Introduction to Hardware Discription Languages for FPGA Design

Professors Tim Scherr and Benjamin Spriggs

University of Colorado Boulder

[MUSIC]

FPGA Design for Embedded Systems

Hardware Description Languages for Logic Design

Hello and welcome to FPGA design for embedded systems.

An Introduction to VHDL

- VHDL is a widely used design language.
- VHDL is a structured language for logic design.
- In this Module we will introduce the VHDL language so that you can begin to design logic circuits.

In this course, we will introduce the VHDL language

Why learn VHDL?

- VHDL is a structured language, and can be easier than Verilog to learn as your first Hardware Description Language.
- Since VHDL is language based, designs created earlier can be re-used.

VHDL is a structured language
and can be easier than

How can you learn VHDL?

Like any new language

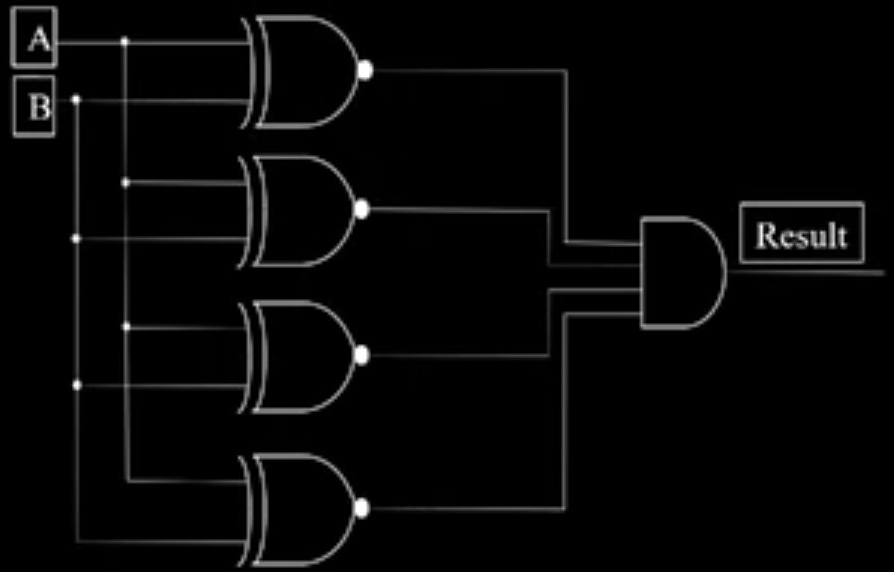
If you were going to learn Spanish :

- Learn key phrases
- Learn Grammar and Syntax
- Make more complex sentences
- Practice, Practice, Practice

if you are going
to learn Spanish,

How would you write the VHDL code?

- A 4-bit Comparator



In this course, we will ask the question: How would you

Videos in this Module

1. Why Learn VHDL (this video)
2. FPGA Design Flow
3. Intro to VHDL : Finite State Machine
4. How to speak VHDL, First phrases
5. VHDL Assignments, Operators, Types
6. VHDL Rules and Syntax, Interface Ports
7. Using VHDL in ModelSim : Download and Install
8. Using VHDL in Modelsim : Adding to your Toolkit

The upcoming videos, we will cover FPGA design flow,

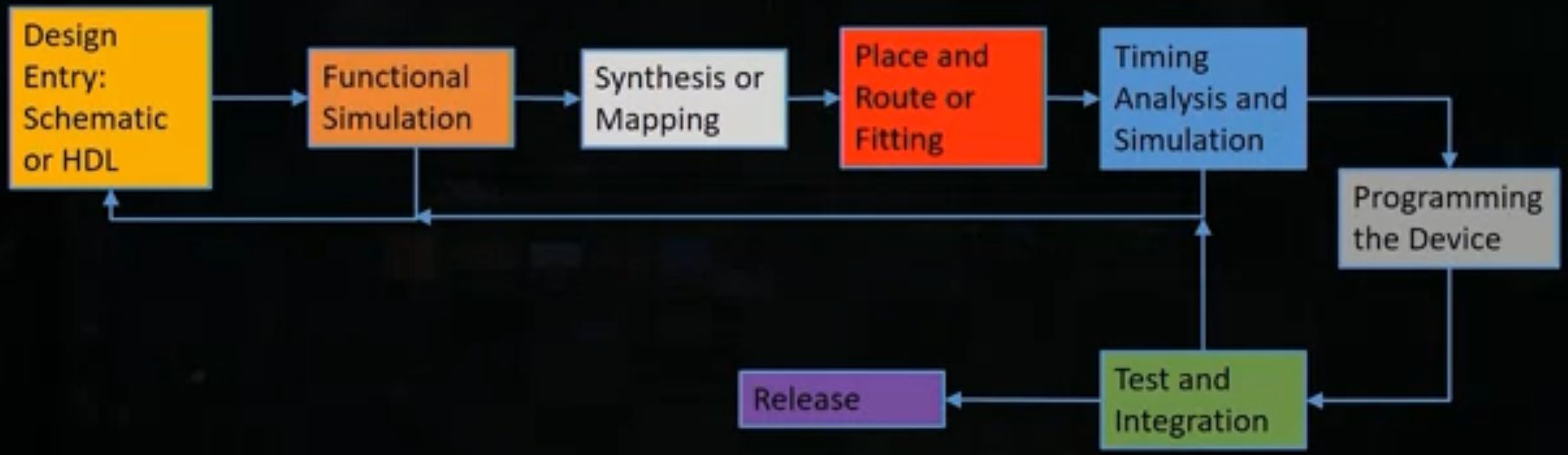
Objectives

- Acquire proficiency designing digital circuits with Field Programmable Gate Arrays (FPGA)s
- Employ industry-standard tools to design and implement programmable logic solutions using several Hardware Description Language (HDL) design entry methods, including VHDL, Verilog and System Verilog
- Build proficiency by designing basic digital programmable logic circuits in VHDL, Verilog and System Verilog

circuits with field-programmable

gate arrays FPGAs,

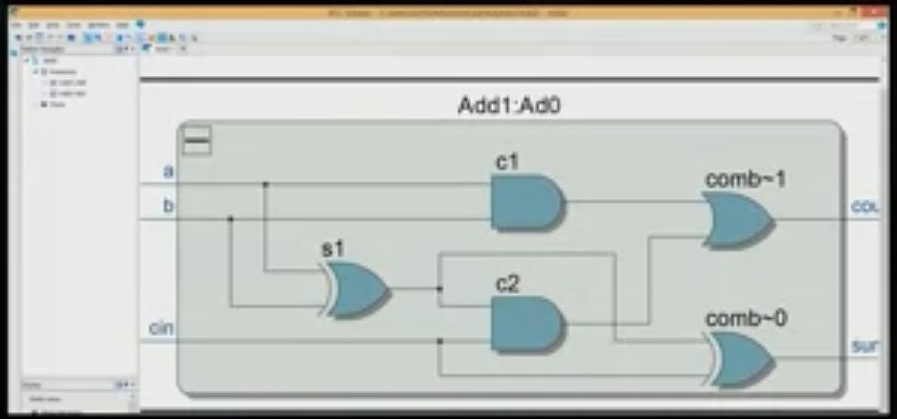
Recap : FPGA Design Flow



As a recap,
here's a flow of the FPGA design process,

FPGA Design Entry Methodologies

Schematic Entry



- Logical correlation between schematic design entry and circuit implementation
- Not always transferrable between different FPGA tools and technologies
- Difficult to scale for larger designs

In schematic entry, we have logical correlation between schematic design and

Using Schematic design entry is easily transferable to other FPGA or ASIC tool flows ?

- False

Correct
Schematic entry tools are specific to a vendor tool data entry flow, but HDLs are text and are easily emailed or transferred to other projects.

- True

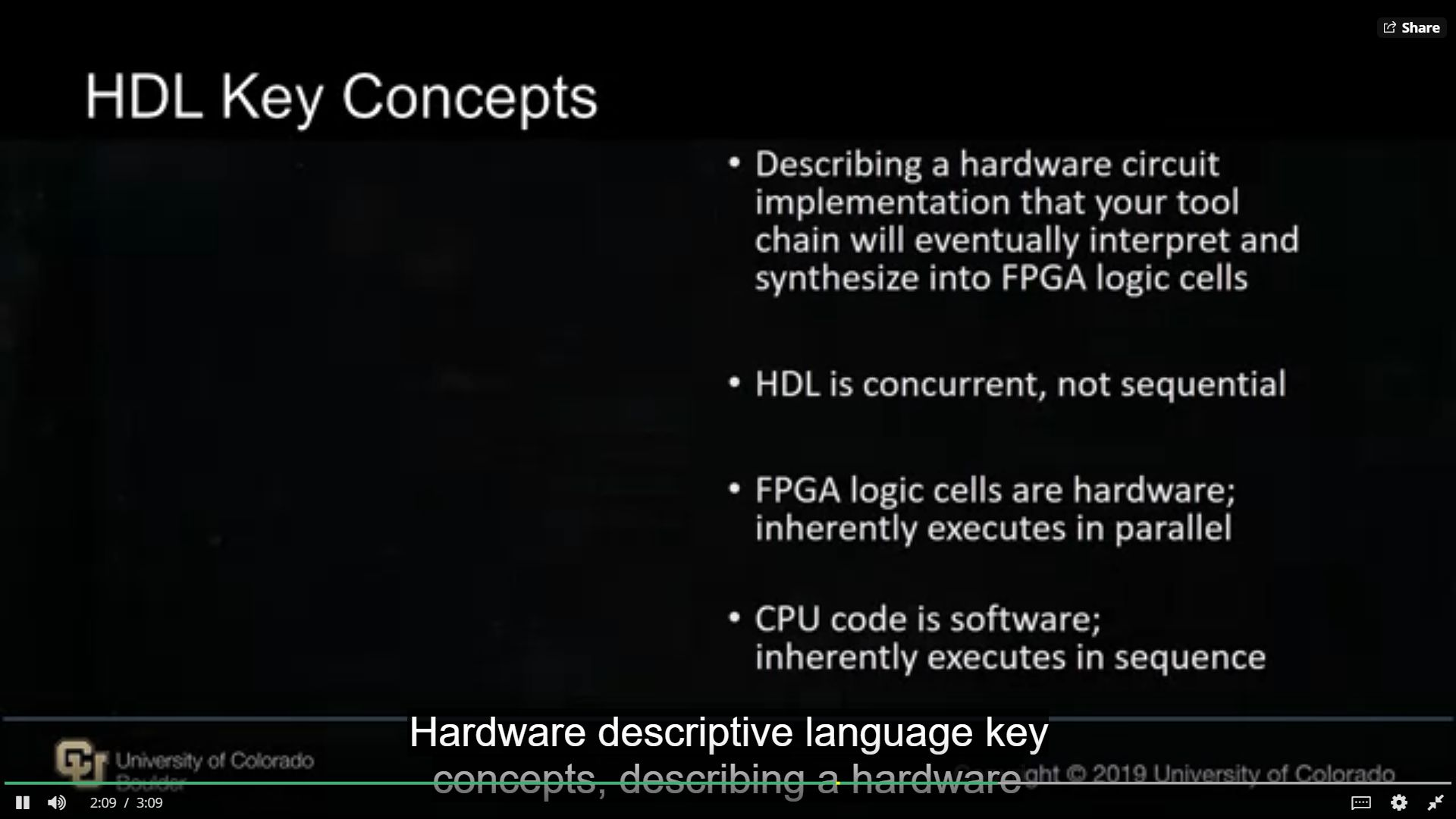
Continue

FPGA Design Entry Methodologies

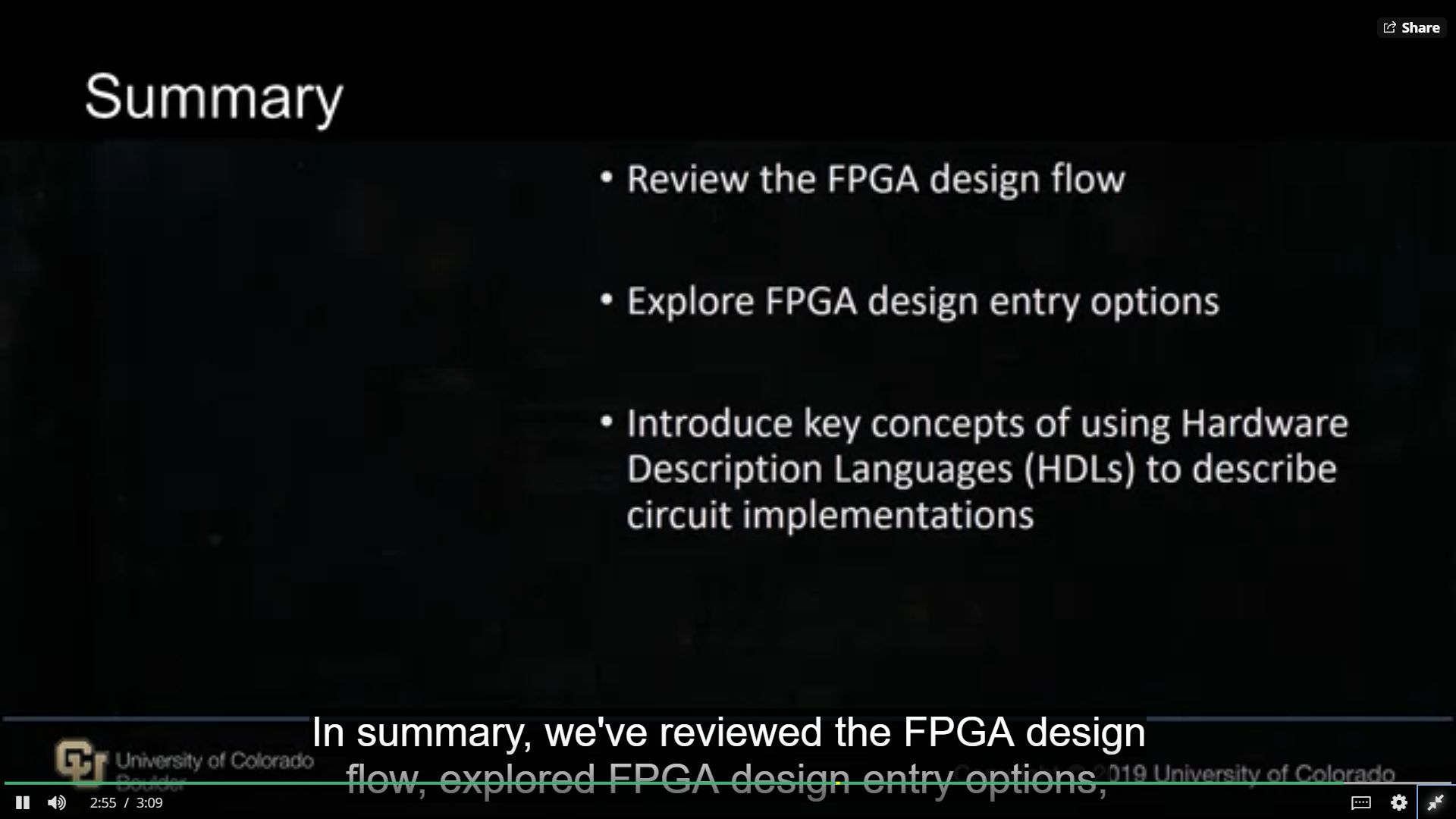
- **Hardware Description Languages (HDLs)**
  - VHDL
  - Verilog
  - System Verilog

```
-- Synchronous process (State FFs)
SyncProcess: process(Reset, Clk)
begin
  if (Reset = '1') then
    CurrentState <= S0;
  elsif (rising_edge(Clk)) then
    CurrentState <= NextState;
  end if;
end process SyncProcess;
```

save it as part of our design archive.

