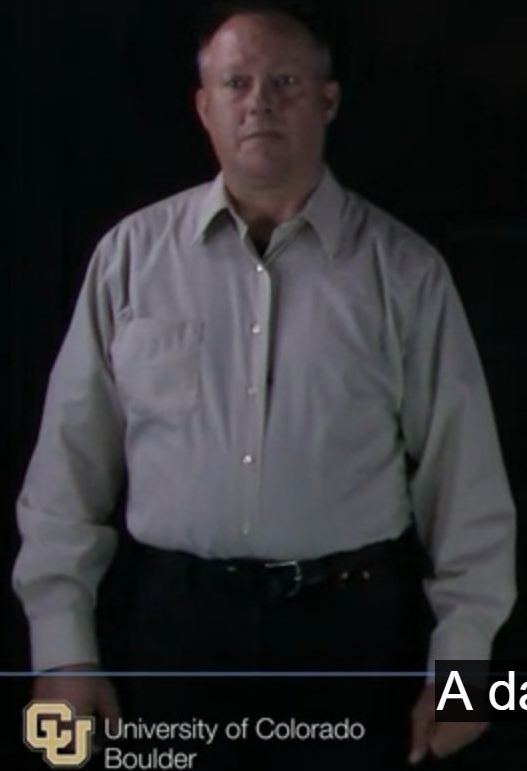
FPGA Design for Embedded Systems

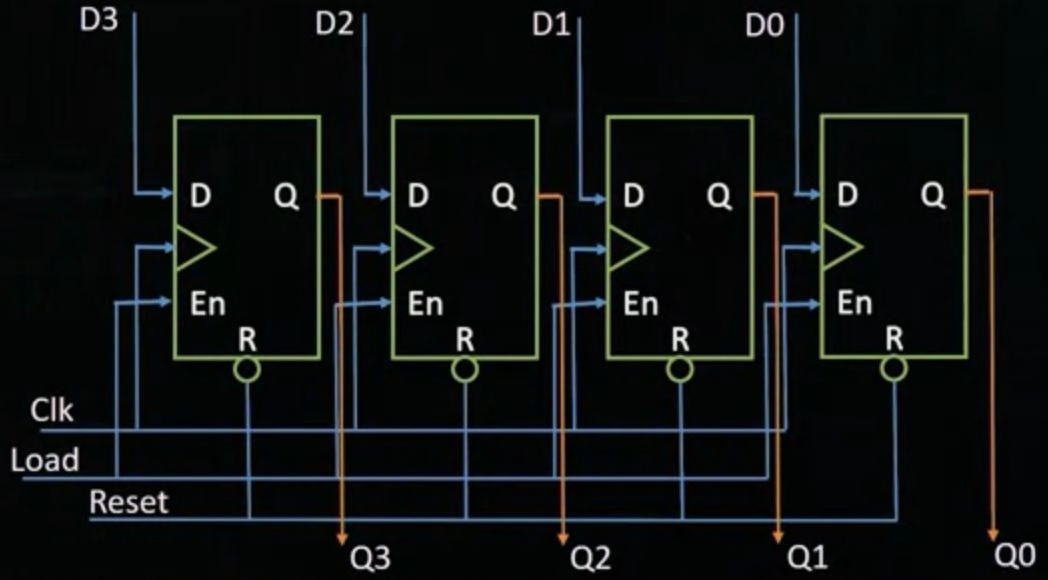
Hardware Description Languages for Logic Design

University of Colorado Boulder

Synchronous Logic : Data Register

Data is stored in a register for later use.
How can we make a data register in VHDL?

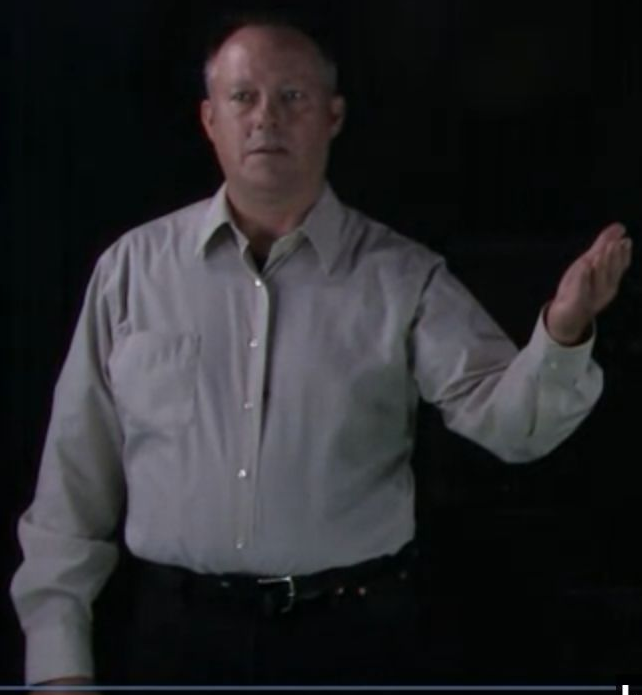




A data register stores data for later use.

University of Colorado Boulder

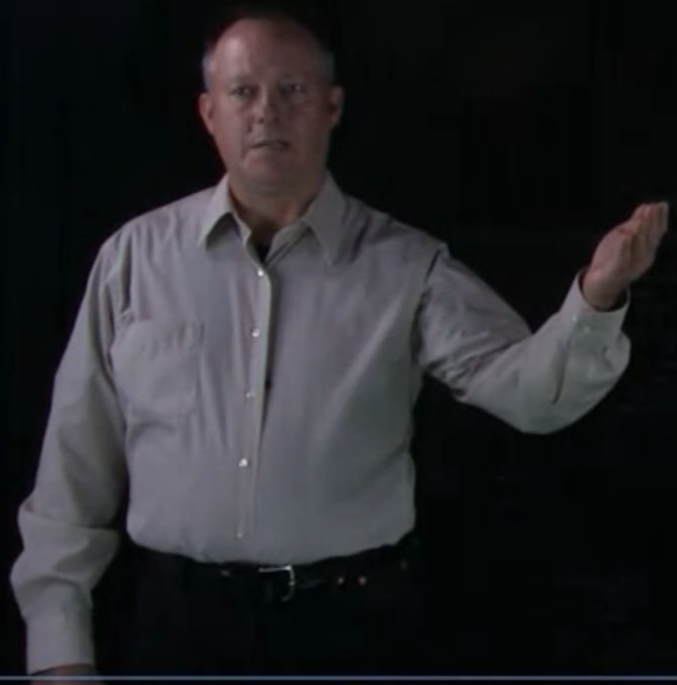
Synchronous Logic : Data Register



```vhdl
-- Entity
entity Data_Reg is port (
  clk, reset, load : in  std_logic;
  d                : in  std_logic_vector(3 downto 0);
  q                : out std_logic_vector(3 downto 0));
end entity Data_Reg;

-- Architecture Next Slide
```

In the case of the VHDL,
we have our entity, clock, reset and

Synchronous Logic : Data Register



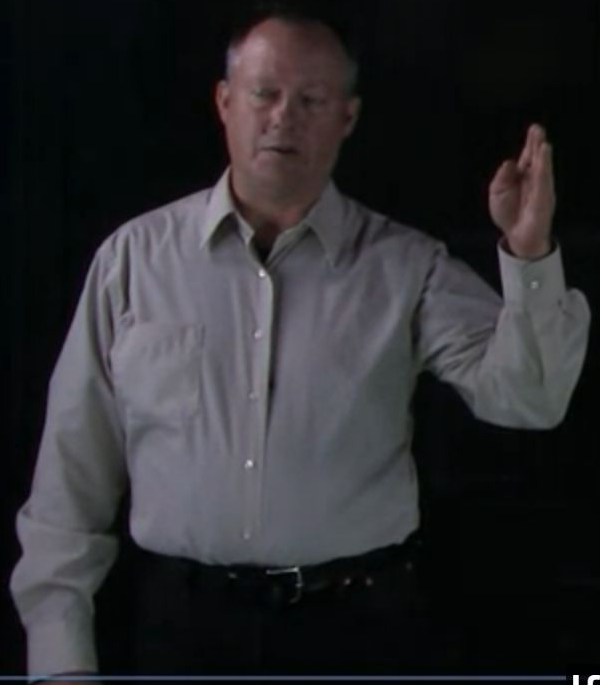
```
-- Architecture
architecture Reg_Arch of Data_Reg is
  begin dreg_proc : process (clk, reset, load)
    begin
      if    (reset='0')          then  q <= "0000";
      elsif (rising_edge(clk)) then
        if (load='1')            then  q <= d;
        end if;
      end if;
  end process dreg_proc;
end architecture Reg_Arch;a
```

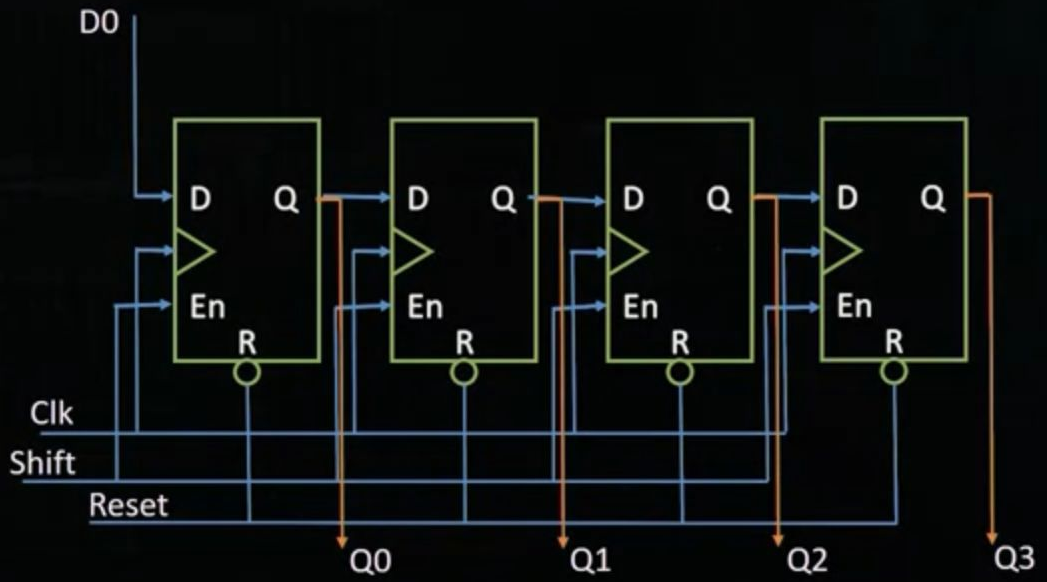
In our if statement,
we have reset equals 0,

Synchronous Logic : Shift Register

How can we make a shift register in VHDL?

