

ECE 351

Verilog and FPGA Design

Week 10_1: FPGA overview (wrap-up)
Review Roy's HW #3 solution
Introduction to Xilinx Vivado
PmodSSD on Nexys A7 project walkthrough

Roy Kravitz
Electrical and Computer Engineering Department
Maseeh College of Engineering and Computer Science

Questions about Homework #4?

Write-up: [ece351sp21_hw4_release_r1_1/docs/ece351sp21_hw4.pdf](#)

FPGA Overview (wrap-up)

Sources:

- ECE 540 lecture notes by David B. and Roy K.

Field-programmable gate array (FPGA)

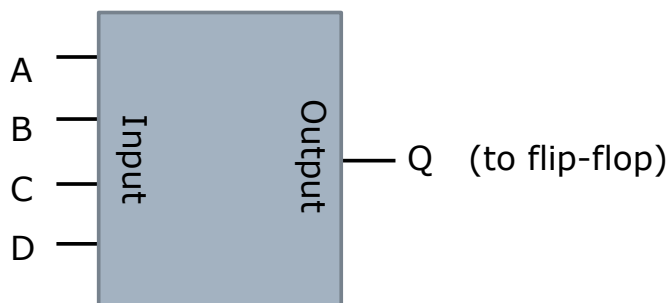
- ☐ A “gate array” is just what it says...an array of logic gates that can be configured by customizing the interconnect according to a netlist
- ☐ FPGAs can (and are) reconfigured even after installed in the end user site
- ☐ Contrast to gate arrays that can't be reconfigured in the field
 - ASIC
 - Erasable programmable read-only memory (EPROM)
 - ☐ Peel back a sticker, expose to UV
 - Electronically-erasable programmable read-only memory (EEPROM)
 - ☐ Could be rewritten in the field but it's memory, not logic

Feature comparison

- ☐ Logic cells (not "real" like 60W light bulbs aren't "real")
- ☐ **C**onfigurable **L**ogic **B**locks
- ☐ **B**lock **R**AM
- ☐ **D**igital **S**ignal **P**rocessing slices
- ☐ Advanced features

A "logic cell"

- "...the logical equivalent of a classic four-input **Lookup Up Table** and a flip-flop."
- 4-input LUT abbreviated as 4LUT. Many other numbers of inputs in commercial products.
 - A 4LUT can implement (and optionally store) any 4-variable logic function



Lookup table

A	B	C	D	Q
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

You program Q column, store in local SRAM

Hypothetical logic cell

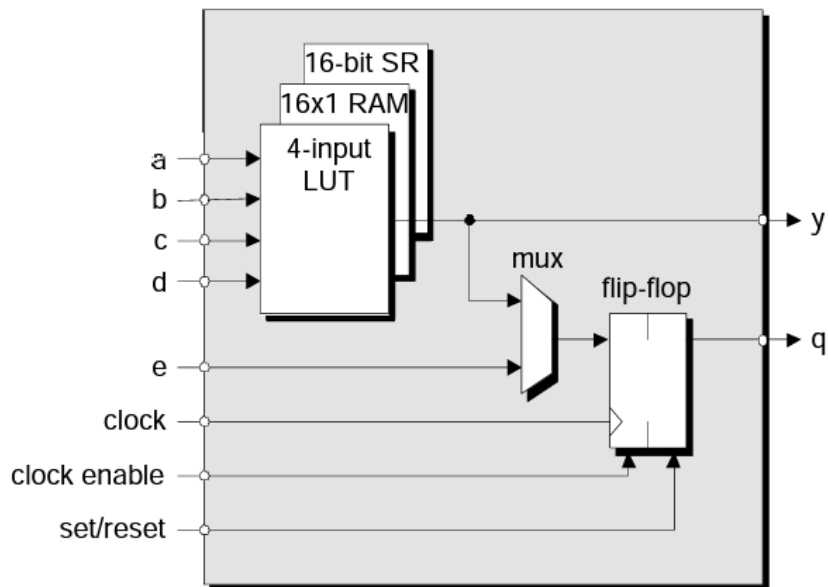
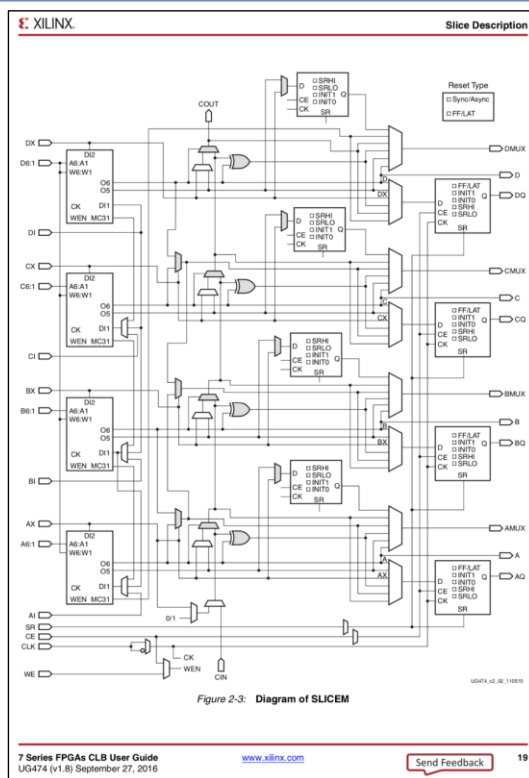


Figure 4-7. A simplified view of a Xilinx LC.

