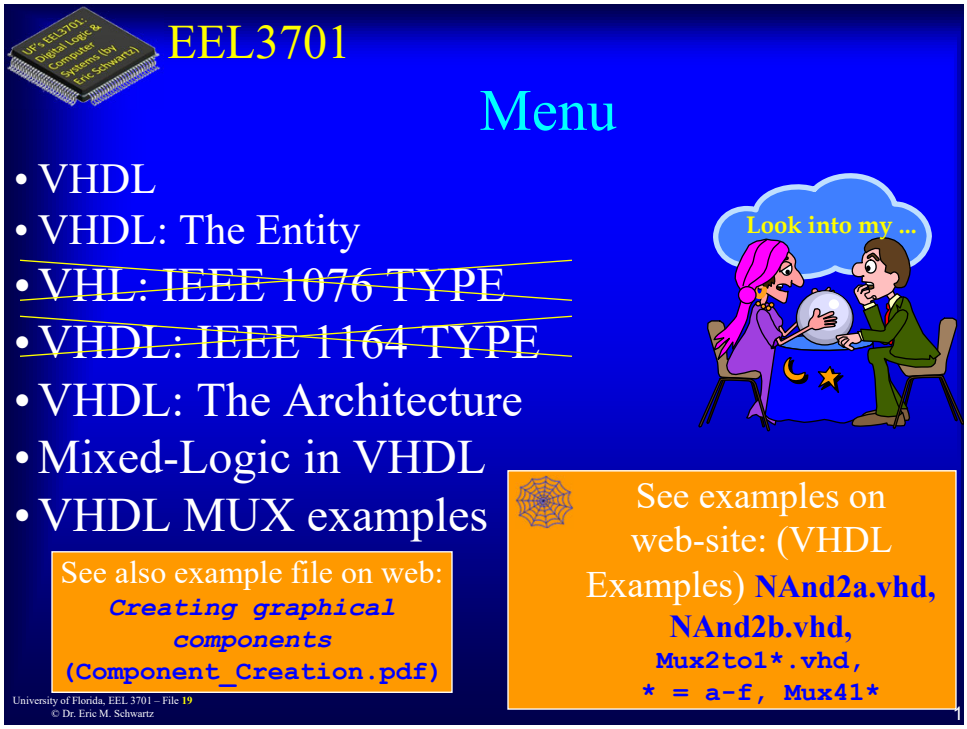
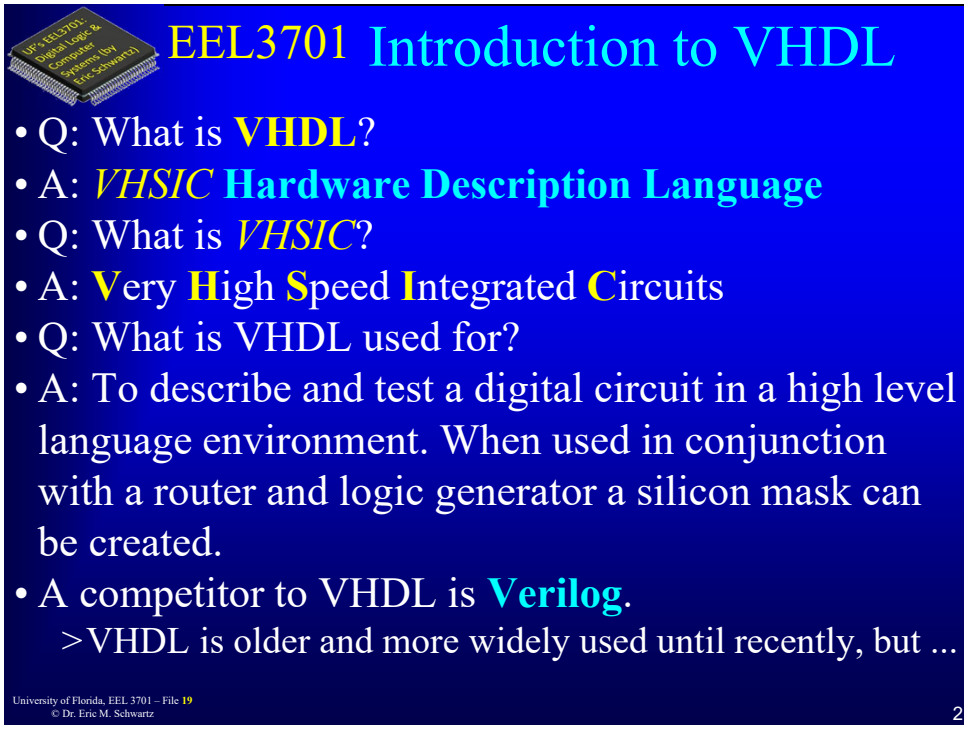

