FPGA Design for Embedded Systems

Hardware Description Languages for Logic Design

University of Colorado Boulder

0:01 / 1:27

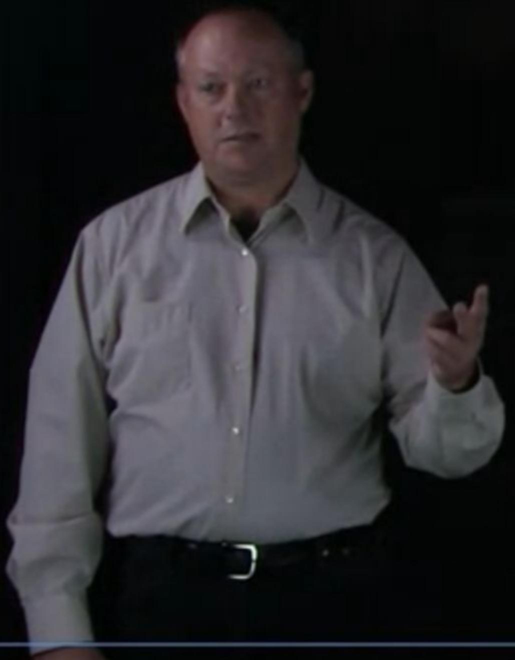
An Introduction to VHDL

VHDL is a powerful and efficient language for logic design.

Now that the basics have been introduced, in this Module we will further explore how to use VHDL to design circuits through many examples and techniques.

circuits through many
examples and techniques.

Learning to Speak VHDL



Like learning any other language...

- Learn key phrases, practice them

✓begun

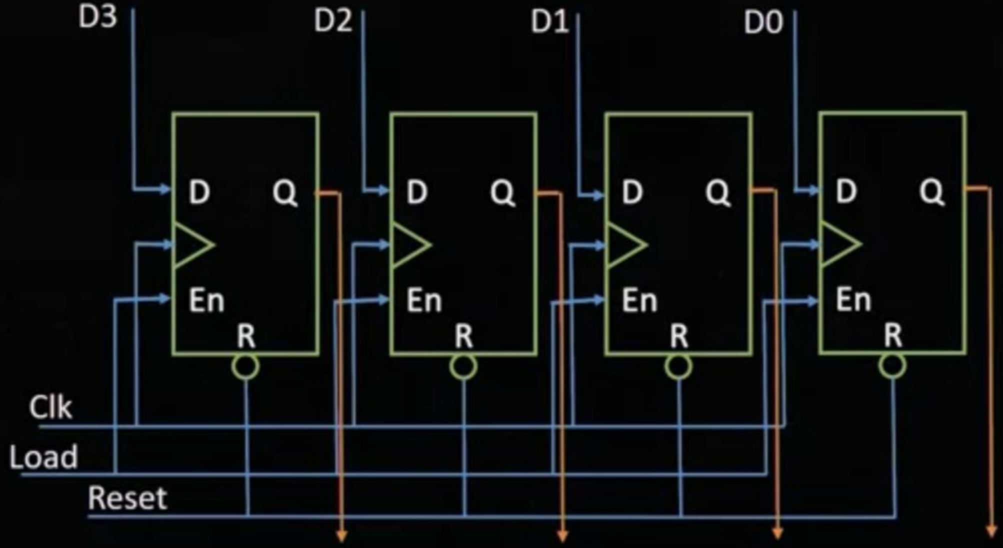
- Learn Grammar and Syntax

✓done

- Make compound sentences
- Practice, Practice, Practice

You learn grammar and
syntax, that you've done,

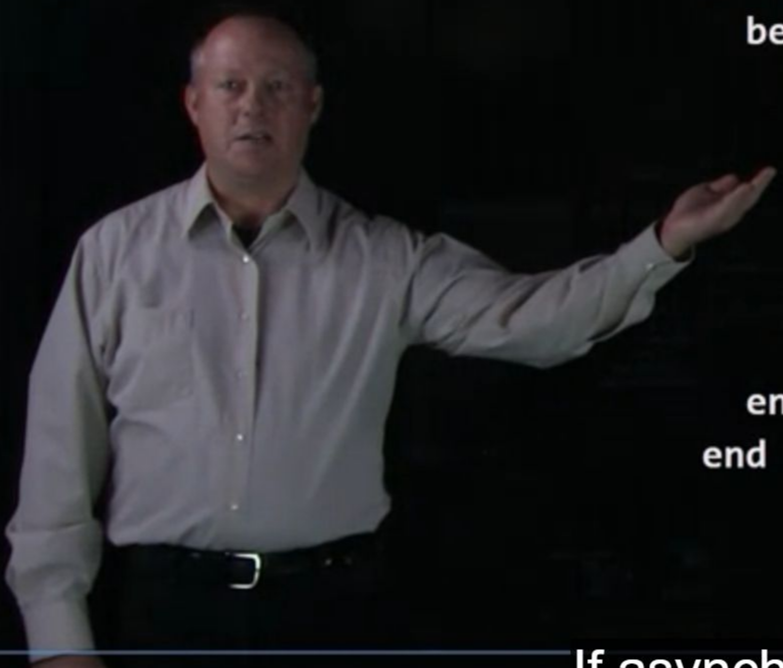
Writing VHDL : A 4-bit Register



write the VHDL to
create this circuit?