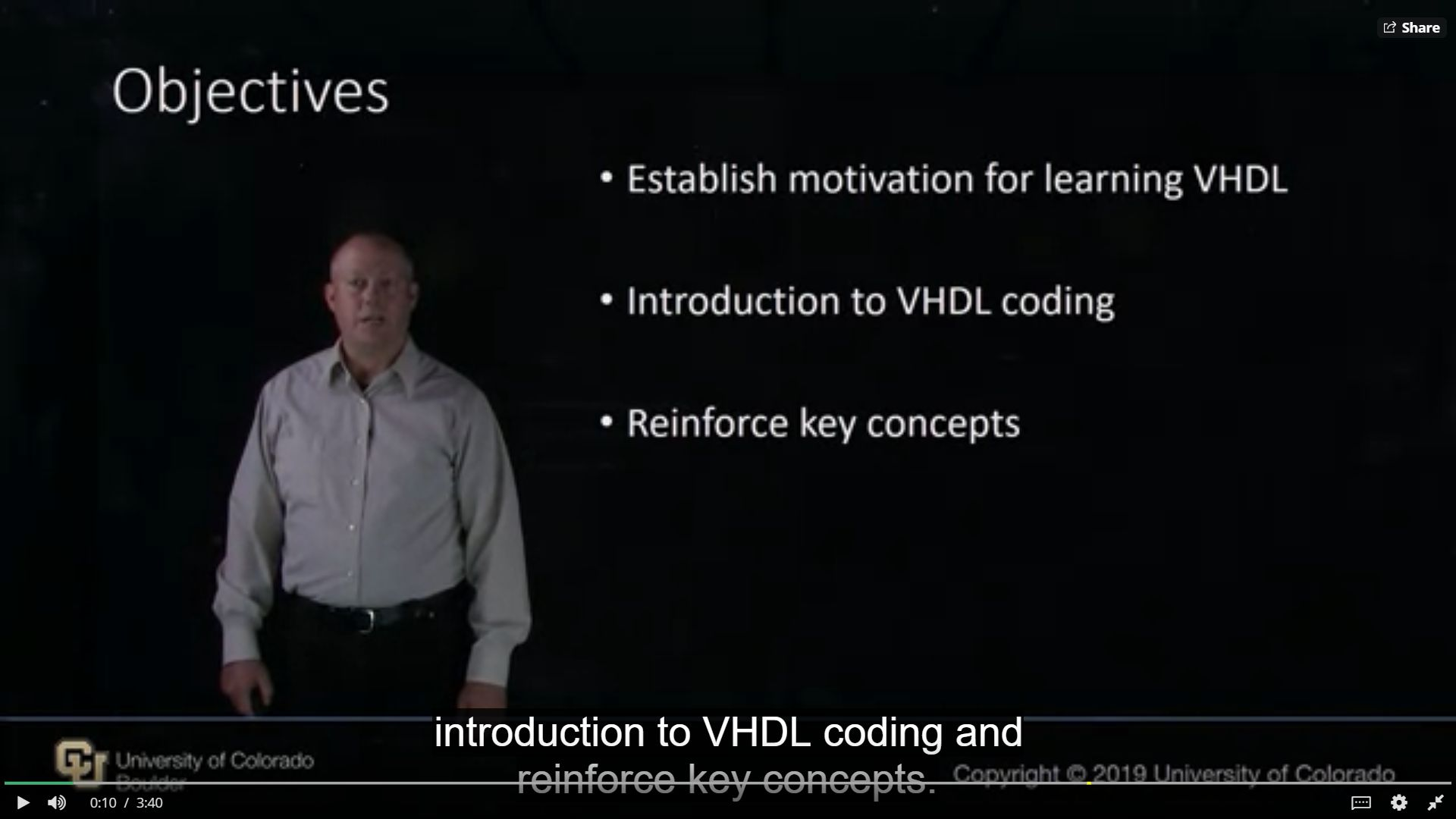


Hardware descriptive language key concepts, describing a hardware



In summary, we've reviewed the FPGA design flow, explored FPGA design entry options,


Share
FPGA Design for Embedded Systems
Hardware Description Languages for Logic Design
Hello and welcome back to FPGA Design for Embedded Systems.
University of Colorado Boulder
Copyright © 2019 University of Colorado
0:03 / 3:40


Share
Objectives
• Establish motivation for learning VHDL
• Introduction to VHDL coding
• Reinforce key concepts
introduction to VHDL coding and
reinforce key concepts.
University of Colorado Boulder
Copyright © 2019 University of Colorado
0:10 / 3:40

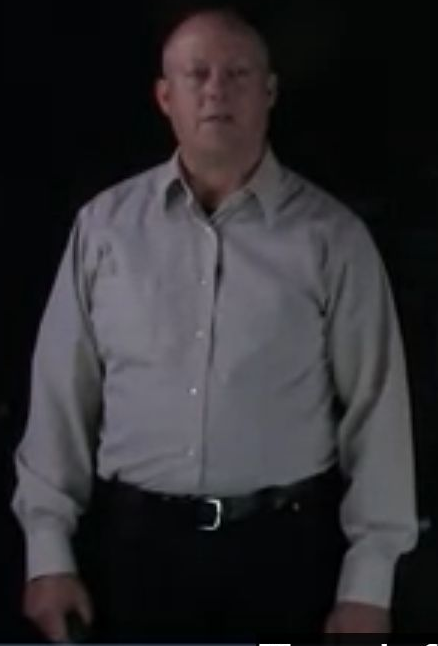
Why Learn VHDL?

- Earn a great income writing VHDL for FPGA and ASIC devices
- VHDL is more structured than Verilog and faster than schematic capture!
- Learn a new language for efficient and reusable problem solving

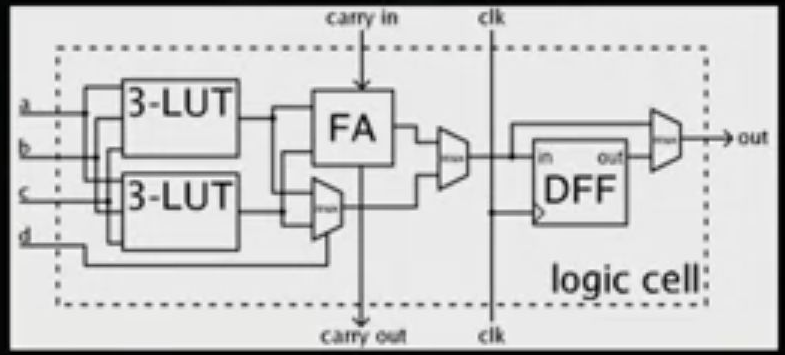
Writing VHDL for FPGA and ASIC devices.

University of Colorado Boulder

HDL Key Concepts

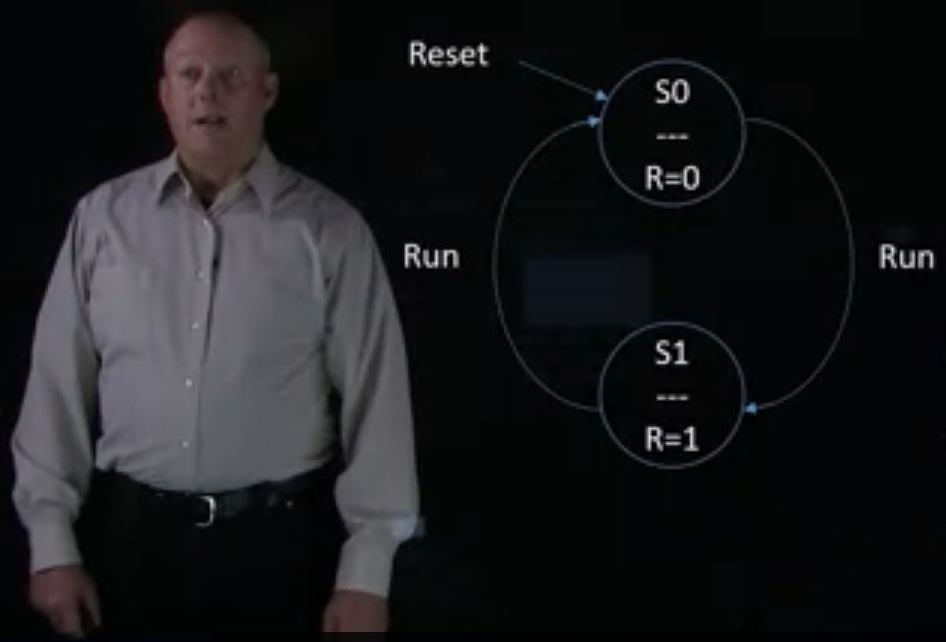


- *Describing* a hardware circuit that will synthesize into FPGA logic cells or gates
- Logic is concurrent, not sequential
  - FPGA gates are hardware; executes in parallel
  - CPU code is software; executes serially



To reinforce those concepts here is a simplified schematic of an FPGA

Finite State Machine Example



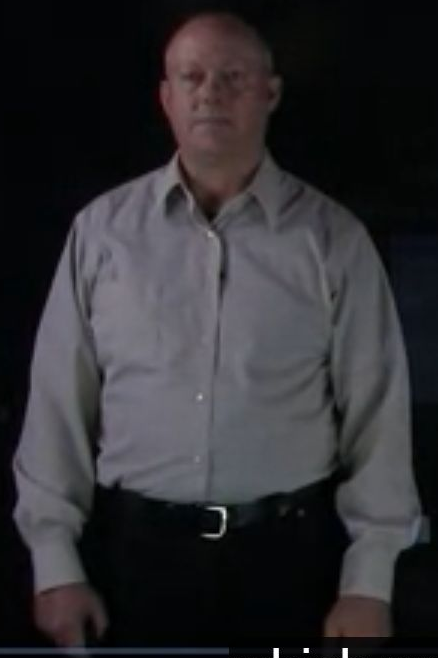
- State encoding
  - S(Q1, Q0) = “00”, “10”
- State table

Current State		IN	Next State		OUT
Q1	Q0	Run	D1	D0	R
0	0	0 1	0 0	1 1	0 0
0	1	0 1	0 1	1 0	0 1
1	0	0 1	1 0	0 1	1 0
1	1	0 1	0 0	1 1	0 0

- INPUT : Reset, Run, Clk
- OUTPUT : R Result
- STATE : Current, Next

For example, in a finite state machine,
we could describe

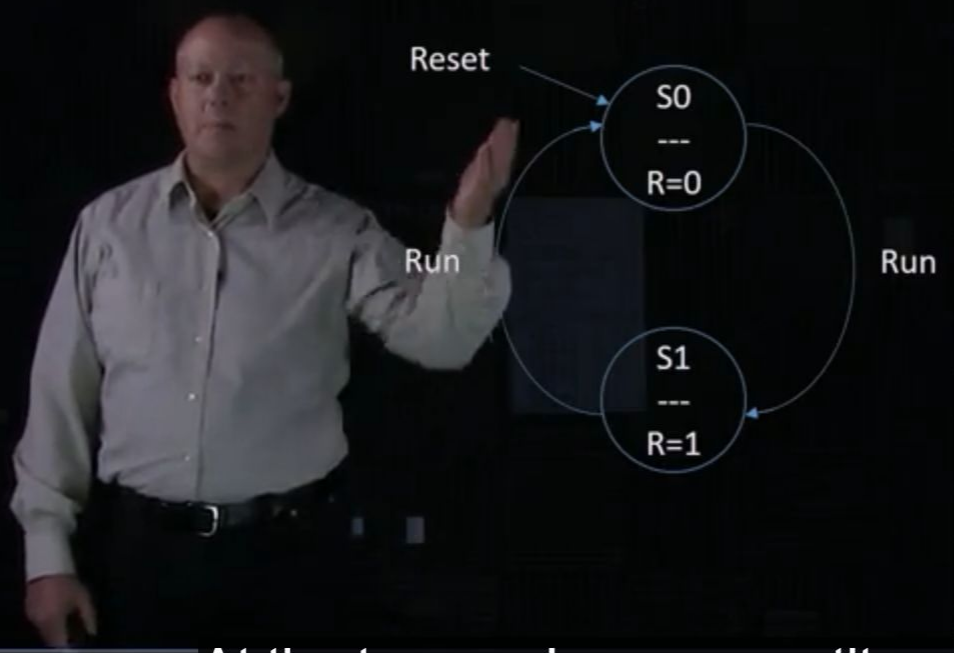
Finite State Machine Example



Current State		IN	Next State		OUT
Q1	Q0	Run	D1	D0	R
0	0	0 1	0 0	1 1	0 0
0	1	0 1	0 1	1 0	0 1
1	0	0 1	1 0	0 1	1 0
1	1	0 1	0 0	1 1	0 0

which would represent the inputs and output of this design.

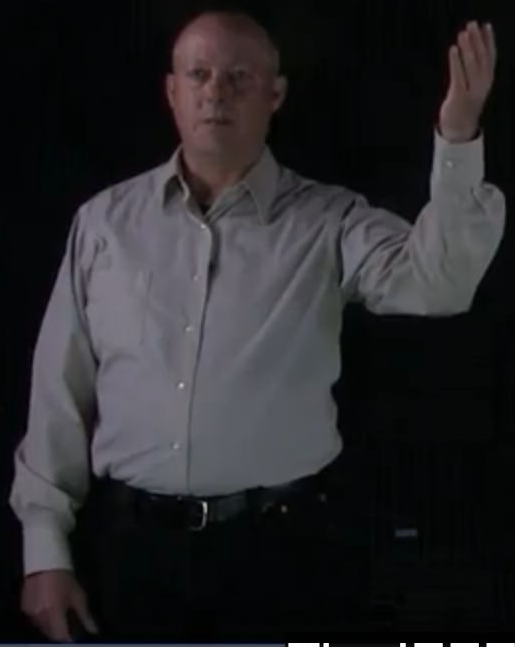
Finite State Machine Example




```
-- Use standard IEEE library
library IEEE;
use IEEE.std_logic_1164.all;

-- Entity port declaration
entity MyFSM is port (
  Run    : in std_logic;
  Clk, Reset  : in std_logic;
  Result  : out std_logic);
end MyFSM;

-- Architecture of FSM
architecture struct of myFSM is
  type stateType is (S0, S1);
  signal CurrentState, NextState : stateType;
begin

  -- Synchronous process (State FFs)
  SyncProcess: process(Reset, Clk)
  begin
    if (Reset = '1') then
      CurrentState <= S0;
    elsif (rising_edge(Clk)) then
      CurrentState <= NextState;
    end if;
  end process SyncProcess;

  -- Combinatorial process (State and output decode)
  CombProcess: process(CurrentState, Run)
  begin
    case CurrentState is
      when S0 =>
        Result <= '0';
        if (Run = '1') then
          NextState <= S1;
        else
          NextState <= S0;
        end if;
      when S1 =>
        Result <= '1';
        if (Run = '1') then
          NextState <= S0;
        else
          NextState <= S1;
        end if;
      when others =>
        Result <= '0';
        NextState <= S0;
    end case;
  end process CombProcess;
end struct;
```

At the top, we have our entity ports and

Introduction to VHDL



```
-- (VHDL comment)
-- Import std_logic library
library IEEE;
use IEEE.std_logic_1164.all;

-- Entity
entity ANDGATE is
        port (  A : in std_logic;
                B : in std_logic;
                Y : out std_logic);
end entity ANDGATE;

-- Architecture
architecture RTL of ANDGATE is
begin
        Y <= A AND B;
end architecture RTL;
```

The IEEE, the entity, AND gate.

