editor” (see next slide)
- Set up the hardware (USB-Blaster)
- Select the instance you want (“Prb1” in this case)
- Click the “Continuously Read Probe Data” toolbar icon:
- Click to change the source values
- Play around with it to discover new features

Using the Sources and Probes

- Compile and program the FPGA
- Open the “In-System Sources and Probes Editor” (see next slide)
- Set up the hardware (USB-Blaster)
- Select the instance you want (“Prb1” in this case)
- Click the “Continuously Read Probe Data” toolbar icon:
- Click to change the source values
- Play around with it to discover new features

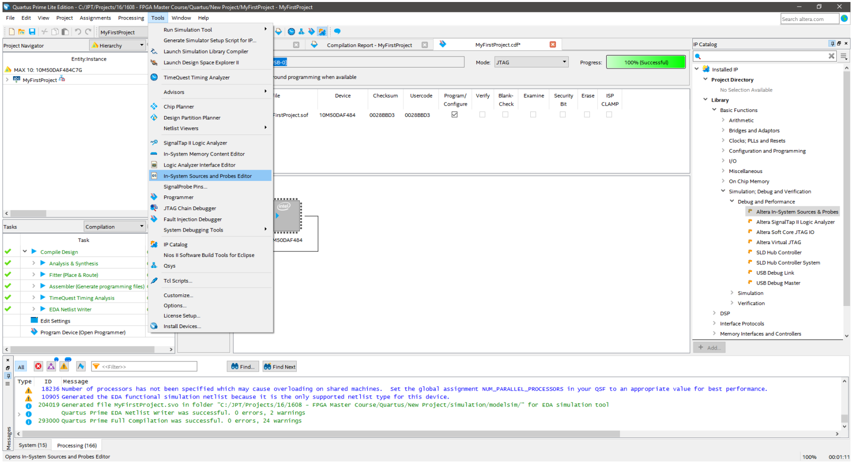
Using the Sources and Probes

- Compile and program the FPGA
- Open the “In-System Sources and Probes Editor” (see next slide)
- Set up the hardware (USB-Blaster)
- Select the instance you want (“Prb1” in this case)
- Click the “Continuously Read Probe Data” toolbar icon:
- Click to change the source values
- Play around with it to discover new features

Using the Sources and Probes

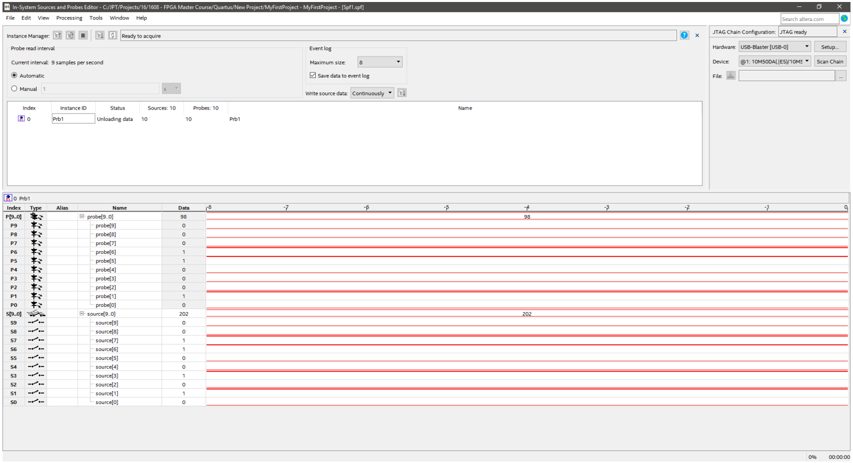
- Compile and program the FPGA
- Open the “In-System Sources and Probes Editor” (see next slide)
- Set up the hardware (USB-Blaster)
- Select the instance you want (“Prb1” in this case)
- Click the “Continuously Read Probe Data” toolbar icon:
- Click to change the source values
- Play around with it to discover new features

