ECE 351
Verilog and FPGA Design

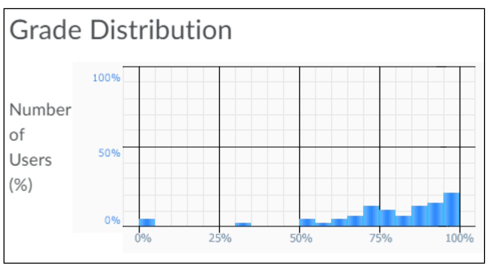
Week 9_2: **Serial communications (wrap-up)**
FPGA overview
Introduction to Xilinx Vivado (if time)

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

1

Midterm Exam Statistics

High:	> 100 (tops at 100)
Low:	< 55 (several)
Average:	81.25
Median:	84
Std Dev:	~15%



Extra credit opportunity (up to 5 pts added to your midterm exam score): Email final exam questions/solution to me by 10:00 PM on Fri, 04-Jun
- I will post the "rules" in Announcements

Portland State UNIVERSITY

2

Serial Communication (wrap-up)

Review: Serial Communication (cont'd)

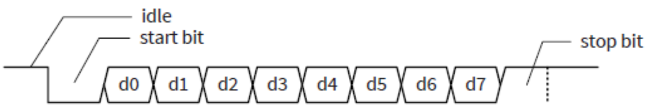


Figure 12.1 Transmission of a byte.

- Serial line is 1 when idle
- Transmission starts with a 1->0 transition called the start bit followed by 6, 7, or 8 data bits and an optional parity bit (odd, even, or none)
  - ex: Odd parity- parity bit set to 0 when data bits have an odd number of 1's
- Transmission ends with a 1, 1.5, or 2 stop bits
- LSB (least significant bit) of data is transmitted first
- No separate clock – receiver uses *oversampling* scheme to retrieve/recover the data bits

Source: FPGA Prototyping by SystemVerilog Examples by Pong Chu

Review: Oversampling procedure

5

- Oversampling => sampling each serial bit multiple times and captures the value ~ ½ way through the bit time
  - Most commonly used sampling rate is 16x the baud rate (number of bits per second)
- Works as follows (sampling rate 16x, N data bits, M stop bits):
  1. Wait until the incoming signal is 0 (beginning of start bit) and start the sampling "tick counter"
  2. When the sampling tick counter reaches 7 (middle point of start bit) clear the counter to 0 and restart
  3. When the tick counter reaches 15 (the middle of the first data bit) shift the value of the data bit into a register and restart the tick counter
  4. Repeat Step 3 N-1 more times to retrieve the remaining data bits
  5. If the optional parity bits is used repeat step 3 one time to obtain the parity bit
  6. Repeat Step 3 M more times to obtain the stop bit(s)

UART block diagram

6

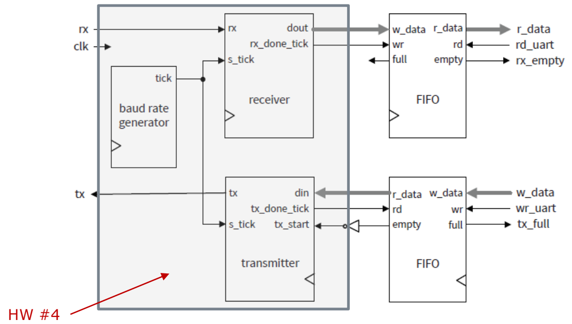
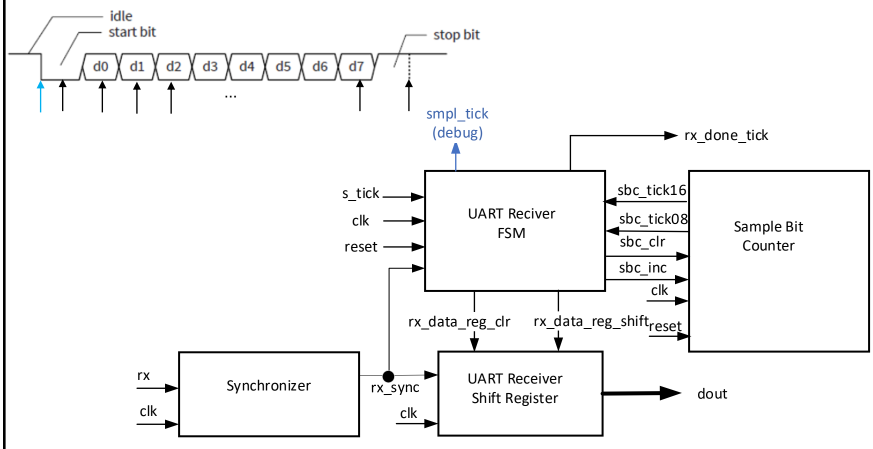


Figure 12.2 Block diagram of a complete UART.