The Design Warrior's Guide to FPGAs by Clive "Max" Maxfield, Elsevier, 2004

Resource example: SLICEM



Configurable Logic Blocks (CLBs)

Table 1-2: Artix-7 FPGA CLB Resources


Device	Slices ⁽¹⁾	SLICEL	SLICEM	6-input LUTs	Distributed RAM (Kb)	Shift Register (Kb)	Flip-Flops
7A12T	2,000 ⁽²⁾	1,316	684	8,000	171	86	16,000
7A15T	2,600 ⁽²⁾	1,800	800	10,400	200	100	20,800
7A25T	3,650	2,400	1,250	14,600	313	156	29,200
7A35T	5,200 ⁽²⁾	3,600	1,600	20,800	400	200	41,600
7A50T	8,150	5,750	2,400	32,600	600	300	65,200
7A75T	11,800 ⁽²⁾	8,232	3,568	47,200	892	446	94,400
7A100T	15,850	11,100	4,750	63,400	1,188	594	126,800
7A200T	33,650	22,100	11,550	134,600	2,888	1,444	269,200

Notes:

1. Each 7 series FPGA slice contains four LUTs and eight flip-flops; only SLICEMs can use their LUTs as distributed RAM or SRLs.
2. Number of slices corresponding to the number of LUTs and flip-flops supported in the device.

7 Series FPGAs CLB User Guide UG474 (v1.8) September 27, 2016 -- ug474_7Series_CLB.pdf

Virtex FPGA clock speeds



Virtex UltraScale FPGAs Data Sheet: DC and AC Switching Characteristics

Clock Buffers and Networks

Table 35: Clock Buffers Switching Characteristics

Symbol	Description	Speed Grades and V _{CCINT} Operating Voltages				Units	
		1.0V		0.95V			
		-3	-1H	-2	-1		
Global Clock Switching Characteristics (Including BUFGCTRL)							
F _{MAX}	Maximum frequency of a global clock tree (BUFG)	850	725	725	630	MHz	
Global Clock Buffer with Input Divide Capability (BUFGCE_DIV)							
F _{MAX}	Maximum frequency of a global clock buffer with input divide capability (BUFGCE_DIV)	850	725	725	630	MHz	
Global Clock Buffer with Clock Enable (BUFGCE)							
F _{MAX}	Maximum frequency of a global clock buffer with clock enable (BUFGCE)	850	725	725	630	MHz	
Leaf Clock Buffer with Clock Enable (BUFCE_LEAF)							
F _{MAX}	Maximum frequency of a leaf clock buffer with clock enable (BUFCE_LEAF)	850	725	725	630	MHz	
GTH/GTY Clock Buffer with Clock Enable and Clock Input Divide Capability (BUFG_GT)							
F _{MAX}	Maximum frequency of a serial transceiver clock buffer with clock enable and clock input divide capability	512	512	512	512	MHz	

850 MHz --> 1.17 ns

Distributed RAM (SLICEM only)

- Faster in-slice memory

Table 2-3: Distributed RAM Configuration

RAM	Description	Primitive	Number of LUTs
32 x 1S	Single port	RAM32X1S	1
32 x 1D	Dual port	RAM32X1D	2
32 x 2Q	Quad port	RAM32M	4
32 x 6SDP	Simple dual port	RAM32M	4
64 x 1S	Single port	RAM64X1S	1
64 x 1D	Dual port	RAM64X1D	2
64 x 1Q	Quad port	RAM64M	4
64 x 3SDP	Simple dual port	RAM64M	4
128 x 1S	Single port	RAM128X1S	2
128 x 1D	Dual port	RAM128X1D	4
256 x 1S	Single port	RAM256X1S	4

7 Series FPGAs CLB User Guide UG474 (v1.8) September 27, 2016 -- ug474_7Series_CLB.pdf

- Ports depend on what needs to access memory and when
- Keep within RAM sizes to take advantage of speed benefits!
- "... read and write operations each have an associated address bus ... [t]his means that the read and write operations can be performed simultaneously."

- The Design Warrior's Guide to FPGAs by Clive "Max" Maxfield, Elsevier, 2004

Block RAM

- ❑ A block of RAM regionally separated from logic blocks
- ❑ 36 kbit blocks in 7-series Xilinx FPGAs
- ❑ 2x (72kb) and 1/2x (18 kb) with special properties
- ❑ True dual port, simple dual port, single port...
- ❑ "Widths greater than 16 bits should use block RAM, if available." (ug474)
 - Good advice but up to 72-bit width is possible
- ❑ Like distributed RAM, to your benefit to stick within bit size, width, port limitations to maximize performance
 - Less important if FPGA is an ASIC prototype vs. final device

Digital signal processing (DSP) slices

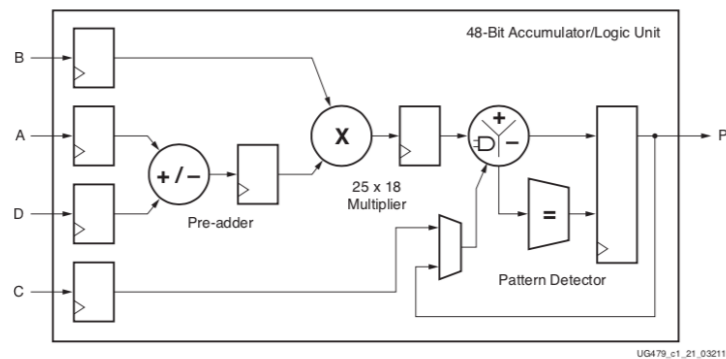


Figure 1-1: Basic DSP48E1 Slice Functionality

- "DSP applications use many binary multipliers and accumulators that are best implemented in dedicated DSP slices."
- Example features:
 - 25 × 18 two's-complement multiplier
 - 48-bit accumulator

