ECE 351
Verilog and FPGA Design

Lecture 1:   **Course overview**
Introduction to SystemVerilog
RTL and gate-level models
FPGAs, ASICs, ASSPs, SoCs

Roy Kravitz

Electrical and Computer Engineering Department

Maseeh College of Engineering and Computer Science

Portland State
UNIVERSITY

Course Overview

Portland State
UNIVERSITY

1

Instructor and T/A

- ☐ Instructor:
  - ■ Roy Kravitz, roy.kravitz@pdx.edu, 503.913.1678 (M)
- ☐ T/A and Grader:
  - ■ T/A: Rishitosh Sawant, risawant@pdx.edu
  - ■ Grader: Dhakshayini Koppad, dkoppad@pdx.edu
- ☐ (Virtual)Office Hours:
  - ■ Roy:
    - ☐ Tue, 10:30 AM – Noon (https://pdx.zoom.us/j/452550907)
    - ☐ Wed, 7:00 PM – 8:00 PM (https://pdx.zoom.us/j/480926265)
  - ■ Rishitosh:
    - ☐ TBD
  - ■ All office hours will be conducted via Zoom meetings
    - ☐ Students will enter the Waiting room to be admitted
    - ☐ We will also take appointments outside normal office hours. Email to set date/time

ECE 351 Verilog and FPGA Design

Portland State
UNIVERSITY

Course Objectives

By the end of this course you should be able to:

- ■ Describe complex digital designs using the SystemVerilog Hardware Description Language
- ■ Implement basic testbenches needed to functional verify digital designs
- ■ Apply design automation tools to synthesize designs in FPGAs
- ■ Describe good coding practices for efficient synthesized designs in FPGAs
- ■ Describe how embedded systems are implemented in FPGAs (including the role of vendor-specific, 3rd party, and custom IP)

ECE 351 Verilog and FPGA Design

Portland State
UNIVERSITY

Course Overview

- ☐ Lecture orientation
  - ■ Lectures covering SystemVerilog for synthesis, testbenches, FPGA design flows, IP block, JTAG, embedded systems in SoC's
- ☐ Assignments:
  - ■ Homework – Problem solving but there may be some short answer, multiple choice, or True/False questions…you will be writing/simulating SystemVerilog programs
  - ■ FPGA mini-project – More effort than the homework. Will rely on commercial EDA tools
    - ☐ My hope is to have you execute your design on the FPGA boards in the labs…if the labs reopen before the end of the term
  - ■ All assignments will be done individually and submitted to D2L for grading
- ☐ Quizzes and Exams:
  - ■ Quizzes and/or graded exercises during class meeting time
  - ■ Midterm exam Thur, 07-May (week 6)
  - ■ Final exam on Mon, 08-Jun from 10:30 AM – 12:05 PM (week 11)
  - ■ Quizzes and exams will be given remotely

ECE 351 Verilog and FPGA Design

Portland State
UNIVERSITY

Course Website

- ☐ Can access from your *my.pdx.edu* login or https://d2l.pdx.edu/
- ☐ Login with your ODIN username and password
  - ■ Announcements and course calendar
  - ■ Lecture notes and reading material
  - ■ Links to Zoom class meetings and recordings
  - ■ Project releases and dropbox(es) for submissions
  - ■ Discussion forums
  - ■ Links

Do NOT send email from D2L – We cannot reply to it. Use your `pdx.edu` email address and send email to our pdx.edu email address
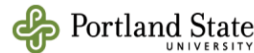
ECE 351 Verilog and FPGA Design

Portland State
UNIVERSITY

Zoom Class Meetings

People/Zoom Meetings on the ECE 351 D2L website

ECE 351 Verilog and FPGA Design

Portland State
UNIVERSITY

Tools and Textbooks

☐ Tools:
- ■ Mentor Graphics QuestaSim
- ■ Xilinx Vivado HLX 2018.2 Update 1 Webpack Edition