- Baud rate generator – generates sampling signal that is 16x the UART's designated baud rate
- Receiver – obtains the data byte from the serial line via oversampling
- Transmitter – shifts the bits in a data byte out one bit at a time at the specified rate
- FIFO – Provide buffers for transmitted and received data for the CPU. Allows CPU to process a *burst* of data

Source: FPGA Prototyping by SystemVerilog Examples by Pong Chu

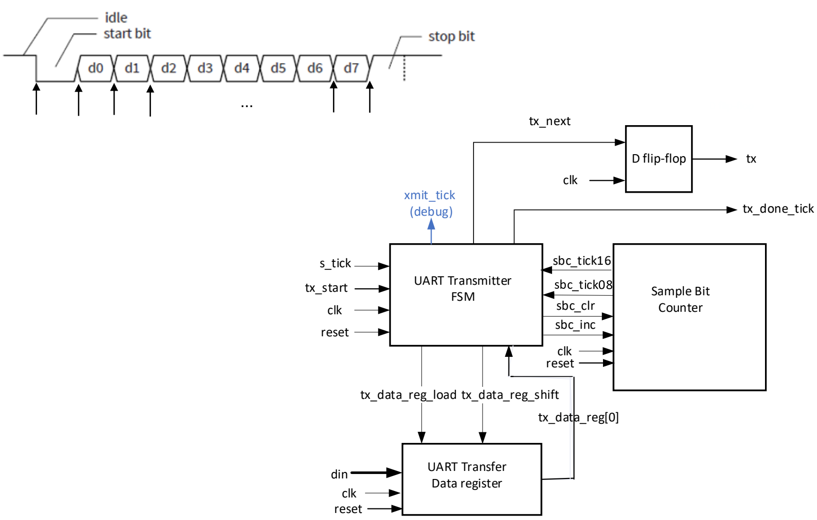
UART serial receiver FSMD



UART Receiver

Source: ..\..\assignments\hw4\hdl\prob1\hdl\uart_rx.sv

UART serial transmitter FSMD



UART Transmitter

Questions about Homework #4

Write-up: ece351sp21_hw4_release_r1_1\docs\ece351sp21_hw4.pdf

FPGA Overview

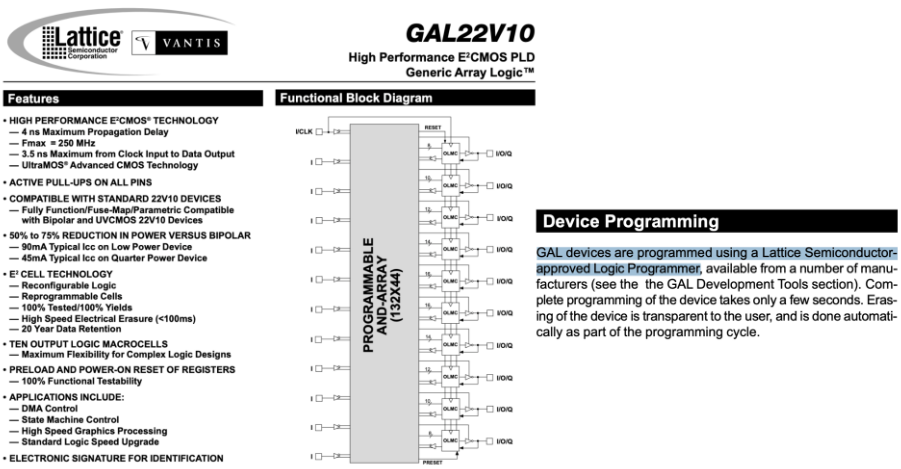
Sources:
- ECE 540 lecture notes by David B. and Roy K.

Field-**p**rogrammable **g**ate **a**rray (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

Programmable logic controllers (PLCs)

- Close but not quite "field" programmable; only logic



GAL22V10

High Performance E²CMOS PLD
Generic Array Logic™

Features

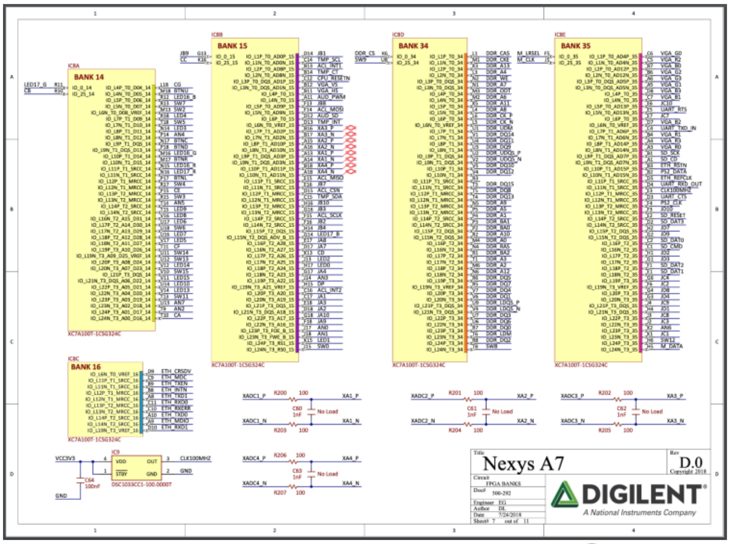
- HIGH PERFORMANCE E²CMOS® TECHNOLOGY
  - 4 ns Maximum Propagation Delay
  - Fmax = 250 MHz
  - 3.5 ns Maximum from Clock Input to Data Output
  - UltraMOS® Advanced CMOS Technology
- ACTIVE PULL-UPS ON ALL PINS
- COMPATIBLE WITH STANDARD 22V10 DEVICES
  - Fully Function/Fuse-Map/Parametric Compatible with Bipolar and UVCMOS 22V10 Devices
- 50% to 75% REDUCTION IN POWER VERSUS BIPOLAR
  - 90mA Typical Icc on Low Power Device
  - 45mA Typical Icc on Quarter Power Device
- E² CELL TECHNOLOGY
  - Reconfigurable Logic
  - Reprogrammable Cells
  - 100% Tested/100% Yields
  - High Speed Electrical Erasure (<100ms)
  - 20 Year Data Retention
- TEN OUTPUT LOGIC MACROCELLS
  - Maximum Flexibility for Complex Logic Designs
- PRELOAD AND POWER-ON RESET OF REGISTERS
  - 100% Functional Testability
- APPLICATIONS INCLUDE:
  - DMA Control
  - State Machine Control
  - High Speed Graphics Processing
  - Standard Logic Speed Upgrade
- ELECTRONIC SIGNATURE FOR IDENTIFICATION