EEL3701
Menu
• VHDL
• VHDL: The Entity
• ~~VHL: IEEE 1076 TYPE~~
• ~~VHDL: IEEE 1164 TYPE~~
• VHDL: The Architecture
• Mixed-Logic in VHDL
• VHDL MUX examples
See also example file on web:
Creating graphical components
(Component_Creation.pdf)
Look into my ...
See examples on web-site: (VHDL Examples) NAnd2a.vhd, NAnd2b.vhd,
Mux2to1*.vhd,
* = a-f, Mux41*
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
1

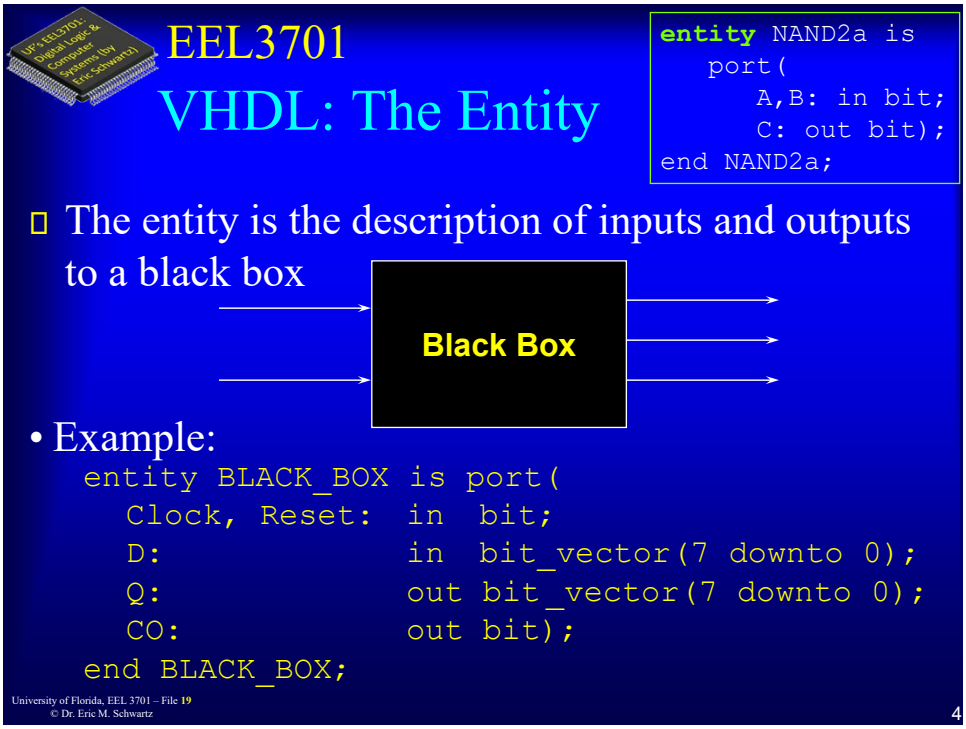
1


EEL3701 Introduction to VHDL
• Q: What is VHDL?
• A: VHSIC Hardware Description Language
• Q: What is VHSIC?
• A: Very High Speed Integrated Circuits
• Q: What is VHDL used for?
• A: To describe and test a digital circuit in a high level language environment. When used in conjunction with a router and logic generator a silicon mask can be created.
• A competitor to VHDL is Verilog.
>VHDL is older and more widely used until recently, but ...
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
2