D0 -> D1 -> D2 -> D3

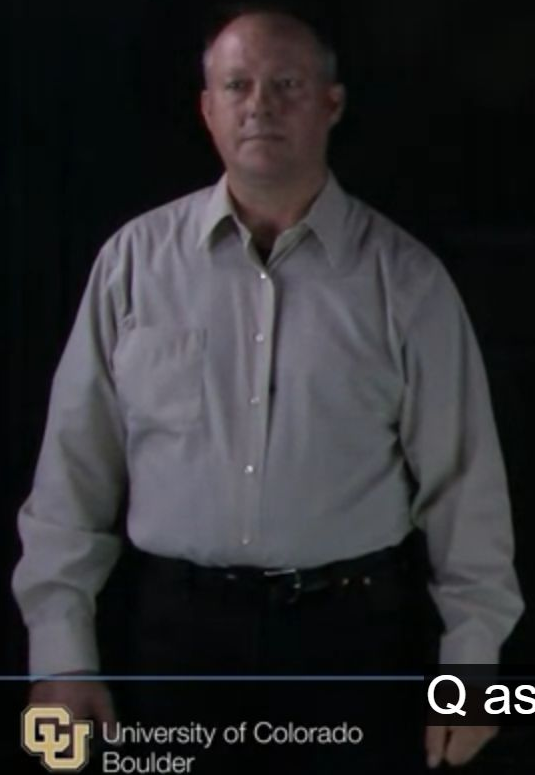




If we wanted to create a shift register from D0 input to D1

Synchronous Logic : Shift Register

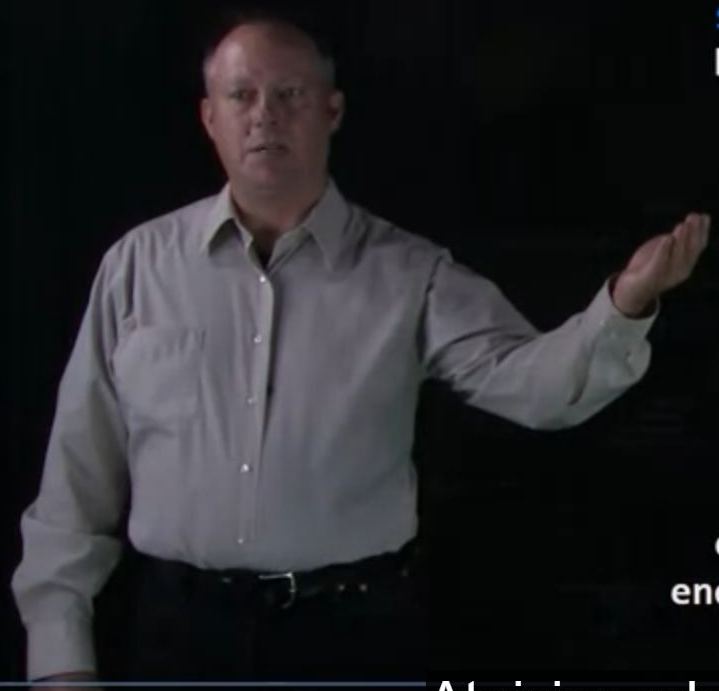
```
-- Entity
entity Shift_Reg is port (
  clk, reset, shift, d0 : in  std_logic;
  q     : out std_logic_vector(3 downto 0));
end entity Shift_Reg;
-- Architecture Next Slide
```



Q as our bus output, three down to zero.

University of Colorado Boulder

Synchronous Logic : Shift Register

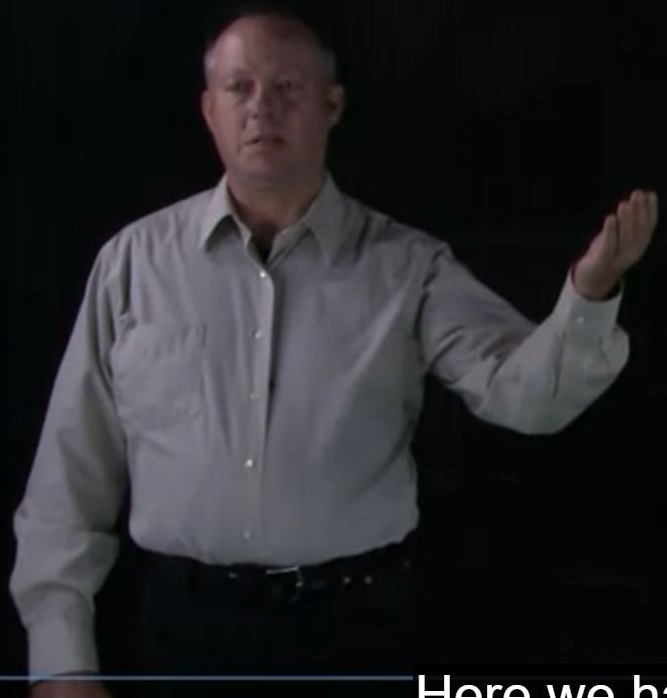


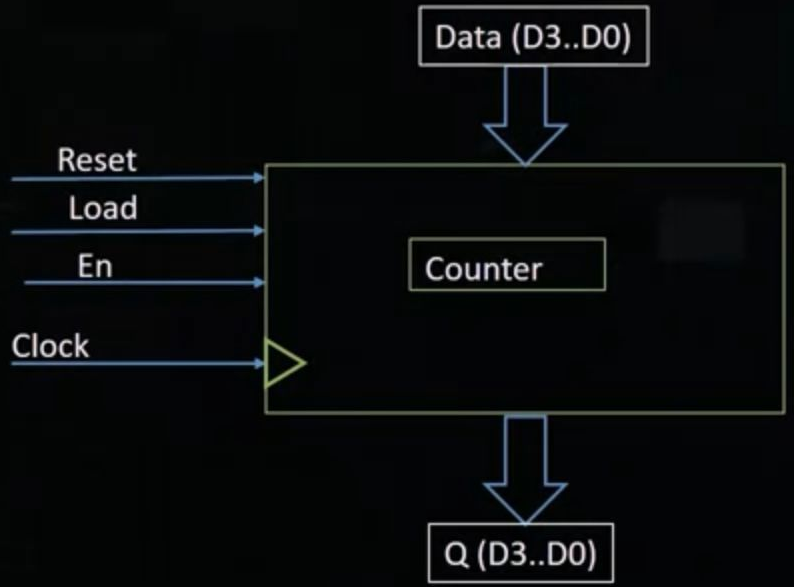
```
-- Architecture, could SLL or shift_left(q,1)
architecture SREG_Arch of Data_Reg is begin
  sreg_proc : process (clk, reset)
  begin
      if    (reset='0')         then  q <= "0000";
      elsif (rising_edge(clk)) then
        if  (shift='1')         then
                  q(0) <= d0;
                  q(1) <= q(0);
                  q(2) <= q(1);
                  q(3) <= q(2);
        end if;
      end if;
  end process sreg_proc;
end architecture SREG_Arch;
```

At rising edge each time, the rising edge occurs, we will evaluate this circuit.

Synchronous Logic : Binary Counter

How can we make a binary counter in VHDL?

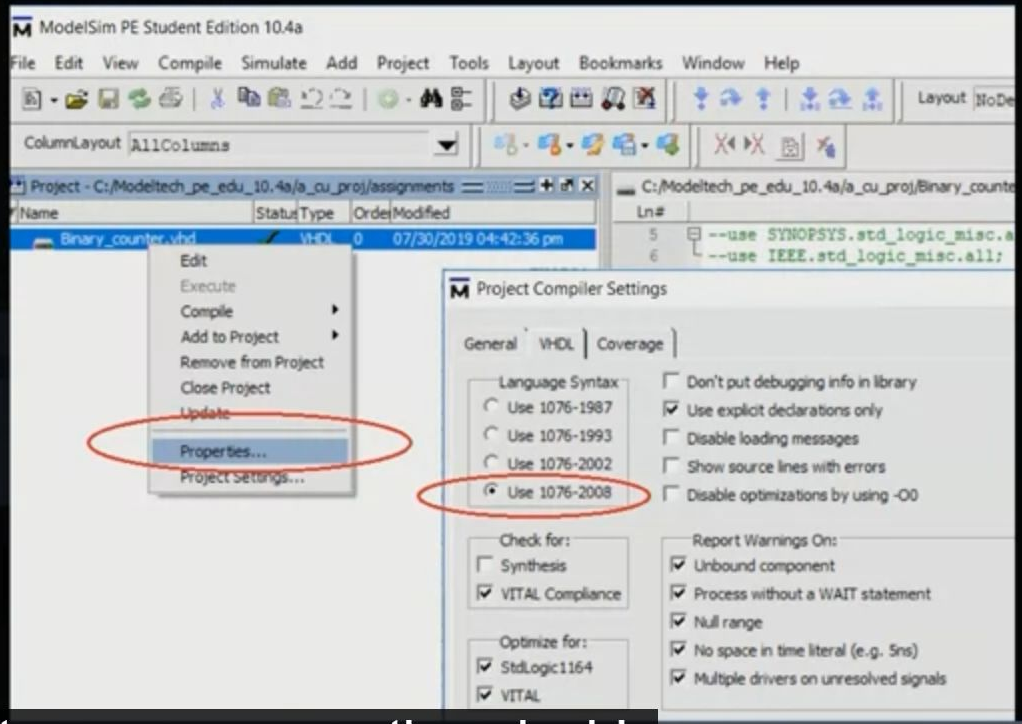




Here we have data input and data output,
reset, load, enable and clock.

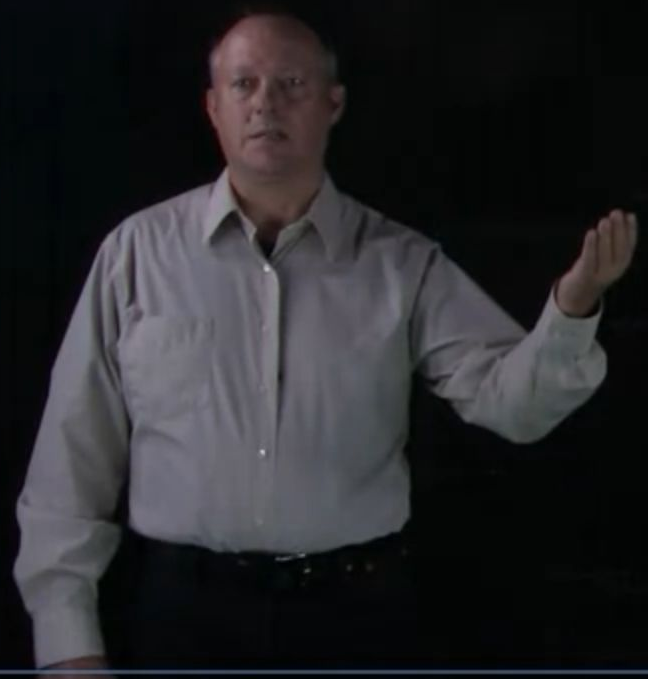
Synchronous Logic : Binary Counter

Compile : Use ModelSim Library - 2008



we need to turn on some options inside of the ModelSim Library for 2008.

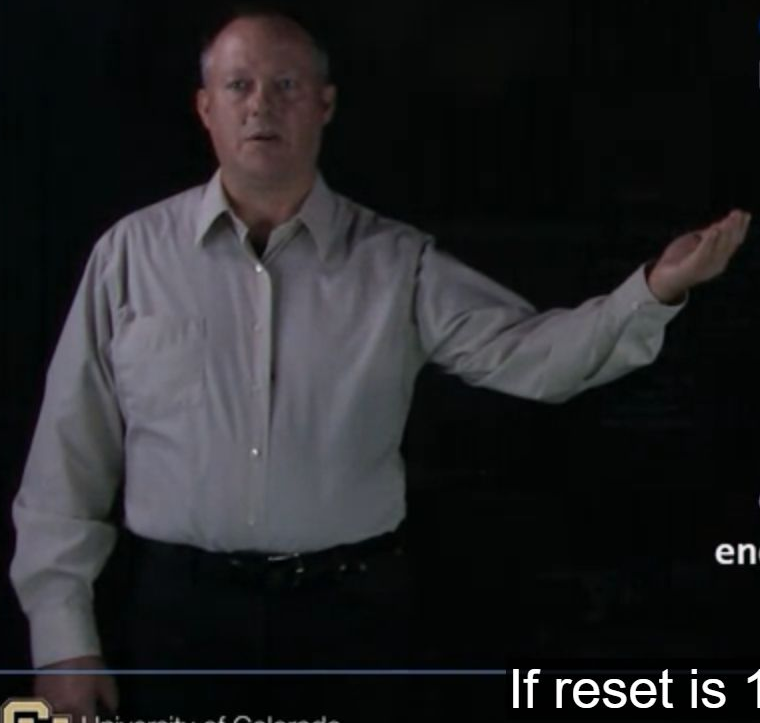
Synchronous Logic : Binary Counter



```
-- Entity
entity Counter is port (
  clk, reset, load, en : in  std_logic;
  d                    : in  std_logic_vector(3 downto 0);
  q                    : out std_logic_vector(3 downto 0));
end entity Counter;

-- Architecture Next Slide
```

D, data bus.