Using the Sources and Probes


Quartus Prime Lite Edition - C:/JPT/Projects/16/1608 - FPGA Master Course/Quartus/New Project/MyFirstProject - MyFirstProject
File Edit View Project Assignments Processing Tools Window Help
Run Simulation Tool
Generate Simulator Setup Script for IP...
Launch Simulation Library Compiler
Launch Design Space Explorer II
TimeQuest Timing Analyzer
Advisors
Chip Planner
Design Partition Planner
Netlist Viewers
SignalTap II Logic Analyzer
In-System Memory Content Editor
Logic Analyzer Interface Editor
In-System Sources and Probes Editor
SignalProbe Pins...
Programmer
JTAG Chain Debugger
Fault Injection Debugger
System Debugging Tools
IP Catalog
Nios II Software Build Tools for Eclipse
Qsys
Tcl Scripts...
Customize...
Options...
License Setup...
Install Devices...
Project Navigator
Hierarchy
Entity:Instance
MAX 10: 10M50DAF484C7G
MyFirstProject
Compilation Report - MyFirstProject
MyFirstProject.cdf*
Mode: JTAG
Progress: 100% (Successful)
File Device Checksum Usercode Program/Configure Verify Blank-Check Examine Security Bit Erase ISP CLAMP
MyFirstProject.sof 10M50DAF484 0028BBD3 0028BBD3
Tasks
Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files)
TimeQuest Timing Analysis
EDA Netlist Writer
Edit Settings
Program Device (Open Programmer)
IP Catalog
Installed IP
Project Directory
No Selection Available
Library
Basic Functions
Arithmetic
Bridges and Adaptors
Clocks; PLLs and Resets
Configuration and Programming
I/O
Miscellaneous
On Chip Memory
Simulation; Debug and Verification
Debug and Performance
Altera In-System Sources & Probes
Altera SignalTap II Logic Analyzer
Altera Soft Core JTAG IO
Altera Virtual JTAG
SLD Hub Controller
SLD Hub Controller System
USB Debug Link
USB Debug Master
Simulation
Verification
DSP
Interface Protocols
Memory Interfaces and Controllers
Add...
All
<<Filter>>
Find...
Find Next
System (15)
Processing (166)
Opens In-System Sources and Probes Editor
100%
00:01:11

```
Type  ID     Message
      18236 Number of processors has not been specified which may cause overloading on shared machines.  Set the global assignment NUM_PARALLEL_PROCESSORS in your QSF to an appropriate value for best performance.
      10905 Generated the EDA functional simulation netlist because it is the only supported netlist type for this device.
     204019 Generated file MyFirstProject.svo in folder "C:/JPT/Projects/16/1608 - FPGA Master Course/Quartus/New Project/simulation/modelsim/" for EDA simulation tool
            Quartus Prime EDA Netlist Writer was successful. 0 errors, 2 warnings
     293000 Quartus Prime Full Compilation was successful. 0 errors, 24 warnings
```

Using the Sources and Probes



In-System Sources and Probes Editor - C:/JPT/Projects/16/1608 - FPGA Master Course/Quartus/New Project/MyFirstProject - MyFirstProject - [Spf1.spf]

Instance Manager: Ready to acquire

Probe read interval
Current interval: 9 samples per second
Automatic
Manual 1 s

Event log
Maximum size: 8
Save data to event log
Write source data: Continuously

JTAG Chain Configuration: JTAG ready
Hardware: USB-Blaster [USB-0]
Device: @1: 10M50DA(.|ES)/10M5
File:

Index	Instance ID	Status	Sources: 10	Probes: 10	Name
0	Prb1	Unloading data	10	10	Prb1

0 Prb1

Index	Type	Alias	Name	Data
P[9..0]			probe[9..0]	98
P9			probe[9]	0
P8			probe[8]	0
P7			probe[7]	0
P6			probe[6]	1
P5			probe[5]	1
P4			probe[4]	0
P3			probe[3]	0
P2			probe[2]	0
P1			probe[1]	1
P0			probe[0]	0
S[9..0]			source[9..0]	202
S9			source[9]	0
S8			source[8]	0
S7			source[7]	1
S6			source[6]	1
S5			source[5]	0
S4			source[4]	0
S3			source[3]	1
S2			source[2]	0
S1			source[1]	1
S0			source[0]	0

0% 00:00:00

THE RADAR MASTERS COURSE

Select References

- Stephen Brown and Zvonko Vranesic
  Fundamentals of Digital Logic with Verilog Design, 2nd Edition
  ISBN 978-0-07-721164-6
- Merrill L Skolnik
  Introduction to RADAR Systems
  ISBN 978-0-07-288138-7
- Mark A. Richards and James A. Scheer
  Principles of Modern Radar: Basic Principles
  ISBN 978-1-89-112152-4
- Deepak Kumar Tala
  World of ASIC
  `http://www.asic-world.com/`
- Jean P. Nicolle
  FPGA 4 Fun
  `http://www.fpga4fun.com/`

FPGA Development for Radar, Radio-Astronomy and Communications

THE RADAR MASTERS COURSE

Dept. Electrical Engineering, University of Cape Town
Private Bag, Rondebosch, 7701, South Africa

http://www.rrsg.uct.ac.za

Presented by John-Philip Taylor

Convened by Prof Daniel O'Hagan

Tutored by Stephen Paine and Randy Cheng

Day 1 – 17 July 2017