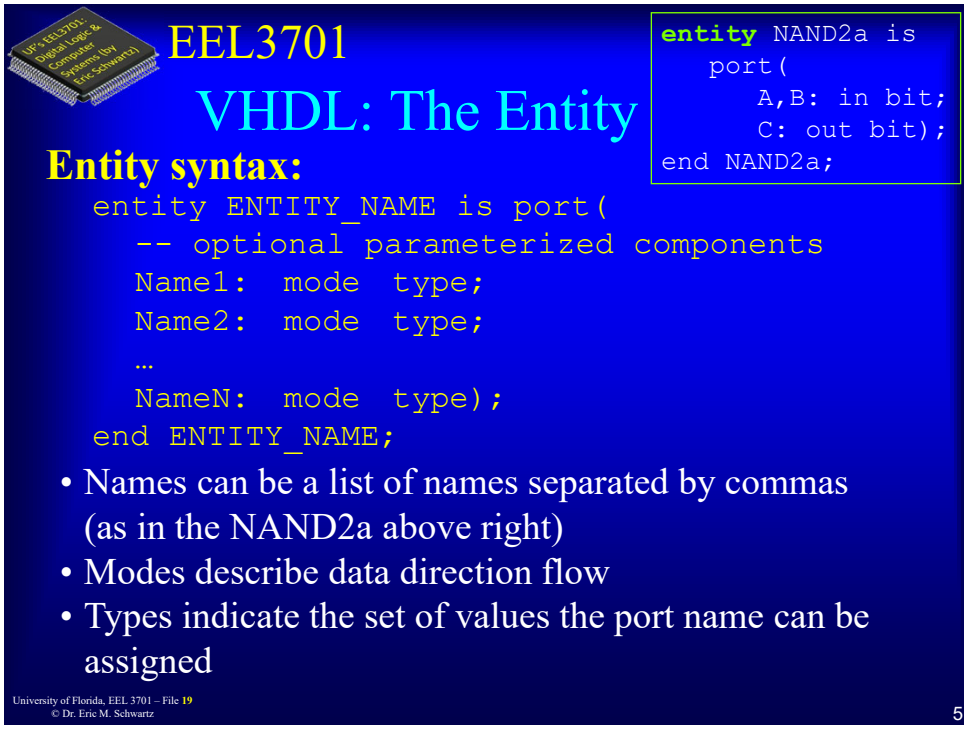
2


EEL3701
VHDL Syntax
• VHDL is case insensitive
> GOOD = Good = good = gOOD
• Everyone has their own conventions; mine follows:
> Keyword of VHDL are all lower case
> Entity and architecture names are all upper case
> Identifiers start with a capital
> All new words in a given identifier is again capitalized
• White space (spaces or tabs) is fine anywhere as separators
• The semicolon is a statement terminator
• Two dashes ("--") indicate a comment follows
• Identifiers must begin with a letter; subsequent characters are alphanumeric or "_"
• No precedence in VHDL; resolves left-to-right; use paren.
(except not has precedence over logical operators)
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
3