Synchronous Logic : Binary Counter

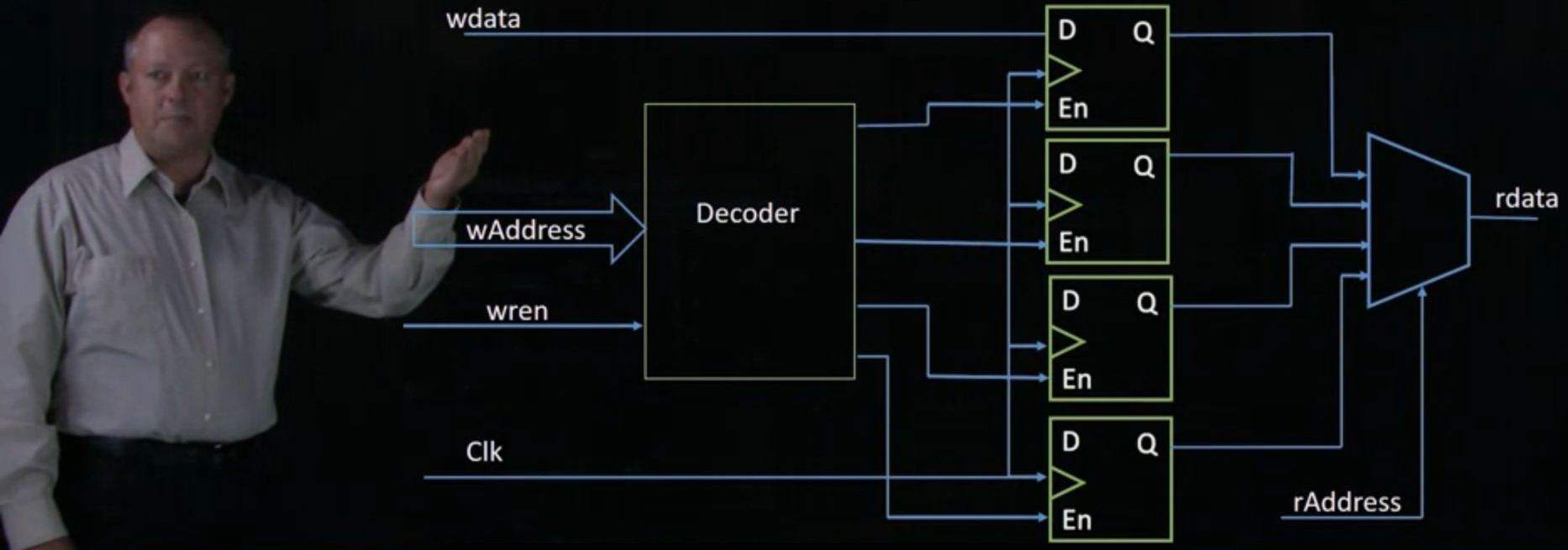


```
-- Architecture
architecture Counter_Arch of Counter is begin
  count_proc : process (clk, reset, load, en)
  begin
      if    (reset='1')       then  q  <= "0000";
      elsif (rising_edge(clk)) then
         if (load='1')        then  q  <= d;
         elsif (en='1')       then  q  <= q + 1;
                                    --2008
         end if;
      end if;
  end process count_proc;
end architecture Counter_Arch;
```

If reset is 1, then q output get 0.

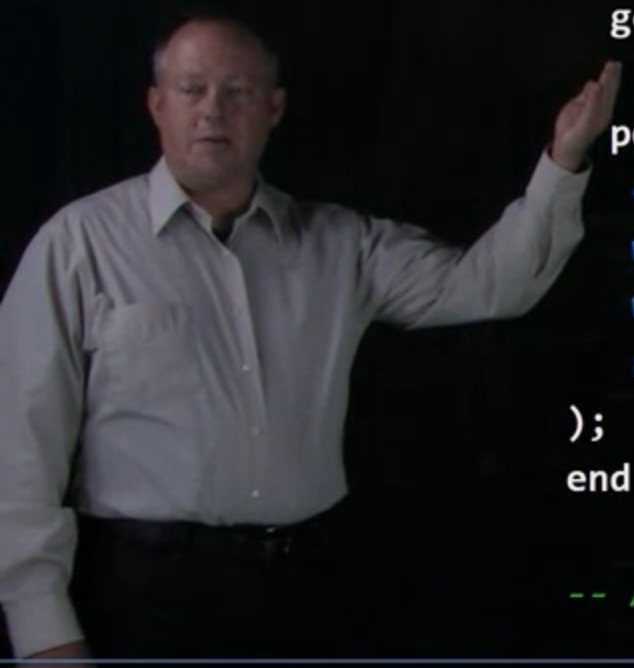
Synchronous Logic : Register File

Register Files are useful constructs that allow addressing of registers. Each flop is assumed as an n-bit register.



If we wanted to create a register file, here is a circuit diagram.

Synchronous Logic : Register File



```
-- Entity --  use IEEE.numeric_std.all; integer conversion
entity regFile is
  generic (Dwidth : integer := 8;
           Awidth : integer := 2 );
  port (
   clk, wren     : in  std_logic;
   wdata         : in  std_logic_vector(Dwidth-1 downto 0);
   waddr, raddr  : in  std_logic_vector(Awidth-1 downto 0);
   rdata         : out std_logic_vector(Dwidth-1 downto 0)
);
end entity regFile;

-- Architecture Next Slide
```

And in this case,
here's a new item called a generic.

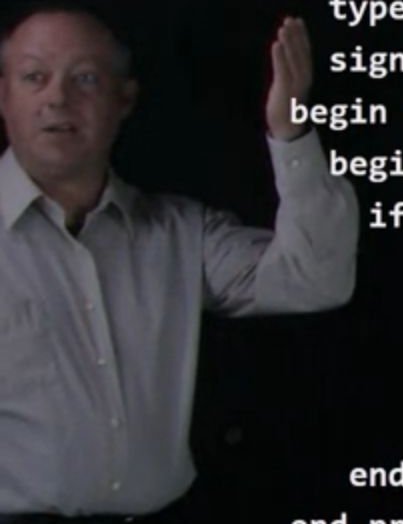
Generics are defined/declared in the Architecture and are used in the Entity.

- True.
- False.

Correct
Right, generics are defined/declared in the Entity and are then used in the Entity and Architecture.

Continue

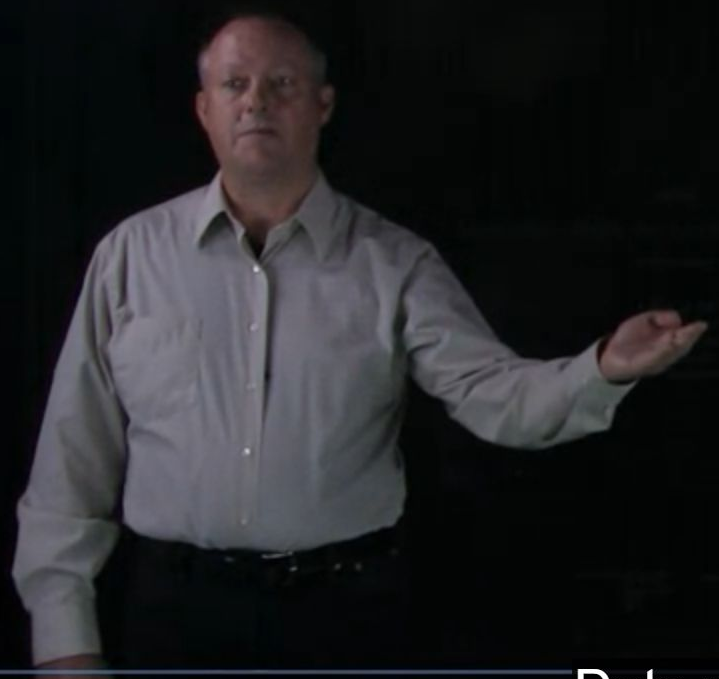
Synchronous Logic : Register File



```
-- Architecture
architecture RFile_Arch of regFile is
    type array_type (0 to 2**Awidth-1) of std_logic_vector (Dwidth-1 downto 0);
    signal array_reg : array_type;
  begin rf_proc : process (clk, wren, wdata, waddr, raddr)
    begin
       if    (rising_edge(clk))      then
          if (wren='1')              then
              array_reg(to_integer(unsigned(waddr)))  <= wdata;
          end if;
          rdata <= array_reg(to_integer(unsigned(raddr)));
       end if;
  end process rf_proc;
end architecture RFile_Arch;
```

In this case,
we're going to define a brand new type.

Summary – VHDL for Synchronous Circuits



In this video, you have learned:

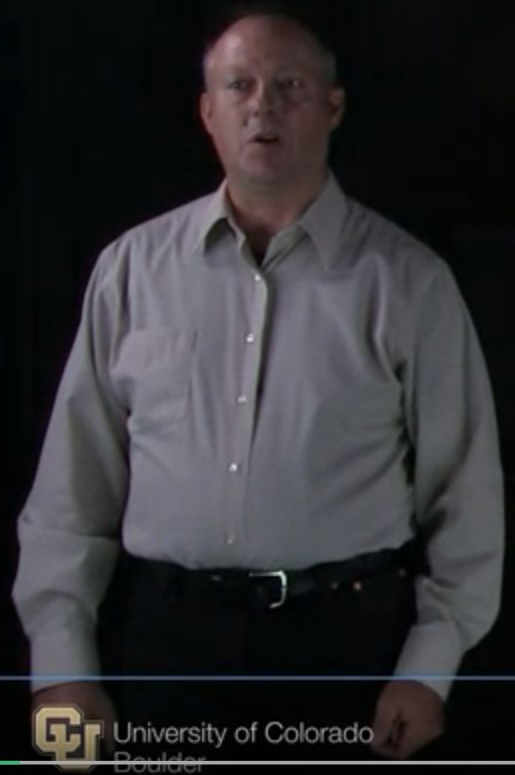
- How to describe synchronous circuits :
  - Data Register
  - Shift Register
  - Counter
  - Register File

Data registers, shift registers,
a counter and a register file.