Share

Writing VHDL : A 4-bit Register



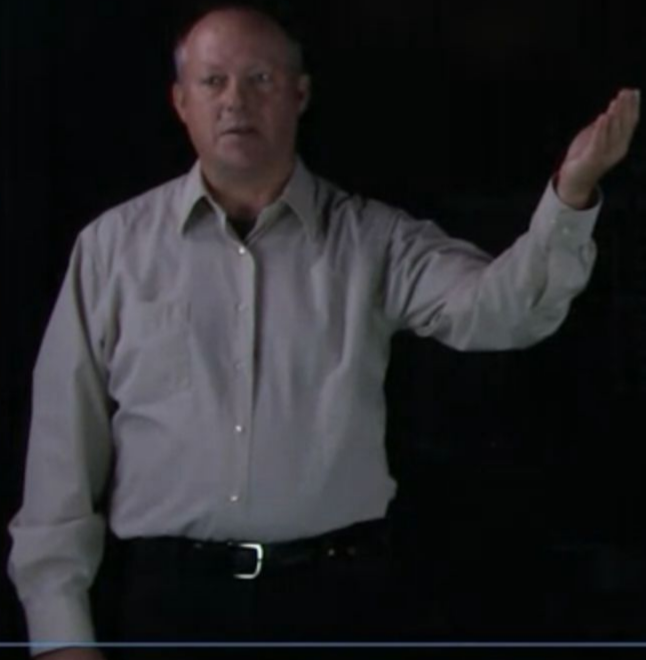
```
-- Architecture
architecture Reg_Arch of Data_Reg is
  begin dreg_proc : process (clk, reset, load)
    begin
      if    (reset='0')          then  q <= "0000";
      elsif (rising_edge(clk)) then
        if (load='1')            then  q <= d;
        end if;
      end if;
  end process dreg_proc;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

If asynchronous reset then
Q output gets four zeros,

University of Colorado Boulder

0:45 / 1:27

Videos in this VHDL Module



1. Learning to Speak VHDL (this video)
2. Combinatorial Circuits
3. Synchronous Logic: Latches and Flip Flops
4. Synchronous Logic: Counters and Registers
5. Interface: Buses and Tri-state Buffers
6. Modular Design in VHDL
7. Test Benches in VHDL - Combinatorial