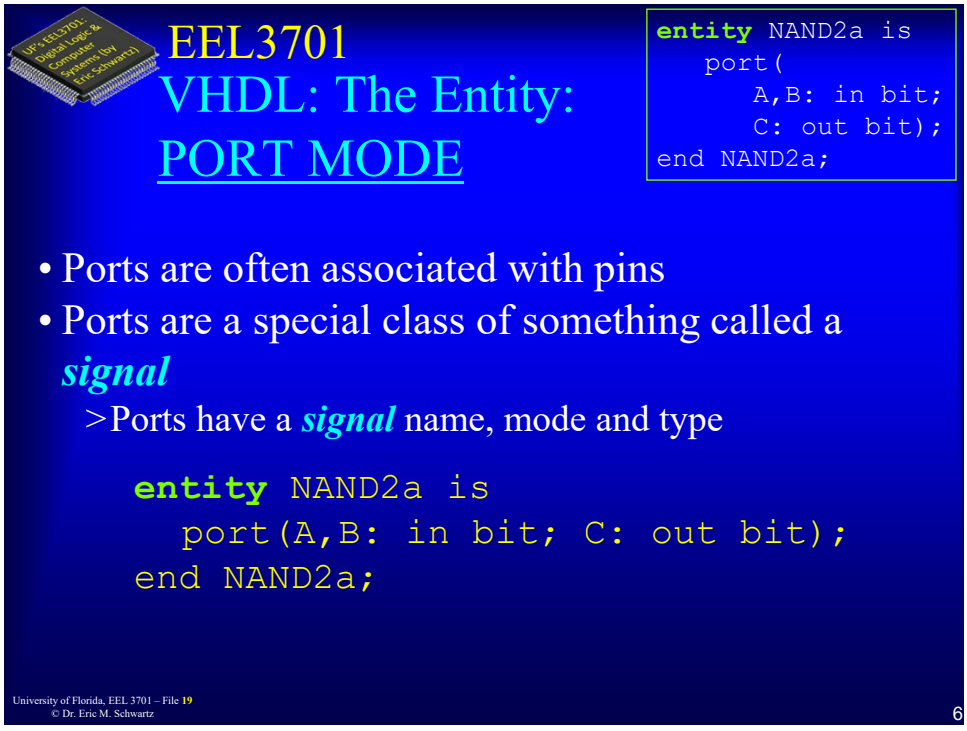
3


EEL3701
VHDL: The Entity
entity NAND2a is
port(
A,B: in bit;
C: out bit);
end NAND2a;
The entity is the description of inputs and outputs to a black box
Black Box
• Example:
entity BLACK_BOX is port(
Clock, Reset: in bit;
D: in bit_vector(7 downto 0);
Q: out bit_vector(7 downto 0);
CO: out bit);
end BLACK_BOX;
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
4

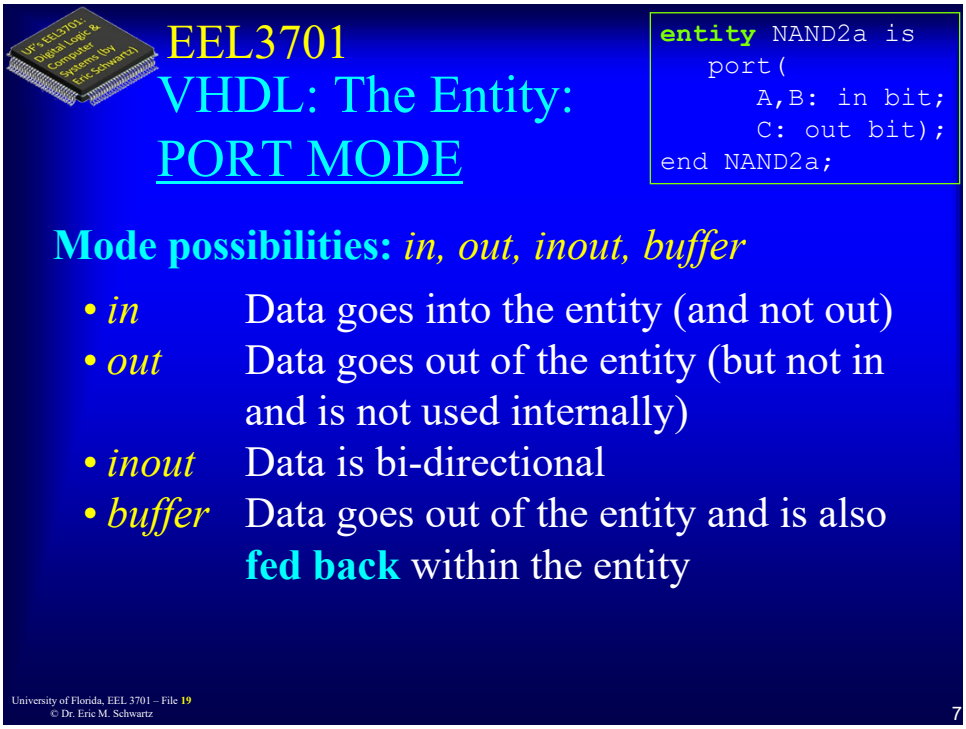
4


EEL3701
VHDL: The Entity
Entity syntax:
entity ENTITY_NAME is port(
-- optional parameterized components
Name1: mode type;
Name2: mode type;
…
NameN: mode type);
end ENTITY_NAME;
• Names can be a list of names separated by commas (as in the NAND2a above right)
• Modes describe data direction flow
• Types indicate the set of values the port name can be assigned
entity NAND2a is
port(
A,B: in bit;
C: out bit);
end NAND2a;
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
5