Advanced features

- ❑ Popular pre-made functions: CMTs, PCIe, GTP, XADC ...
- ❑ Complete pre-made CPUs
 - Microblaze, picoblaze CPU cores are defined in software, "soft"
 - ARM core(s) in Zynq are "hard"
- ❑ "Slices" vs. "blocks" vs. "cores"
 - Slices are the most fundamental component in an FPGA
 - ❑ Probably have $\gg 1$ "logic cell"
 - ❑ Form the "FPGA fabric"
 - Blocks are monolithic and often unique
 - Cores are advanced unique functions (e.g., UART controller)
 - Block/core somewhat interchangeable

Review Roy's Homework #3 solution

Write_up: [ece351sp21_hw3_release_r1_0/docs/hw3_write_up.pdf](#)

Solution: [..\..\assignments\hw3\hdl](#)

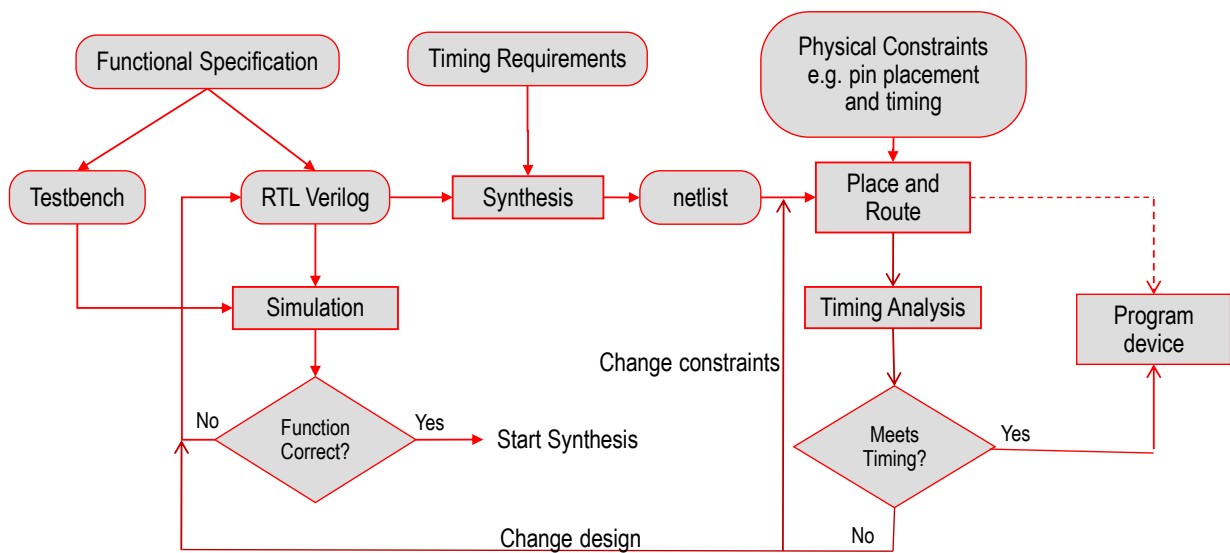
Results: [..\..\assignments\hw3\results](#)

Introduction to Xilinx Vivado

Sources:

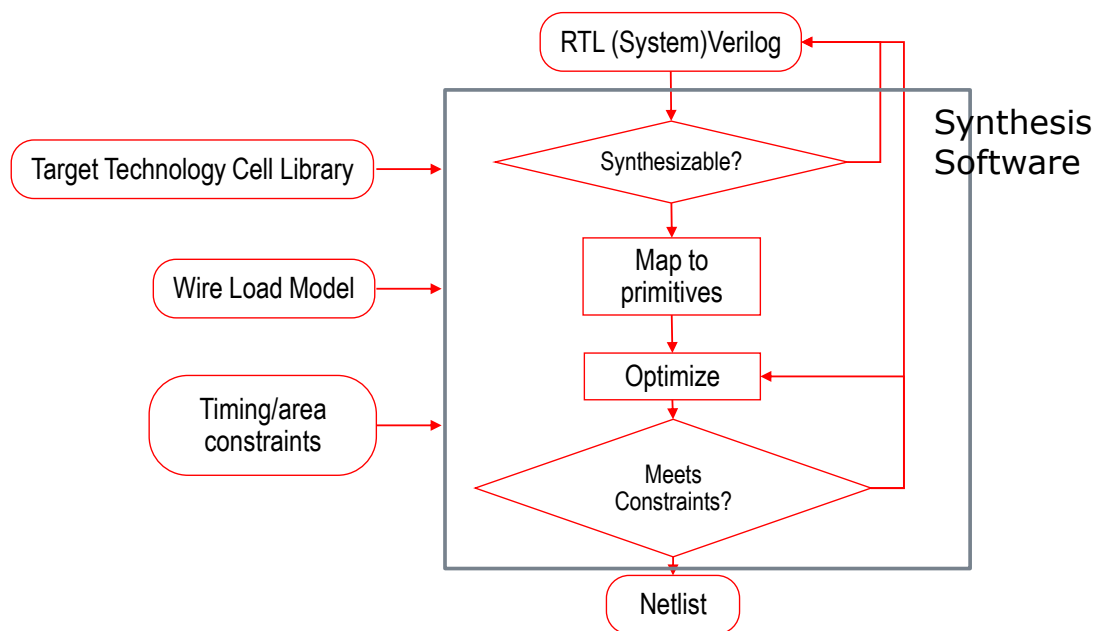
- FPGA design flow using Vivado workshop, Xilinx Corp.
- ECE 540 lecture notes by Brian Cruickshank
- ECE 540 lecture notes by Roy K. and David B.