Functional Block Diagram

Device Programming

GAL devices are programmed using a Lattice Semiconductor-approved Logic Programmer, available from a number of manufacturers (see the the GAL Development Tools section). Complete programming of the device takes only a few seconds. Erasing of the device is transparent to the user, and is done automatically as part of the programming cycle.

Nexys A7 schematic (only the FPGA)



https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-a7/nexys-a7-sch.pdf

XC7A100T-1**CSG324**C Ball Grid Array (BGA)

CS/CSG324 and CS/CSG325 (Artix-7 FPGAs)
Wire-Bond Chip-Scale BGA (0.8 mm Pitch)

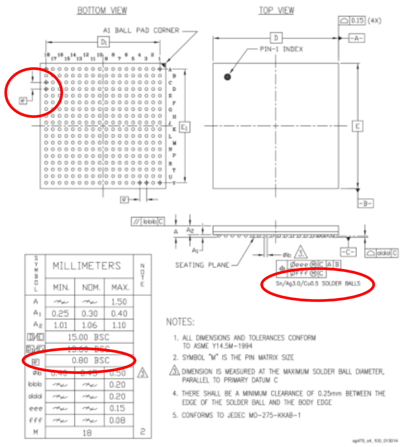


Figure 4-9: **CS/CSG324 and CS/CSG325 Wire-Bond Chip-Scale BGA Package Specifications for Artix-7 FPGAs**

- 0.8 mm ~= 31.5 mil = 0.0315"
- ("Basic Spacing between Centers")
- Typical PCB manufacturing
  - 0.004" trace/space ($$$)
  - https://www.sunstone.com/pcb-products/pcb-manufacturing/pcbpro-full-feature
  - Fit, at most, 3 wires between solder balls
- Solder composition given

https://www.xilinx.com/support/documentation/user_guides/ug475_7Series_Pkg_Pinout.pdf

Multi-function pins

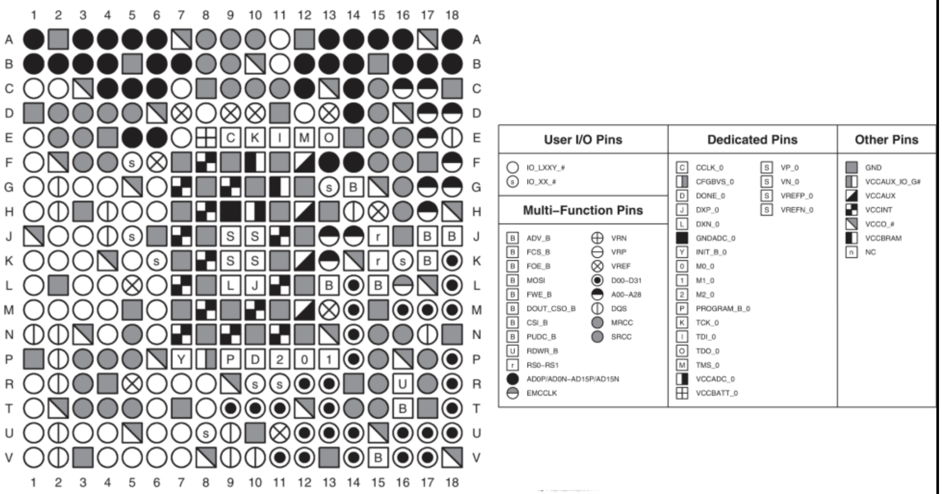


Figure 3-49: **CS324 and CSG324 Packages—XC7A15T, XC7A35T, XC7A50T, XC7A75T, and XC7A100T CSG324 Packages (only)—XA7A15T, XA7A35T, XA7A50T, XA7A75T, and XA7A100T Pinout Diagram**

https://www.xilinx.com/support/documentation/user_guides/ug475_7Series_Pkg_Pinout.pdf

"Pinout Planning"

- "The best approach is to let the tools choose the I/O locations based on the FPGA requirements. Results can be adjusted if necessary for board layout considerations. The timing constraints should be set so that the tools can choose optimal placement for the design requirements."

…but if you can't

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

Specifying constraints

XDC Constraints XDC constraints are a combination of:

- Industry standard Synopsys Design Constraints (SDC), and
- Xilinx proprietary physical constraints

- XDC constraints have the following properties:
  - They are not simple strings but are commands that follow the Tcl semantic.
  - They can be interpreted like any other Tcl command by the Vivado Tcl interpreter.
  - They are read in and parsed sequentially the same as other Tcl commands.