A VHDL design file typically has 3 parts, which includes : Library IEEE.std_logic_1164, Entity with in and out ports, and an Architecture with logic definitions.

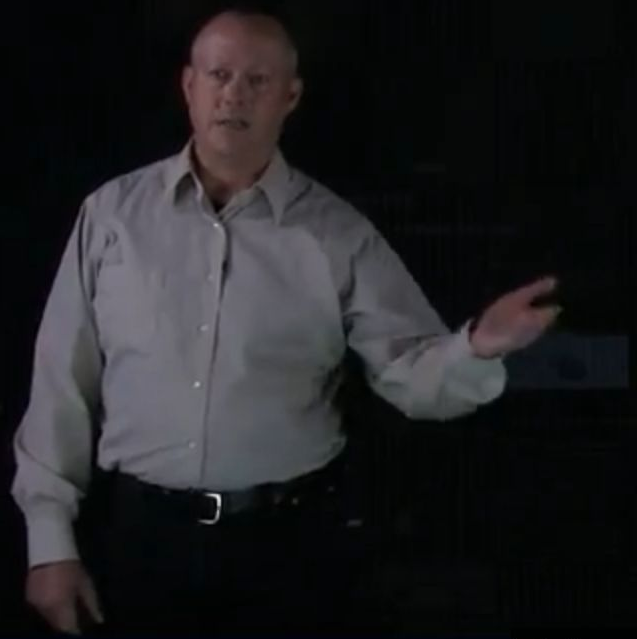
- False
- True

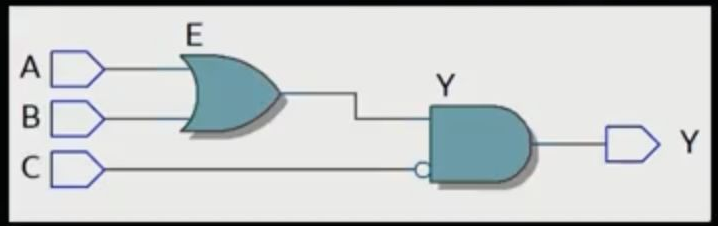
Correct
Yes, a VHDL design file includes a Library, Entity, and an Architecture.

Continue

Introduction to VHDL

- Example: Basic Logic





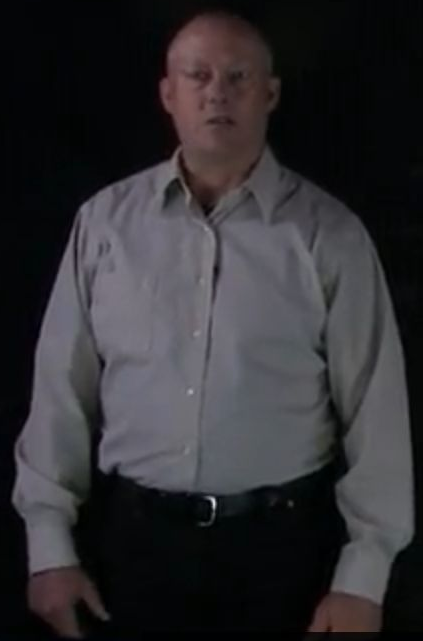
D is not of C.

Introduction to VHDL

```
-- (VHDL comment)
-- Import std_logic library
library IEEE;
use IEEE.std_logic_1164.all;

-- Entity
entity BasicLogic is
  port (
    A : in std_logic;
    B : in std_logic;
    C : in std_logic;
    Y : out std_logic);
end entity BasicLogic;

-- Architecture
architecture RTL of BasicLogic is
  Signal D, E : std_logic;begin
    D <= NOT C;
    E <= A OR B;
    Y <= D AND E;
end architecture RTL;
```

The VHDL code which goes along with this in our port declaration entity,

Summary



- VHDL is a structured language
  - Entity and Architecture pairs
- Gates synthesized into hardware operate concurrently and in parallel

In summary, VHDL is a structured language with entity and architecture pairs.

FPGA Design for Embedded Systems

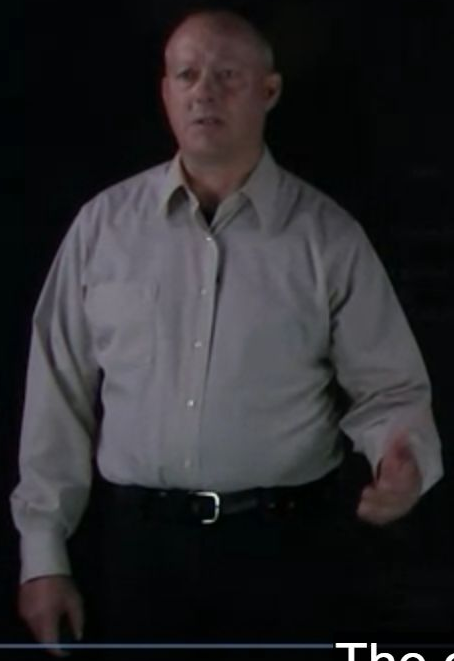
Hardware Description Languages for Logic Design

Hello, and welcome back to FPGA

University of Colorado Boulder

Objectives

- Explore the history of VHDL
- Methods for learning VHDL
- Syntax and practice



The objectives of this are to

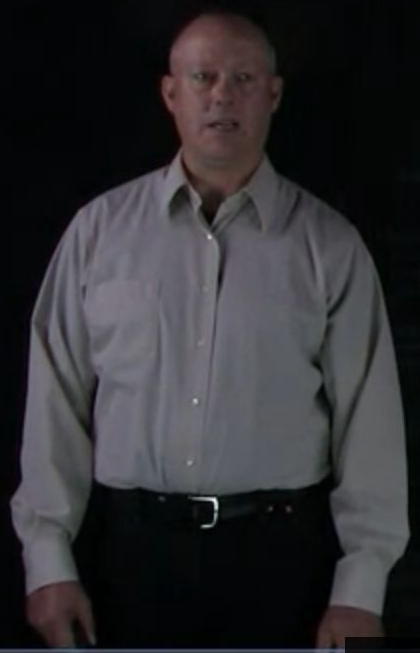
What is VHDL?

- VHDL is a programming language
- VHDL is structured to
  - Describe
  - Model (Simulate)
  - Synthesize (Translate) into digital electronics
- VHDL has abstraction levels
  - Transistor (AND, Flip-Flop)
  - Complete System (Hierarchy Levels)

VHDL is a programming language.

University of Colorado Boulder

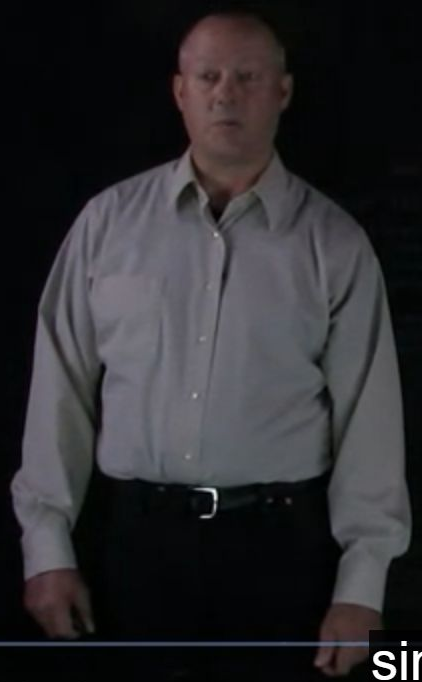
What is VHDL?



- VHDL is an IEEE standard
  - V = VHSIC
  - VHSIC = Very High Speed Integrated Circuit
  - HDL = Hardware Description Language
- VHDL was a project sponsored by the US Government and the Air Force which began in 1980

and has become an IEEE standard called 1076-1987.

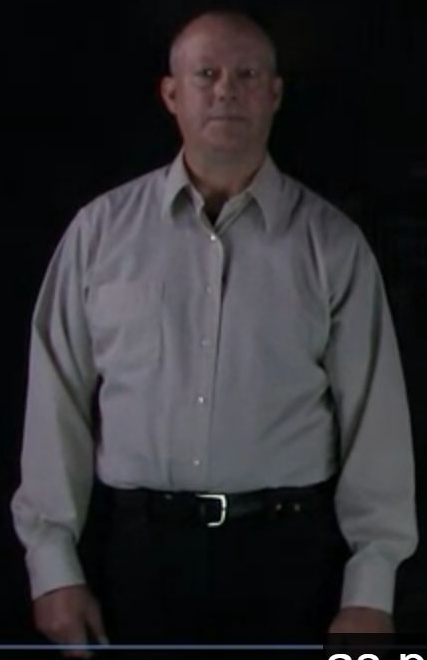
Learning to “Speak” VHDL



- Strategies for learning any language
  - Reduction (Basic elements)
  - Immersion (Simulation, Synthesis)
  - Repetition (Practice)
- First phrases
  - Start with useful phrases (Skeleton)
  - Deconstruct “grammar” (syntax) elements

similar to the examples
we've already shown,

VHDL Keywords

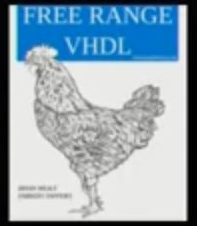


- Reserved words (short list)
- Can not be used as identifiers
  - Ex. do not use “access” as a bus name

access	after	alias	all	attribute	block
body	buffer	bus	constant	exit	file
for	function	generic	group	in	is
label	loop	mod	new	next	null
of	on	open	out	range	rem
return	signal	shared	then	to	type
until	use	variable	wait	while	with

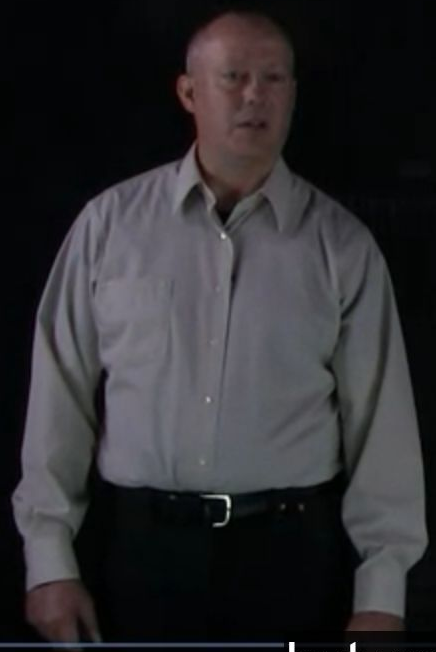
Reference : Bryan Mealy, Fabrizio Tappero, 2012/01/13,
FREE RANGE VHDL : freerangefactory.org

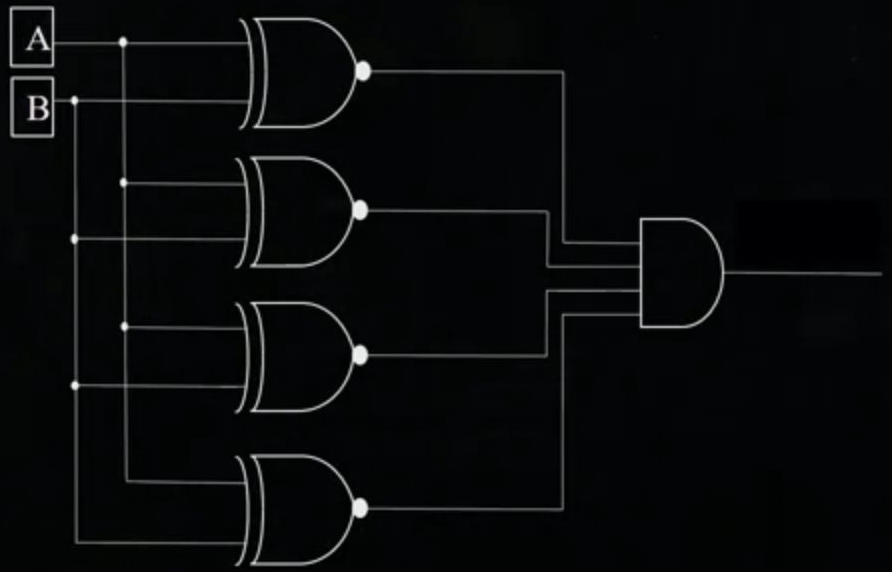
free ebook



as provided by the link here.

Example: 4-bit Comparator in VHDL



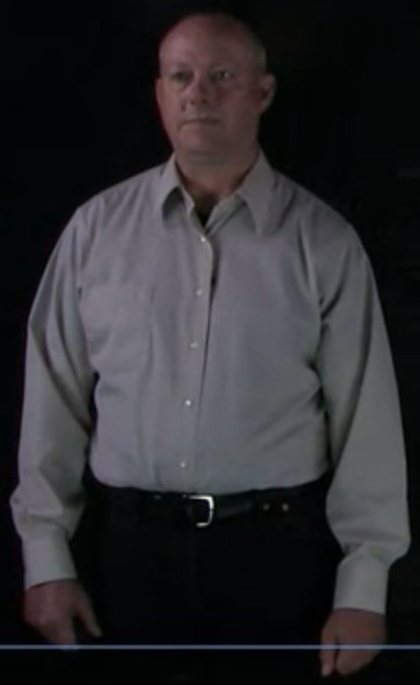


but we can also implement it in

Share

Press Esc to exit full screen

VHDL Modeling



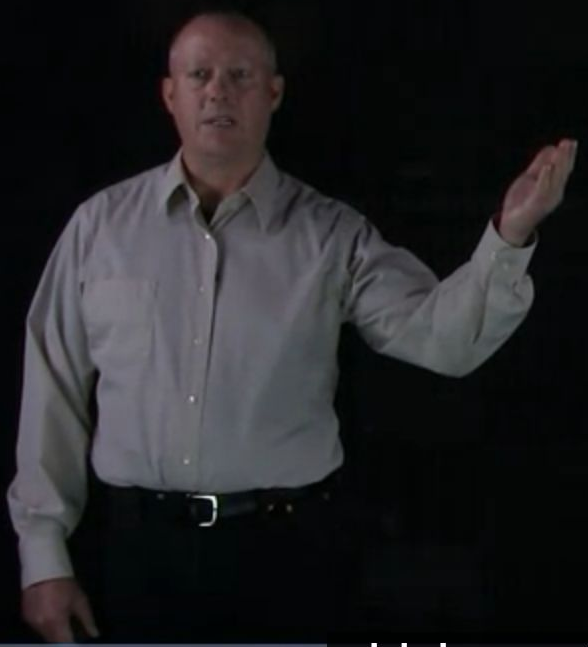
- Structural modeling (Gate-level)
  - Library defined primitive gates (and2/or2)
  - Boolean, bitwise logical (and/or)
  - Library user defined functions (and17)
- Dataflow modeling
  - Use assignment and select statements
- Behavioral modeling
  - Use assignments within a process
  - process(A, B) , sensitivity list of A, B signals

the FPGA in many different fashions.