8. Test Benches in VHDL – Synchronous
9. Memories in VHDL
10. Finite State Machines in VHDL

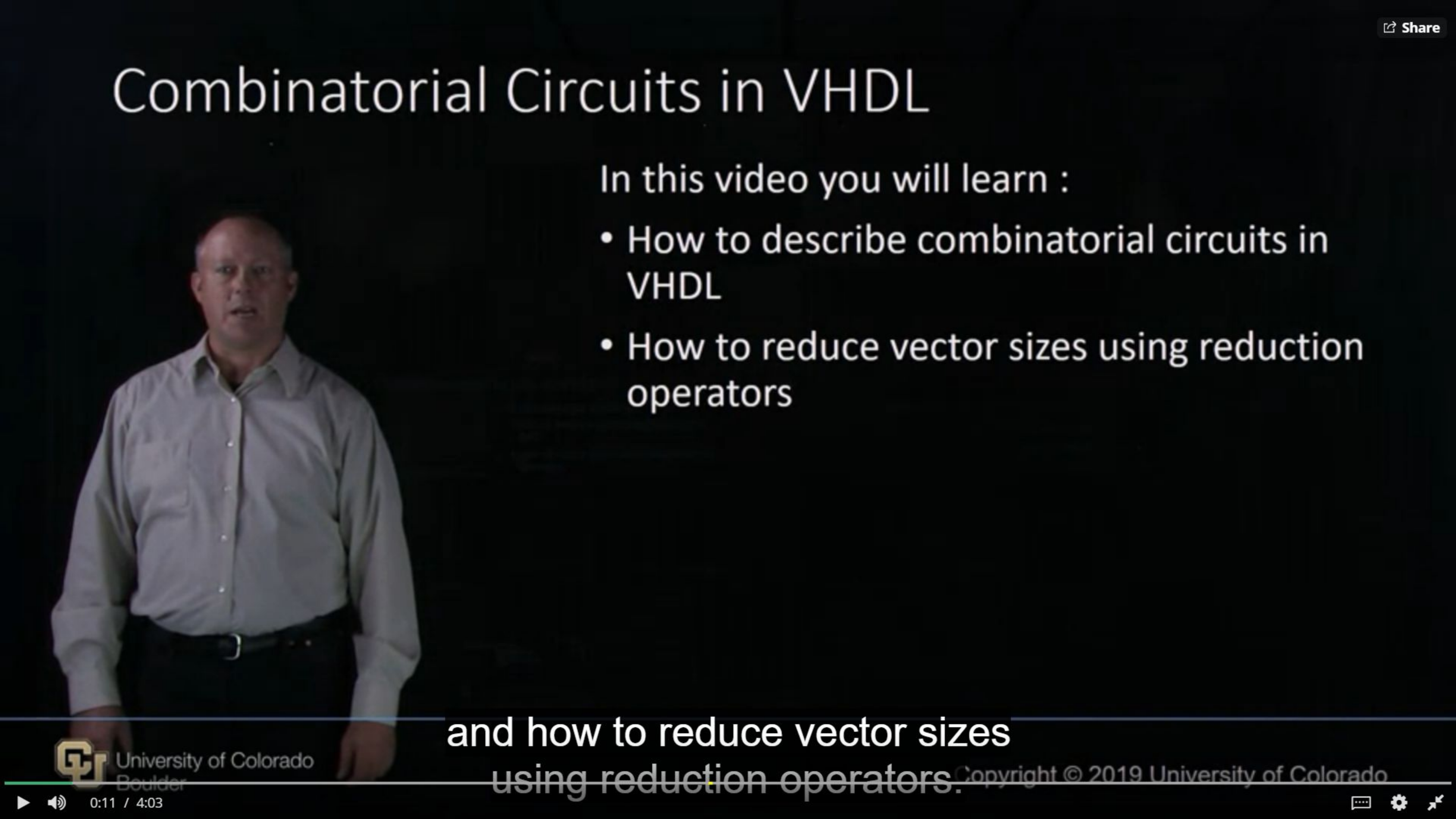
and flip flops,
counters and registers,

University of Colorado
Boulder

FPGA Design for Embedded Systems

Hardware Description Languages for Logic Design

Hello and welcome to FPGA design for embedded systems.



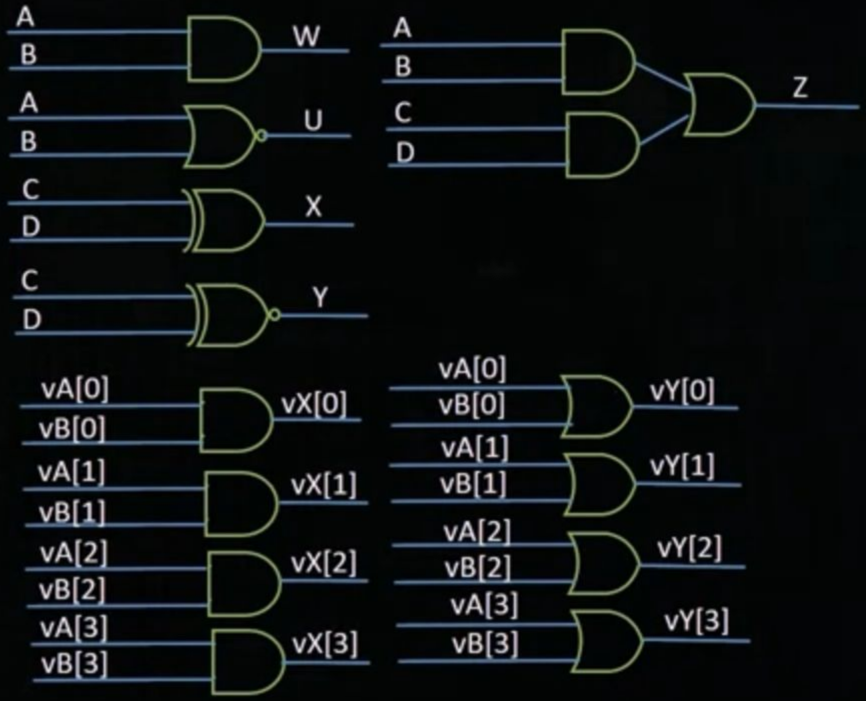
Combinatorial Circuits in VHDL

In this video you will learn :

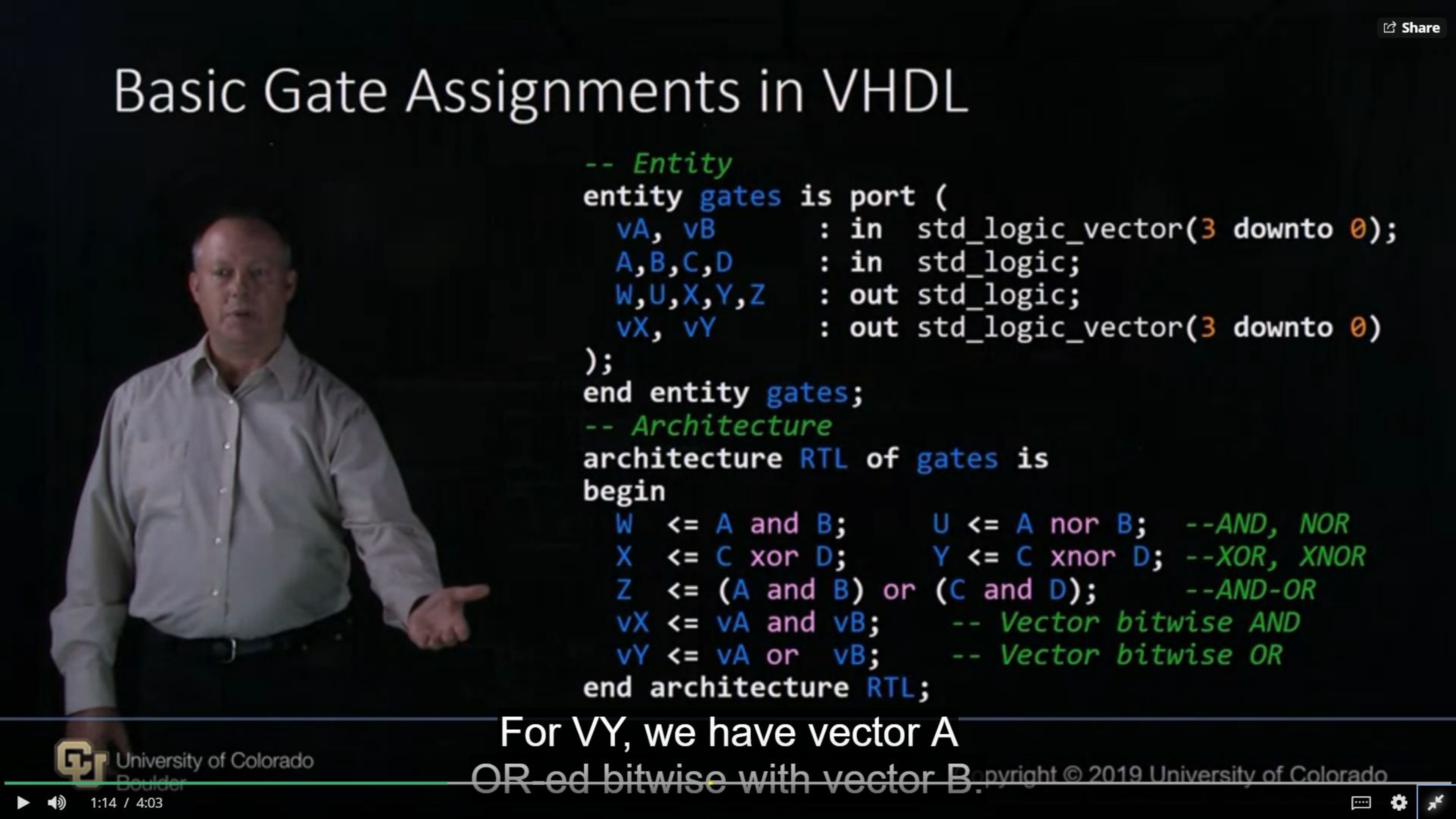
- How to describe combinatorial circuits in VHDL
- How to reduce vector sizes using reduction operators

and how to reduce vector sizes
using reduction operators.

Basic Gate Assignments in VHDL



Here we have a set of gates: AND gates, OR gates,

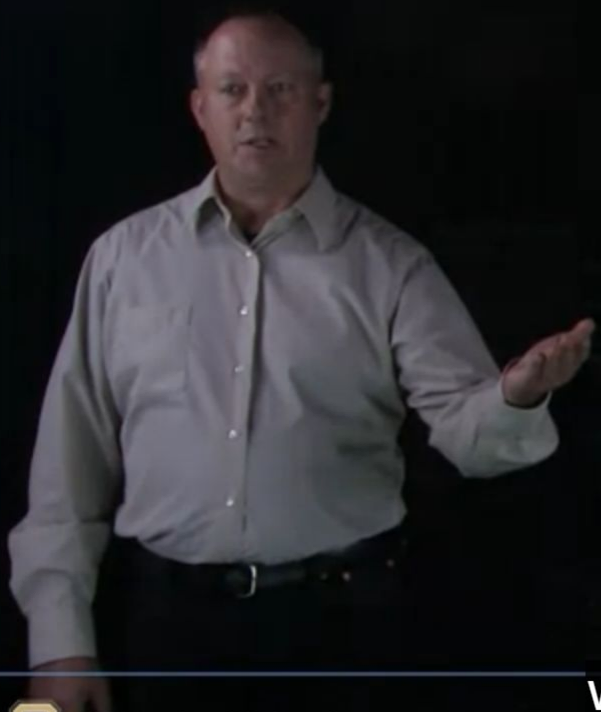


Basic Gate Assignments in VHDL