Nexys A7 constraints file (for PmodSSD demo): nexysA7fpga.xdc

Xilinx FPGA families

Table 1: **7 Series Families Comparison**

Max. Capability	Spartan-7	Artix-7	Kintex-7	Virtex-7
Logic Cells	102K	215K	478K	1,955K
Block RAM[1]	4.2 Mb	13 Mb	34 Mb	68 Mb
DSP Slices	160	740	1,920	3,600
DSP Performance[2]	176 GMAC/s	929 GMAC/s	2,845 GMAC/s	5,335 GMAC/s
MicroBlaze CPU[3]	260 DMIPs	303 DMIPs	438 DMIPs	441 DMIPs
Transceivers	–	16	32	96
Transceiver Speed	–	6.6 Gb/s	12.5 Gb/s	28.05 Gb/s
Serial Bandwidth	–	211 Gb/s	800 Gb/s	2,784 Gb/s
PCIe Interface	–	x4 Gen2	x8 Gen2	x8 Gen3
Memory Interface	800 Mb/s	1,066 Mb/s	1,866 Mb/s	1,866 Mb/s
I/O Pins	400	500	500	1,200
I/O Voltage	1.2V–3.3V	1.2V–3.3V	1.2V–3.3V	1.2V–3.3V
Package Options	Low-Cost, Wire-Bond	Low-Cost, Wire-Bond, Bare-Die Flip-Chip	Bare-Die Flip-Chip and High-Performance Flip-Chip	Highest Performance Flip-Chip

Artix-7 FPGA Feature Summary

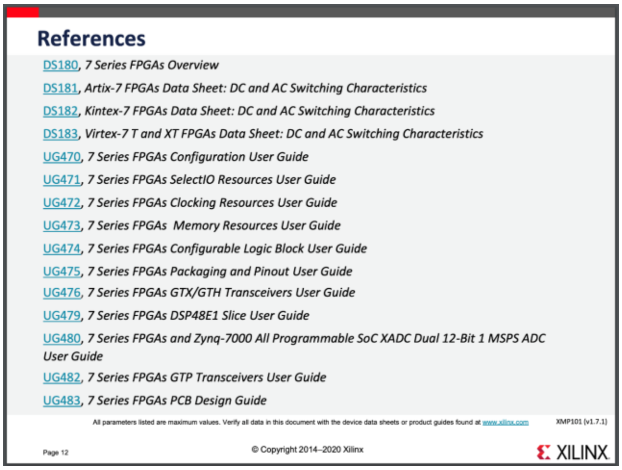
Table 4: **Artix-7 FPGA Feature Summary by Device**

Device	Logic Cells	Configurable Logic Blocks (CLBs)		DSP48E1 Slices[2]	Block RAM Blocks[3]			CMTs[4]	PCIe[5]	GTPs	XADC Blocks	Total I/O Banks[6]	Max User I/O[7]
		Slices[1]	Max Distributed RAM (Kb)		18 Kb	36 Kb	Max (Kb)						
XC7A12T	12,800	2,000	171	40	40	20	720	3	1	2	1	3	150
XC7A15T	16,640	2,600	200	45	50	25	900	5	1	4	1	5	250
XC7A25T	23,360	3,650	313	80	90	45	1,620	3	1	4	1	3	150
XC7A35T	33,280	5,200	400	90	100	50	1,800	5	1	4	1	5	250
XC7A50T	52,160	8,150	600	120	150	75	2,700	5	1	4	1	5	250
XC7A75T	75,520	11,800	892	180	210	105	3,780	6	1	8	1	6	300
XC7A100T	101,440	15,850	1,188	240	270	135	4,860	6	1	8	1	6	300
XC7A200T	215,360	33,650	2,888	740	730	365	13,140	10	1	16	1	10	500

https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf

Documentation aplenty

19



References

DS180, *7 Series FPGAs Overview*

DS181, *Artix-7 FPGAs Data Sheet: DC and AC Switching Characteristics*

DS182, *Kintex-7 FPGAs Data Sheet: DC and AC Switching Characteristics*

DS183, *Virtex-7 T and XT FPGAs Data Sheet: DC and AC Switching Characteristics*

UG470, *7 Series FPGAs Configuration User Guide*

UG471, *7 Series FPGAs SelectIO Resources User Guide*

UG472, *7 Series FPGAs Clocking Resources User Guide*

UG473, *7 Series FPGAs Memory Resources User Guide*

UG474, *7 Series FPGAs Configurable Logic Block User Guide*

UG475, *7 Series FPGAs Packaging and Pinout User Guide*

UG476, *7 Series FPGAs GTX/GTH Transceivers User Guide*

UG479, *7 Series FPGAs DSP48E1 Slice User Guide*

UG480, *7 Series FPGAs and Zynq-7000 All Programmable SoC XADC Dual 12-Bit 1 MSPS ADC User Guide*

UG482, *7 Series FPGAs GTP Transceivers User Guide*

UG483, *7 Series FPGAs PCB Design Guide*

All parameters listed are maximum values. Verify all data in this document with the device data sheets or product guides found at www.xilinx.com XMP101 (v1.7.1)

Page 12 © Copyright 2014–2020 Xilinx XILINX

Xilinx FPGA product selection guide:
https://www.xilinx.com/support/documentation/selection-guides/7-series-product-selection-guide.pdf