FPGA Design for Embedded Systems

Hardware Description Languages for Logic Design

Bus Systems in VHDL



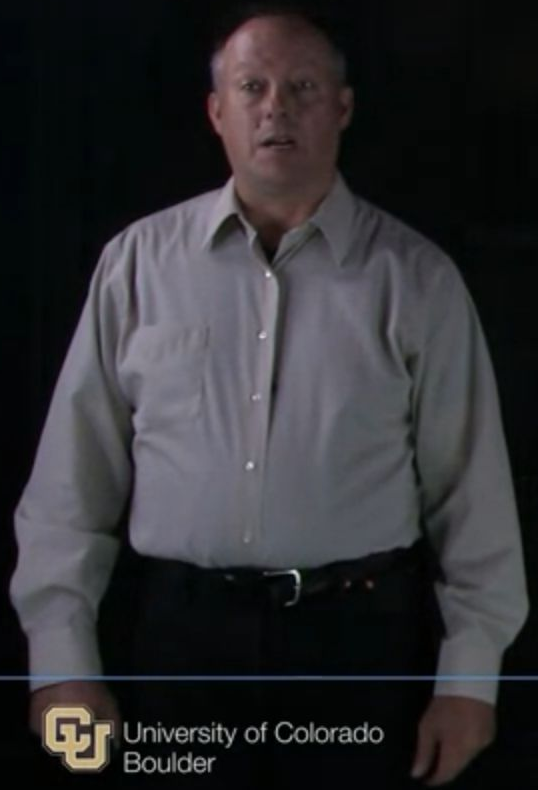
In this video you will learn :

- How to make tri-state and bi-directional bus systems
- How to join and separate buses

University of Colorado Boulder

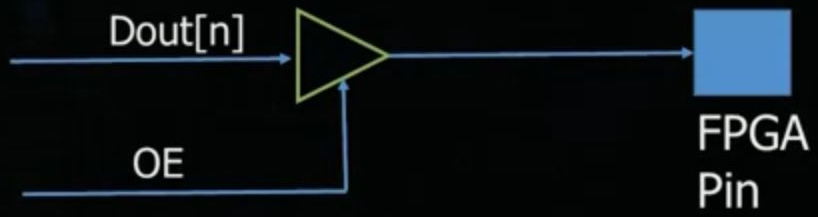
In this video, you will learn how to make

Tri-state Buses



External connections on FPGA pins are often in a group of related signals known as a bus.

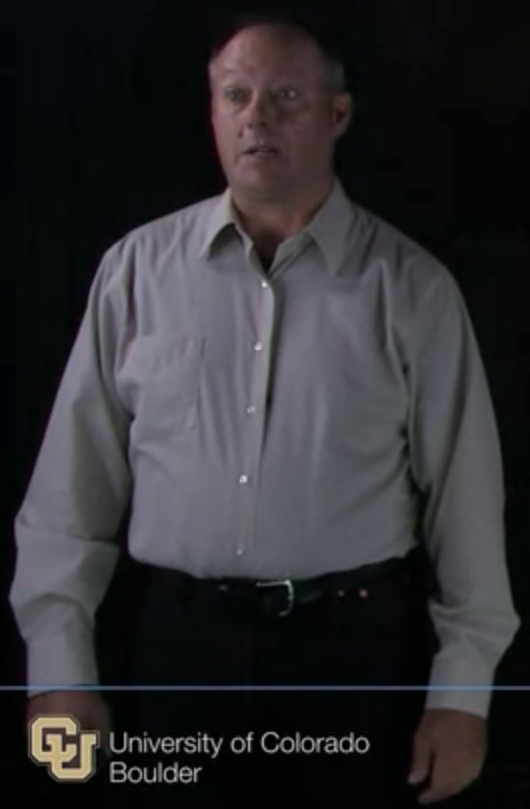
The I/O structure of FPGAs often allow the bus to be tri-stated, so that multiple drivers can be attached to the IO at the same time, with only one driver active.



University of Colorado Boulder

External connections on FPGA pins,

Tri-state in VHDL

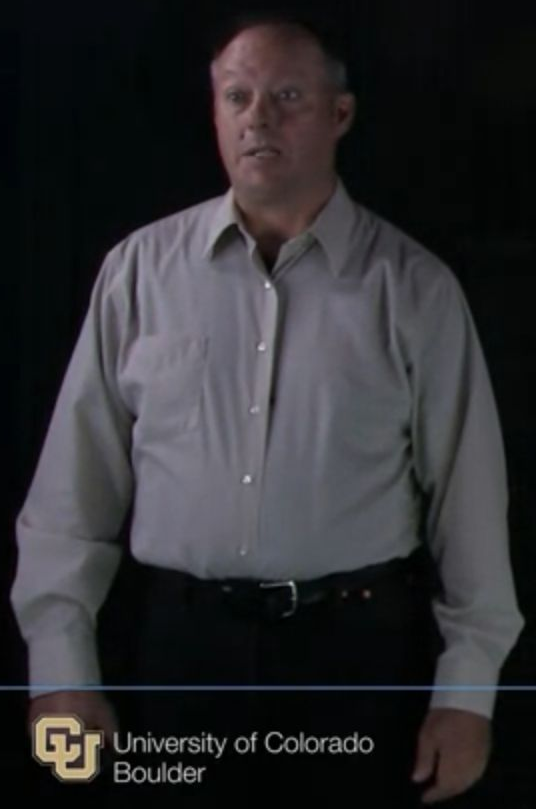


```vhdl
-- Entity
entity y_tri is port (
  OE        : in  std_logic;
  Dout      : in  std_logic_vector(3 downto 0);
  Pinout    : out std_logic_vector(3 downto 0) );
end entity y_tri;

-- Architecture
architecture tri_arch of y_tri is
begin
      Pinout <= Dout when (OE='1') else "ZZZZ";
end architecture tri_arch;
```

University of Colorado Boulder

How would we create a tri-state bus in VHDL?

Tri-state in VHDL

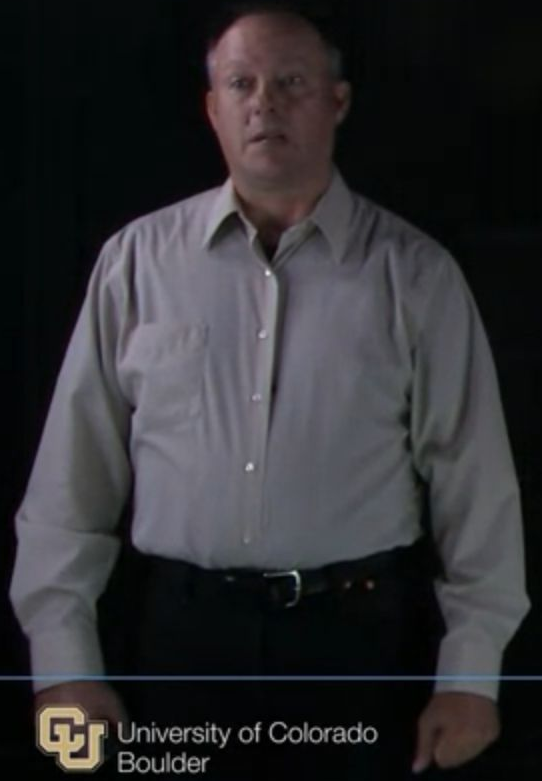


```vhdl
-- Entity
entity y_tri is port (
  OE         : in  std_logic;
  Dout       : in  std_logic_vector(3 downto 0);
  Pinout     : out std_logic_vector(3 downto 0) );
end entity y_tri;

-- Architecture
architecture tri_arch of y_tri is
begin
      Pinout <= Dout when (OE='1') else "ZZZZ";
end architecture tri_arch;
```

University of Colorado Boulder

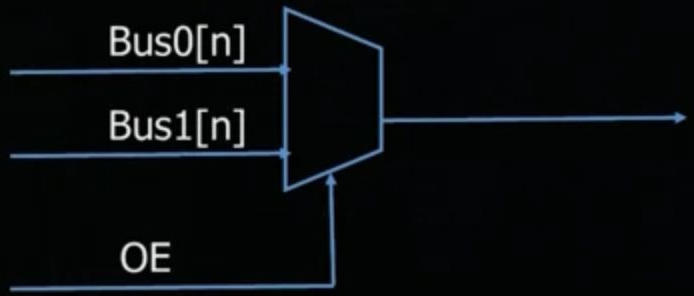
How would we create a tri-state bus in VHDL?

Tri-state Buses



When driving external tri-stated buses, some protocol should be used to assure that only one drive is active on the bus at a time.

VHDL code can define tri-state buses, but tri-state buses are typically not implemented inside of an FPGA but rather as a mux/multiplexer.

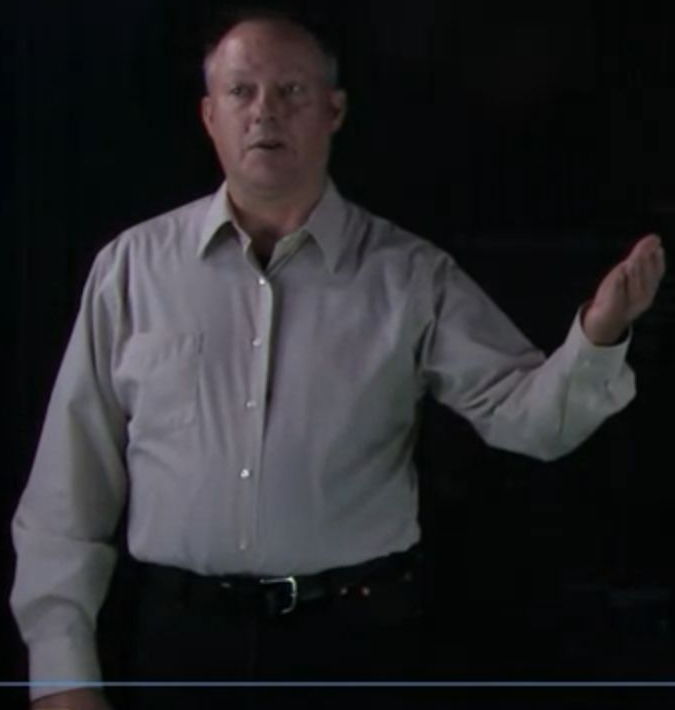


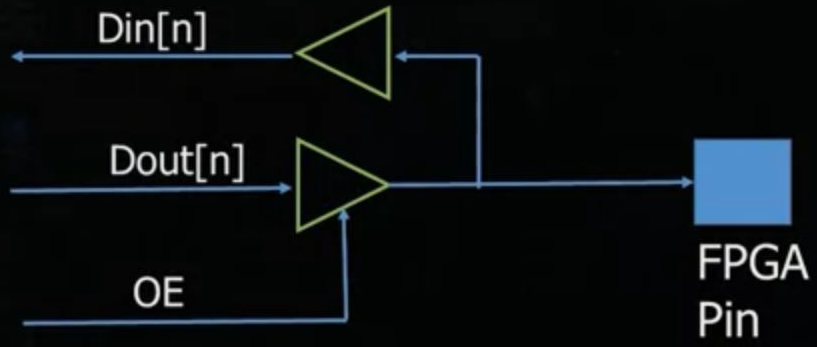
When driving external tri-stated buses,

University of Colorado Boulder

Bi-directional Buses

The I/O structure of the FPGA will also allow us to create Bi-directional buses, in which the external pin can be treated as either an input or an output, depending on the state of the enable signal.





we can create bidirectional buses.