☐ Required Textbook:
- ☐ *RTL Modeling with SystemVerilog for Simulation and Synthesis* by Stuart Sutherland. Publisher: CreateSpace, An Amazon Company; ISBN-13: 978-154677634

☐ Technical Documentation:
- ■ Mentor Graphics *QuestaSim* user documentation
- ■ Xilinx Vivado user documentation
- ■ Digilent Nexys A7 documentation
- ■ Technical papers, reports, datasheets, and application notes as posted to class web page

ECE 351 Verilog and FPGA Design

Portland State
UNIVERSITY
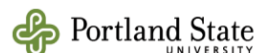
Tools and Textbooks (cont'd)

- ☐ Reference (optional – no readings assigned):
  - ☐ *Logic Design and Verification Using SystemVerilog (Revised)* by Donald Thomas. Publisher: CreateSpace, An Amazon.com Company; ISBN-13: 978-1523364022 (Available from the PSU Bookstore, Amazon.com and other retail outlets)

  - ☐ *Designing Digital Systems with SystemVerilog:v2.0)* by Brent E. Nelson. Independently published (June 2019); ISBN-13: 978-1075968433 (Available from Amazon.com)

ECE 351 Verilog and FPGA Design

Portland State UNIVERSITY

QuestaSim on MCECS Systems

- ☐ At PSU downtown campus
  - ■ Intel PC Lab: FAB Basement
  - ■ Capstone/Digital Circ... B Basement
- ☐ At Willow Creek
  - ■ Works... CC 313)
  - ■ Log in ... your MCECS ID/Password
  - ■ Network ... slow (it could take a few minutes to login)
    - ☐ Suggestion: Store your projects on a Flash drive instead of using your N: drive
- ☐ Remote access through VPN (Virtual Private Network) and Remote Desktop
  - ■ https://cat.pdx.edu/services/network/vpn-services/
  - ■ Can set up local drives, drag/drop to copy, etc.
    - ☐ Secure shell and FTP (or scp)
- ☐ Remote access through Spark View (RDP)
  - ■ Web-based – no VPN connection needed
  - ■ https://rdp.cecs.pdx.edu/

ECE 351 Verilog and FPGA Design

Portland State UNIVERSITY

FPGA Platform: Digilent Nexys A7*

Available from Digilent Inc. (see syllabus for URL's and part numbers) for $198.75 (Academic pricing) + S&H

Digilent Nexys A7-100T

- □ *Xilinx Artix-7 FPGA in a 324-pin BGA package w/ integrated Analog-to-digital converter*
- □ *100MHz CMOS oscillator*
- □ *128MiB DDR2*
- □ *Serial Flash*
- □ *10/100 Ethernet PHY*
- □ *On-board USB-JTAG*
- □ *USB-UART and USB-HID port (for mouse/keyboard)*
- □ *12-bit VGA port*
- □ *72 I/O's routed to expansion connectors*
- □ *GPIO includes 16 LEDs, 3 tri-color LEDs, 6 buttons,16 slide switches and a 8-digit seven-segment display*
- □ *Accelerometer, PDM microphone, PWM audio out, temperature sensor, micro-SD connector*

*Digilent renamed Nexys4 DDR but it's the same board
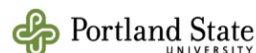
ECE 351 Verilog and FPGA Design

Portland State UNIVERSITY

Details

Syllabus: ..\misc\ece351sp20_syllabus_R1_0.pdf

Weekly Schedule: ..\misc\ece351sp20_tentative_weekly_schedule_r1_0.pdf

ECE 351 Verilog and FPGA Design

Portland State UNIVERSITY

Introduction to System Verilog

Sources:
- Alex Pearson, Mark Faust and Roy Kravitz ECE 571 Lecture Slides
- *Logic Design and Verification with System Verilog (Revised)* by Donald Thomas

---

What is SystemVerilog?