```
-- Entity
entity gates is port (
  vA, vB        : in  std_logic_vector(3 downto 0);
  A,B,C,D       : in  std_logic;
  W,U,X,Y,Z     : out std_logic;
  vX, vY        : out std_logic_vector(3 downto 0)
);
end entity gates;
-- Architecture
architecture RTL of gates is
begin
  W  <= A and B;      U <= A nor B;  --AND, NOR
  X  <= C xor D;      Y <= C xnor D; --XOR, XNOR
  Z  <= (A and B) or (C and D);      --AND-OR
  vX <= vA and vB;    -- Vector bitwise AND
  vY <= vA or  vB;    -- Vector bitwise OR
end architecture RTL;
```

For VY, we have vector A
OR-ed bitwise with vector B.

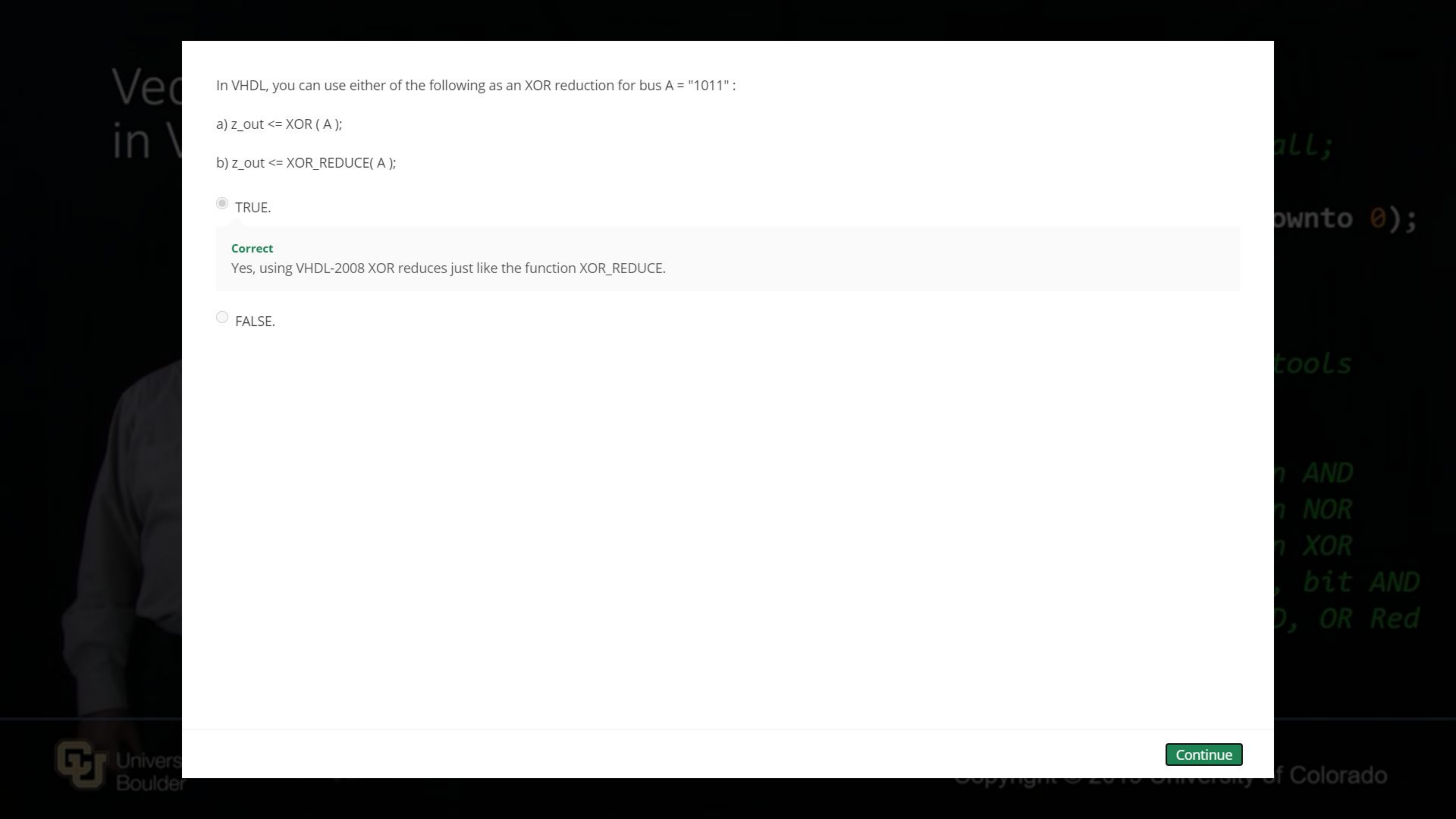
Vector Reduction in VHDL