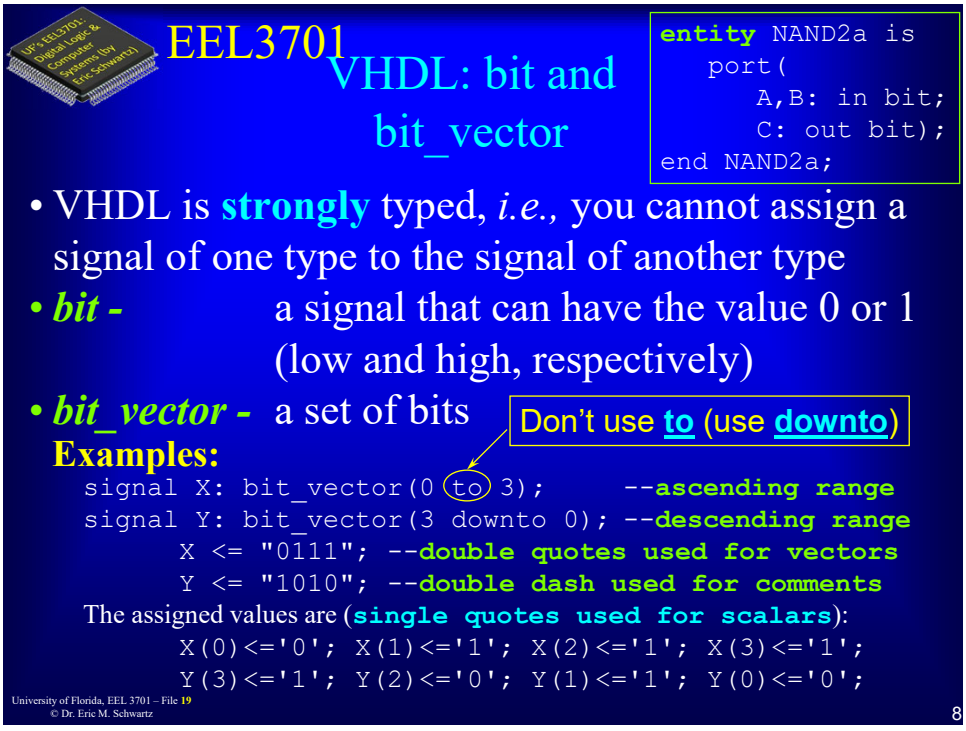
5


EEL3701
VHDL: The Entity:
PORT MODE
• Ports are often associated with pins
• Ports are a special class of something called a signal
>Ports have a signal name, mode and type
entity NAND2a is
port(A,B: in bit; C: out bit);
end NAND2a;
entity NAND2a is
port(
A,B: in bit;
C: out bit);
end NAND2a;
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
6