- □ SystemVerilog is a language and a simulator for digital systems
  - ■ Enables a designer to model a digital system as interconnected blocks of logic (incl. primitive gates like AND, OR, …)
  - ■ Specifies how logic values are propagated through digital system as a function of time
- □ Single design language to describe, model, synthesize (turn into optimized logic blocks) and simulate simple and complex electronic digital systems
  - ■ Attempts to merge the rich verification features and abstraction features of VHDL with the relative simplicity of describing hardware in Verilog
- □ Provides rich environment for verifying that a digital system functions correctly

ECE 351 Verilog and FPGA Design

Portland State
UNIVERSITY

7

Why SystemVerilog?

- ☐ New features for high level design, RTL design, and verification
- ☐ Additional constructs, features
  - ■ Reusability, Conciseness
  - ■ Improved error detection and reporting
- ☐ Greatly improved/enhanced support for verification of complex designs (assertions, constrained randomization, support for **O**bject **O**riented **P**rogramming, …
- ☐ More consistent behavior between synthesized hardware and simulated result

…all this…and more…yet it retains backwards compatibility w/ Verilog 2001 syntax

ECE 351 Verilog and FPGA Design

Portland State
UNIVERSITY

Why SystemVerilog in ECE 351?

- ☐ There is no more "Verilog"
  - ■ Verilog spec was incorporated into SystemVerilog spec
  - ■ Last standalone specification was Verilog 2001
  - ■ SystemVerilog is the new name for what used to be Verilog
- ☐ ECE department has standardized on SystemVerilog as the **H**ardware **D**escription **L**anguage we teach
  - ■ ECE 571, 4/581, ECE 590, ECE Design Verification and Validation graduate track all teach/use it
  - ■ SystemVerilog is a "better" language for expressing hardware behavior concisely
  - ■ New SystemVerilog constructs and simulation kernel pretty much eliminate the race conditions that made simulation frustrating at times (…exactly how do I interpret this result?)
- ☐ Commercial simulation and synthesis tools support SystemVerilog constructs

ECE 351 Verilog and FPGA Design

Portland State
UNIVERSITY

Key Features: Design

- ☐ Data types:
  - ■ `wire`, `reg`, `integer`, `tri`, … (inherited from Verilog 2001)
  - ■ `logic` (the more generic, go-to type)
  - ■ C-like data types (ex: `int`)
  - ■ User-defined types (ex: `typedef`)
  - ■ Enumerated types (ex: `enum`)
  - ■ Structures and unions (ex: `struct` and `union`)
  - ■ Typecasting
- ☐ Logical and arithmetic operations
  - ■ ~, +, -, *, /, %, |, &, ^, !, &&, ||, …
  - ■ ++, --, += and other assignment operators
- ☐ Explicit constructs for combinatorial, latch-based, and flip-flop based designs
- ☐ Packages for definitions shared by multiple design blocks
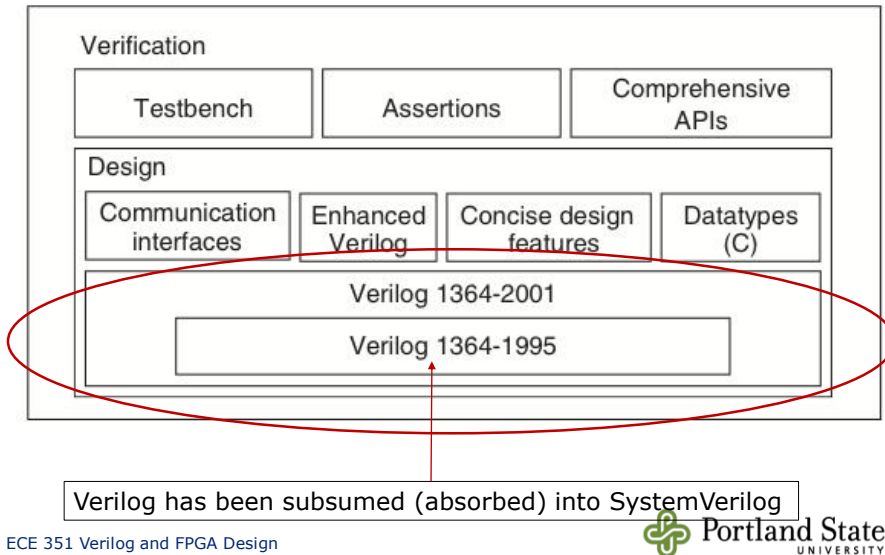- ☐ Interfaces to encapsulate communication and protocol checking