FPGA design flow



Synthesis flow

20



Vivado IDE solution

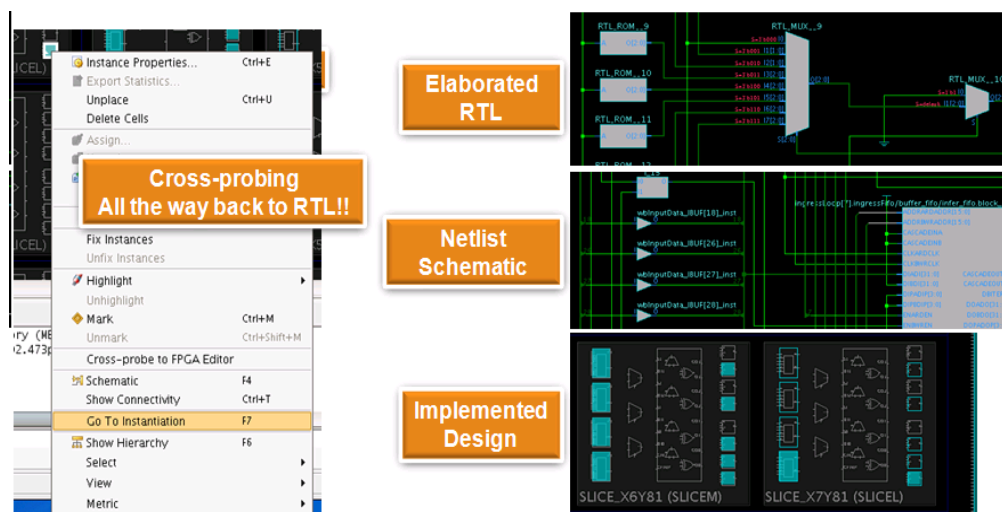
- Interactive design and analysis
 - Timing analysis, connectivity, resource utilization, timing constraint analysis, and entry
- RTL development and analysis
 - Elaboration of HDL
 - Hierarchical exploration
 - Schematic generation
- XSIM logic simulator integration
- Synthesis and implementation in one package
- I/O pin planning
 - Interactive rule-based I/O assignment

Who should use Vivado?

- ☐ Designers needing an interactive design approach
 - Analysis and area constraints to drive place & route
- ☐ Challenging designs
 - Large devices, complex constraints, and high device utilization
 - Advantages are also seen with small devices
- ☐ Designs experiencing implementation issues
 - Performance, capacity, run time, and repeatability
- ☐ Designs requiring implementation control
 - Users looking for options other than just a pushbutton flow
 - Visualize design issues from many aspects
 - Block-based design constraints
- ☐ Designs targeting the 7-Series (or newer) devices

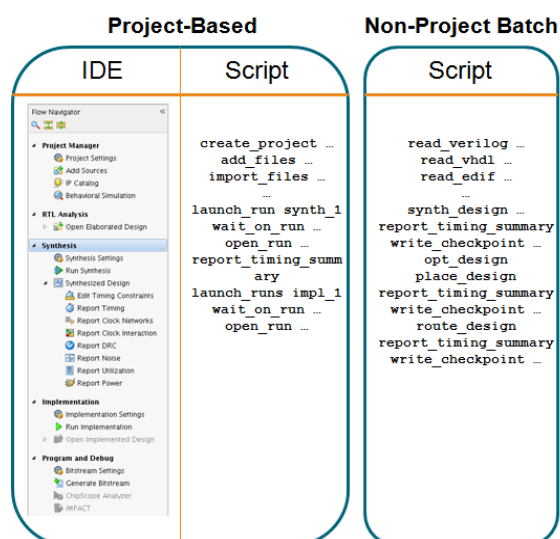
Vivado's visualization feature

- Visualize and debug your design at any flow stage
 - Cross-probing between netlist/schematic/RTL



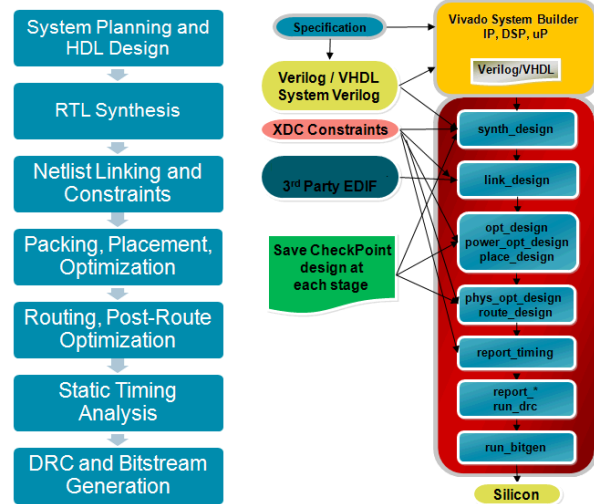
Project and non-project batch flows

- Vivado tools support two flows
 - Project based
 - Non-project batch
- Non-project batch flow
 - No project infrastructure
 - Tcl based
 - Can use GUI for visualization via the start_gui command
 - Must manually create reports and checkpoints via commands
- Project-based flow
 - Project infrastructure is saved in *.XPR file
 - Reports/state/runs/cross-probing is available
 - IDE GUI or Tcl script both available