```
-- Entity :  Note:  use IEEE.std_logic_misc.all;
entity gates is port (
  vA, vB, vC, vD  : in  std_logic_vector(3 downto 0);
  W,U,X,Y,Z       : out std_logic  );
end entity gates;

-- Architecture : reduction after VHDL-2008 tools
architecture RTL of gates is
begin
  W  <=  AND_REDUCE(vA);  -- Vector Reduction AND
  U  <=  NOR_REDUCE(vB);  -- Vector Reduction NOR
  X  <=  XOR_REDUCE(vD);  -- Vector Reduction XOR
  Y  <=  OR_REDUCE(vA) and vB(0);  -- OR Red, bit AND
  Z  <=  OR_REDUCE(vA and vB);     -- Bit AND, OR Red
end architecture RTL;
```

we have a built in function
in the VHDL library.



In VHDL, you can use either of the following as an XOR reduction for bus A = "1011" :

a) z_out <= XOR (A);

b) z_out <= XOR_REDUCE(A);

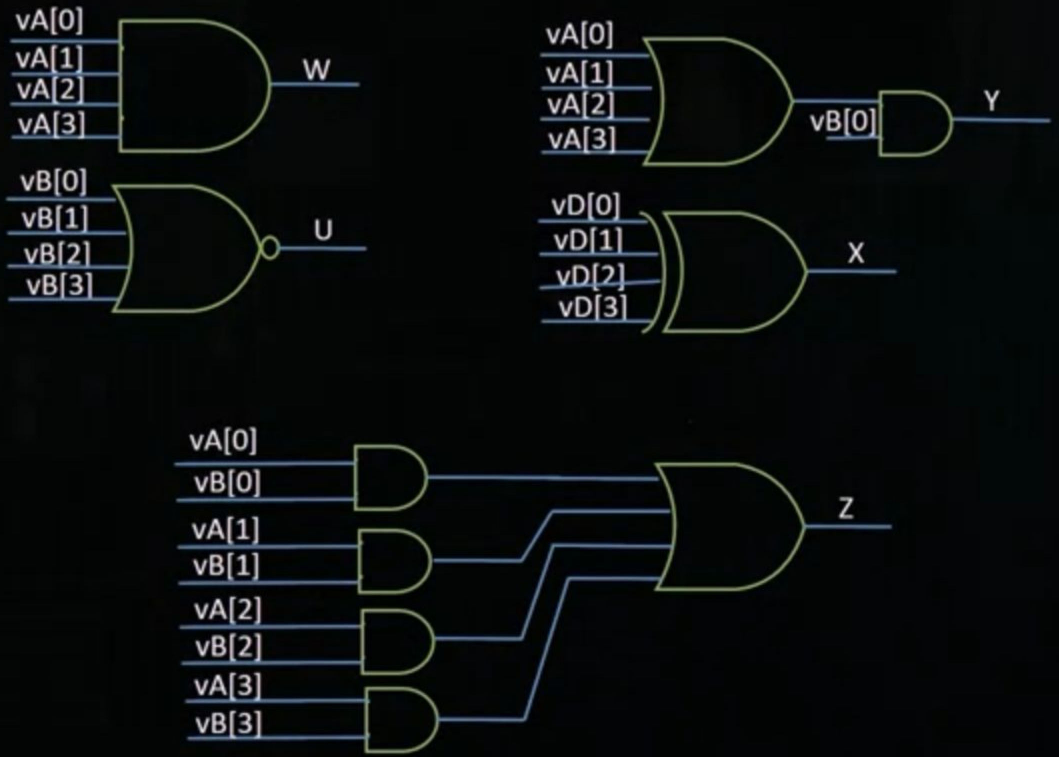
[✓ TRUE.]

Correct

Yes, using VHDL-2008 XOR reduces just like the function XOR_REDUCE.

FALSE.

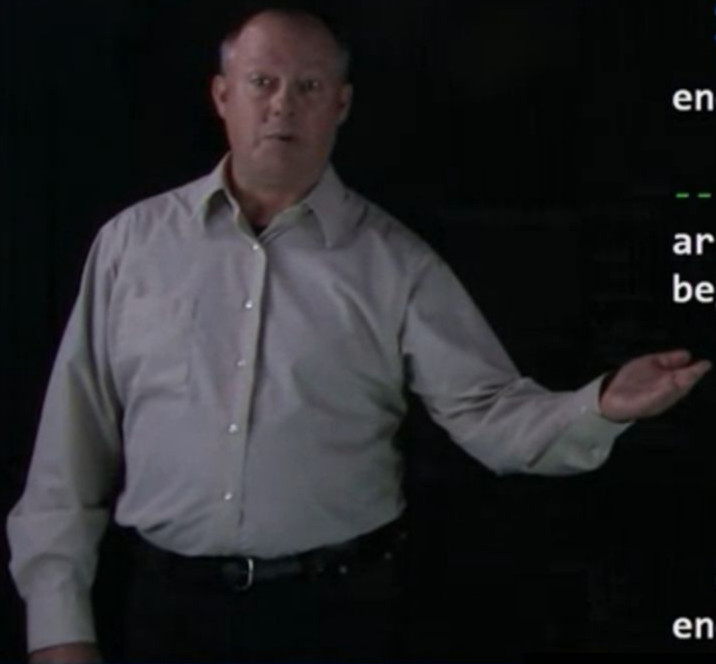
Vector Reduction in VHDL



So we have vector A
bitwise AND-ed for W,

Share

Procedural Logic in VHDL



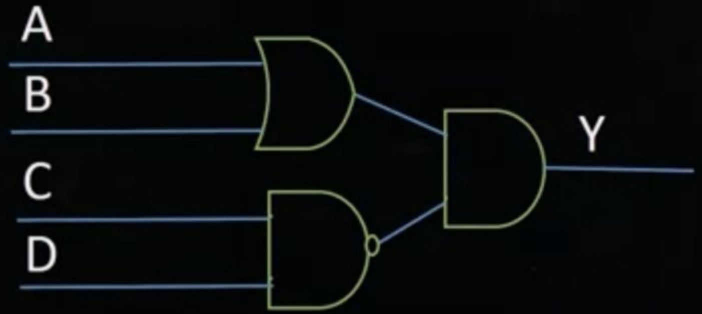
```
-- Entity
entity gates is port (
  A,B,C,D      : in  std_logic;
  Y            : out std_logic    );
end entity gates;