ECE 351 Verilog and FPGA Design

Portland State UNIVERSITY

Key Features: Verification

- ☐ Data Types
  - ■ Strings
  - ■ Dynamic arrays, associative arrays, queues
- ☐ Object Oriented Programming (e.g. Classes)
- ☐ Constrained Random Generation
- ☐ Assertions
- ☐ Coverage
- ☐ Synchronization

ECE 351 Verilog and FPGA Design
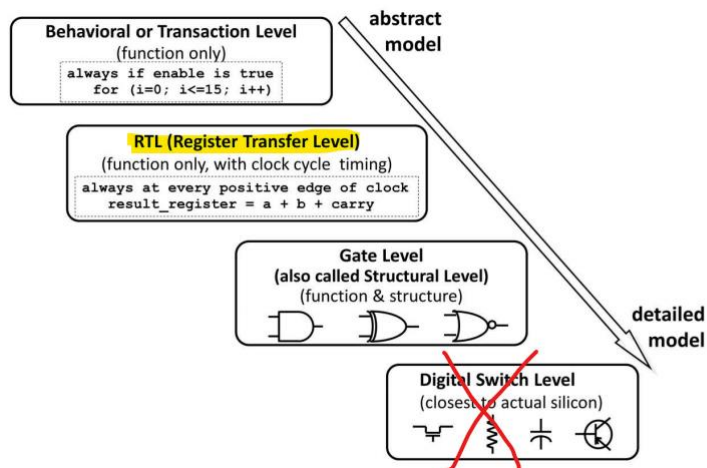
Portland State UNIVERSITY

Features/Constructs

Verification

Testbench	Assertions	Comprehensive APIs

Design

Communication interfaces	Enhanced Verilog	Concise design features	Datatypes (C)

Verilog 1364-2001

Verilog 1364-1995

Verilog has been subsumed (absorbed) into SystemVerilog

ECE 351 Verilog and FPGA Design

Portland State UNIVERSITY

SystemVerilog levels of abstraction

abstract model

Behavioral or Transaction Level
(function only)

```
always if enable is true
    for (i=0; i<=15; i++)
```

RTL (Register Transfer Level)
(function only, with clock cycle timing)

```
always at every positive edge of clock
    result_register = a + b + carry
```

Gate Level
(also called Structural Level)
(function & structure)

detailed model

Digital Switch Level
(closest to actual silicon)

ECE 351 Verilog and FPGA Design

Sutherland Figure 1-3

Portland State UNIVERSITY

SystemVerilog levels of abstraction (cont'd)

- **Behavioral** or **Transaction Level** – Highest level of abstraction. A module is implemented with little (or no) concern for hardware-implementation details (function-only)
  - Very similar to C programming
- **R**egister **T**ransfer **L**evel – function-only w/ clock cycle timing
- **Gate Level** - Module is implemented in terms of logic gates and interconnection between the gates
  - Similar to a logic schematic
- **Switch Level** – Lowest level of abstraction. Module is implemented with switches, storage nodes and the interconnect between them

ECE 351 Verilog and FPGA Design

Portland State
UNIVERSITY

D Flip-Flop Gate Level Model

```
module DFF (
  output logic q,
  input logic d,
  input logic clk, reset
);

logic s, sbar, r, rbar, q, qbar;
logic clkbar, cbar;

not  inv1(cbar, clear);
not  inv2(clkbar, clk);

nand sr1g1(sbar,rbar, s);
nand srig2(s, sbar, cbar, clkbar);

nand sr2g1(r, rbar, clkbar,s);
nand sr2g2(rbar, r, cbar, d);

nand sr3g1(q, s, qbar);
nand sr3g2(qbar, q, r, cbar);

endmodule
```
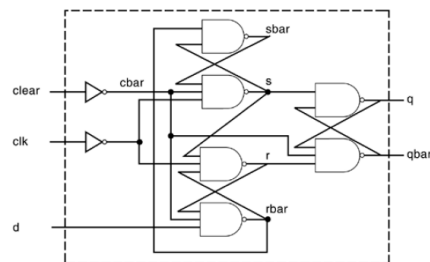