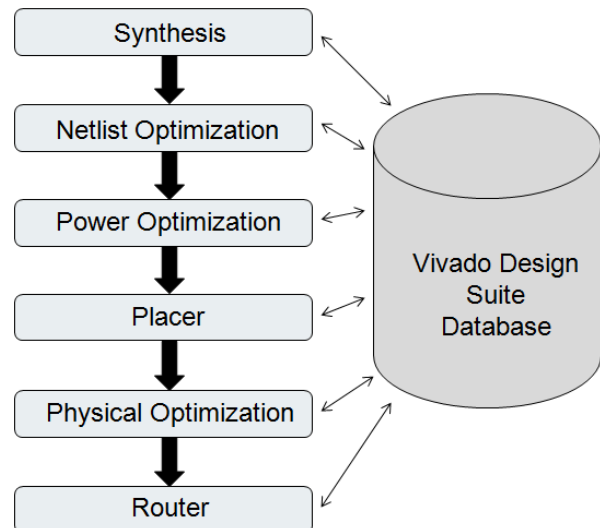
Vivado design flow

- Interactive IP plug-n-play environment
 - AXI4, IP_XACT
- Common constraint language (XDC) throughout flow
 - Apply constraints at any stage
- Reporting at any stage
 - Robust Tcl API
- Common data model throughout the flow
 - "In memory" model improves speed
 - Generate reports at all stages
- Save checkpoint designs at any stage
 - Netlist, constraints, place and route results



Design database

- Processes access the underlying database of your design
 - Each process operates on a netlist and will modify the netlist or create a new netlist
- Different netlists are used throughout the design process
 - Elaborated
 - Synthesized
 - Implemented

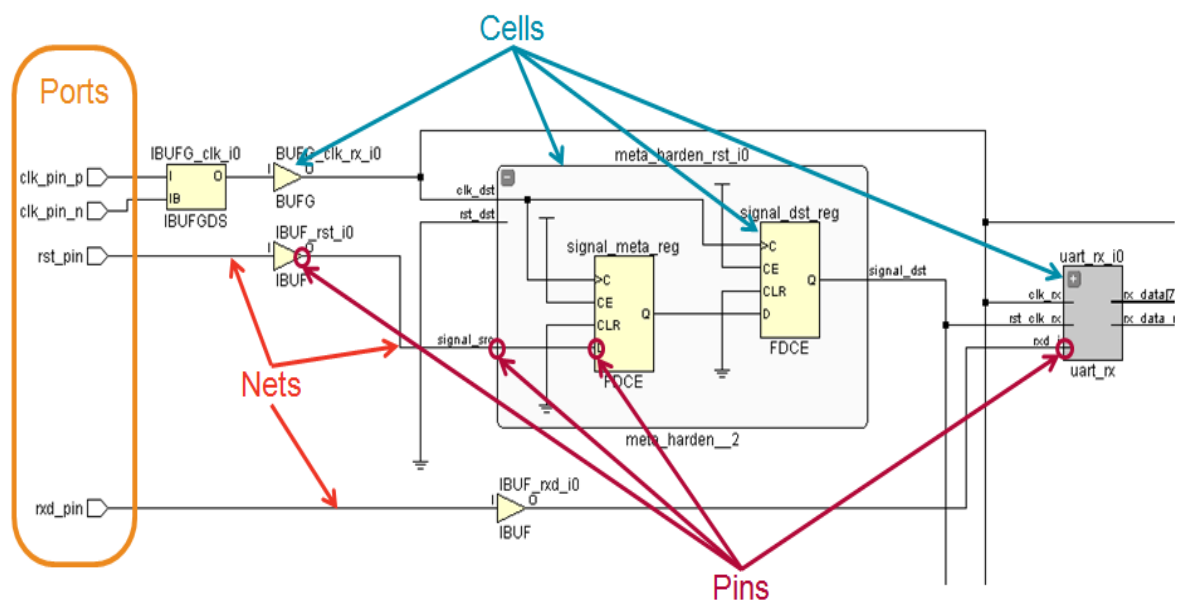


What is a netlist?

- A Netlist is a description of your design
 - Consists of cells, pins, port and nets
 - Cells are design objects
 - Instances of user modules/entities
 - Instances of library Basic Elements (BELs)
 - LUTs, FF, RAMs, DSP cells, etc...
 - Generic technology representations of hardware functions
 - Black boxes
 - Pins are connection points on cells
 - Ports are the top level ports of your design
 - Nets make connections between pins and from pins to ports

Netlist objects

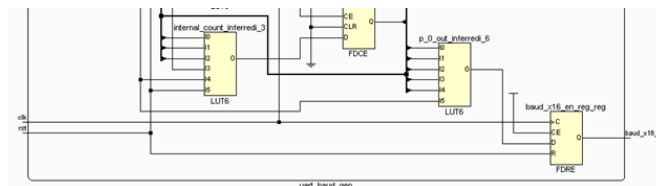
28



-

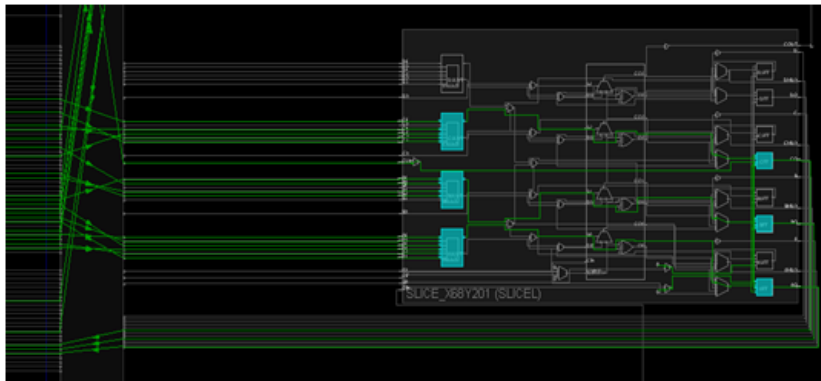
Synthesized design

- Representation of the design after synthesis
 - Interconnected netlist of hierarchical and Basic Elements (BELs)
 - Instances of modules/entities
 - BELs
 - LUTs, flip-flops, carry chain elements, wide MUXes
 - Block RAMs, DSP cells
 - Clocking elements (BUFG, BUFR, MMCM, ...)
 - I/O elements (IBUF, OBUF, I/O flip-flops)
- Object names are the same as names in the elaborated netlist when possible