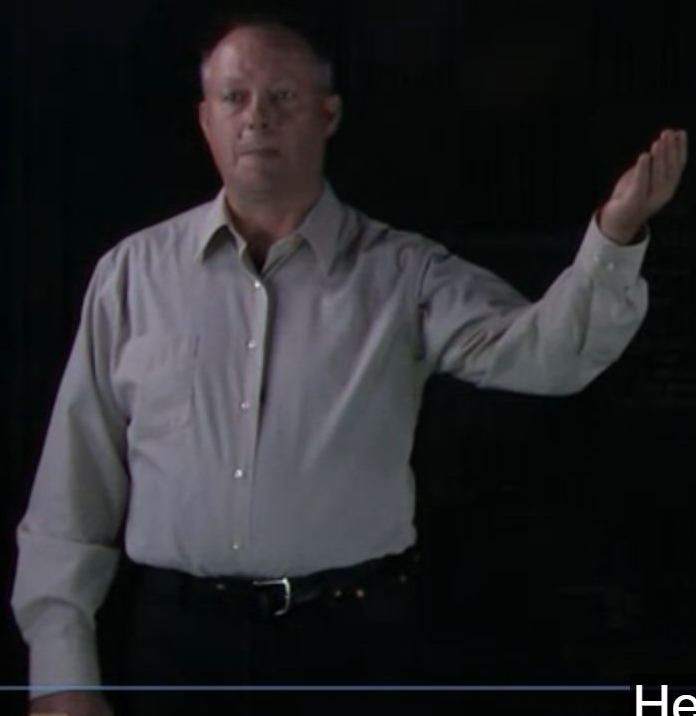
Bi-directional Buses in VHDL

```vhdl
-- Entity
entity bidir is port (
 OE    : in    std_logic;
 Dout  : in    std_logic_vector(3 downto 0);
 Din   : out   std_logic_vector(3 downto 0);
 IOpin : inout std_logic_vector(3 downto 0) );
end entity bidir;

-- Architecture Next Slide
```

we have our output enable,

University of Colorado Boulder

Bi-directional Buses in VHDL



```
-- Architecture
architecture bidir_arch of bidir is
  begin bi_proc : process (OE, Dout)
    begin
      Din <= IOpin;
      if    (OE='1') then IOpin <= Dout;
      elsif (OE='0') then IOpin <= "ZZZZ";
      else                IOpin <= "XXXX";
      end if;
    end process bi_proc;
end architecture bidir_arch;
```

Here's our architecture, data in gets IOpin.

University of Colorado Boulder

Tri stating the output of an IO pin allows for external devices to safely drive the shared pin as an input.

- False.
- True.

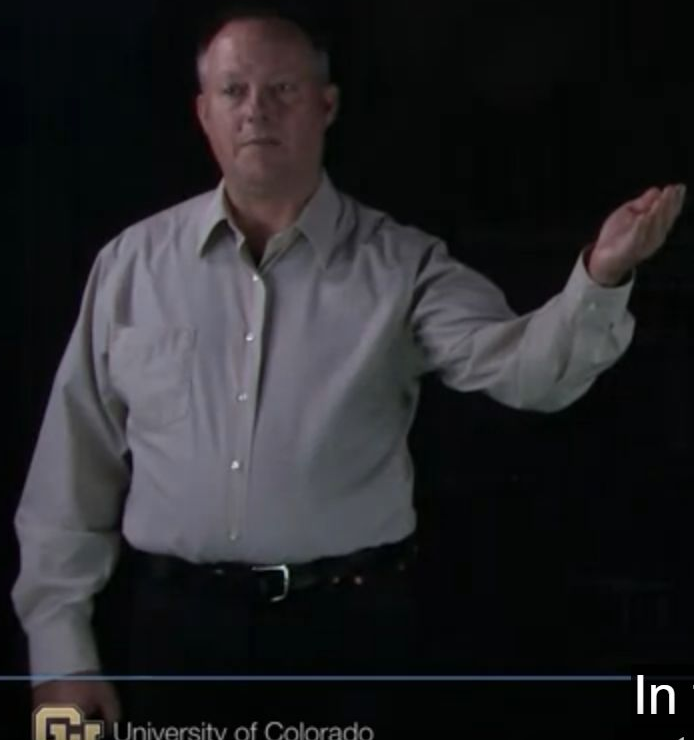
Correct

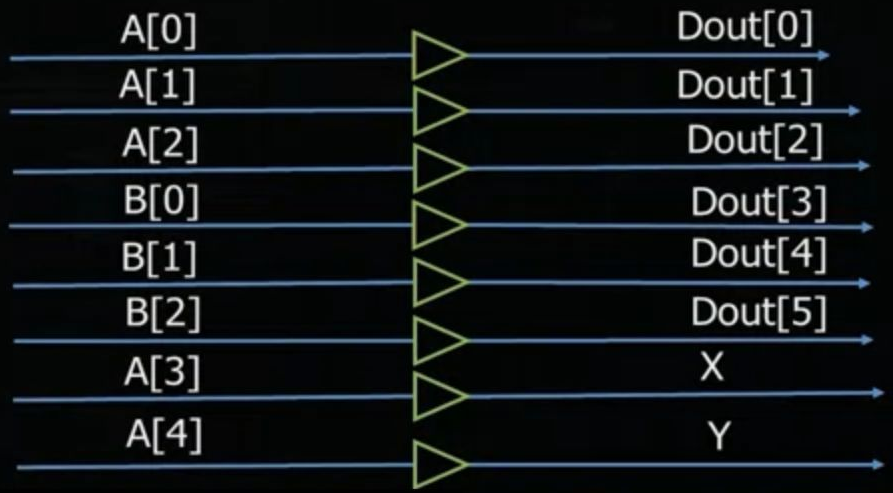
Correct. Driving an pin as output '0' or ground while and external device is driving a high voltage '1' value onto the IO pin could burn out the driver on either device. Setting the IO to high impedance or 'Z' will remove the voltage conflict.

Continue

Joining and Splitting Buses

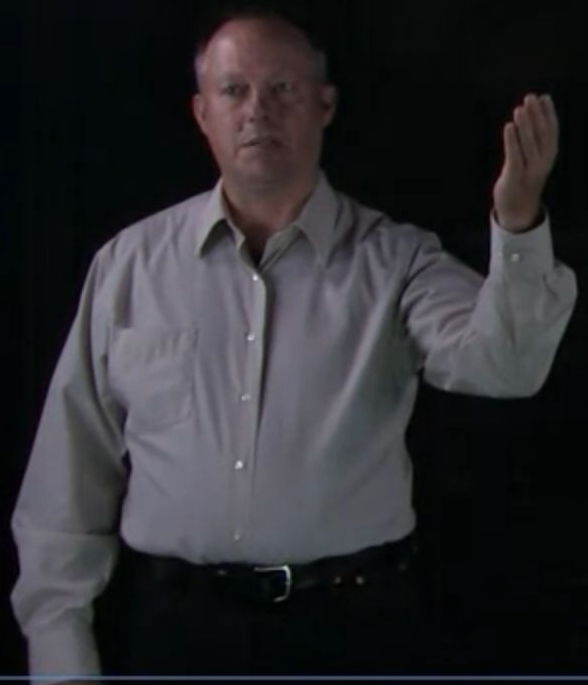
VHDL includes the concatenation operator which allows buses to be combined. Splitting buses can be done using indexing.





In this case, for the data out bus zero through five,

Joining Buses in VHDL

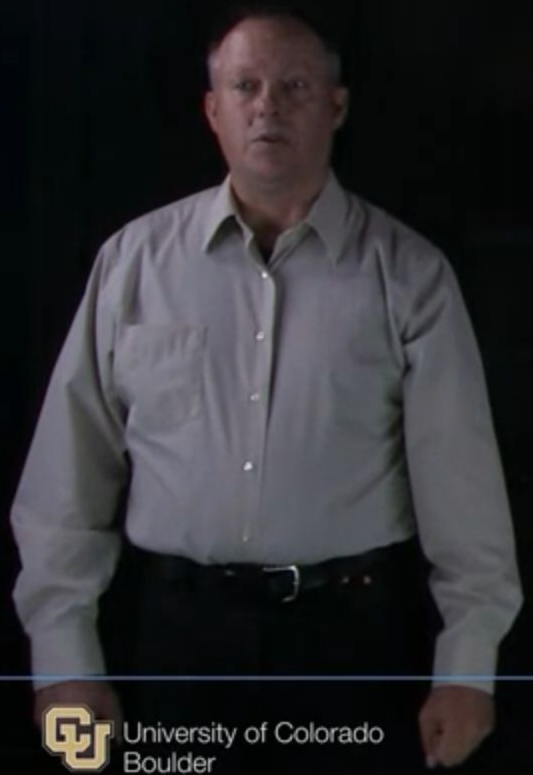


```
-- Entity
entity bus_js is port (
  A      : in    std_logic_vector(4 downto 0);
  B      : in    std_logic_vector(2 downto 0);
  X, Y   : out   std_logic;
  Dout   : out   std_logic_vector(5 downto 0) );
end entity bus_js;
-- Architecture
architecture js_arch of bus_js is
  begin js_proc : process (A,B)  begin
      Dout <= (B(2) & B(1) & B(0) &
                A(2) & A(1) & A(0));
      X <= A(3);
      Y <= A(4);
  end process js_proc;
end architecture js_arch;
```

B, standard logic inputs.

University of Colorado Boulder

Summary – VHDL for Bus Systems



In this video, you have learned:

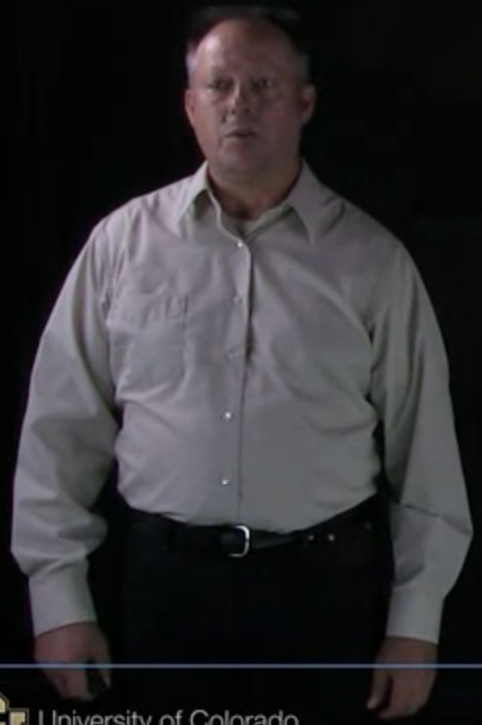
- How to make tri-state and bi-directional bus systems
- How to join and separate buses

University of Colorado Boulder

In this video, you've learned how to make


Share
FPGA Design for Embedded Systems
Hardware Description Languages for Logic Design
University of Colorado Boulder
Copyright © 2019 University of Colorado
0:00 / 3:08

Modular Design in VHDL



In this video, you will learn:

- How to build bigger designs using modular design techniques
- Use of loops in VHDL
- How to use Generate to make copies of circuits

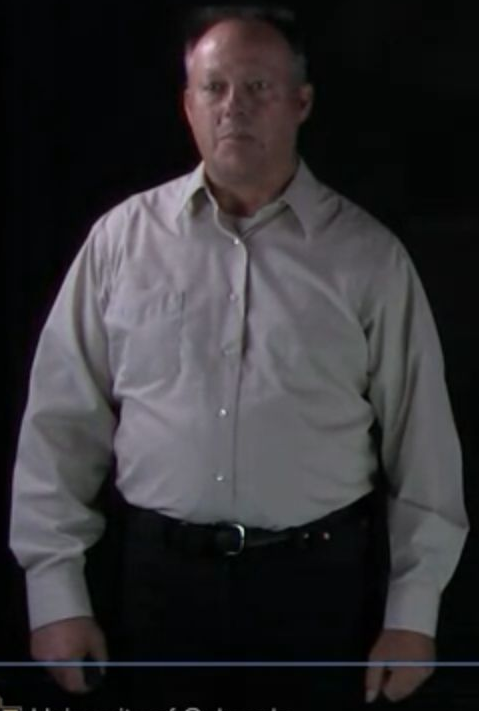
University of Colorado Boulder

In this video, you will learn how to build

Share

Press Esc to exit full screen

Modular Design in VHDL



Many tools are provided in VHDL to make modular designs, including :