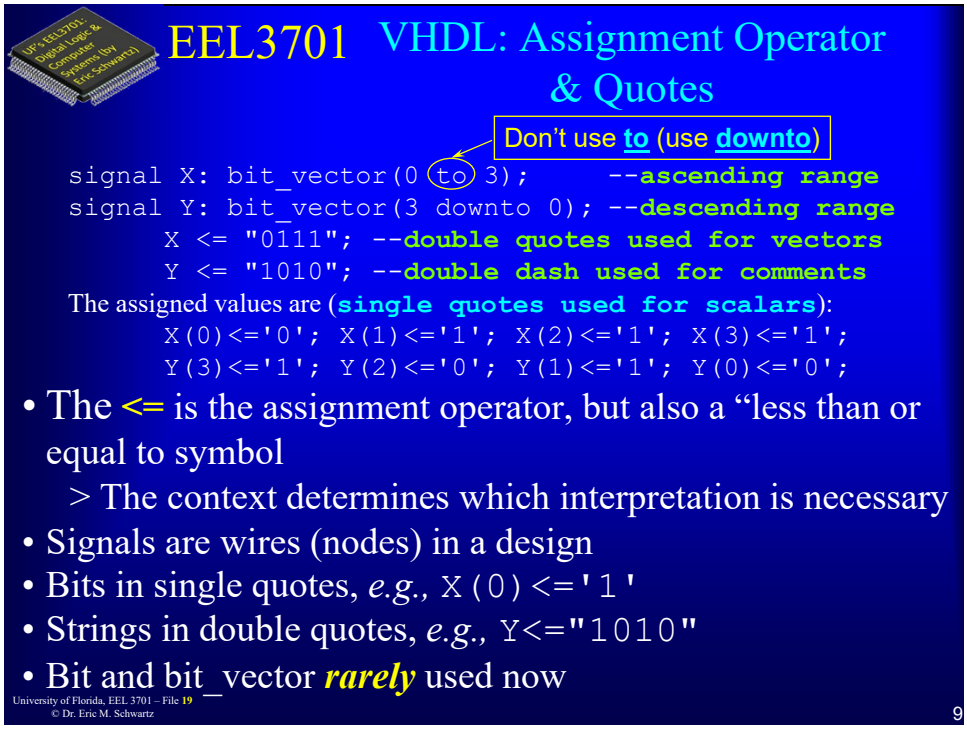
6


EEL3701
VHDL: The Entity:
PORT MODE
entity NAND2a is
port(
A,B: in bit;
C: out bit);
end NAND2a;
Mode possibilities: in, out, inout, buffer
• in Data goes into the entity (and not out)
• out Data goes out of the entity (but not in and is not used internally)
• inout Data is bi-directional
• buffer Data goes out of the entity and is also fed back within the entity
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
7

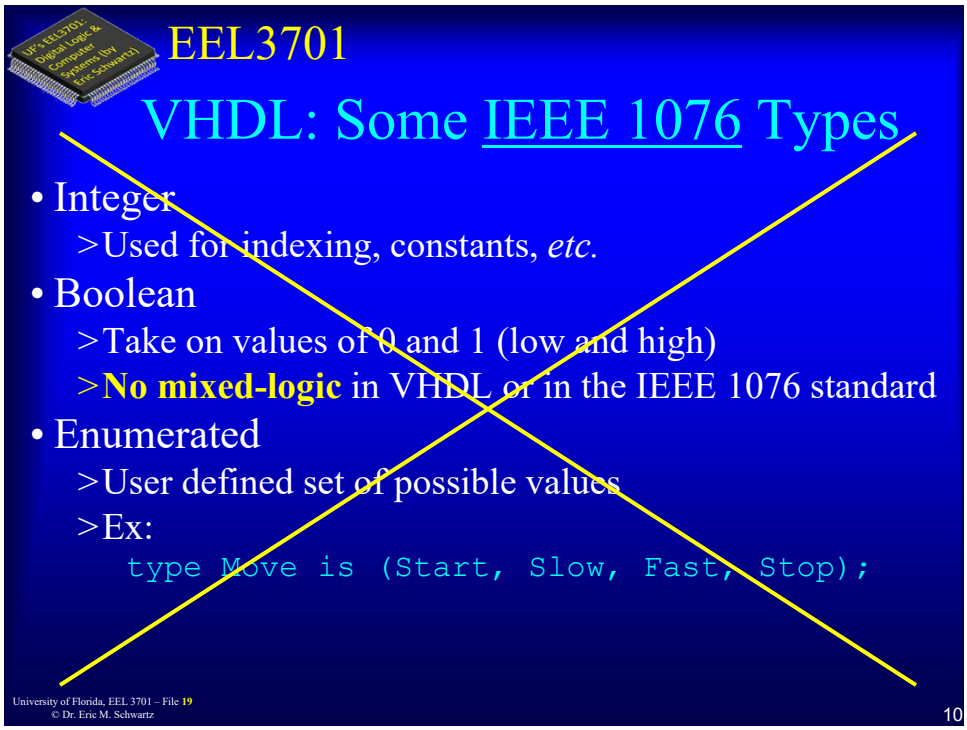
7


EEL3701
VHDL: bit and bit_vector
entity NAND2a is
port(
A,B: in bit;
C: out bit);
end NAND2a;
• VHDL is strongly typed, i.e., you cannot assign a signal of one type to the signal of another type
• bit - a signal that can have the value 0 or 1 (low and high, respectively)
• bit_vector - a set of bits
Don't use to (use downto)
Examples:
signal X: bit_vector(0 to 3); --ascending range
signal Y: bit_vector(3 downto 0); --descending range
X <= "0111"; --double quotes used for vectors
Y <= "1010"; --double dash used for comments
The assigned values are (single quotes used for scalars):
X(0)<='0'; X(1)<='1'; X(2)<='1'; X(3)<='1';
Y(3)<='1'; Y(2)<='0'; Y(1)<='1'; Y(0)<='0';
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
8