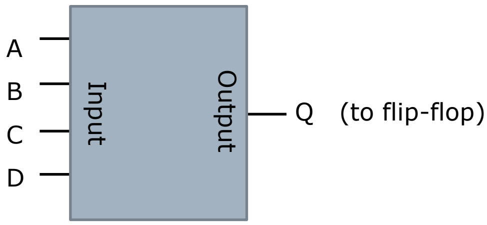
***Feature comparison

20

- 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 **L**ookup **U**p **T**able 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

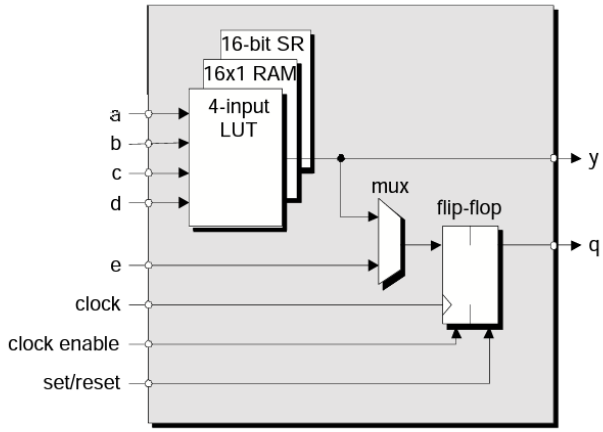
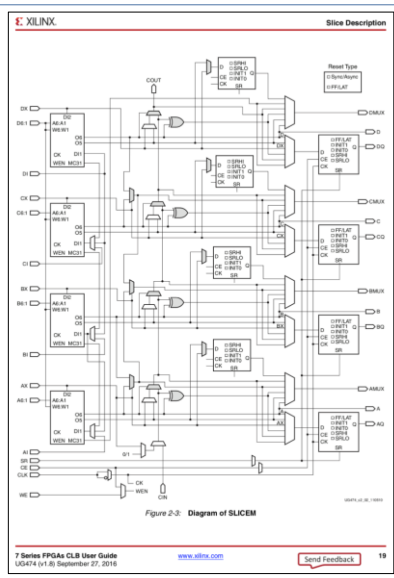


Figure 4-7. A simplified view of a Xilinx LC.

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

XILINX. 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 **s**ignal **p**rocessing (DSP) slices

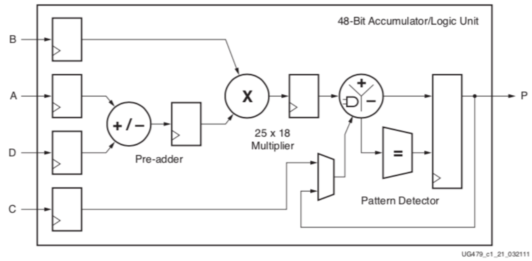


Figure 1-1: **Basic DSP48E1 Slice Functionality**

- "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

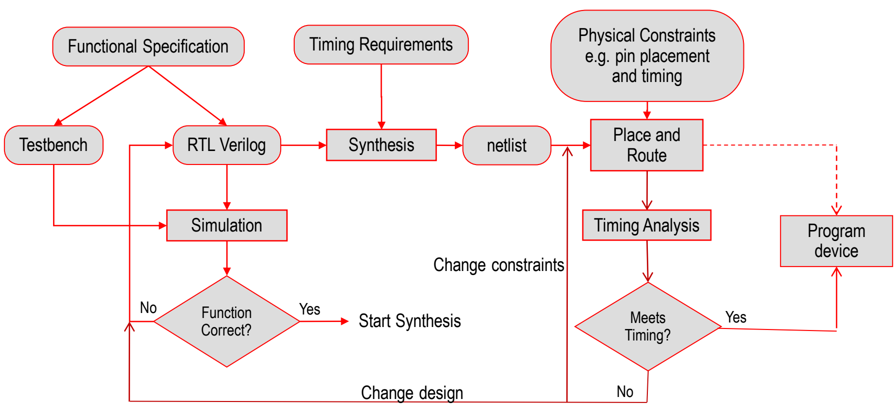
- Popular pre-made functions: CMTs, PCIe, GTP, XADC ...
- Complete pre-made CPUs
  - Microblaze, picoblaze CPU cores are defined in software, "soft"
  - ARM core(s) in Zynq are "hard"
- "Slices" vs. "blocks" vs. "cores"
  - Slices are the most fundamental component in an FPGA
    - Probably have >>1 "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

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.