-- Architecture
architecture combo of gates is
begin combo_process : process (A,B,C,D)
  begin
    if    ((C='1') and (D='1')) then  Y <= '0';
    elsif ((A='1') or  (B='1')) then  Y <= '1’;
    else    Y <= '0';
    end if;
  end process combo_process;
end architecture combo;
```

we can create a statement with

2:37 / 4:03

Procedural Logic in VHDL - Circuit



we have A OR-ed with B,

Procedural Logic – Gotcha, Latch Generate

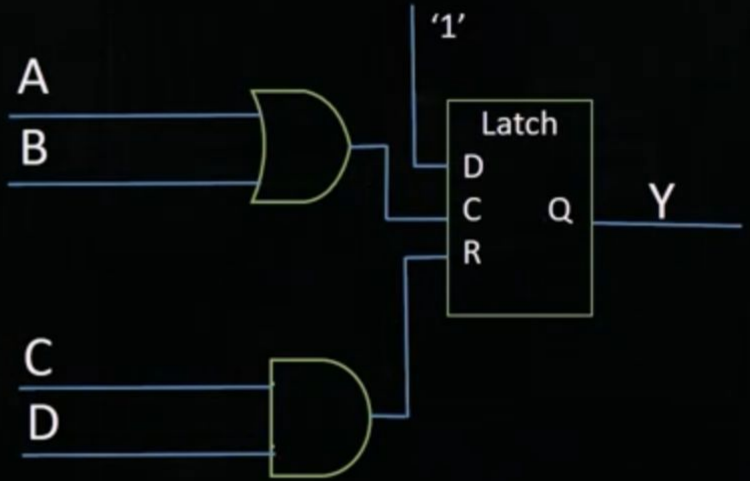
```
-- Entity
entity gates is port (
  A,B,C,D       : in  std_logic;
  Y             : out std_logic    );
end entity gates;

-- Architecture
architecture combo of gates is
begin combo_process : process (A,B,C,D)
  begin
    if    ((C='1') and (D='1')) then  Y <= '0';
    elsif ((A='1') or  (B='1')) then  Y <= '1';
    end if;  -- Missing Else condition for Y
  end process combo_process;
end architecture combo;
```

missing which zeroed out the Y,

University of Colorado Boulder

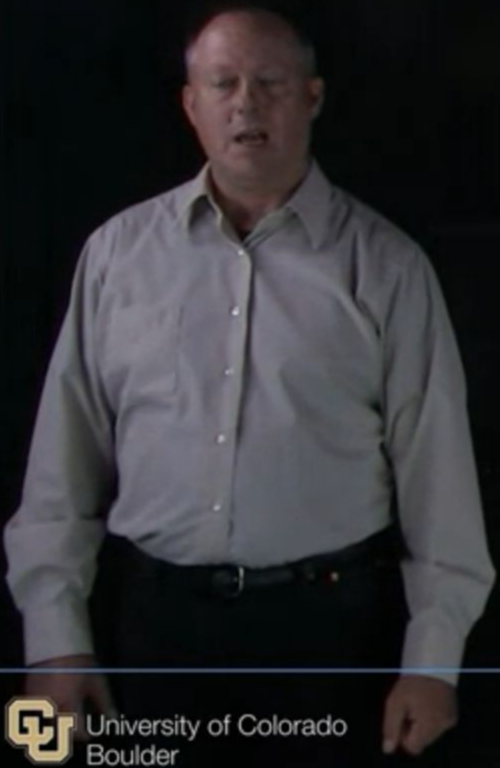
Procedural Logic – Gotcha, Latch Generate Incomplete Assignment



So in the same case,

3:33 / 4:03

Summary – VHDL for Combinatorial Circuits



In this video, you have learned:

- How to describe combinatorial circuits in VHDL using either assignment statements
- How to reduce vector sizes using operators as bus to bit reduction operators
- How to prevent unintentional latches in logic circuits

using assignment statements,

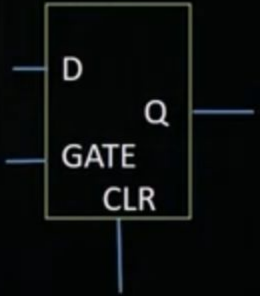
FPGA Design for Embedded Systems

Hardware Description Languages for Logic Design

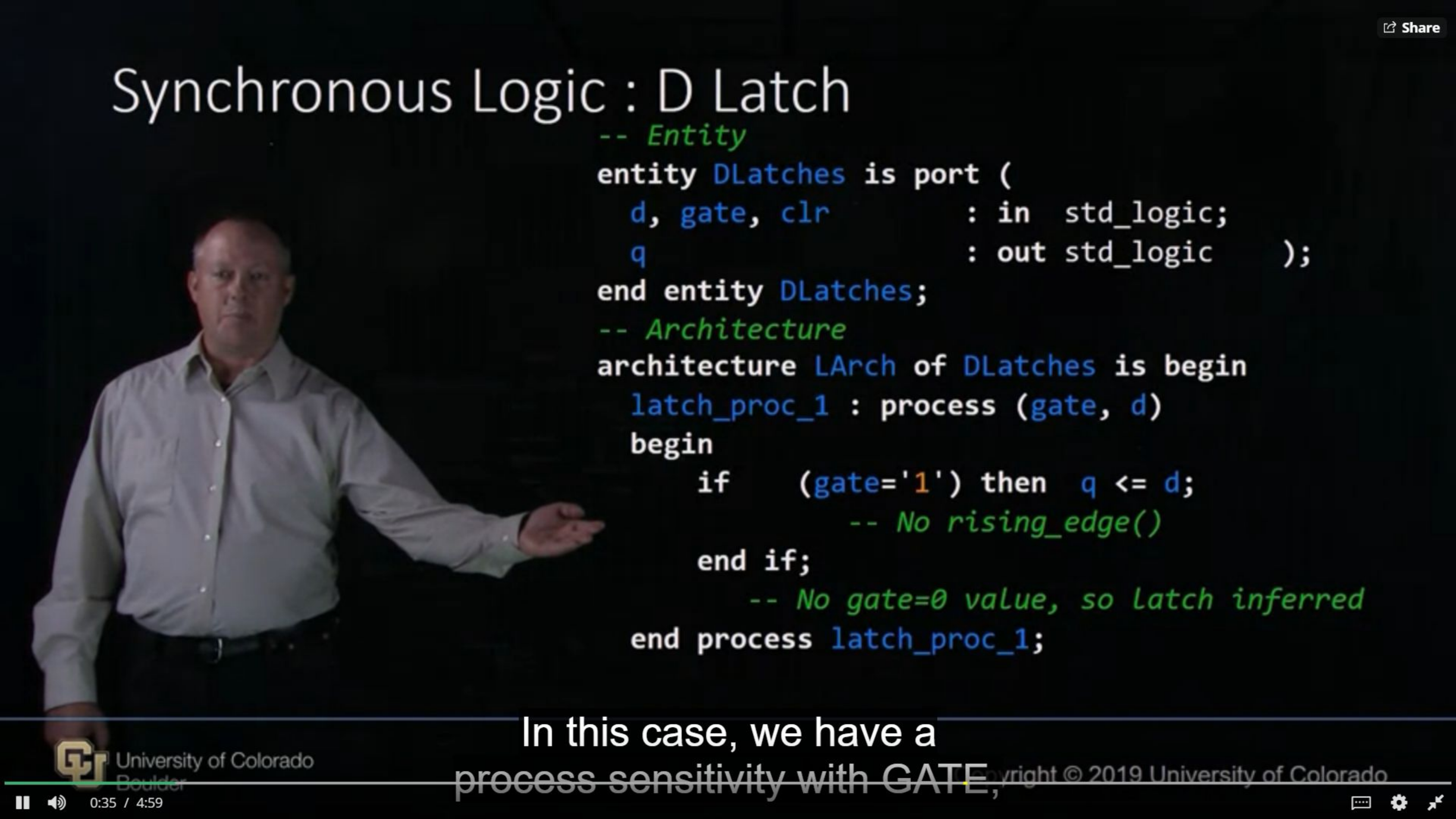
Hello, and welcome back to

Synchronous Logic : D Latch

How can we make a D Latch in VHDL?



synchronous logic as well
as combinatorial logic.

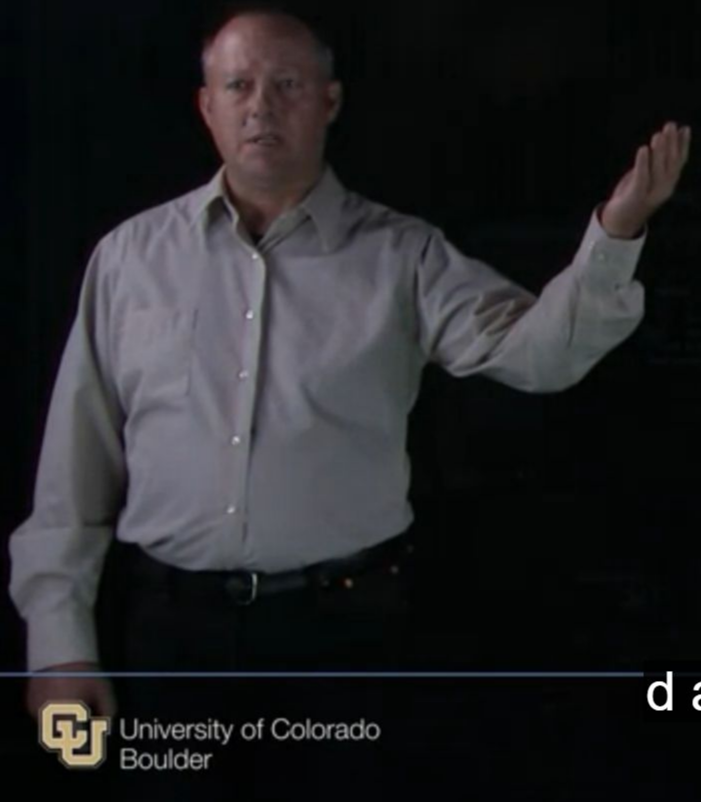


Synchronous Logic : D Latch