University of Colorado Boulder

3:18 / 6:58

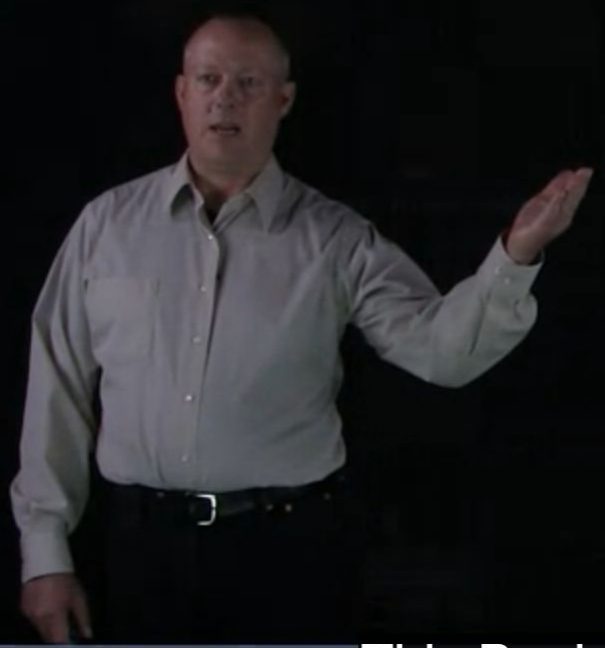
4-bit Comparator – Structural Description



```
-- Use standard IEEE library
library IEEE;
use IEEE.std_logic_1164.all;

-- Entity
entity Comparator is port (
  A,B      : in  std_logic_vector(3 downto 0);
  Result   : out std_logic);
end Comparator;

use work.gatespkg.all;

-- Architecture
-- Structural gate description
architecture struct of Comparator is
  signal x : std_logic_vector(3 downto 0);
begin
  u3: xnor2 port map (A(3), B(3), X(3));
  u2: xnor2 port map (A(2), B(2), X(2));
  u1: xnor2 port map (A(1), B(1), X(1));
  u0: xnor2 port map (A(0), B(0), X(0));

  u4: and4  port map (X(3), X(2), X(1), X(0), Result);

end struct;
```

which are 4-bit wide buses,

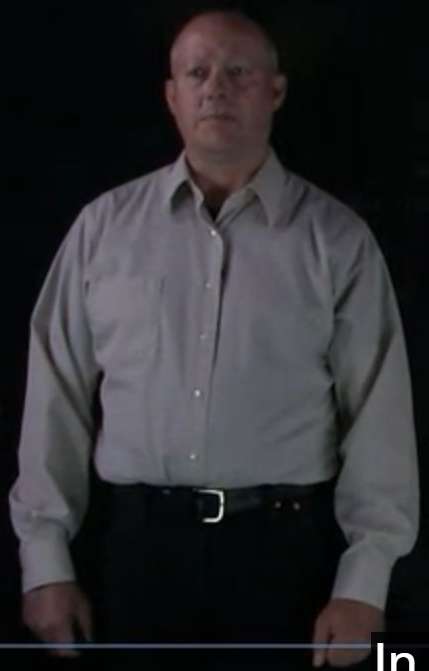
4-bit Comparator – Boolean Description



```
-- Use standard IEEE library
library IEEE;
use IEEE.std_logic_1164.all;

-- Entity
entity Comparator is port (
  A,B    : in  std_logic_vector(3 downto 0);
  Result : out std_logic);
end Comparator;

-- Architecture
-- Boolean logic description
architecture bool of Comparator is
begin
  Result <= not(A(3) xor B(3)) and
            not(A(2) xor B(2)) and
            not(A(1) xor B(1)) and
            not(A(0) xor B(0));
end bool;
```

This Boolean description has the same entity inputs,

4-bit Comparator – Dataflow Description



```
-- Use standard IEEE library
library IEEE;
use IEEE.std_logic_1164.all;

-- Entity
entity Comparator is port (
  A,B    : in  std_logic_vector(3 downto 0);
  Result : out std_logic);
end Comparator;

-- Architecture
-- Dataflow description
architecture dataflow of Comparator is
begin
  Result <= '1' when (A=B) else '0';
end dataflow;
```

In this example, we have a dataflow description.

4-bit Comparator – Behavioral Description

```
-- Use standard IEEE library
library IEEE;
use IEEE.std_logic_1164.all;

-- Entity
entity Comparator is port (
  A,B    : in  std_logic_vector(3 downto 0);
  Result : out std_logic);
end Comparator;

-- Architecture
-- Behavioral description
architecture behavioral of Comparator is
begin
  CompareProcess : process(A, B)
  begin
    if (A=B) then
      Result <= '1';
    else
      Result <= '0';
    end if;
  end process CompareProcess;
end behavioral;
```

the behavioral description is

Structural modeling in VHDL uses the process and sensitivity list.

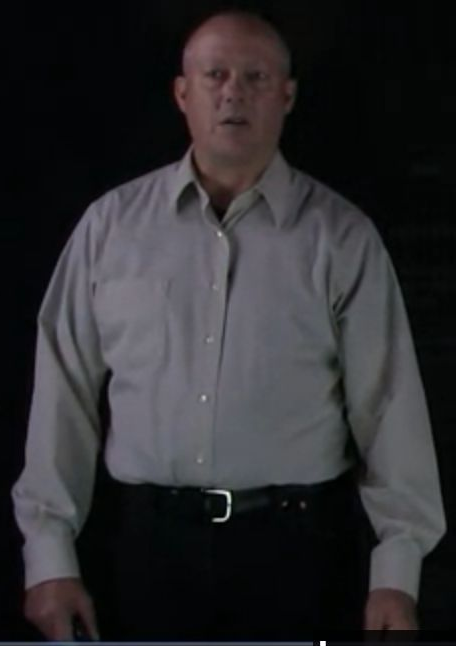
- False

Correct
Correct. Behavioral modeling uses the process sensitivity list.

- True.

Continue

Summary



- History and definition of VHDL
- An approach to learning VHDL, involving assimilation of vocabulary, phrases and syntax
- Initial design examples : 4-bit comparator
  - Structural, Boolean, Dataflow, Behavioral

In summary, we've gone over

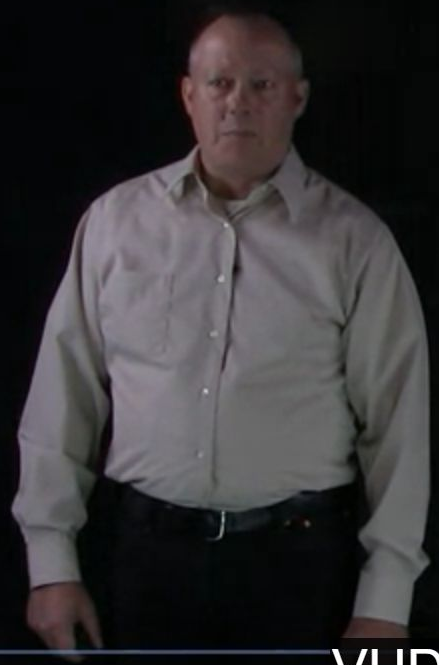

Share
FPGA Design for Embedded Systems
Hardware Description Languages for Logic Design
University of Colorado Boulder
Copyright © 2019 University of Colorado
0:01 / 3:52

Objectives

- VHDL Assignments : signals and variables
- Operators : add, multiply, and everybody shift to the left

multipliers, and everybody
shift to the left.

VHDL Assignments



- Signals
  - Signal wire to the entity : std_logic
  - Scheduled update
  - <= signal assignment operator
  - Z <= A and B; -- concurrent, gates silicon
  - Z <= D after 5ns; -- sequential, simulation
- Variables
  - Variable only within the process : integer
  - Immediate update
  - := variable assignment operator
  - count := count +1; --simulation loop counter
  - a := 27; -- variable assignment

VHDL assignments include the signal assignment operator

The variable assignment := updates in the process immediately.

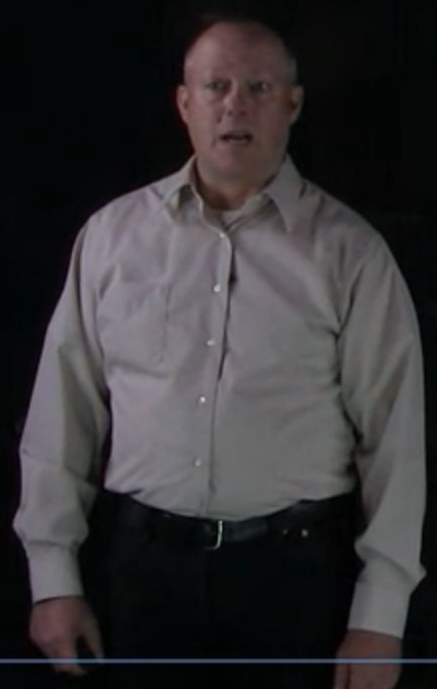
- (•) True.

Correct
Correct. The variable updates immediately and does not need to wait for an event like a clock edge in the process.

- ( ) False.

Continue

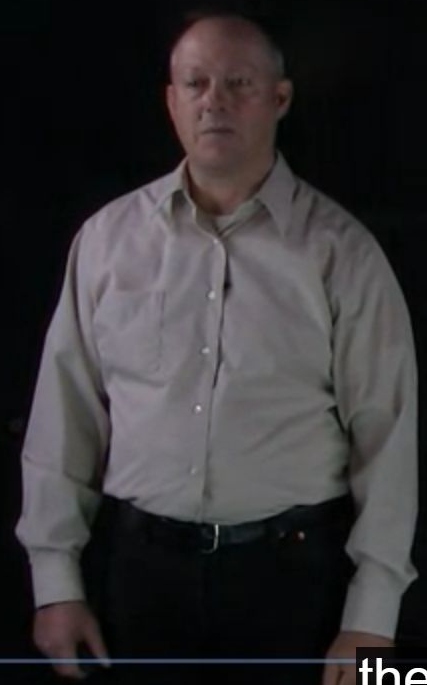
Select Assignment



```
entity mux_4 is port(
        a, b, c, d:  in std_logic_vector(3 downto 0);
        s:           in std_logic_vector(1 downto 0);
        x:          out std_logic_vector(3 downto 0));
end mux_4;

architecture sel_arch of mux_4 is
begin
        with  s select
        x <= a when "00";
                b when "01";
                c when "10";
                d when "11";
                d when others;

end sel_arch;
```

An example of the Select Assignment,

Conditional Assignment

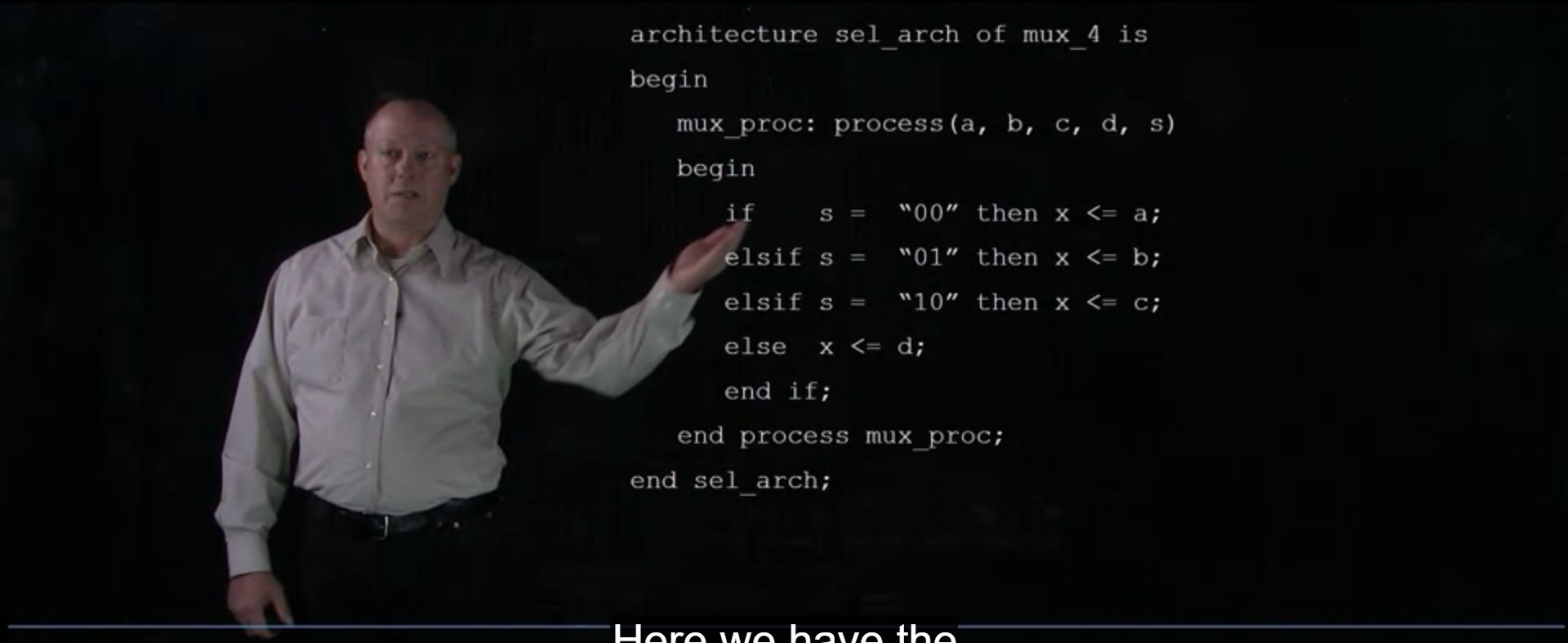


```
architecture sel_arch of mux_4 is
begin
  x <= a when ( s ="00") else
       b when ( s ="01") else
       c when ( s ="10") else
       d;
end sel_arch;
```

the else statement selecting
between each of our options.

University of Colorado Boulder

Process Assignment



Here we have the Process Assignment.