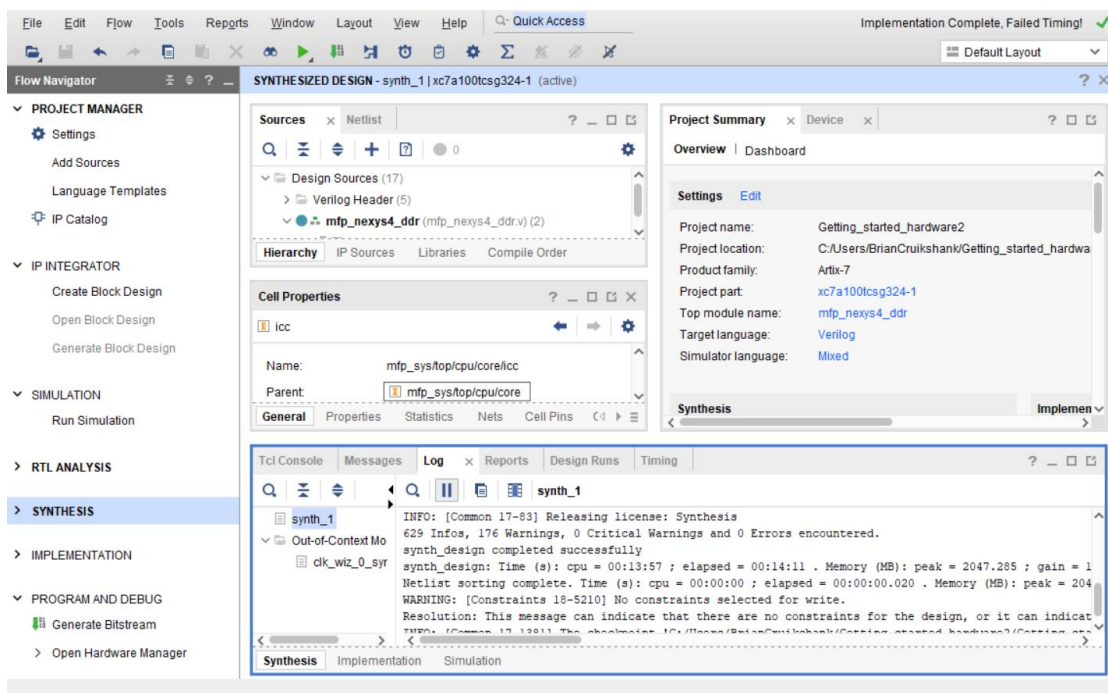
Implemented design

- Representation of the design during and after the implementation process
 - Structurally similar to the Synthesized Design
 - Cells have locations, and nets are mapped to specific routing channels



Main Vivado screen

32



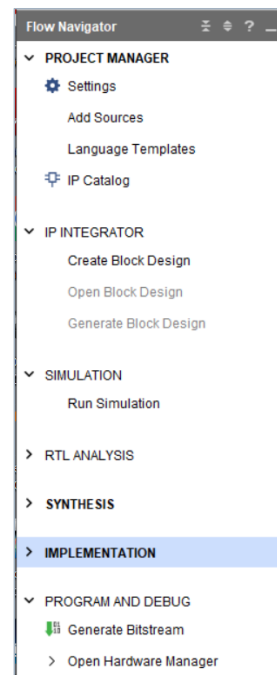
ECE 351 Verilog and FPGA Design

TCL console

- ☐ Vivado also has a TCL console
- ☐ Very similar to Synopsys/Cadence commands
- ☐ There is a Tools->Xilinx TCL store with different options to help you with design and timing convergence

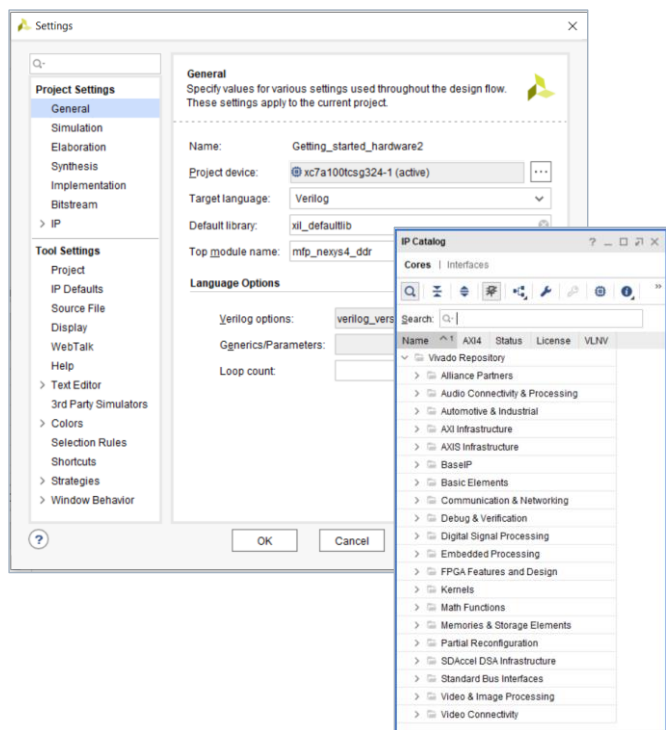
Vivado flow stages

- ☐ Project Manager
- ☐ IP Integrator
- ☐ Simulation
- ☐ RTL Analysis
- ☐ Synthesis
- ☐ Implementation
- ☐ Program and Debug