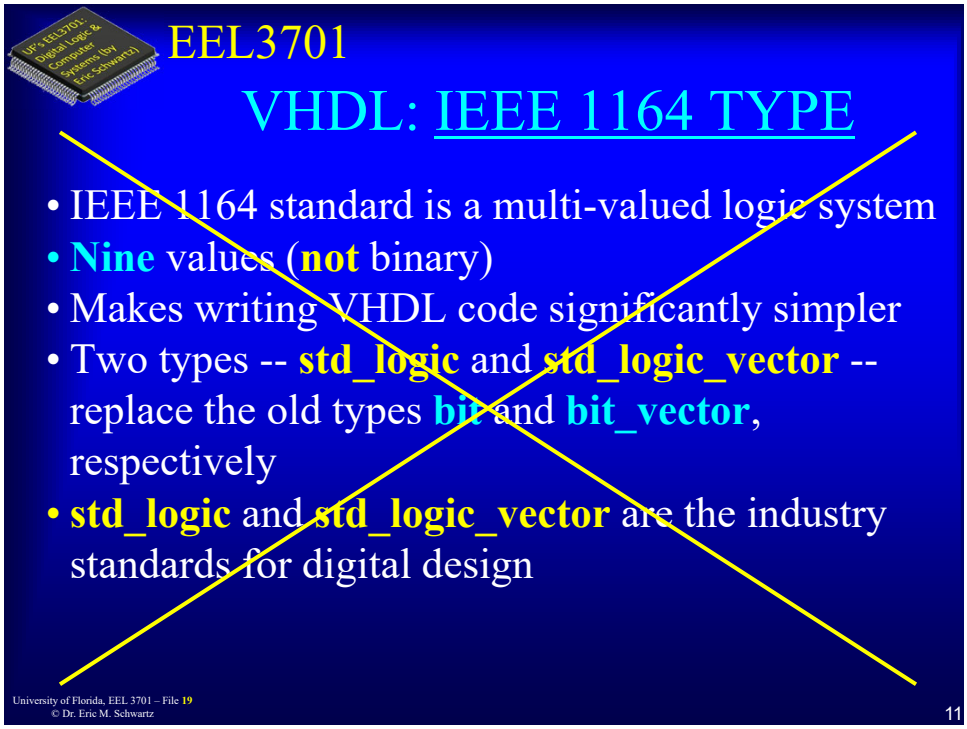
8


EEL3701 VHDL: Assignment Operator & Quotes
Don't use to (use downto)
signal X: bit_vector(0 to 3); --ascending range
signal Y: bit_vector(3 downto 0); --descending range
X <= "0111"; --double quotes used for vectors
Y <= "1010"; --double dash used for comments
The assigned values are (single quotes used for scalars):
X(0)<='0'; X(1)<='1'; X(2)<='1'; X(3)<='1';
Y(3)<='1'; Y(2)<='0'; Y(1)<='1'; Y(0)<='0';
• The <= is the assignment operator, but also a "less than or equal to symbol
> The context determines which interpretation is necessary
• Signals are wires (nodes) in a design
• Bits in single quotes, e.g., X(0)<='1'
• Strings in double quotes, e.g., Y<="1010"
• Bit and bit_vector rarely used now
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
9