VHDL Operators

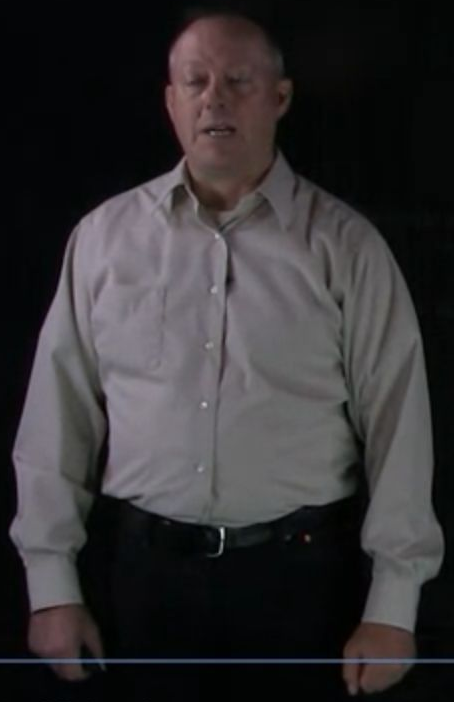
- Operators

```
**    exponent          abs    absolute value
not   complement        + -    add or subtract
*     multiply            /    divide
mod   modulo            rem    remainder

sll, srl       shift left, right
rol, ror       rotate left, right
=, /=, <, <=, >, >=       equality, greater, less than
and, or, nand, nor, xor, xnor
```

- Order precedence
  - Left to right, Parenthesis
  - Unary single operand on right mod A
  - Binary operators on both sides A+B

In VHDL we have a variety of operators such as;

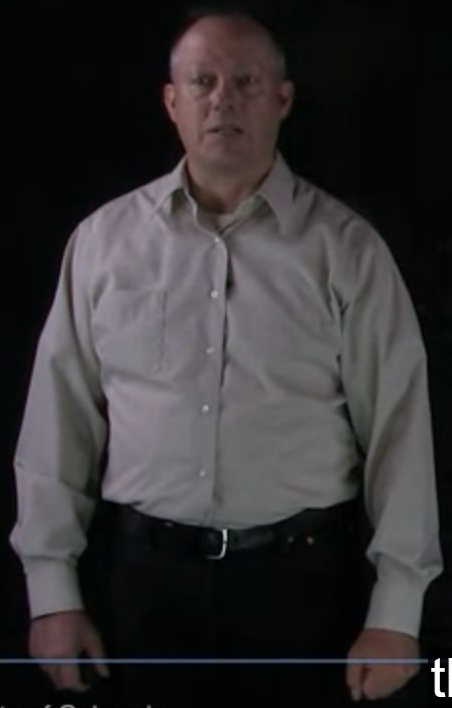
Data Types - Array and Scalar



- Array
  - string “abc”
  - bit_vector “1001”
  - std_logic_vector “101Z”
- Scalar
  - character ‘a’
  - bit ‘1’, ‘0’
  - std_logic ‘1’, ‘0’, ‘X’, ‘Z’
  - boolean true, false
  - real, integer 3.14 1E+0, 27
  - time fs, ps, ns, us, ms

When we look at
Data Types in VHDL,

Summary



- VHDL Assignments : signal and variable
  - Logical Select
- Operators - Dial 0
- Scalar and Array data types

the VHDL assignments:
signal and variable,

FPGA Design for Embedded Systems

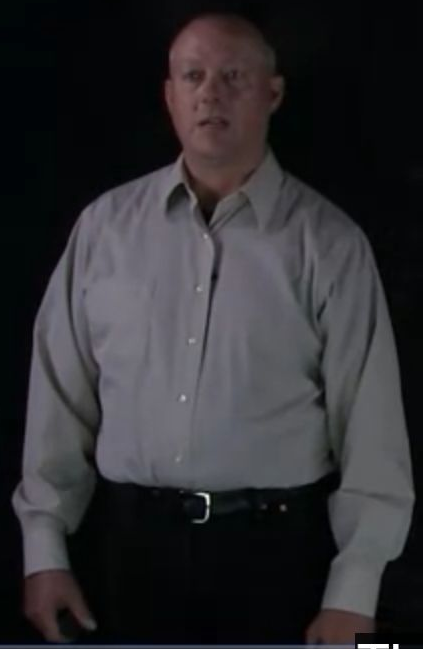
Hardware Description Languages for Logic Design

University of Colorado Boulder

Hello and welcome back to fpga design for embedded systems.

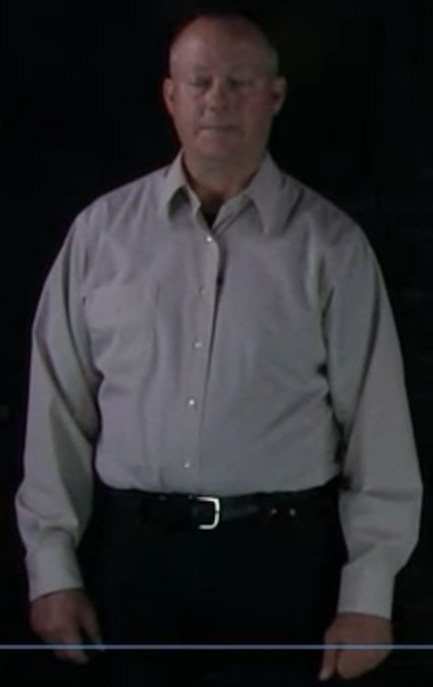
Objectives

- VHDL rules and syntax
- Understand VHDL Constructs



The objective of this class
are vhdl rules and syntax and

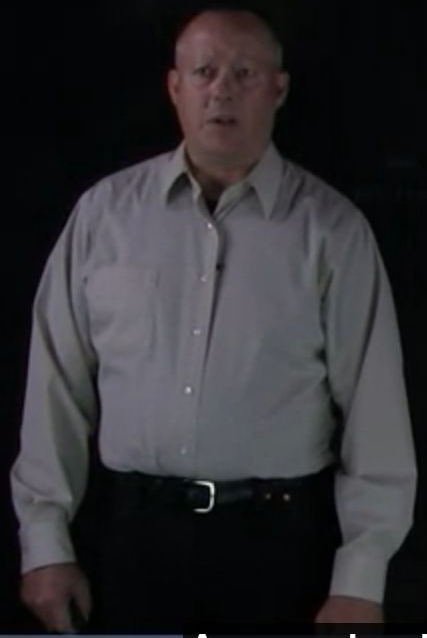
VHDL Rules and Syntax



- Case Sensitivity
  - VHDL is not case sensitive. Equivalent :
    - `DATA_out <= A_in and B_in;`
    - `data_OUT <= a_IN AND b_IN;`
- Tabs and Spaces
  - Use for readable coding style. Equivalent :
    - `D_out <= a_in or b_in;`
    - `D_out <= a_in  or  b_in;`
- Comments
  - -- dash dash at the beginning or end of line
- Semicolon ; Terminates each statement

every statement is
terminated with a semicolon.

Deconstructing VHDL Components

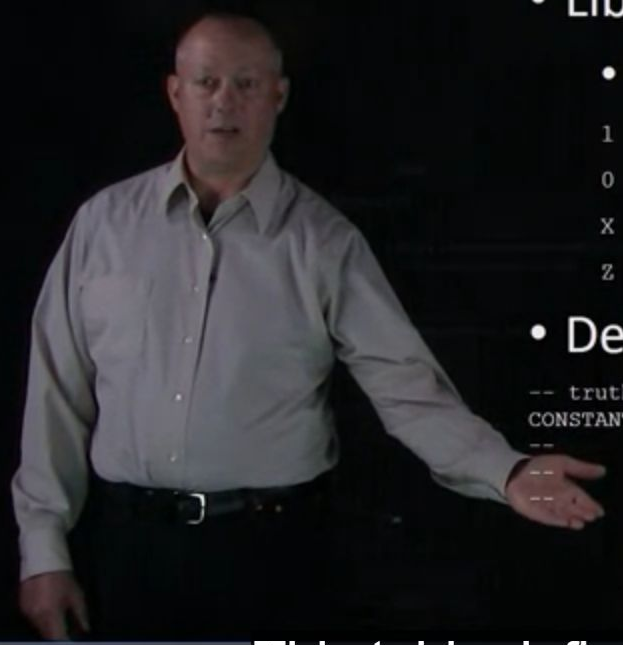


```
-- Use standard IEEE library
library IEEE;
use IEEE.std_logic_1164.all;

-- Entity
entity Comparator is port (
  A,B     : in  std_logic_vector(3 downto 0);
  Result  : out std_logic);
end Comparator;

-- Architecture
-- Dataflow description
architecture dataflow of Comparator is
begin
  Result <= '1' when (A=B) else '0';
end dataflow;
```

As we look at the main components of vhdl we can see that we have a standard

Library : IEEE Standard

```
-- Use standard IEEE library
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;  -- arithmetic
```



- Library is text readable (take a look inside)
  - Defines standard types : std_ulogic

```
1    logic one,       strong drive
0    logic zero,      strong drive
X    unknown value,   strong drive
Z    high impedance, tri-state  . . .
```

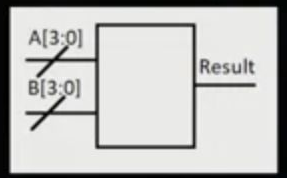
  - Defines Functions : and

```
-- truth table for "and" function
CONSTANT and_table : stdlogic_table := (
--      ---------------------------------------------------
--      |  U    X    0    1    Z    W    L    H    -     |   |
--      ---------------------------------------------------
        ( 'U', 'U', '0', 'U', 'U', 'U', '0', 'U', 'U' ),  -- | U |
        ( 'U', 'X', '0', 'X', 'X', 'X', '0', 'X', 'X' ),  -- | X |
        ( '0', '0', '0', '0', '0', '0', '0', '0', '0' ),  -- | 0 |
        ( 'U', 'X', '0', '1', 'X', 'X', '0', '1', 'X' ),  -- | 1 | . . .
```

This table defines all the possible values for the outputs of an ant

Entity : Interface ports

```
-- Entity
entity Comparator is port (
  A,B     : in  std_logic_vector(3 downto 0);
  Result  : out std_logic);
end Comparator;
```

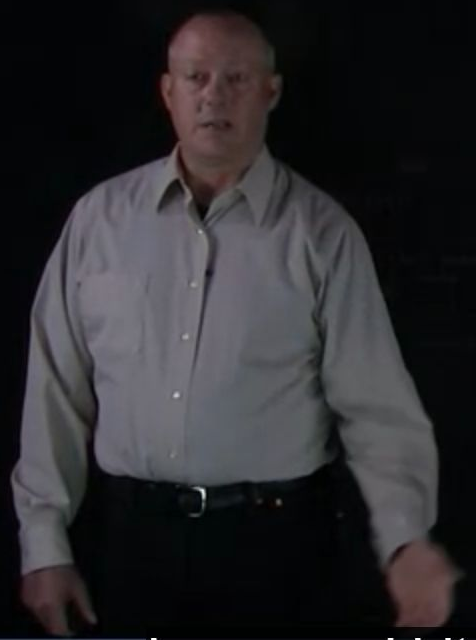


- Ports : blackbox
  - Direction

```
in      input
out     output
inout   bi-directional
```

  - Vector (bus, bundle) and bit

```
std_logic_vector (3 downto 0)     bus  1,0,X,Z,W,L,H,U,-
std_logic                         bit  1,0,X,Z,W,L,H,U,-
unsigned (127 downto 0)           bus (full 128 bits data)
```



buses and bits can be defined as single or

The type std_logic_vector has 4 value settings of 1,0,X,Z.

- True.
- False.

Correct
Correct. There are Quantity 9 values of 1,0,X,Z,W,L,H,U,-

Continue

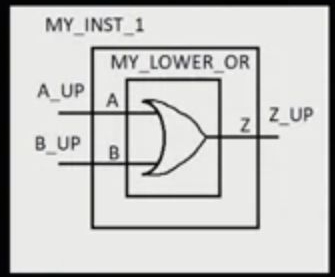
Share

Press Esc to exit full screen

Architecture : Design

```
-- Architecture
architecture dataflow of Comparator is
begin
  Result <= '1' when (A=B) else '0';
end dataflow;
```

- Functional definition of circuit design
  - Simple to Complex
  - Code re-usable
  - Chip top to lowest hierarchy
  - Instantiate



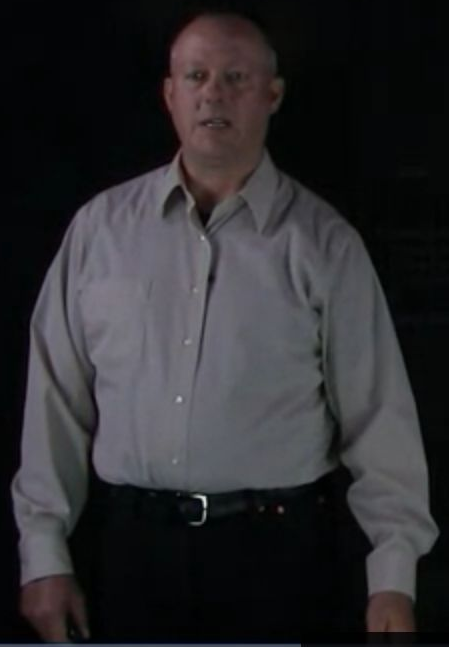

```
architecture  MY_HIER  of MY_UPPER is
  component  MY_LOWER_OR  port (A,B: in std_logic; Z: out std_logic);
  end component;
begin
  MY_INST_1: MY_LOWER_OR  port map (A=>A_UP, B=>B_UP, Z=>Z_UP);
end architecture  MY_HIER;
```

We're instantiating my instance from
a lower level definition of my lower or

University of Colorado Boulder

3:21 / 3:50

Summary



- Create readable code style by using :
  - Spaces, tabs
  - Other readable code sources as examples
  - Use comments before and on each line
- Library, Entity, Architecture

every vhdl block a library and
an entity and architecture.

FPGA Design for Embedded Systems

Hardware Description Languages for Logic Design

VHDL Simulation Example : 4-bit Adder

```
-- Entity
entity Add4 is port (
  Data1,Data2 : in  std_logic_vector(3 downto 0);
  Cin         : in  std_logic;
  Cout        : out std_logic;
  Sum         : out std_logic_vector(3 downto 0) );
end entity Add4;

-- Architecture
architecture RTL of Add4 is
    signal Out5bit : unsigned(4 downto 0);
begin
  Out5bit <= ('0' & Data1) + ('0' & Data2) + Cin;
  Sum     <= Out5bit(3 downto 0); -- 4 bits
  Cout    <= Out5bit(4);          -- 5th bit
end architecture RTL;
```

Hello and welcome to FPGA design for embedded systems.

Downloading ModelSim : Go to Link :

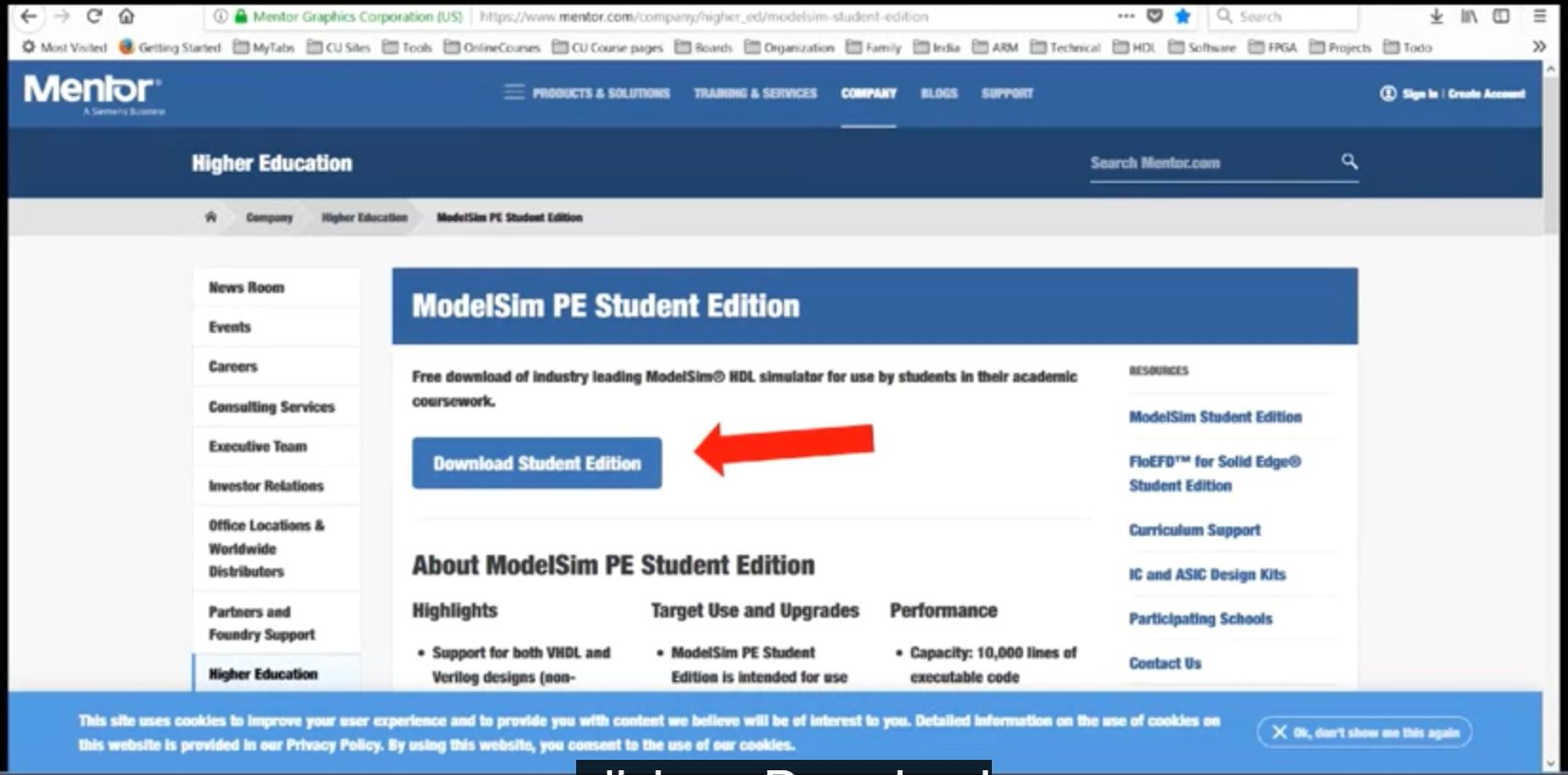
https://www.mentor.com/company/higher_ed/modelsim-student-edition

Downloading ModelSim Alternative

- If you took the Introduction to FPGA Design Course, Course 1 of this video series, you may already have : ModelSim-Altera version of ModelSim installed.
- If you did not take Course 1 the Altera FPGA version is still a good alternative.
- Follow the directions in ALTERA QUARTUS DOWNLOAD AND INSTALLATION.docx to install the Altera version of ModelSim.