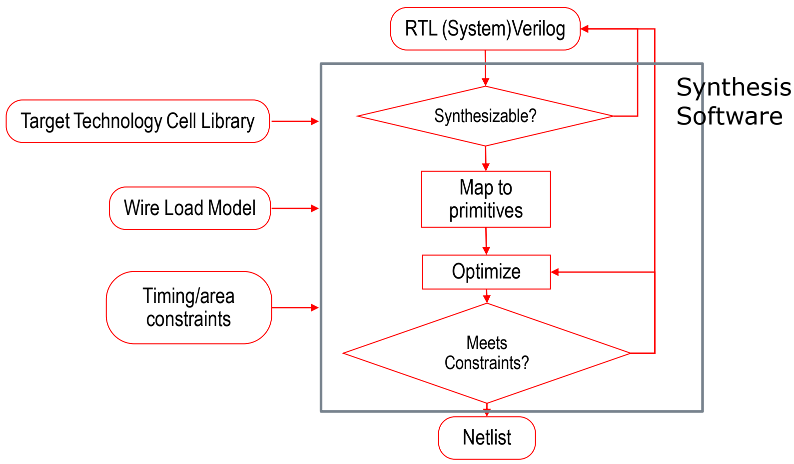
Portland State UNIVERSITY

FPGA design flow



Portland State UNIVERSITY

Synthesis flow



Vivado IDE solution

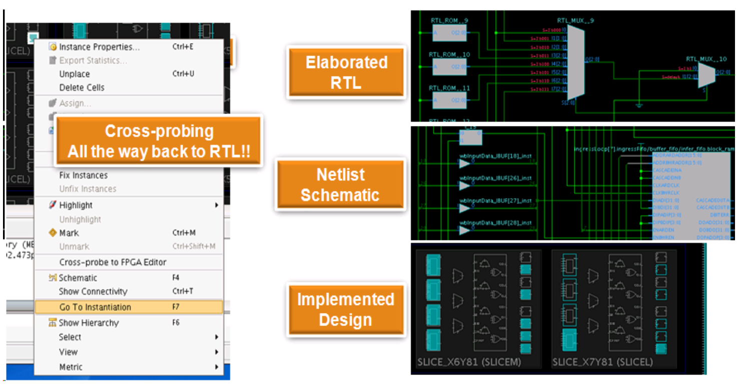
- Interactive design and analysis
  - Timing analysis, connectivity, resource utilization, timing constraint analysis, and entry
- RTL development and analysis
  - Elaboration of HDL
  - Hierarchical exploration
  - Schematic generation
- XSIM logic simulator integration
- Synthesis and implementation 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

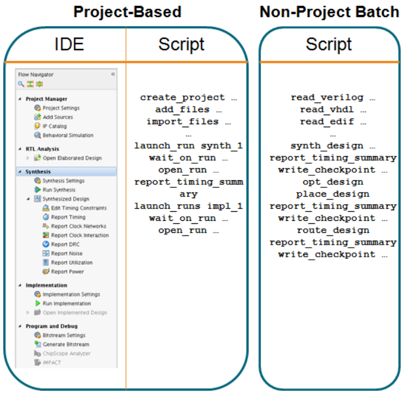
Vivado's visualization feature

- 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



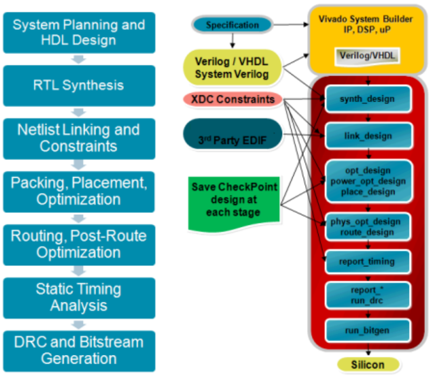
ECE 351 Verilog and FPGA Design

Portland State UNIVERSITY

38

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



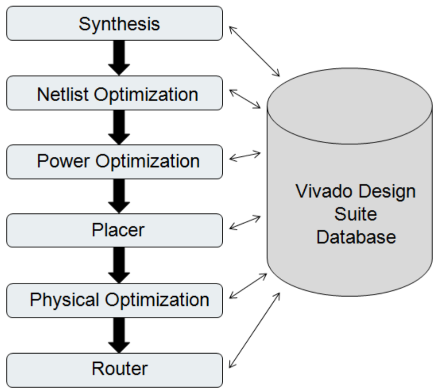
ECE 351 Verilog and FPGA Design

Portland State UNIVERSITY

39

Design database

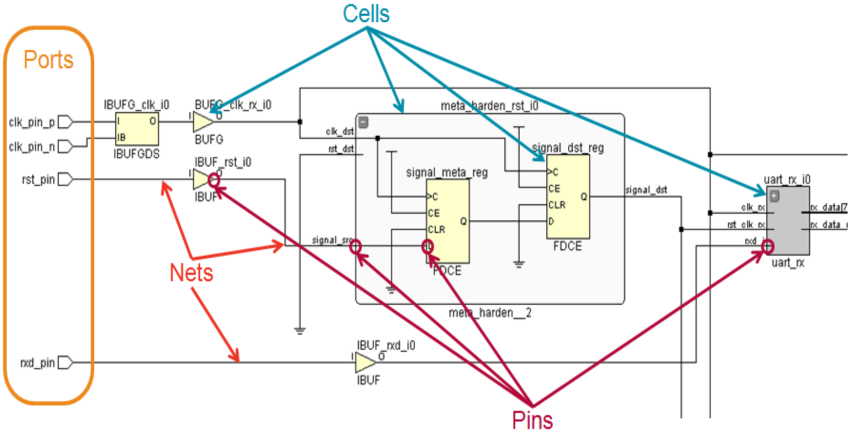
- Processes access the underlying database of your design
  - Each process operates on a netlist and will modify the netlist or create a new netlist
- Different netlists are used throughout the design process
  - Elaborated
  - Synthesized
  - Implemented



What is a Netlist?