ECE 351 Verilog and FPGA Design

Portland State
UNIVERSITY

D Flip-Flop RTL Model

```
// module DFF with asynchronous reset
module DFF(
    output logic q,
    input logic d, clk, reset,
    logic q
);

always_ff @(negedge clk or posedge reset) begin
  if (reset)
    q <= 1'b0;
  else
    q <= d;
end

endmodule
```

Portland State
UNIVERSITY

Gate Level Modeling

Source material drawn from:
- *Palnitkar (some figures from online edition)*
- *Brown and Veranasic*
- ECE 351 lecture slides

Gate Level Modeling

- ☐ SystemVerilog supports gate level modeling
    - ■ Can replace gate level schematics and you can simulate and debug the design!
    - ■ Provides explicit directions for synthesis tool
- ☐ Basic Gate Types primitives
    - ■ and, or, xor, nand, nor, xnor - logic gates
    - ■ buf, not - non-inverting and inverting buffers
    - ■ bufif1, notif1, bufif0, notif0 - tri-state buffers
    - ■ pulldown, pullup, nmos, pmos, cmos,…

ECE 351 Verilog and FPGA Design

Portland State UNIVERSITY

Gate Level Modeling (cont'd)

- ☐ Syntax is the same for all instantiations
    - ■ <gate_type> <gate_instance>(out, in1, in2, …, in_n);

```
nand n1(OUT, IN1, IN2);
```

Inputs
Output
Gate instance name
Gate type
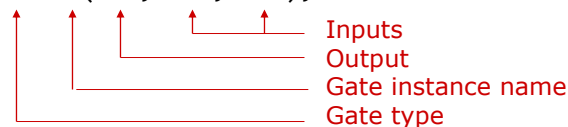
- ■ Single output, but gates w/ more than 2 inputs (NAND, NOR, etc) are instantiated by simply adding more input ports
- ■ Gate instance name is optional

ECE 351 Verilog and FPGA Design

Portland State UNIVERSITY

Verilog Gate Types

Name	Description	Usage
and	$f = (a \cdot b \cdot)$	**and**$(f,a,b,\ldots)$
nand	$f = \overline{(a \cdot b \cdot)}$	**nand**$(f,a,b,\ldots)$
or	$f = (a + b + \cdots)$	**or** $(f,a,b,\ldots)$
nor	$f = \overline{(a + b + \cdots)}$	**nor**$(f,a,b,\ldots)$
xor	$f = (a \oplus b \oplus \cdots)$	**xor**$(f,a,b,\ldots)$
xnor	$f = \overline{(a \odot b \odot \cdot)}$	**xnor**$(f,a,b,\ldots)$
not	$f = \overline{a}$	**not**(f,a)
buf	$f = a$	**buf** (f,a)
notif0	$f = (!e?\overline{a}:'bz)$	**notif0**(f,a,e)
notif1	$f = (e?a:'bz)$	**notif1**(f,a,e)
bufif0	$f = (!e?\overline{a}:'bz)$	**bufif0** (f,a,e)
bufif1	$f = (e?a:'bz)$	**bufif1** (f,a,e)