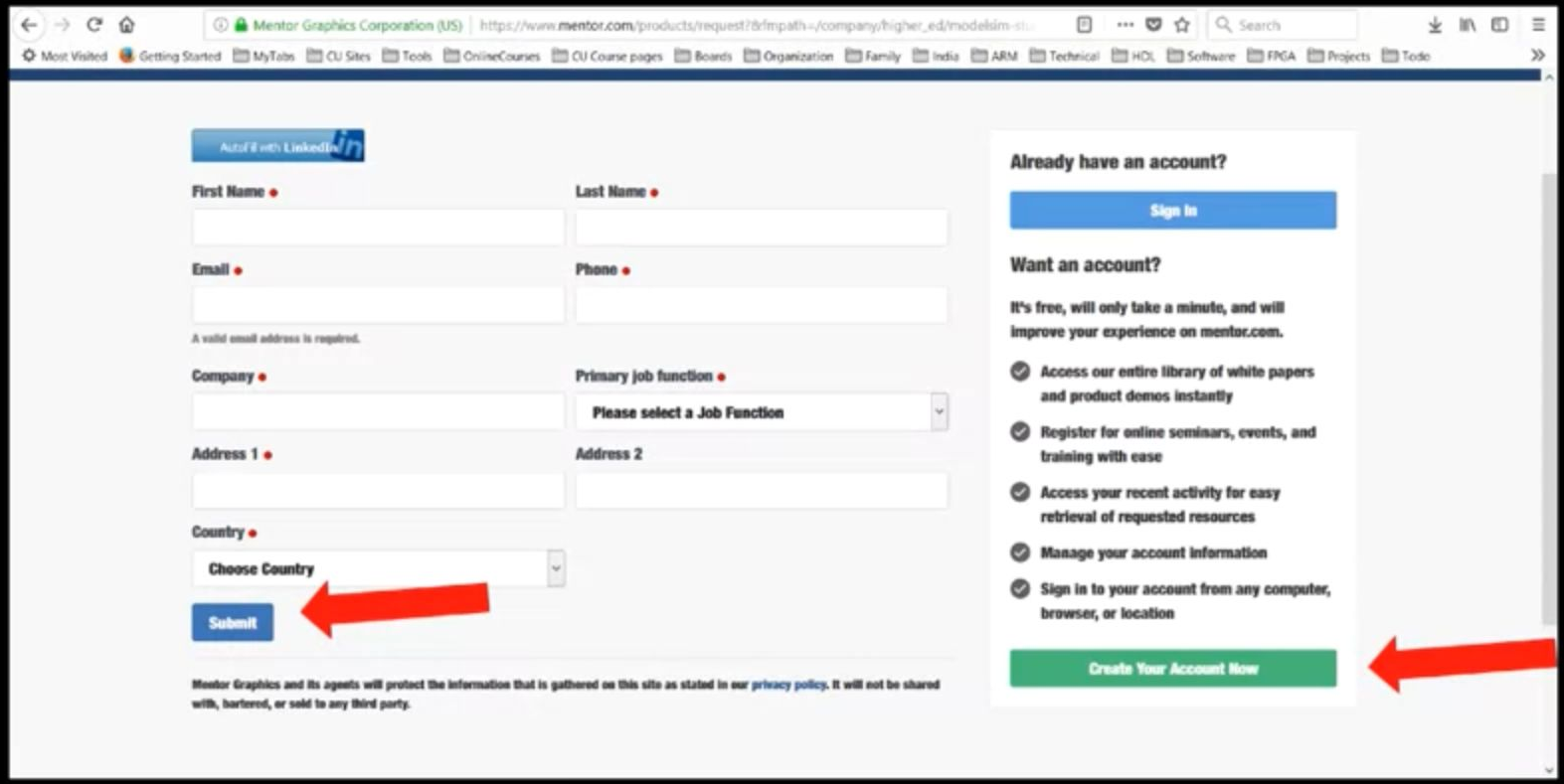
installation.docx document.

Downloading Modelsim



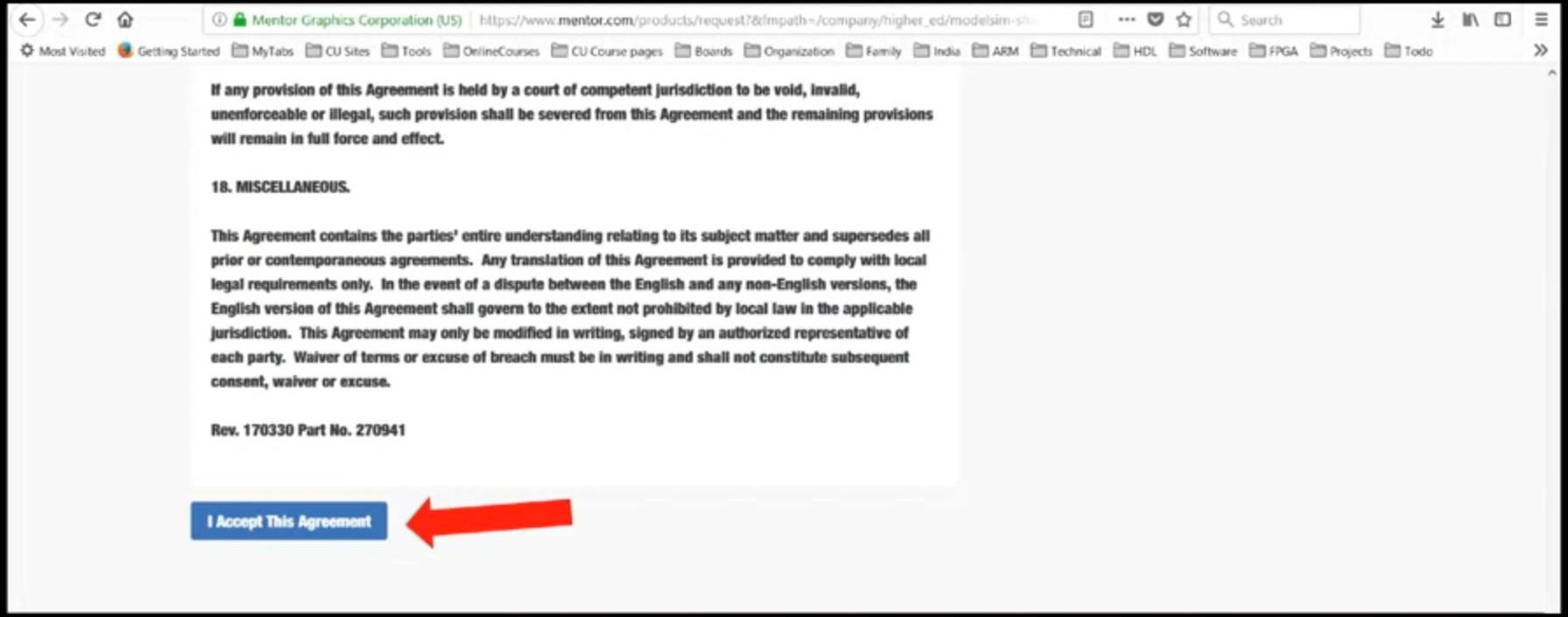
click on Download Student Edition.

Downloading Modelsim



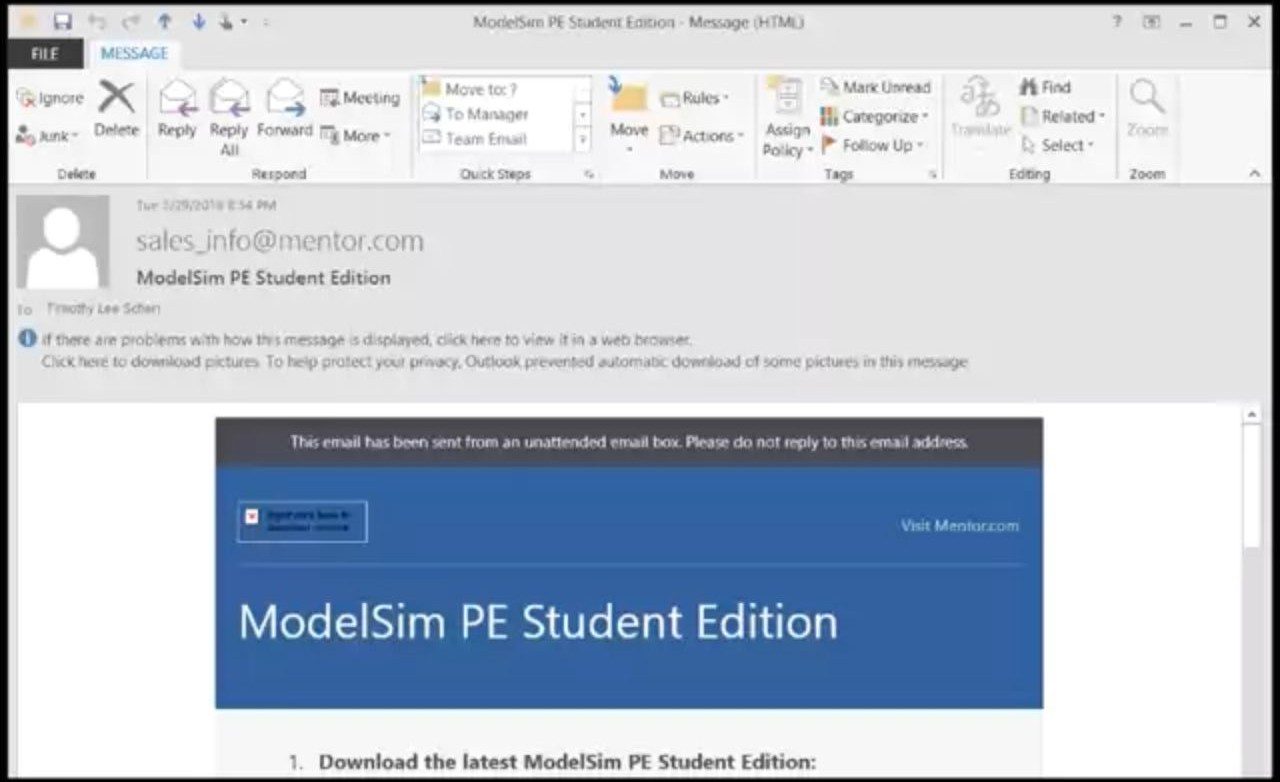
Please fill out your information and click Submit.

Downloading Modelsim



Click I Accept This Agreement.

Downloading Modelsim



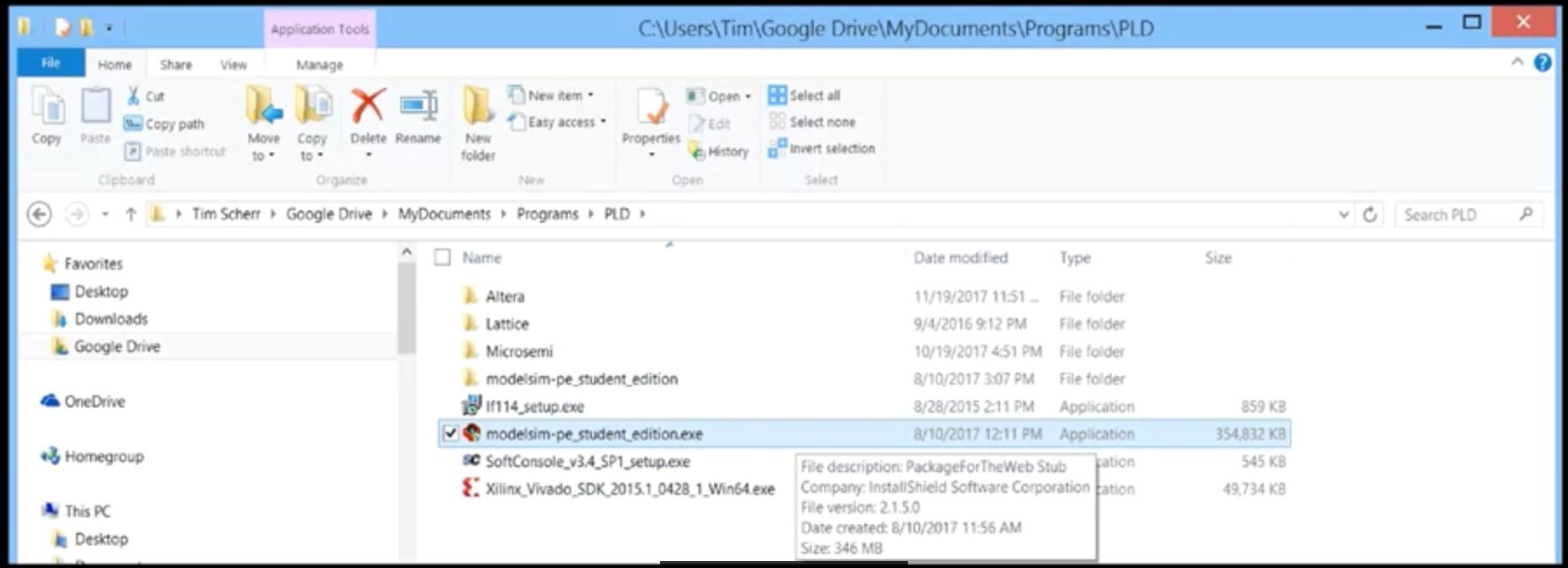
Installing ModelSim

- After the file downloads completely, double-click on the .exe file to begin the installation process.
- You must agree to the Mentor Graphics End-User License agreement during installation to continue.

during the installation process.