```
-- Entity
entity DLatches is port (
  d, gate, clr          : in  std_logic;
  q                     : out std_logic    );
end entity DLatches;
-- Architecture
architecture LArch of DLatches is begin
  latch_proc_1 : process (gate, d)
  begin
      if   (gate='1') then  q <= d;
                -- No rising_edge()
      end if;
          -- No gate=0 value, so latch inferred
  end process latch_proc_1;
```

In this case, we have a
process sensitivity with GATE,

Synchronous Logic : D Latch



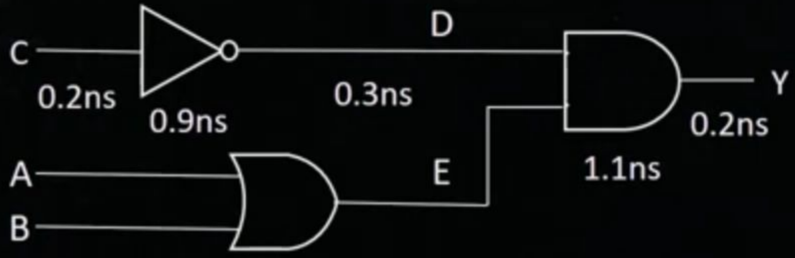
```
  -- another Latch example

    latch_proc_2 : process (gate, d, clr)
    begin
        if    (clr ='1') then  q <= '0';
        elsif (gate='1') then  q <= d;
        end if;
    end process latch_proc_2;
  end architecture LArch;
```

d and clear as an input.

Combinatorial Logic

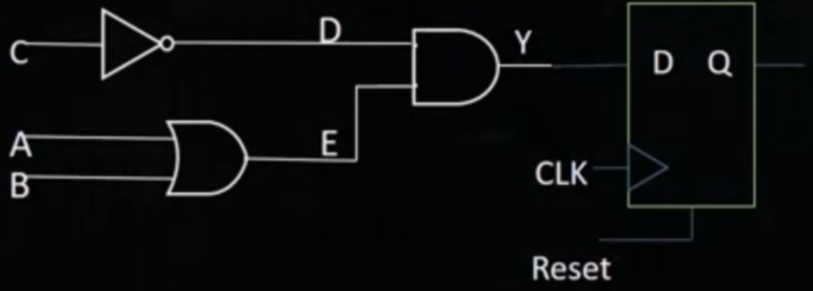
- Finite time for best case or worst case logic value to propagate a result through wires and logic cells across the chip
- 0 -> 1 transition for path C to Y
  - 0.2 + 0.9 + 0.3 + 1.1 + 0.2 = 2.7nano seconds

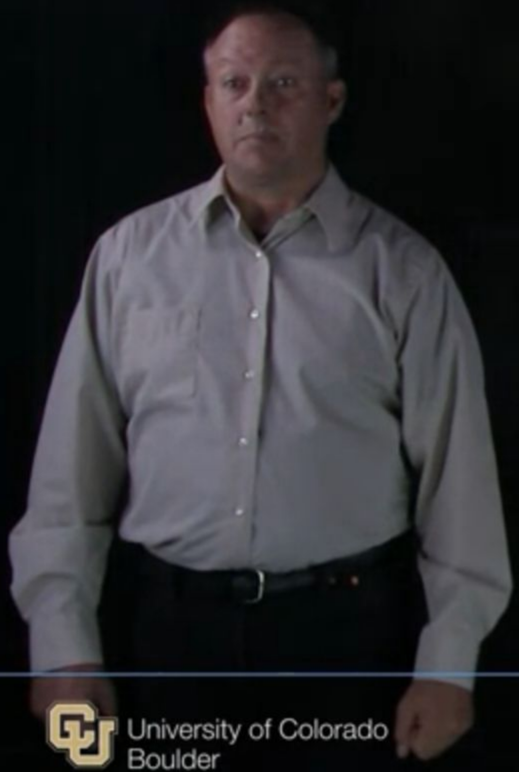


Combinatorial logic is END gates,

Synchronous Logic

- Logic path synchronized to CLK

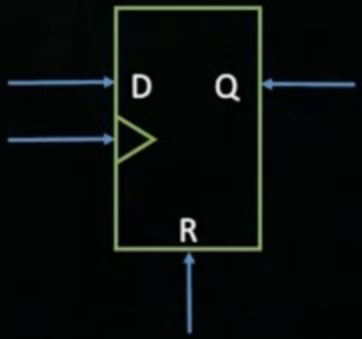




If we wanted to
synchronize this path,

Synchronous Logic : D Flip Flop

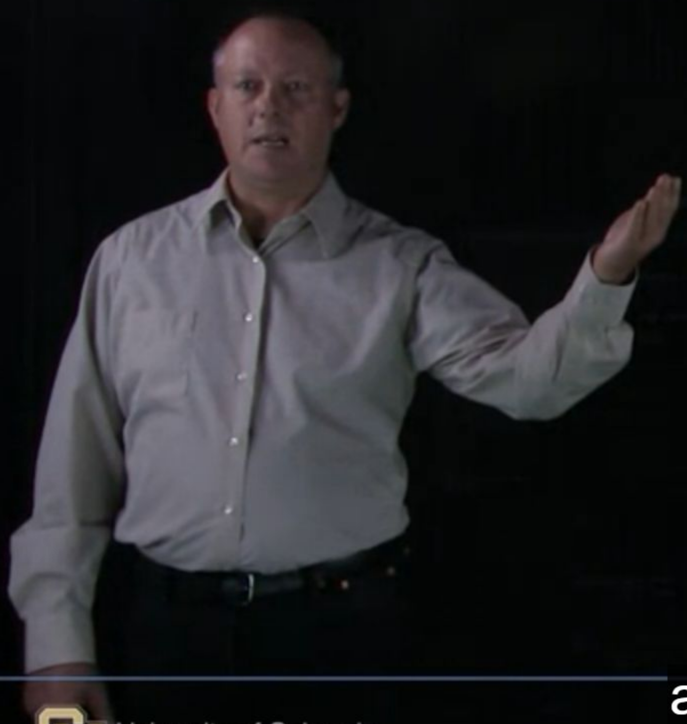
How can we make a D Flip Flop in VHDL?



D input, clock trigger,

Synchronous Logic : D Flip Flop – Sync Reset

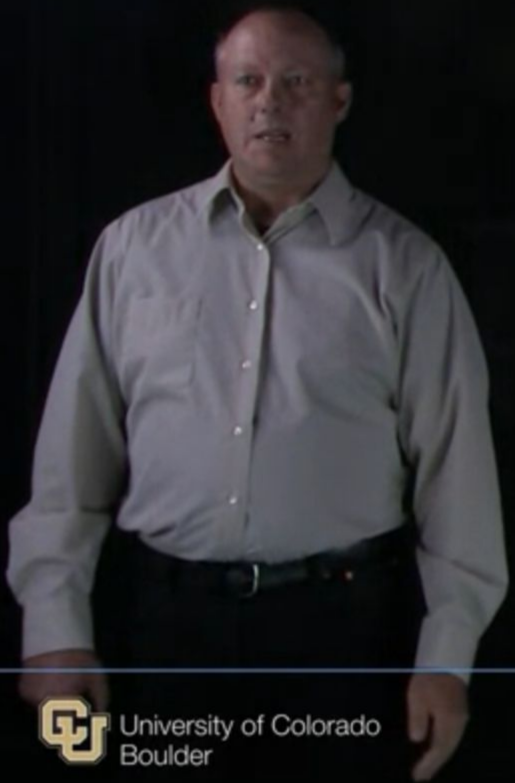
```
-- Entity
entity DFF is port (
  d, clk, reset               : in  std_logic;
  q                           : out std_logic    );
end entity DFF;
```



and a q as an output.

University of Colorado Boulder

Synchronous Logic : D Flip Flop – Sync Reset



```
-- Architecture,
-- could use (clk'event and clk='1')
architecture DFF_Arch of DFF is
  begin dff_proc_1 : process (clk)
    begin
       if (rising_edge(clk))  then
         if (reset='1') then q <=  '0';
                 --  Sync Reset
         else                     q <= d;
         end if;
       end if;
  end process dff_proc_1;
end architecture DFF_Arch;
```

In this case, we could
also use clock tick

University of Colorado Boulder

Either of the following constructs with create a Flip-Flop :

a) if (clk'event and clk='1') then

b) if (rising_edge(clk)) then

[✓ True.]

Correct
Right. Either method will create a Flip-Flop.

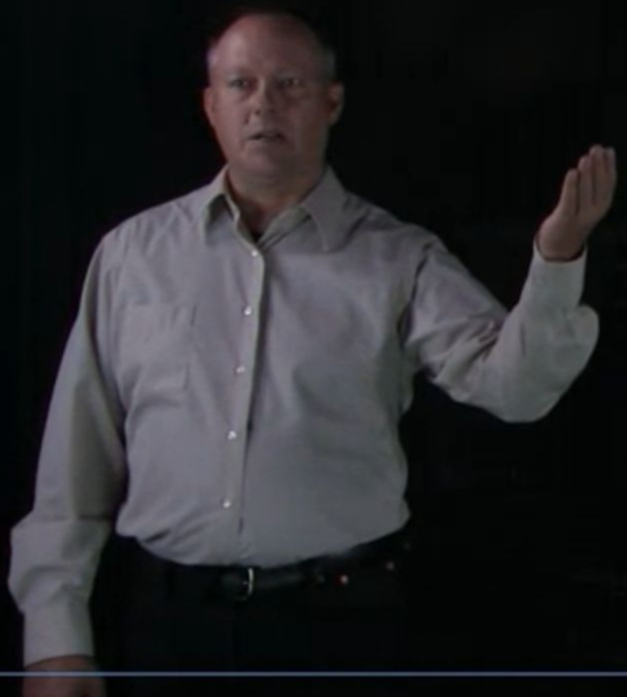
False.

Continue

Share

Synchronous Logic : D Flip Flop – Async Reset