ECE 351 Verilog and FPGA Design

Portland State UNIVERSITY

Truth Tables for Gates

- □ SystemVerilog is a 4 valued logic simulator
  - ■ 0, 1, x, z
- □ x is unknown that means it can either be a 1 or a 0
- □ z essentially mean *open circuit*, so, for example, a z input to a nand() gate has an uncertain output

and	0	1	x	z
0	0	0	0	0
1	0	1	x	x
x	0	x	x	x
z	0	x	x	x

xnor	0	1	x	z
0	1	0	x	x
1	0	1	x	x
x	x	x	x	x
z	x	x	x	x

ECE 351 Verilog and FPGA Design

Portland State UNIVERSITY
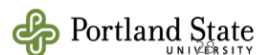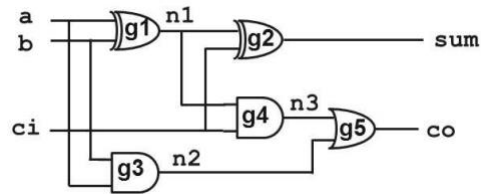
Example

```
module gate_adder (
   input  wire a, b, ci,
   output wire sum, co
);

wire n1, n2, n3;

xor            g1 (n1, a, b);
xor #1.3       g2 (sum, n1, ci);
and            g3 (n2, a, b);
and            g4 (n3, n1, ci);
or  #(1.5,1.8) g5 (co, n2, n3);

endmodule: gate_adder
```

ECE 351 Verilog and FPGA Design

Portland State

Sutherland Figure 1-4, Example 1-1

Technology Overview

Source material drawn from:
- *ASIC Design Methodology Primer*, IBM ASIC Products Application Note, Initial publication 5/98
- Wikipedia
- Slideware provided by Xilinx
- Dr Song's lecture notes
- Verilog workshop and other lecture material by Roy

ASIC's, ASPP's and Full custom IC's

- □ **A**pplication **S**pecific **I**ntegrated **C**ircuit - a custom chip designed for a very specific purpose.
  - ■ Ex: a chip with a DSP front-end designed for a specific model of cardiac monitor made by only one manufacturer
  - ■ Companies implement their ASIC designs in a single silicon die by mapping their functionality to a set of predesigned and verified library of circuits provided by the ASIC vendor. The components of the library are described in the ASIC vendor's databook
- □ **A**pplication **S**pecific **S**tandard **P**roduct - a semi-custom chip designed for a specific application and sold to multiple customers
  - ■ Ex: an integrated circuit that does video or audio encoding/decoding
- □ Full custom chip – an integrated circuit designed by a single company for a specific application, usually huge volumes
  - ■ Ex: an Intel CPU
  - ■ Full-custom chips are developed for a specific semiconductor technology, often with huge teams of architects, logic designers, circuit designers, validation engineers, layout designers, et. al. and take several years to design

ECE 351 Verilog and FPGA Design

Portland State
UNIVERSITY

FPGA's

- □ **F**ield **P**rogrammable **G**ate **A**rray - An integrated circuit designed to be configured by a customer or a designer after manufacturing—hence field-programmable
  - ■ The FPGA configuration is generally specified using a hardware description language (HDL)
  - ■ FPGAs can be used to implement any logical function that an ASIC can perform but the ability to update the functionality after shipping and the low non-recurring engineering costs relative to an ASIC design offer advantages for many applications
  - ■ FPGAs contain programmable logic components called "configurable logic blocks" and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together" when the FPGA is configured (and reconfigured…and reconfigured…and reconfigured…and…)

ECE 351 Verilog and FPGA Design

Portland State
UNIVERSITY

SoC's

- **S**ystem **o**n **C**hip or SOC – An integrated circuit (IC) that integrates all components of a computer or other electronic system into a single chip.
  - Ex: an embedded system such as a cell phone or cable box
  - SoC's may contain digital, analog, mixed-signal, and often radio-frequency functions—all on a single chip substrate
- An SoC can be implemented as an ASIC or as an ASSP or in an FPGA or as a full custom chip
  - Roy's definition: An SoC contains at least one embedded CPU running an application interfaced to one or more vendor-supplied, 3rd party or custom IP blocks
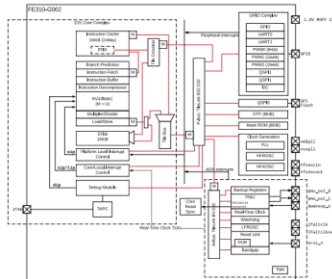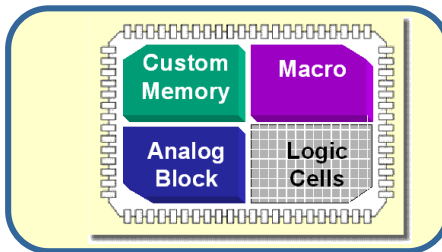