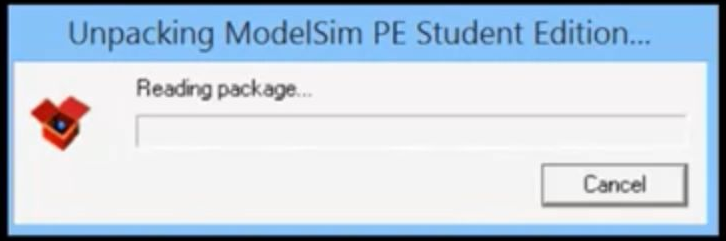
Installing ModelSim

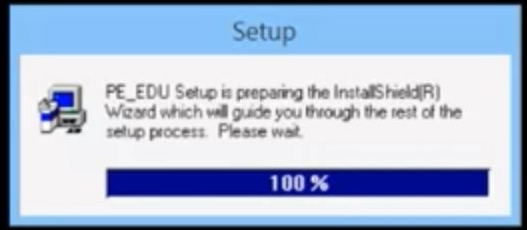
- Double-click on the .exe file to begin the installation process.



to begin the installation process.

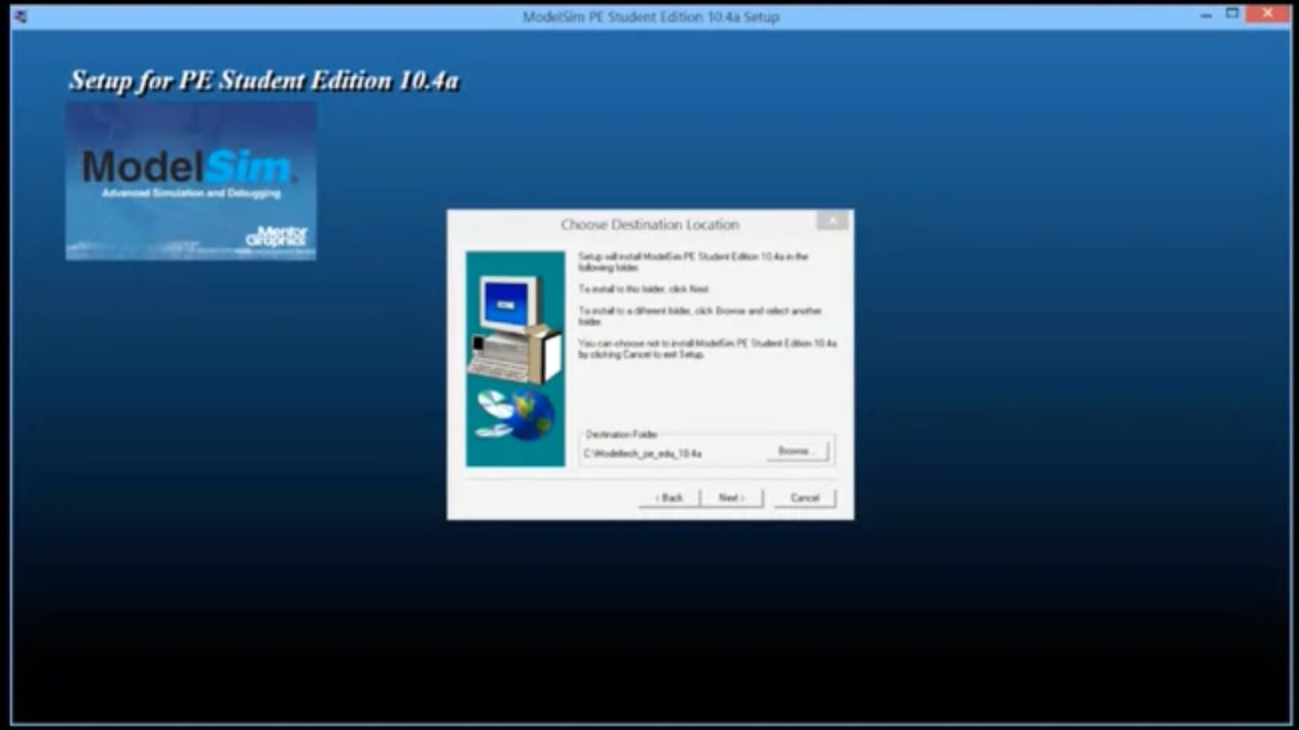
Installing ModelSim





setup and accept the default locations

Installing ModelSim



setup and accept the default locations

Summary –

In this video, you have learned:

- How to download an HDL Simulator, ModelSim from Mentor Graphics
- How to install ModelSim on your PC

download the HDL simulator

FPGA Design for Embedded Systems

Hardware Description Languages for Logic Design

VHDL Evaluation

In this video, you will learn:

- How to test code using an HDL simulator.
- How to evaluate the correctness of your design using waveform analysis.
- How to control the simulator to examine the code and its output.

In this video, you will learn how to test

VHDL Evaluation: 4-bit Adder

```
-- Entity
entity Add4 is port (
  Data1,Data2 : in  std_logic_vector(3 downto 0);
  Cin         : in  std_logic;
  Cout        : out std_logic;
  Sum         : out std_logic_vector(3 downto 0) );
end entity Add4;

-- Architecture
architecture RTL of Add4 is
    signal Out5bit : std_logic_vector(4 downto 0);
begin
  Out5bit <= ('0' & Data1) + ('0' & Data2) + Cin;
  Sum     <= Out5bit(3 downto 0); -- 4 bits
  Cout    <= Out5bit(4);          -- 5th bit
end architecture RTL;
```

a sum four bit output and a carry out bit.

Using ModelSim

Let's Simulate !

We will use the provided Add4.vhd code in ModelSim.

Start ModelSim now and follow the steps as we simulate this code together.

Pause the Video when needed.

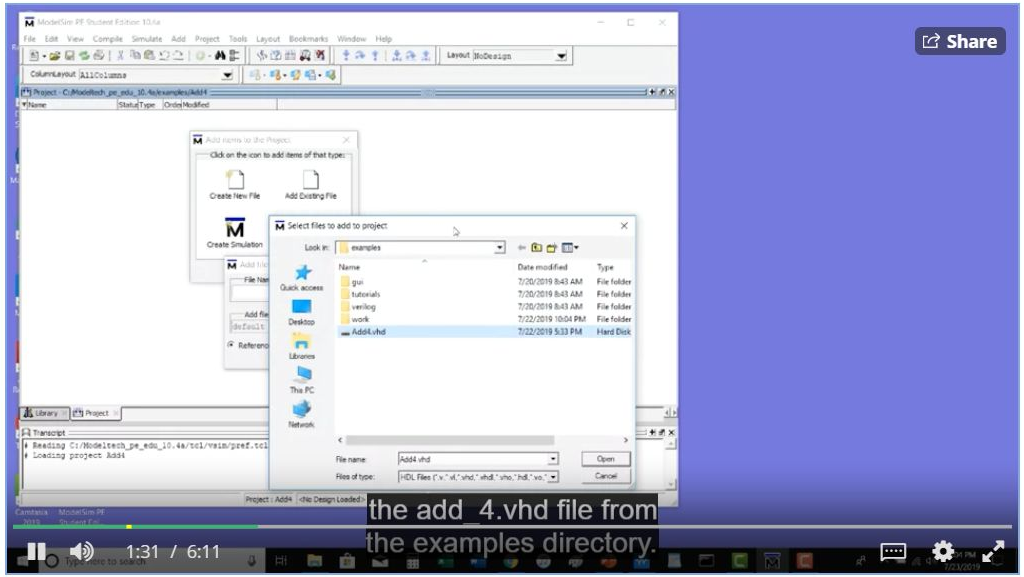
the provided add for
VHDL code in ModelSim.

ATTENTION: You can create the Add4.vhd by typing the code given in previous slide using an editor of your choice, or Modesim file editor. Use the following libraries:

```
library ieee;

use ieee.std_logic_1164.all;

use ieee.std_logic_unsigned.all;
```