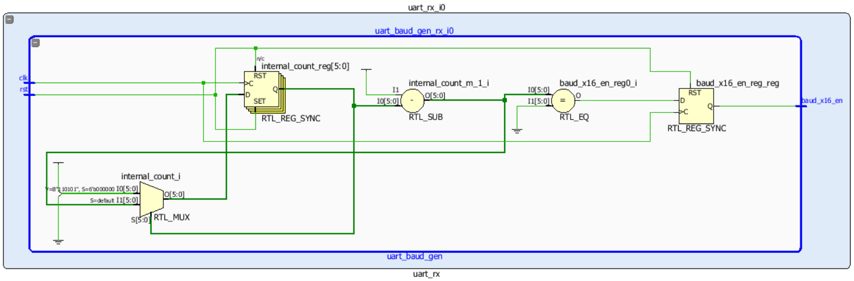
- A Netlist is a description of your design
  - Consists of cells, pins, port and nets
  - Cells are design objects
    - Instances of user modules/entities
    - Instances of library Basic Elements (BELs)
      - LUTs, FF, RAMs, DSP cells, etc…
    - Generic technology representations of hardware functions
    - 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



Elaborated design

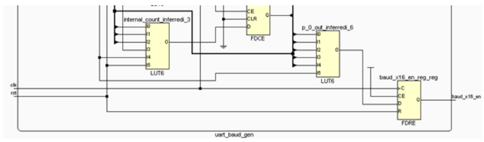
- Representation of the design before synthesis
  - Interconnected netlist of hierarchical and generic technology cells
    - Instances of modules/entities
    - Generic technology representations of hardware components
      - AND, OR, buffer, multiplexers, adders, comparators, etc…



44

Synthesized design

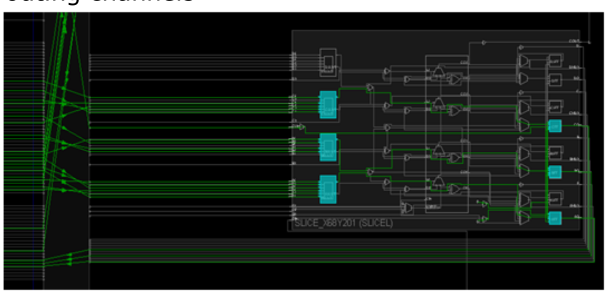
- ☐ Representation of the design after synthesis
  - Interconnected netlist of hierarchical and Basic Elements (BELs)
    - ☐ Instances of modules/entities
    - ☐ BELs
      - LUTs, flip-flops, carry chain elements, wide MUXes
      - Block RAMs, DSP cells
      - Clocking elements (BUFG, BUFR, MMCM,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



45

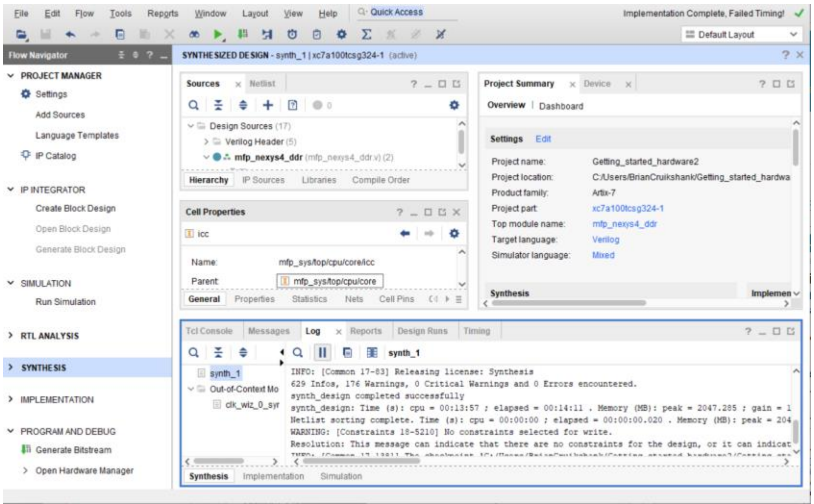
Implemented design

- ☐ 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



46

Main Vivado screen



Portland State UNIVERSITY

46

47

TCL console

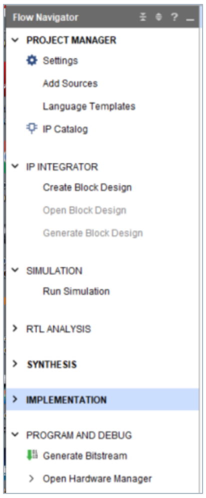
- Vivado also has a TCL console
- Very similar to Synopsys/Cadence commands
- There is a Tools->Xilinx TCL store with different options to help you with design and timing convergence

Portland State UNIVERSITY

47

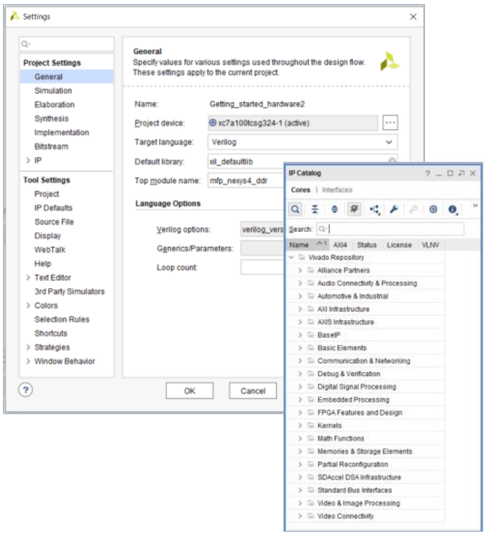
Vivado flow stages

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



Vivado project manager

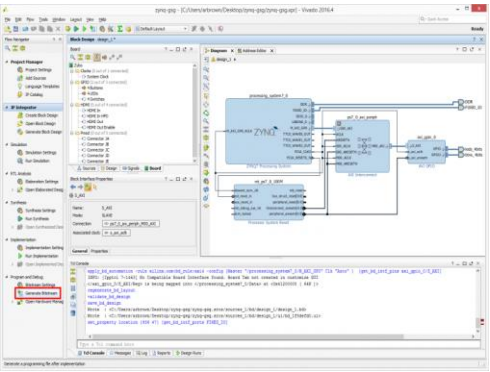
- Settings
- Add Sources
- Language Templates
- IP Catalog