ECE 351 Verilog and FPGA Design

SiFive FE310-G002 top-level block diagram

Portland State
UNIVERSITY
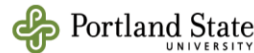
ASIC: Standard Cell

- Advantages
  - Lowest price for high-volume production (greater than 200K per year)
  - Fastest clock frequency (performance)
  - Unlimited size
  - Integrated analog functions
    - Custom ASICs
  - Low power
- Disadvantages
  - Highest non-recurring engineering costs
  - Longest design cycle
 - Limited vendor IP with high cost
  - High cost for ECO's

ECE 351 Verilog and FPGA Design

Portland State
UNIVERSITY
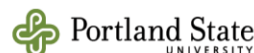
FPGA

- ☐ Advantages
  - ■ Lowest cost for low-volume to medium-volume production
  - ■ No non-recurring engineering costs
  - ■ Standard product
  - ■ Fastest time to market
  - ■ Extensive library of IP
    - ☐ Inexpensive compared to ASIC vendors
  - ■ Ability to make bug fixes quickly and inexpensively
- ☐ Disadvantages
  - ■ Slower performance
  - ■ Size limited to ~25 million system gates
  - ■ Digital only

ECE 351 Verilog and FPGA Design

Portland State UNIVERSITY
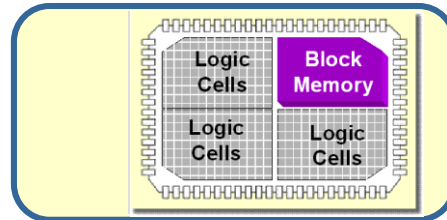
Design Flow Comparison

ECE 351 Verilog and FPGA Design

Portland State UNIVERSITY
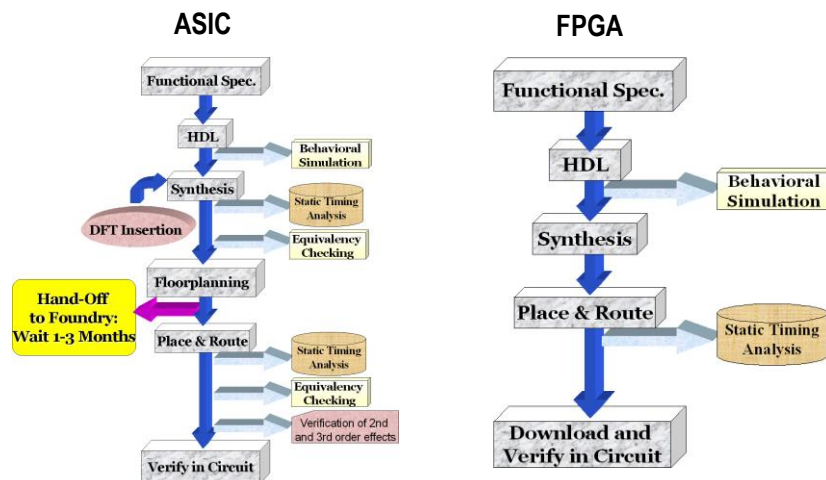
Next Time

- ☐ Topics:
  - ■ Simulation w/ QuestaSim
  - ■ SystemVerilog language rules
- ☐ You should:
  - ■ Read Sutherland Ch 1, and Ch 2
- ☐ Homework, projects and quizzes
  - ■ Homework #1 will be assigned Tue, 14-Apr

ECE 351 Verilog and FPGA Design

Portland State
UNIVERSITY