Continue

VHDL in ModelSim: Adding to your Toolkit



Save Note | Discuss | Download | Share

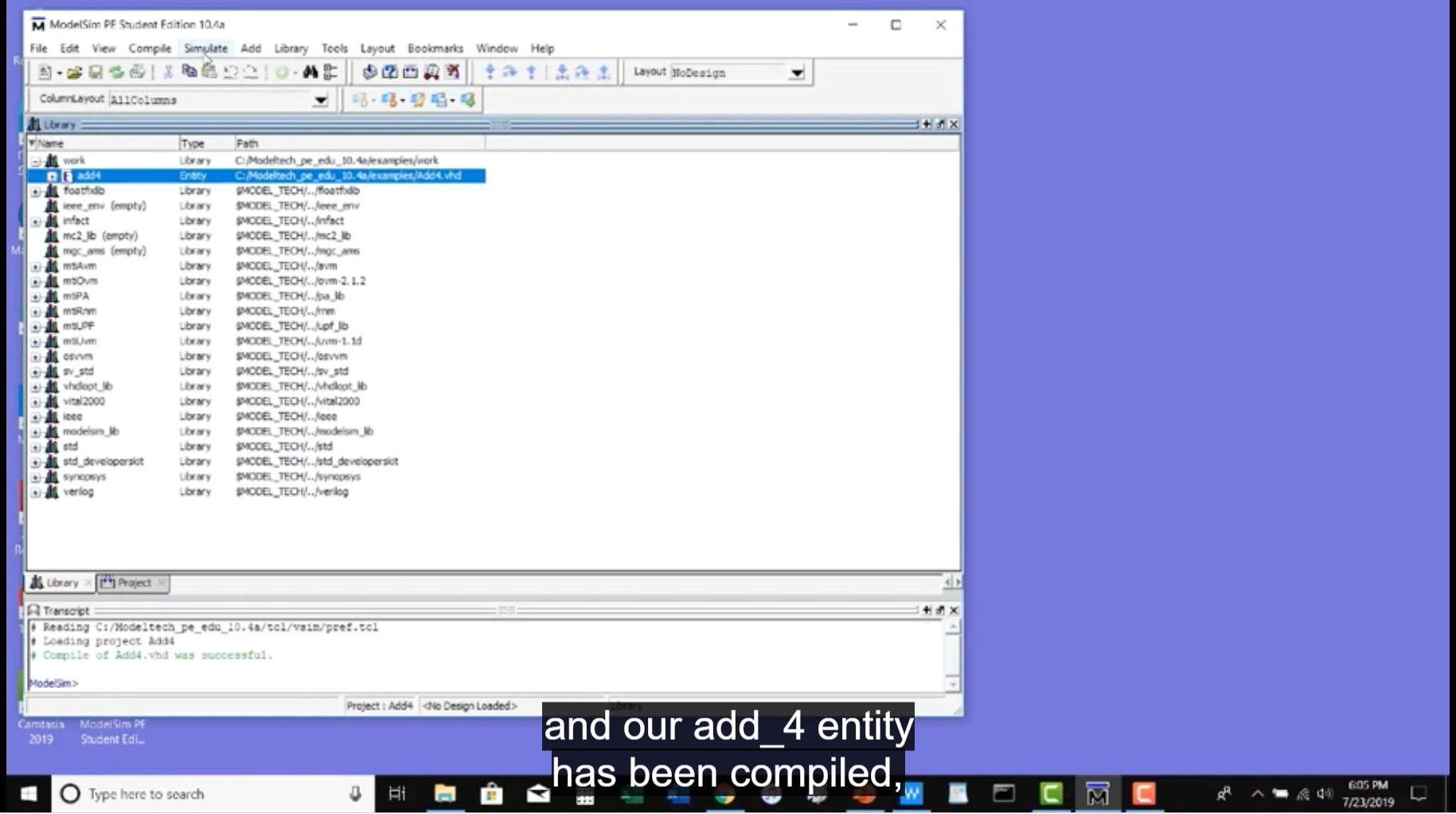
English

Help Us Translate

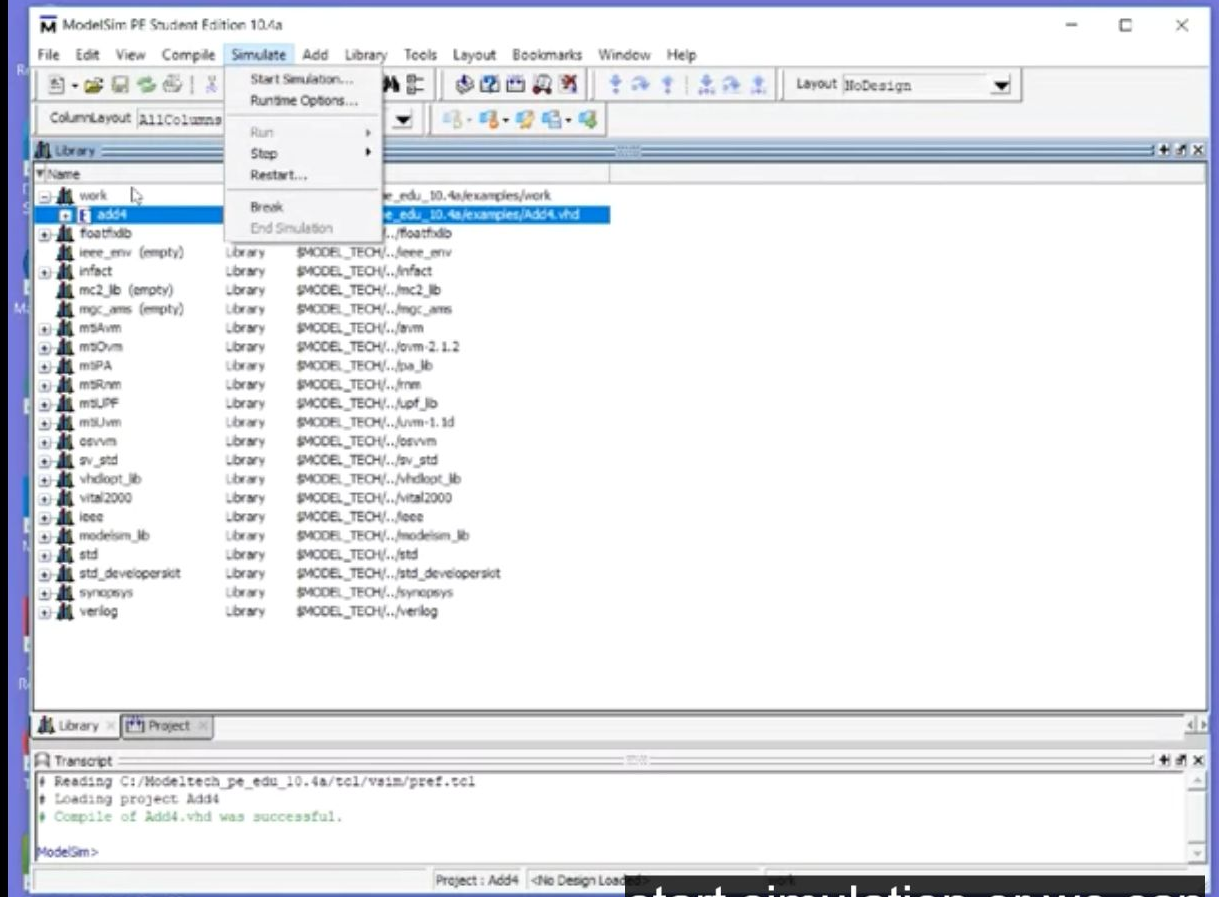
Notes

All notes

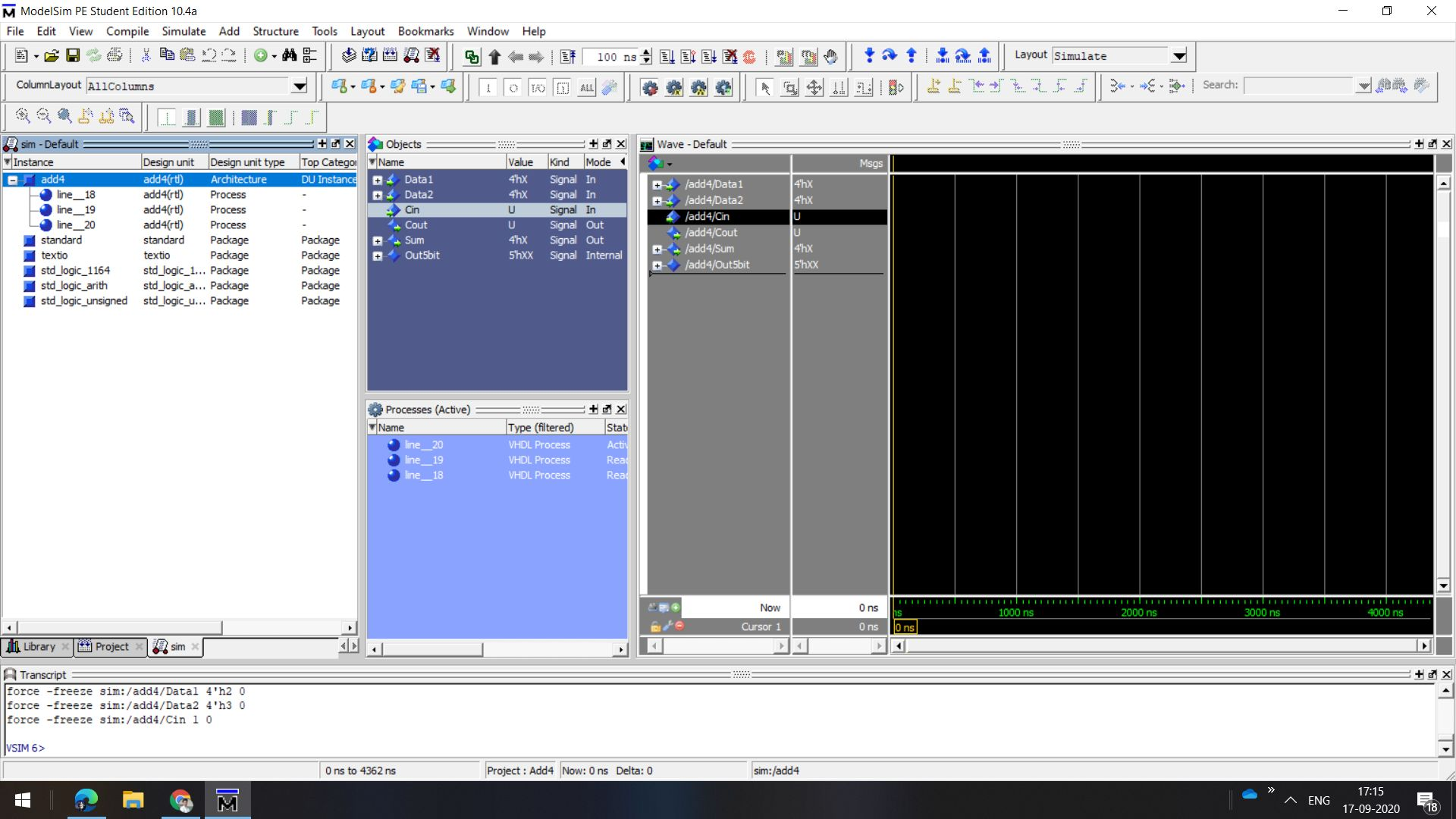
Click the "Save Note" button when you want to capture a screen. You can also highlight and save lines from the transcript below. Add your own notes to anything you've captured.

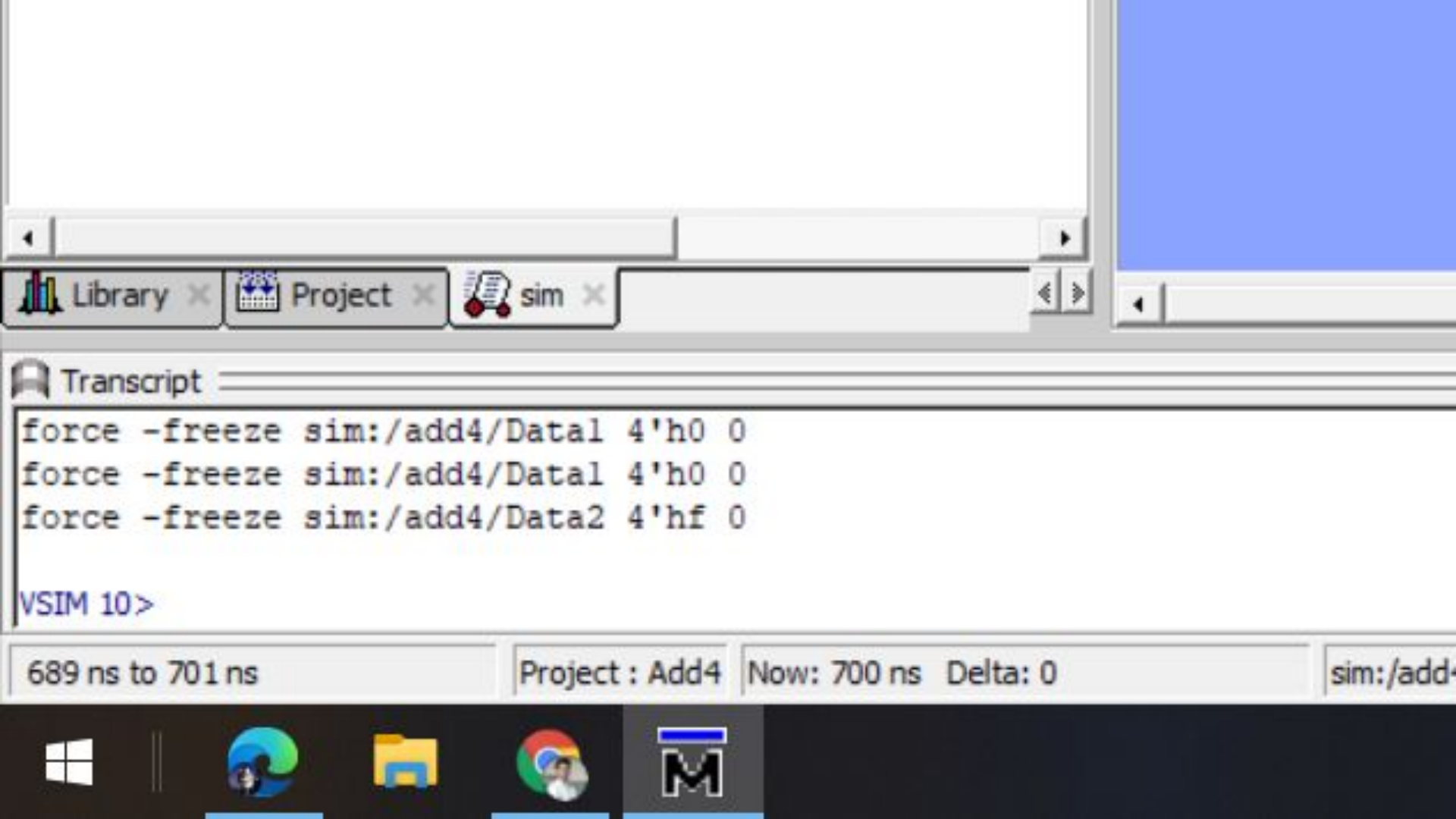


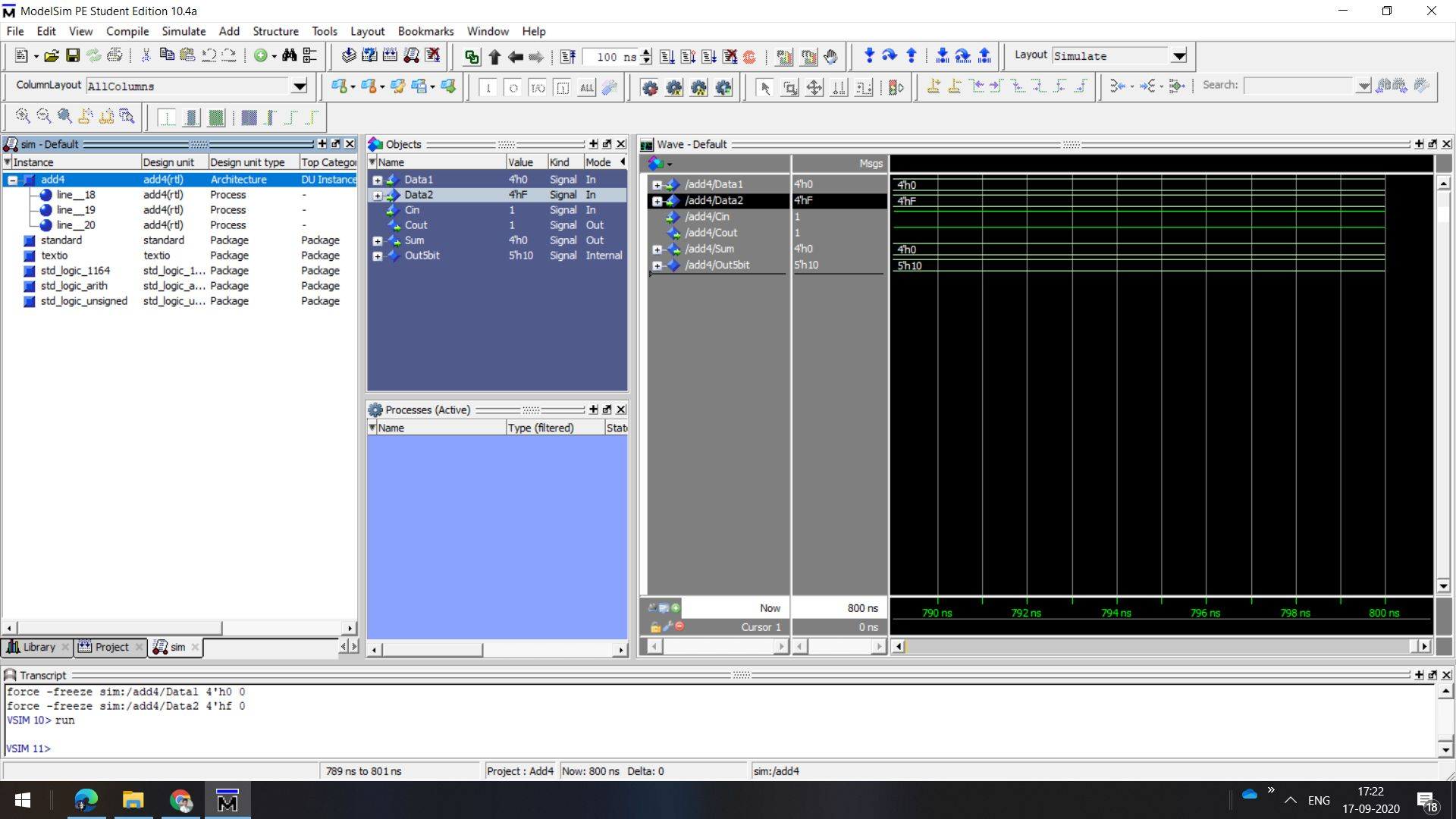
and our add_4 entity has been compiled,



start simulation or we can right-click "Simulate."


ModelSim PE Student Edition 10.4a
File Edit View Compile Simulate Add Structure Tools Layout Bookmarks Window Help
sim - Default
Instance Design unit Design unit type Top Categor
add4 add4(rtl) Architecture DU Instance
line__18 add4(rtl) Process -
line__19 add4(rtl) Process -
line__20 add4(rtl) Process -
standard standard Package Package
textio textio Package Package
std_logic_1164 std_logic_1... Package Package
std_logic_arith std_logic_a... Package Package
std_logic_unsigned std_logic_u... Package Package
Objects
Name Value Kind Mode
Data1 4'hX Signal In
Data2 4'hX Signal In
Cin U Signal In
Cout U Signal Out
Sum 4'hX Signal Out
Out5bit 5'hXX Signal Internal
Processes (Active)
Name Type (filtered)
line__20 VHDL Process
line__19 VHDL Process
line__18 VHDL Process
Wave - Default
Msgs
/add4/Data1 4'hX
/add4/Data2 4'hX
/add4/Cin U
/add4/Cout U
/add4/Sum 4'hX
/add4/Out5bit 5'hXX
Now 0 ns
Cursor 1 0 ns
1000 ns 2000 ns 3000 ns 4000 ns
0 ns
Library Project sim
Transcript
force -freeze sim:/add4/Data1 4'h2 0
force -freeze sim:/add4/Data2 4'h3 0
force -freeze sim:/add4/Cin 1 0
VSIM 6>
0 ns to 4362 ns
Project : Add4
Now: 0 ns Delta: 0
sim:/add4
ENG
17:15
17-09-2020


Library
Project
sim
Transcript
force -freeze sim:/add4/Data1 4'h0 0
force -freeze sim:/add4/Data1 4'h0 0
force -freeze sim:/add4/Data2 4'hf 0
VSIM 10>
689 ns to 701 ns
Project : Add4
Now: 700 ns Delta: 0
sim:/add4


ModelSim PE Student Edition 10.4a
File Edit View Compile Simulate Add Structure Tools Layout Bookmarks Window Help
sim - Default
Instance Design unit Design unit type Top Categor
add4 add4(rtl) Architecture DU Instance
line__18 add4(rtl) Process -
line__19 add4(rtl) Process -
line__20 add4(rtl) Process -
standard standard Package Package
textio textio Package Package
std_logic_1164 std_logic_1... Package Package
std_logic_arith std_logic_a... Package Package
std_logic_unsigned std_logic_u... Package Package
Objects
Name Value Kind Mode
Data1 4'h0 Signal In
Data2 4'hF Signal In
Cin 1 Signal In
Cout 1 Signal Out
Sum 4'h0 Signal Out
Out5bit 5'h10 Signal Internal
Processes (Active)
Name Type (filtered)
Wave - Default
Msgs
/add4/Data1 4'h0
/add4/Data2 4'hF
/add4/Cin 1
/add4/Cout 1
/add4/Sum 4'h0
/add4/Out5bit 5'h10
4'h0
4'hF
4'h0
5'h10
Now 800 ns
Cursor 1 0 ns
790 ns 792 ns 794 ns 796 ns 798 ns 800 ns
Library Project sim
Transcript
force -freeze sim:/add4/Data1 4'h0 0
force -freeze sim:/add4/Data2 4'hf 0
VSIM 10> run
VSIM 11>
789 ns to 801 ns
Project : Add4
Now: 800 ns Delta: 0
sim:/add4
ENG
17:22
17-09-2020

Summary – VHDL Evaluation

In this video, you have learned:

- How to test the code using an HDL simulator, ModelSim.
- How to evaluate the correctness of the design using waveform analysis.
- How to control the Inputs and see the resulting Outputs, using different number combinations to get good test coverage.

and how to control the inputs