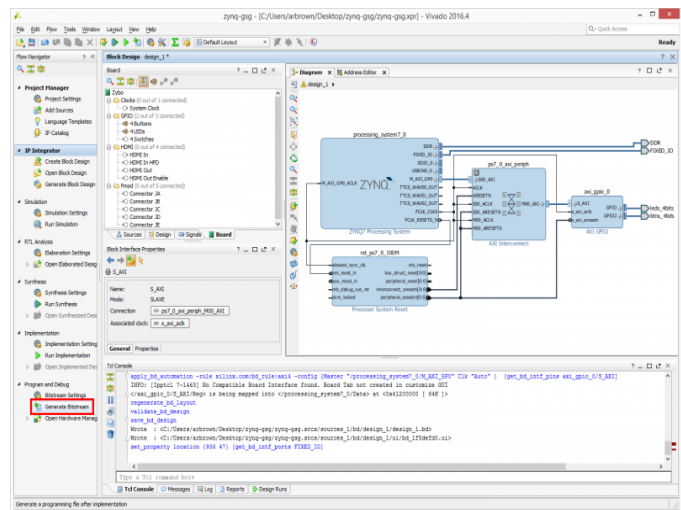
Vivado project manager

- ☐ Settings
- ☐ Add Sources
- ☐ Language Templates
- ☐ IP Catalog



Vivado **IP** Integrator

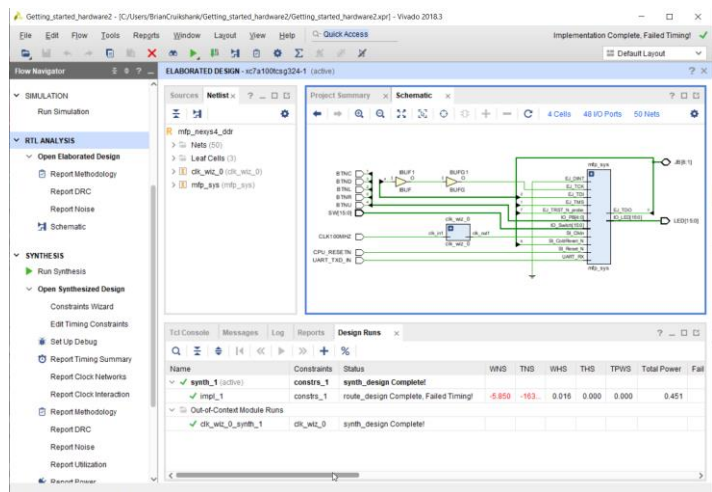
- ☐ Create Block Design
- ☐ Open Block Design
- ☐ Generate Block Design



<https://reference.digilentinc.com/vivado/getting-started-with-ipi/2018.2>

Vivado RTL analysis

- Open Elaborated design
 - Report Methodology
 - Report DRC
 - Report Noise
 - Schematic
- Schematics at RTL Analysis
 - No optimization, just reading RTL



Vivado synthesis

- Run Synthesis
- Open Synthesized Design
 - Constraints Wizard
 - Edit Timing Constraints
 - Set Up Debug
 - Report Timing Summary
 - Report Clock Networks
 - Report Clock Interaction
 - Report Methodology
 - Report DRC
 - Report Noise
 - Report Utilization
 - Report Power
 - Schematic

Schematics at Vivado synthesis level

- Schematics are mapped to a netlist
 - A netlist is a Verilog netlist mapping to LUT cells, Carry Logic, DSP, RAMs, Ios, clock circuits, etc standard elements in the FPGA.
 - No behavioral code is left.
 - No always blocks. No functions. No ternary operators.
- Schematics/netlist is optimized for timing
 - Serial constructs will be made parallel for area or timing
 - For example, XOR parity structure can be made into a tree instead of xoring each bit in series
- But it is optimized with no physical knowledge
 - Net timing is only approximated
- You can write_verilog from TCL console

Solve problems early

- Finding timing problem at Synthesis saves time
 - Implementation takes a long time to run
 - Many times, the timing problem exists at Synthesis
- Debug the timing problems at Synthesis first

Report timing summary

- Many categories of reports, lots of terms
 - WNS
 - TNS
 - Failing Endpoints
 - Check Timing

The screenshot shows the 'Design Timing Summary' report. The left sidebar lists various report categories, with 'Design Timing Summary' selected. The main content area displays a table with three columns: Setup, Hold, and Pulse Width. The table contains summary statistics for Worst Negative Slack (WNS), Total Negative Slack (TNS), Number of Failing Endpoints, and Total Number of Endpoints for each category. A red status message at the bottom indicates that timing constraints are not met.

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): -6.938 ns	Worst Hold Slack (WHS): 0.013 ns	Worst Pulse Width Slack (WPWS): 3.000 ns
Total Negative Slack (TNS): -8231.170 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 5450	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 18058	Total Number of Endpoints: 18058	Total Number of Endpoints: 8155

Timing constraints are not met.