```
-- Entity
entity DFF is port (
  d, clk, set, reset      : in  std_logic;
  q                       : out std_logic    );
end entity DFF;
-- Architecture, next slide
```



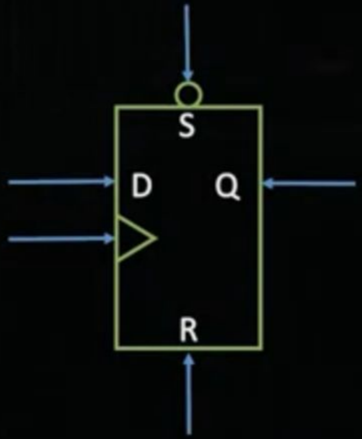
Our entity looks
similar d clock set,

University of Colorado Boulder

3:24 / 4:59

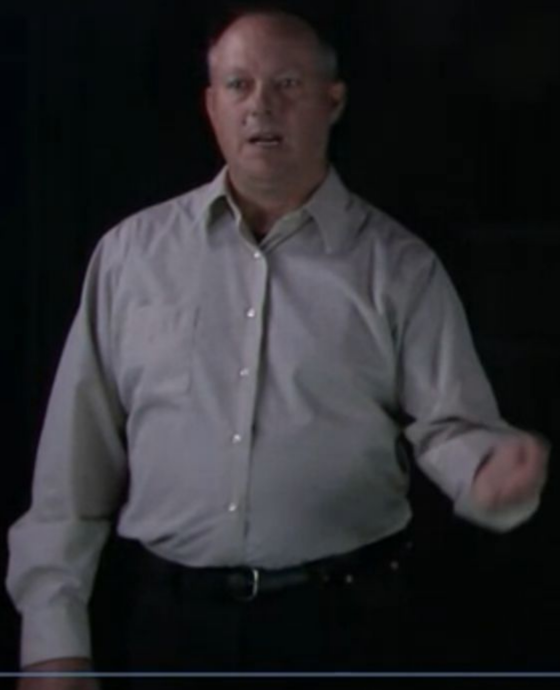
Synchronous Logic : D Flip Flop – Async Reset

How can we make a D Flip Flop in VHDL?



create a D flip flop with
an asynchronous reset?

Synchronous Logic : D Flip Flop – Async Reset

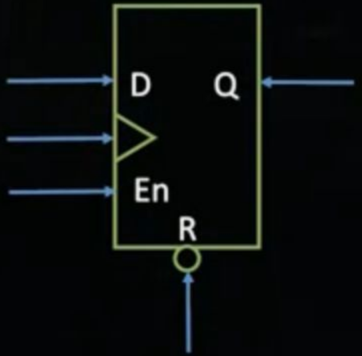


```
-- Architecture
architecture DFF_Arch of DFF is
  begin dff_proc_2 : process (clk, set, reset)
    begin
       if    (reset='1')        then  q <= '0';
              -- Async
       elsif (rising_edge(clk)) then
          if    (set='0')       then  q <=  '1';
              -- Sync
          else                        q <=   d;
              -- Sync
          end if;
       end if;
  end process dff_proc_2;
end architecture DFF_Arch;
```

and here's our architecture.

Synchronous Logic : D Flip Flop

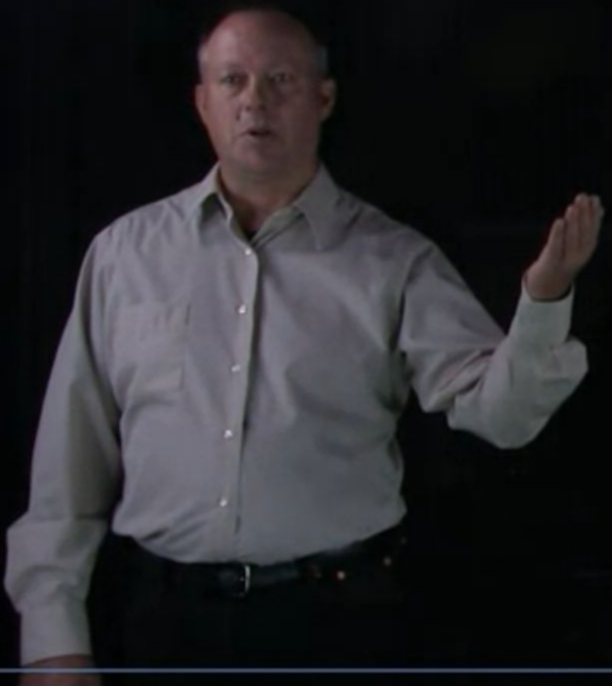
How can we make a Clock Enable in VHDL?



How would we make a d flip flop with a clock enable?

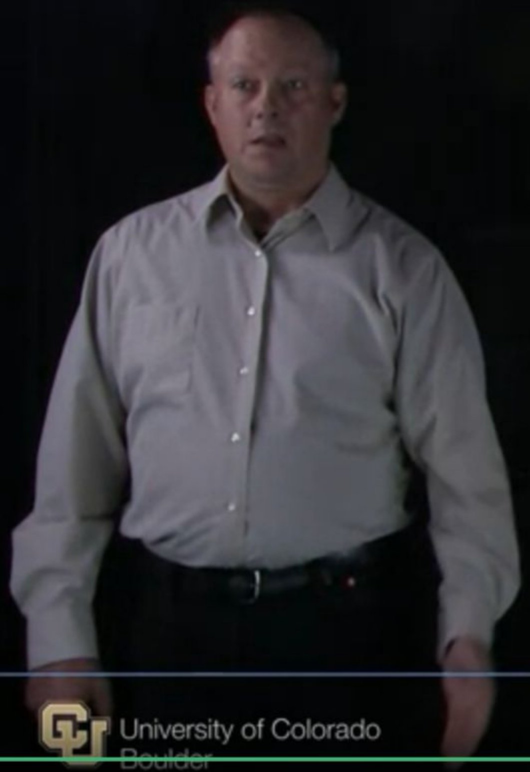
Synchronous Logic : D Flip Flop – Clock Enable

```
-- Entity
entity DFF is port (
  d, clk, ce, reset        : in  std_logic;
  q                        : out std_logic    );
end entity DFF;
-- Architecture Next Slide
```



Similar entity, d clock,

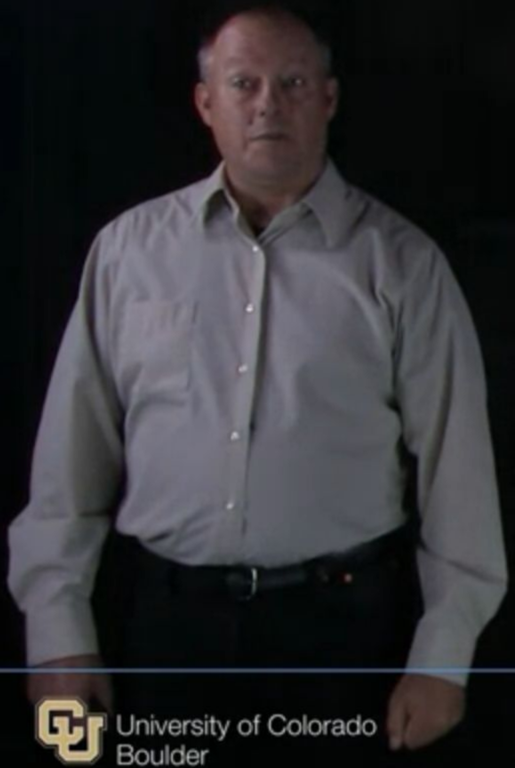
Synchronous Logic : D Flip Flop – Clock Enable



```vhdl
-- Architecture
architecture DFF_Arch of DFF is
  begin dff_proc_3 : process (clk, ce, reset)
    begin
       if    (reset='1')        then  q <= '0';
                -- Async
       elsif (rising_edge(clk)) then
          if    (ce='1')        then  q <=   d;
                -- Sync
          end if;
       end if;
   end process dff_proc_3;
end architecture DFF_Arch;
```

Here's our architecture for

Summary – VHDL for Synchronous Circuits



In this video, you have learned:

- How to describe synchronous circuits in VHDL
- How to design flip flops with synchronous and asynchronous set and reset

In this video, you've
learned how to

University of Colorado
Boulder