50

Vivado **IP I**ntegrator

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



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

ECE 351 Verilog and FPGA Design

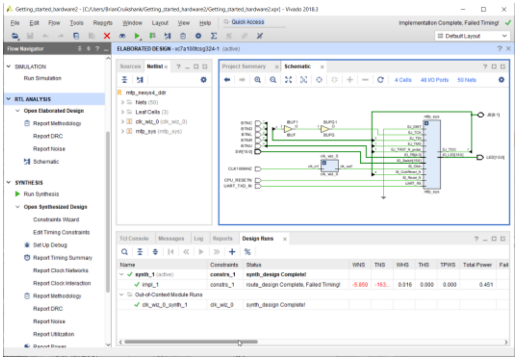
Portland State UNIVERSITY

50

51

Vivado RTL analysis

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



ECE 351 Verilog and FPGA Design

Portland State UNIVERSITY

51

Vivado synthesis

52

- 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

53

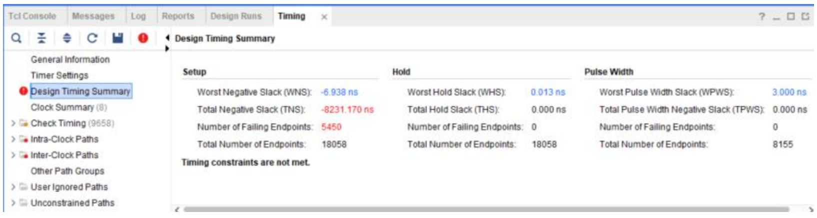
- Schematics are mapped to a netlist
  - A netlist is a Verilog netlist mapping to LUT cells, Carry Logic, DSP, RAMs, Ios, clock circuits, etc standard elements in the FPGA.
  - No behavioral code is left.
    - No always blocks. No functions. No ternary operators.
- Schematics/netlist is optimized for 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



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

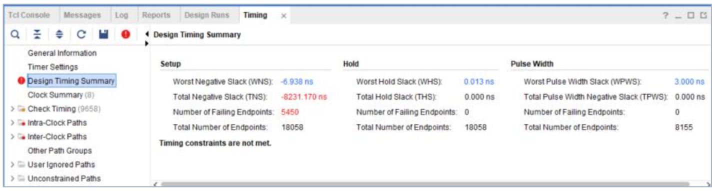
- Maps the LUTs, SRAMs, I/O's, etc. to physical locations on the desired FPGA
- Nets are mapped to global, section, local nets
- Real net delays are used.
- Locations of components are moved around to improve timing
- Nets are split, re-driven
- Iteration happens on the worst paths until nothing more 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

58

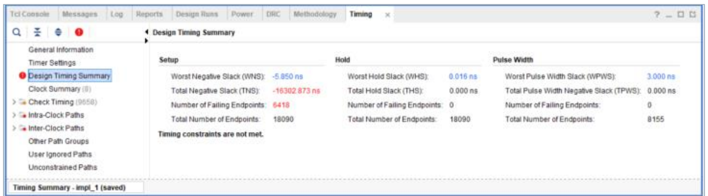
Report timing summary

- Many categories of reports, lots of terms

Synthesis



Implementation

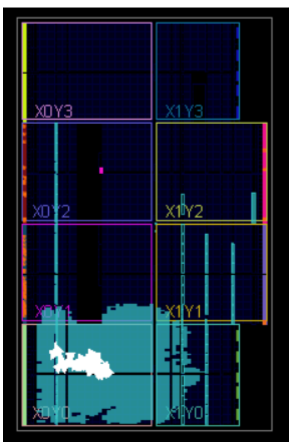


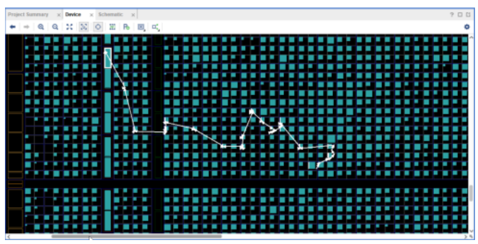
58

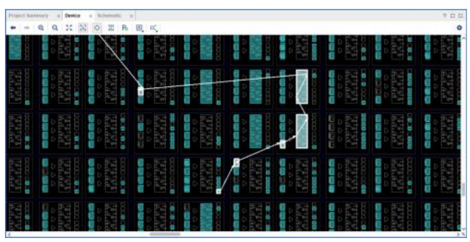
59

Implementation

- Zoom in different levels







59

Next Time

- Topics:
  - FPGA Introduction (wrap-up_
  - Introduction to Vivado
  - Review my HW #3 solution (incl. PmodSSD Nexys A7 demo)
  - JTAG, SoC's, etc.
- 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)