- Component Instantiation
- Looping
- Generate Blocks
- Tasks and Functions

VHDL to make modular designs including,

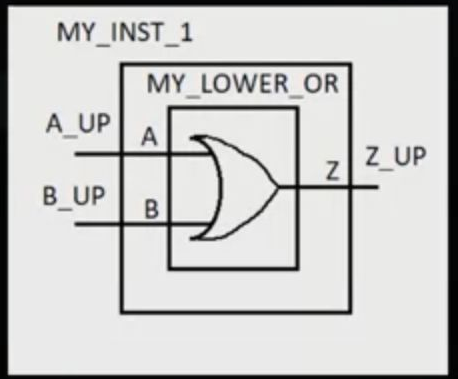
University of Colorado Boulder

0:18 / 3:08

Component Instantiation in VHDL

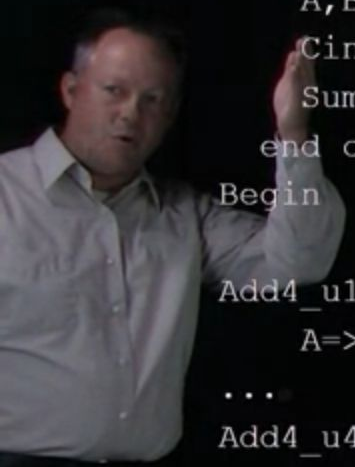
A modular design in VHDL at the architecture level oftentimes consists only of component instantiations, which is the fundamental way to build hierarchy in a design.

```
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);
MY_INST_N : ...

end architecture  MY_HIER;
```



MY_LOWER_OR block into the upper level.

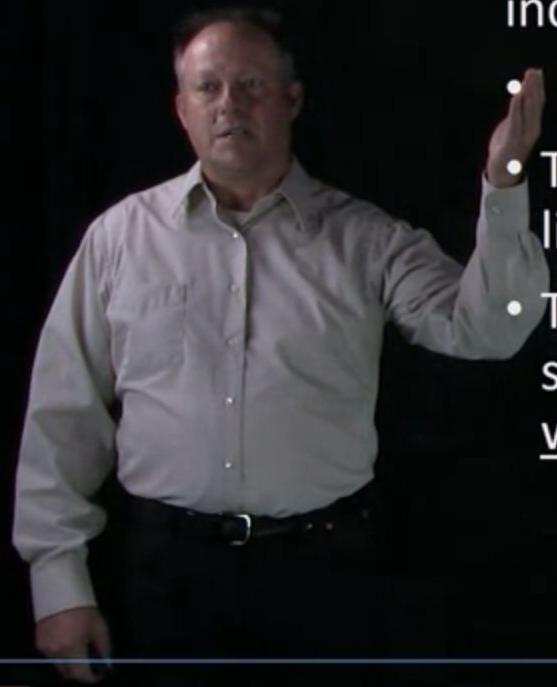
Component Instantiation in VHDL



```
architecture  Add16_Arch  of Add16 is
     component  Add4 port (
       A,B : in std_logic_vector(3 downto 0);
       Cin : in std_logic;  Cout : out std_logic);
       Sum : in std_logic_vector(3 downto 0) );
     end component;
Begin

Add4_u1 : Add4 port_map (
    A=> A(3 downto 0),   B=>(3 downto 0),   Z=> Cin(0), Sum(3 downto 0) );
...
Add4_u4 : Add4 port_map (
    A=> A(15 downto 12), B=>(15 downto 12), Z=> Cin(3), Sum(15 downto 12) );

end architecture  Add16_Arch;
```

we have our component Add4,

Looping in VHDL

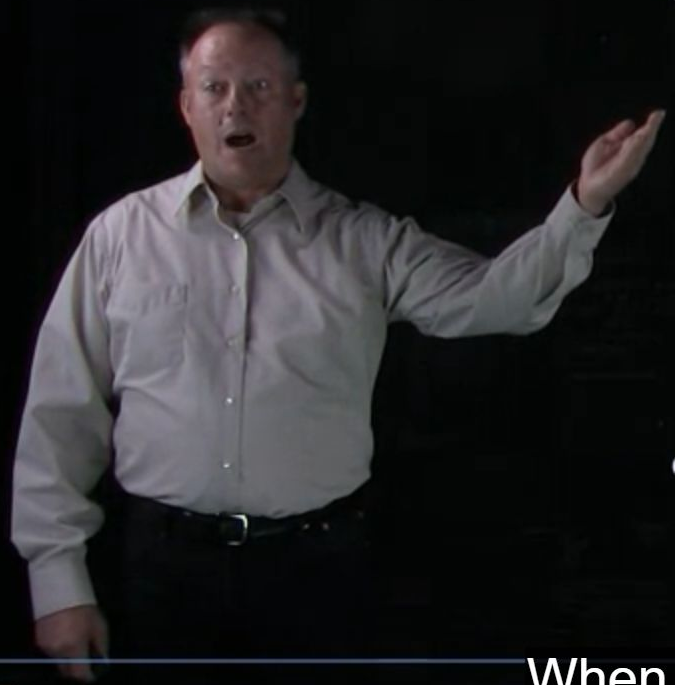


There are several looping constructs in VHDL, including :

- while, for
- The for loop statement is written and behaves just like it does in C, as is the while loop.
- The while loop executes a statement or block of statements until a condition like less than or a when compare is equal to exit or wait.

There are several looping constructs in

Looping in VHDL



```
-- Architecture
architecture Loop_Arch of my_loop is begin

    while (I <= 8) loop
      if (B = '1') then
        Z(I) <= A(I);
      end if;
      I := I + 1;
  end loop;

end architecture Loop_Arch;a
```

When I is less than or equal to eight, perform the loop.

In the while statement : while (I <= 8) loop, the variable I is assigned to the value 8?

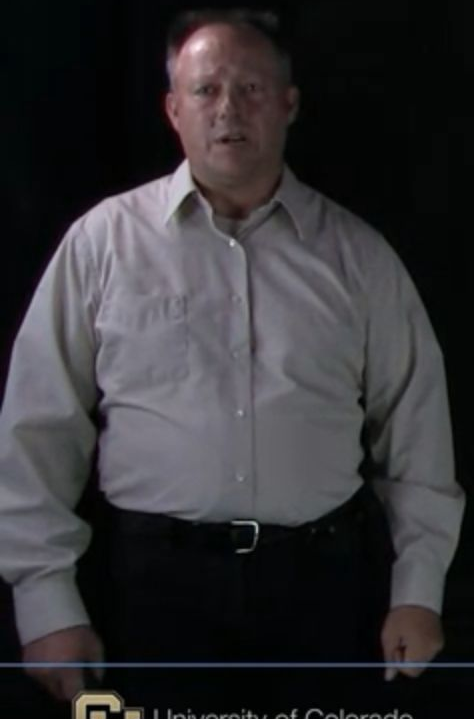
- (•) False.

Correct
Correct. This is a less than or equals comparison for the loop counter.

- ( ) True.

Continue

Generate in VHDL



Although loops can be used to generate data or test patterns, a common use of loops for synthesis is replication of many identical circuits within generate blocks.

The generate … end generate block specifies now an object is to be repeated.

The index variable of a for loop sets the number of elements to generate.

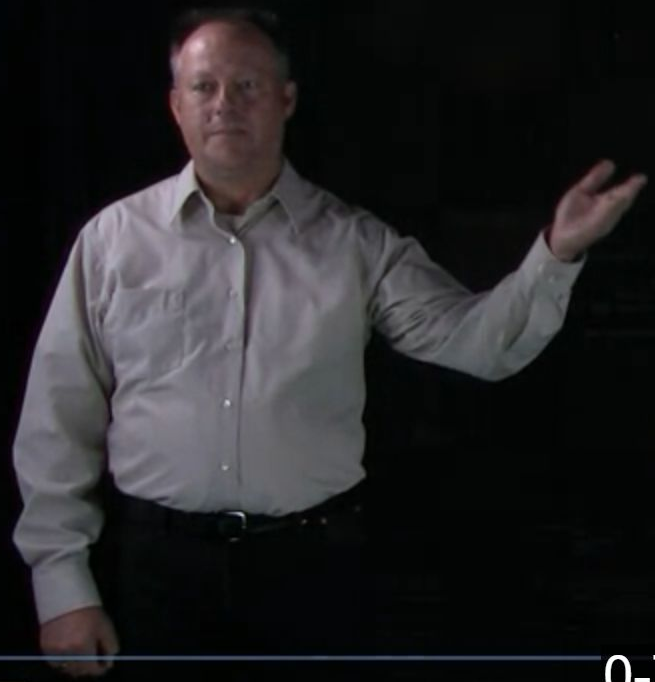
although loops can be used to

University of Colorado Boulder

Share

Press Esc to exit full screen

Generate in VHDL



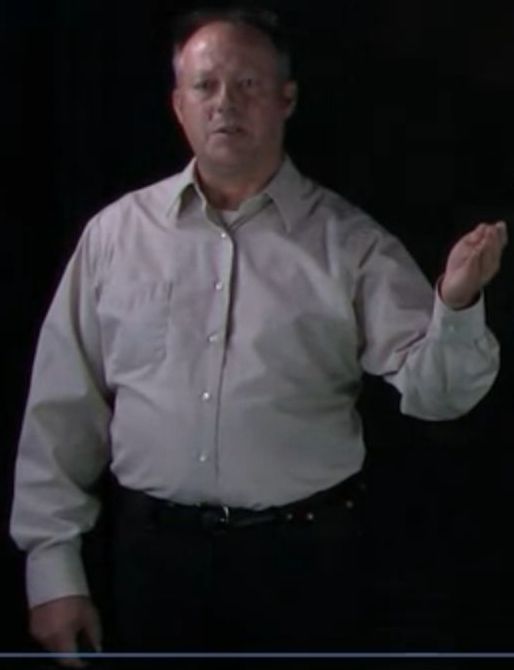
```
-- Architecture
architecture XorGen_Arch of XorGen is begin

  Gen_proc : for i in 0 to 7 generate
      Xout(i) <= Ain(i) xor  Bin(i);
  end generate Gen_proc;

end architecture XorGen_Arch;a
```

0-7 generate Xout of I,

University of Colorado Boulder

2:33 / 3:08

Summary - Modular Design in VHDL

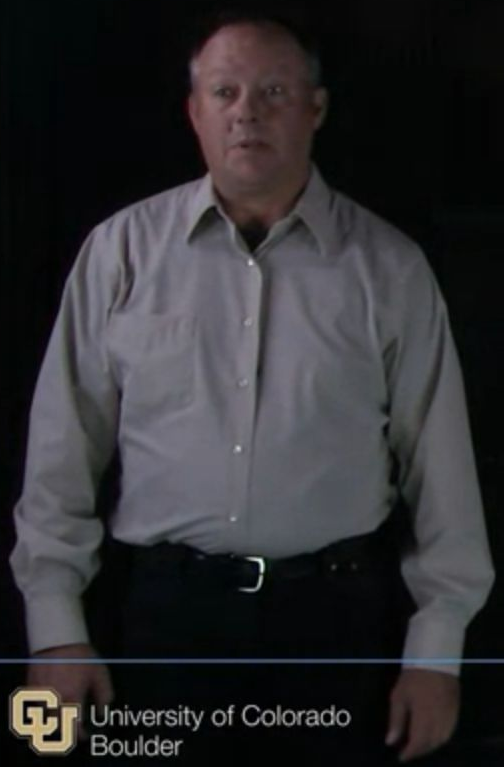


In this video, you have learned:

- How to build bigger designs using modular design techniques
- Use of loops in VHDL
- How to use for ... Generate to make copies of circuits

bigger designs using
modular design techniques,

Testbenches in VHDL



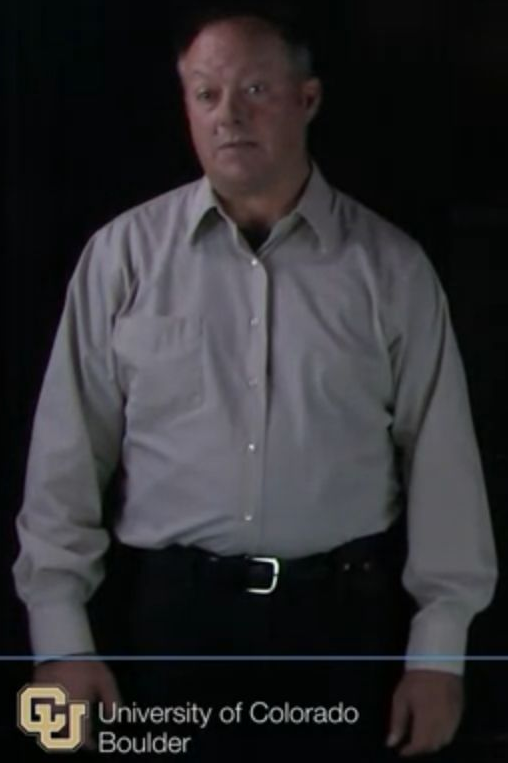
In this video you will learn :

- The concept of using a VHDL program, called a testbench, to test another VHDL program (your code)
- How to write simple testbenches
- How to use loops to generate stimulus
- How to use assertions to determine and report test results

University of Colorado Boulder

the concept of using
a VHDL program,

What is a Testbench?



A testbench is a program written in any language for the purposes of exercising and verifying the functional correctness of the hardware model as coded.

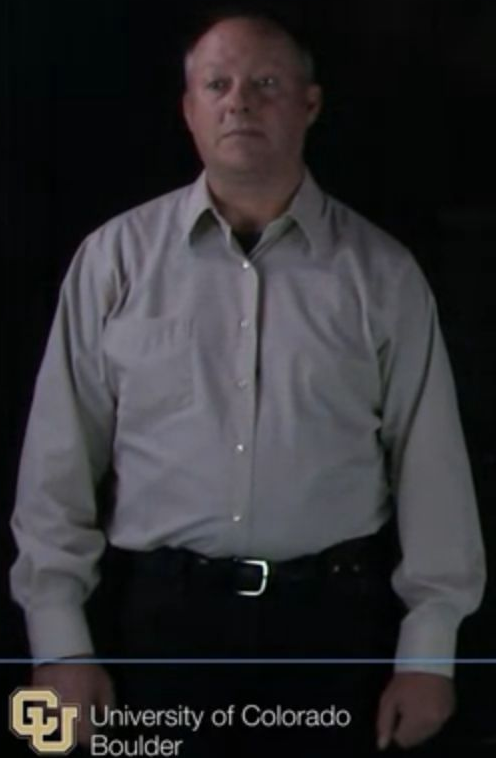
Also known as a test fixture or test harness.

A testbench is a powerful tool for generating test stimulus and test results

University of Colorado Boulder

This is also known as a test fixture or a test harness.

Testbench Functional Sections



A testbench can have several functional sections, including:

1. Top-level testbench declaration
2. Stimulus, Response, and Component Signal declarations
3. Component (Device Under Test) instantiations
4. Test Monitor which logs results and reports mis-compares

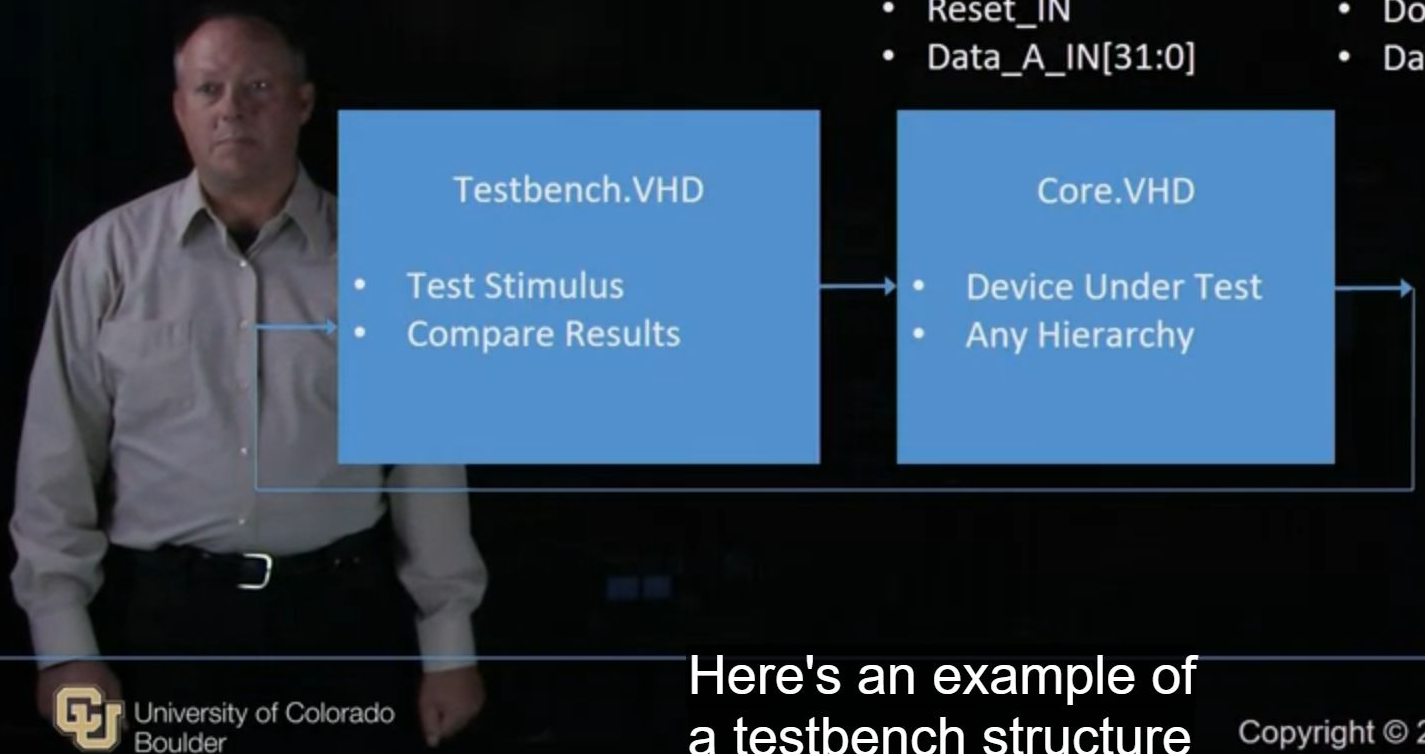
University of Colorado Boulder

a top-level testsbench declaration, stimulus,

Testbench Structure : Exercise the Core

INPUT :
- Clock_IN
- Reset_IN
- Data_A_IN[31:0]

OUTPUT :
- Div_CLK_OUT
- Done_OUT
- Data_B_OUT[31:0]

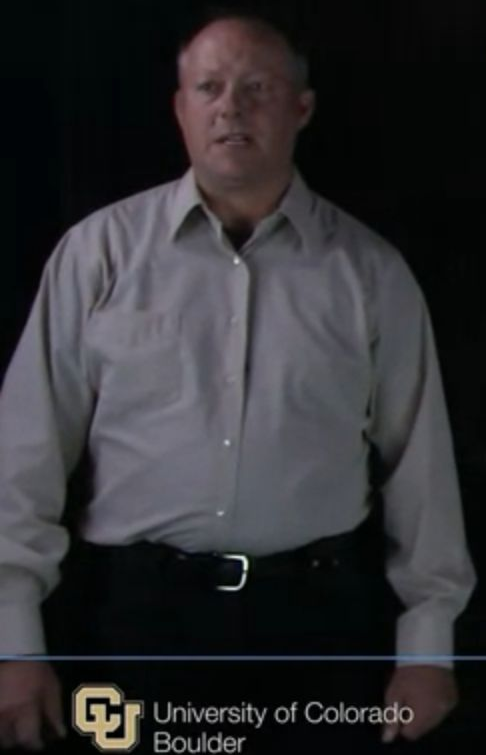


LOG FILE :
- Compare Expected
- Count Mis-Compares
- Pass/Fail

University of Colorado Boulder

Here's an example of a testbench structure

Adder Test Bench



```
-- Entity : no port list !
entity tb_adder is   end entity tb_adder;
-- Architecture
architecture test_arch of tb_adder is
  component Add4 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 component Add4;

  signal a_tb, b_tb : std_logic_vector(3 downto 0); -- INPUT
  signal Cin        : std_logic;                    -- INPUT
  signal Sum_tb     : std_logic_vector(3 downto 0); -- OUTPUT
  signal Cout_tb    : std_logic;                    -- OUTPUT
  signal expect     : std_logic_vector(3 downto 0); --expected
```

University of Colorado Boulder

Here's an example of an adder testbench.

The Entity of a testbench contains the port list.

- True.
- False.

Correct
Correct. There are no ports in a testbench.

Continue

Adder Test Bench

```
begin
  -- DUT Instantiation
  DUT : Add4 port map (
        Data1 => a_tb,   Data2 => b_tb,
        Cin   => Cin,    Cout  => Cout_tb, Sum => Sum_tb);
  -- Stimulus by hand drawn waves, poor coverage
  stim_proc : process begin
   wait for 0ns;
     a_tb   <= "0010"; b_tb <= "0010"; Cin <= '0'; expect <= "0100";
   wait for 10ns;
     a_tb   <= "1111"; b_tb <= "0000"; Cin <= '1'; expect <= "0000";
   wait for 10ns;
     a_tb   <= "0010"; b_tb <= "0100"; Cin <= '1'; expect <= "0111";
   wait;
  end process stim_proc;
```

we're going to
declare our instance,

Adder Test Bench

```
-- Monitor, use ieee.std_logic_textio.all;
--          use std.textio.all;
txt_out : process (Sum_tb, Cout_tb)
  variable str_o : line;
begin
  write(str_o, string'(" a="));      write(str_o, a_tb);
  write(str_o, string'(" b="));      write(str_o, b_tb);
  write(str_o, string'(" cin="));    write(str_o, Cin);
  write(str_o, string'(" sum="));    write(str_o, Sum_tb);
  write(str_o, string'(" cout="));   write(str_o, Cout_tb);
  write(str_o, string'(" expect=")); write(str_o, expect);
  assert false report time'image(now) & str_o.all
         severity note;
end process txt_out;

end architecture test_arch;
```