Vivado implementation

- Run Implementation
- Open Implemented Design
 - Constraints Wizard
 - Edit Timing Constraints
 - Report Timing Summary
 - Report Clock Networks
 - Report Clock Interaction
 - Report Methodology
 - Report DRC
 - Report Noise
 - Report Utilization
 - Report Power
 - Schematic

Implementation

- ❑ Maps the LUTs, SRAMs, I/O's, etc. to physical locations on the desired FPGA
- ❑ Nets are mapped to global, section, local nets
- ❑ Real net delays are used.
- ❑ Locations of components are moved around to improve timing
- ❑ Nets are split, re-driven
- ❑ Iteration happens on the worst paths until nothing more can be improved.
 - Sometimes only the worst is optimized (WNS based), sometimes all paths down to passing is optimized (TNS based)
- ❑ Different settings to trade-off runtime, area, performance

Report timing summary

- Many categories of reports, lots of terms

Synthesis

The screenshot shows the 'Design Timing Summary' report for the Synthesis stage. The report is divided into three main sections: Setup, Hold, and Pulse Width. The Setup section shows Worst Negative Slack (WNS) at -6.938 ns, Total Negative Slack (TNS) at -8231.170 ns, and 5450 failing endpoints. The Hold section shows Worst Hold Slack (WHS) at 0.013 ns and 0 failing endpoints. The Pulse Width section shows Worst Pulse Width Slack (WPWS) at 3.000 ns and 0 failing endpoints. The total number of endpoints is 18058 for Setup and Hold, and 8155 for Pulse Width. A note at the bottom states 'Timing constraints are not met.'

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): -6.938 ns	Worst Hold Slack (WHS): 0.013 ns	Worst Pulse Width Slack (WPWS): 3.000 ns
Total Negative Slack (TNS): -8231.170 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 5450	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 18058	Total Number of Endpoints: 18058	Total Number of Endpoints: 8155

Timing constraints are not met.

Implementation

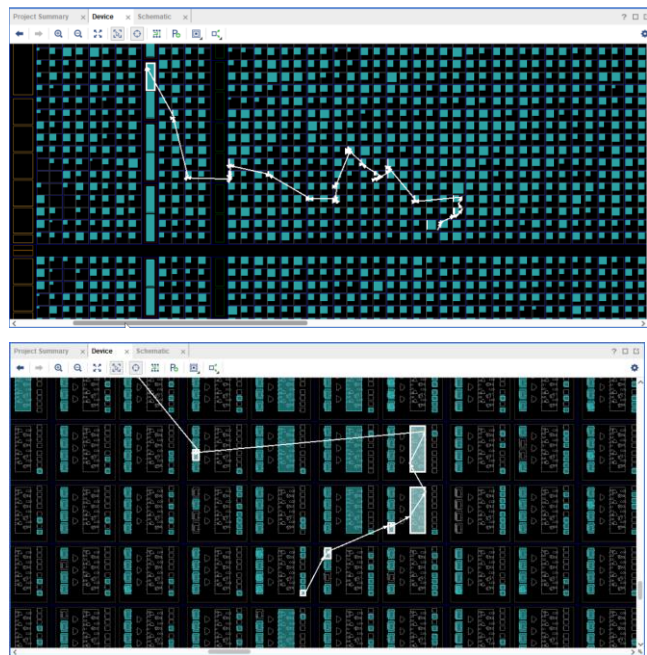
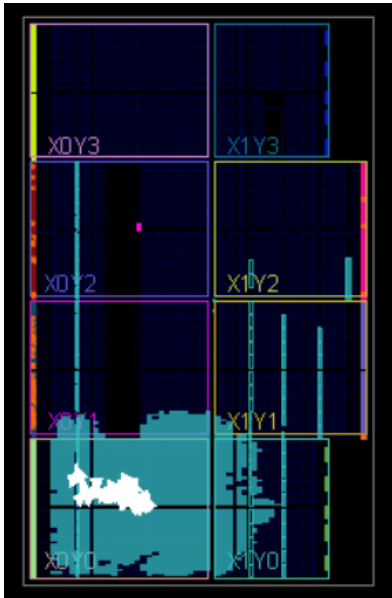
The screenshot shows the 'Design Timing Summary' report for the Implementation stage. The report is divided into three main sections: Setup, Hold, and Pulse Width. The Setup section shows Worst Negative Slack (WNS) at -5.850 ns, Total Negative Slack (TNS) at -16302.873 ns, and 6418 failing endpoints. The Hold section shows Worst Hold Slack (WHS) at 0.016 ns and 0 failing endpoints. The Pulse Width section shows Worst Pulse Width Slack (WPWS) at 3.000 ns and 0 failing endpoints. The total number of endpoints is 18090 for Setup and Hold, and 8155 for Pulse Width. A note at the bottom states 'Timing constraints are not met.'

Setup	Hold	Pulse Width
Worst Negative Slack (WNS): -5.850 ns	Worst Hold Slack (WHS): 0.016 ns	Worst Pulse Width Slack (WPWS): 3.000 ns
Total Negative Slack (TNS): -16302.873 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 6418	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 18090	Total Number of Endpoints: 18090	Total Number of Endpoints: 8155

Timing constraints are not met.

Implementation

- Zoom in different levels





Vivado project: C:\Users\roykr\psuproj\ece351sp21\pmodssd_demo

ECE 351 Verilog and FPGA Design

Next Time

- ☐ Topics (last lecture):
 - JTAG, SoC's, etc.
 - Review for final exam (incl. trying to answer D2L questions)
 - Catch-up, wrap-up
- ☐ You should:
 - Be working on your HW #4 – you only have a week
 - Add your questions for the final review class meeting (03-Jun) to the Discussion forum in Ask the Instructor/For the final review
- ☐ Homework, projects and quizzes
 - Homework #4 has been released
 - ☐ Due to D2L by 10:00 PM on 02-Jun. No late assignments accepted after Noon on Thu, 03-Jun
- ☐ **Final exam scheduled for Mon 07-Jun from 10:15 AM – 12:05 PM**
 - Poll: Should I increase the final exam time to 10:15 AM – 12:45 PM?
 - Extra credit opportunity: Submit final exam questions/solutions to D2L by 10:00 PM on Friday, 04-Jun (up to 8 points added to your midterm exam score)