So this output will create

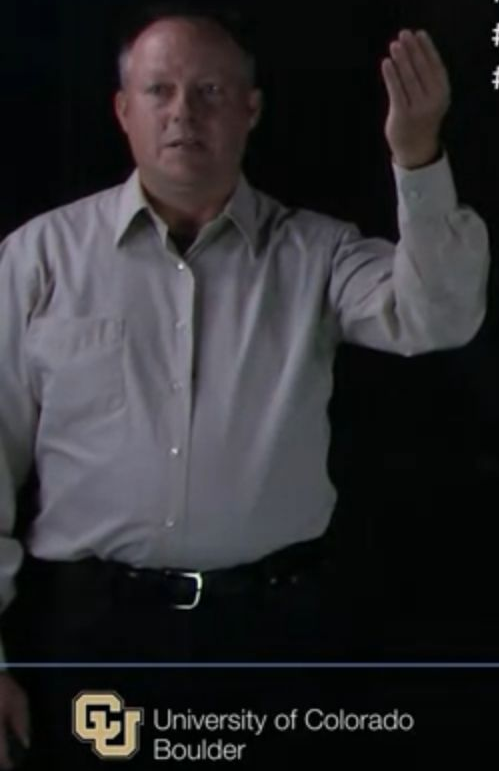
Adder Test Bench => Text Output ModelSim

```
VSIM 10 > run
# ** Note: 0 ns a=UUUU b=UUUU cin=U sum=UUUU cout=U expect=UUUU
#    Time: 0 ns  Iteration: 0  Instance: /tb_adder
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic
operand, the result will be 'X'(es).
#    Time: 0 ns  Iteration: 0  Instance: /tb_adder/DUT
# ** Warning: There is an 'U'|'X'|'W'|'Z'|'-' in an arithmetic
operand, the result will be 'X'(es).
#    Time: 0 ns  Iteration: 0  Instance: /tb_adder/DUT
# ** Note: 0 ns a=0010 b=0010 cin=0 sum=XXXX cout=X expect=0100
#    Time: 0 ns  Iteration: 2  Instance: /tb_adder
# ** Note: 0 ns a=0010 b=0010 cin=0 sum=0100 cout=0 expect=0100
#    Time: 0 ns  Iteration: 4  Instance: /tb_adder
# ** Note: 10 ns a=1111 b=0000 cin=1 sum=0000 cout=1 expect=0000
#    Time: 10 ns  Iteration: 3  Instance: /tb_adder
# ** Note: 20 ns a=0010 b=0100 cin=1 sum=0111 cout=0 expect=0111
#    Time: 20 ns  Iteration: 3  Instance: /tb_adder
write(str_o, string'(" cout=")); write(str_o, Cout_tb);
```

we can see quite a few
messages on the screen,

Adder Test Bench => Text Output ModelSim

```
  (ModelSim : Time messages removed)
# ** Note: 0 ns  a=0010 b=0010 cin=0 sum=XXXX cout=X expect=0100
# ** Note: 0 ns  a=0010 b=0010 cin=0 sum=0100 cout=0 expect=0100
# ** Note: 10 ns a=1111 b=0000 cin=1 sum=0000 cout=1 expect=0000
# ** Note: 20 ns a=0010 b=0100 cin=1 sum=0111 cout=0 expect=0111
```

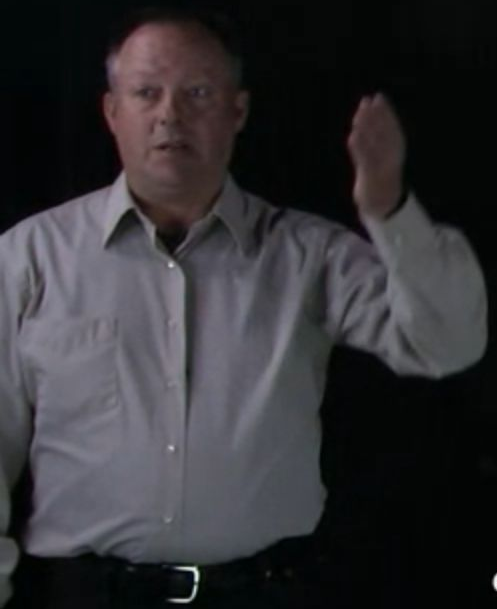


here I've cleaned up
some of those messages.

Adder Test Bench => Wave Output ModelSim

In a, 2, F,

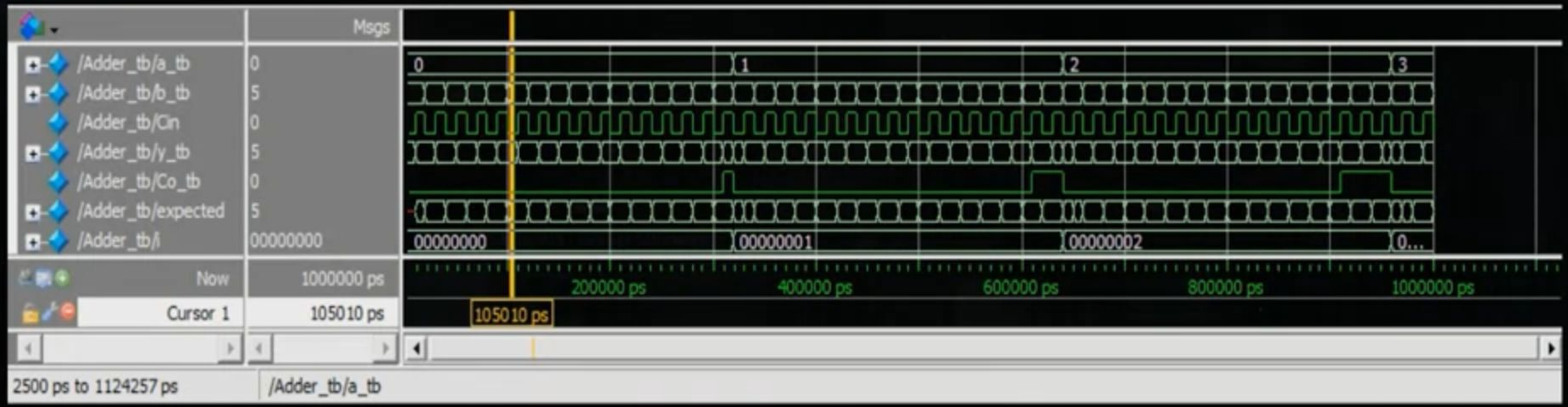
Adder Test Bench : Automation with Loops



```
-- Architecture : Generates coverage and expected stimulus
Loop_proc: process
    variable i, j, k : integer;
  begin
    for i in 0 to 15 loop  a_tb <= i;
      for j in 0 to 15 loop  b_tb <= j;
        for k in 0 to 1 loop
          Cin <= k;
          wait for 10ns;
          expect <= a_tb + b_tb + Cin;
        end loop;
      end loo;
    end loop;
  end process Loop_proc;
end architecture test_arch;
```

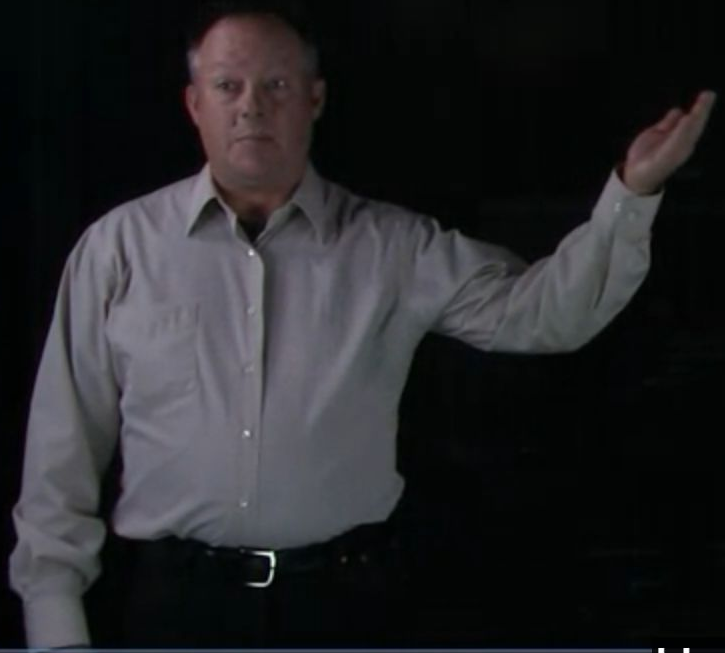
Next, we'll show some automation by creating

Adder Test Bench : Waves for Loops



We've got our a's toggling, b's toggling,

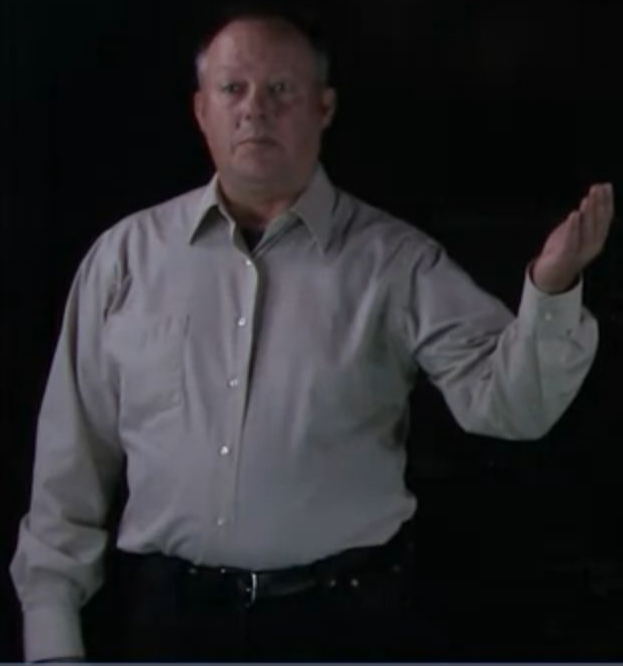
Adder Test Bench : Self Checking



```
expect <= a_tb + b_tb + Cin;


if (sum_tb /= a_tb + b_tb + Cin) then
    write(str_o, string'("Error - Sum"));
    writeline(output, str_o);
    wait;
end if;
```

Here's an example of a self-checking statement.

Summary – VHDL for Combinatorial Circuits



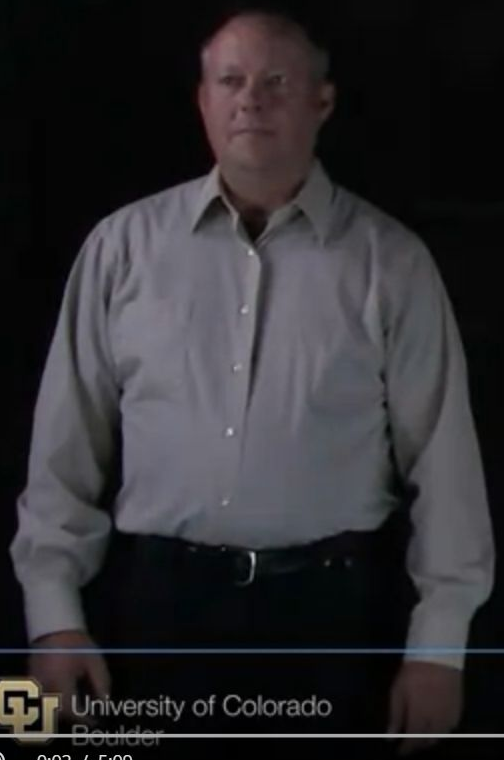
In this video, you have learned:

- How to use loops to generate stimulus
- How to use assertions to report test results

create loops to
generate stimulus,

University of Colorado Boulder

Testbenches in VHDL II



In this video you will learn :

- How to write a synchronous circuit testbench
- How to use external signal generators to create stimulus for circuits

University of Colorado Boulder