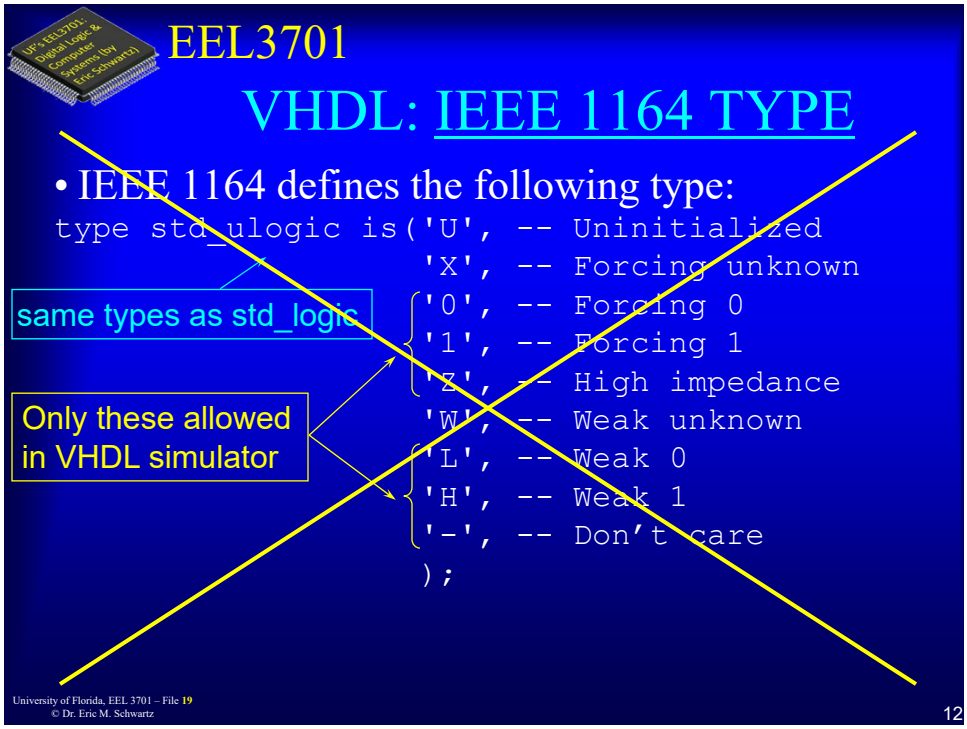
9


EEL3701
VHDL: Some IEEE 1076 Types
• Integer
>Used for indexing, constants, etc.
• Boolean
>Take on values of 0 and 1 (low and high)
>No mixed-logic in VHDL or in the IEEE 1076 standard
• Enumerated
>User defined set of possible values
>Ex:
type Move is (Start, Slow, Fast, Stop);
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
10

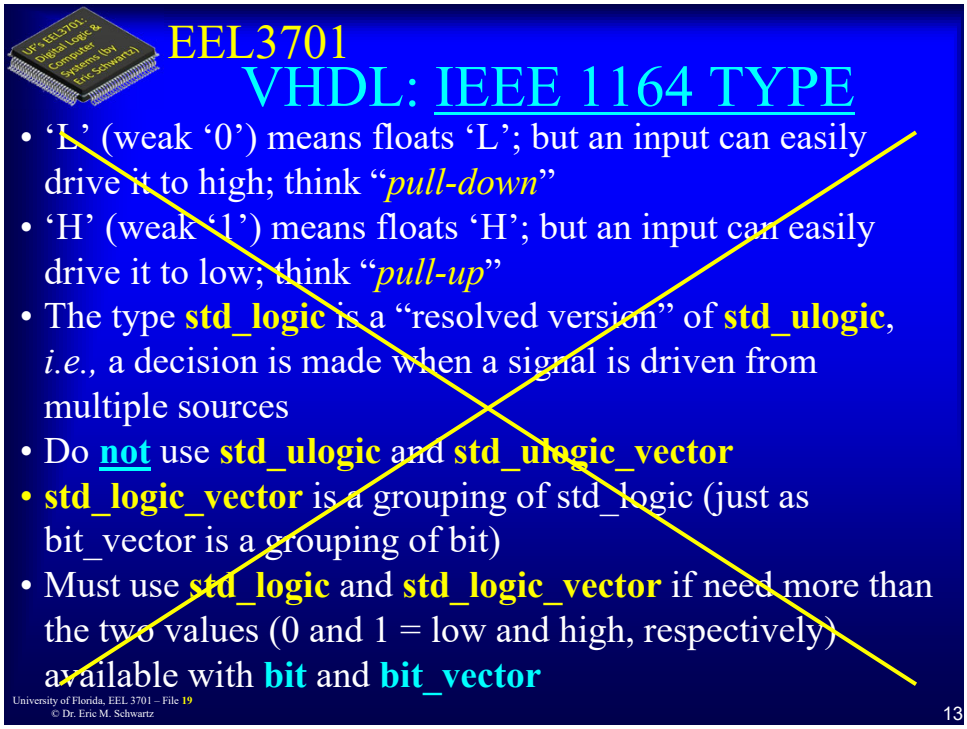
10


EEL3701
VHDL: IEEE 1164 TYPE
• IEEE 1164 standard is a multi-valued logic system
• Nine values (not binary)
• Makes writing VHDL code significantly simpler
• Two types -- std_logic and std_logic_vector -- replace the old types bit and bit_vector, respectively
• std_logic and std_logic_vector are the industry standards for digital design
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
11

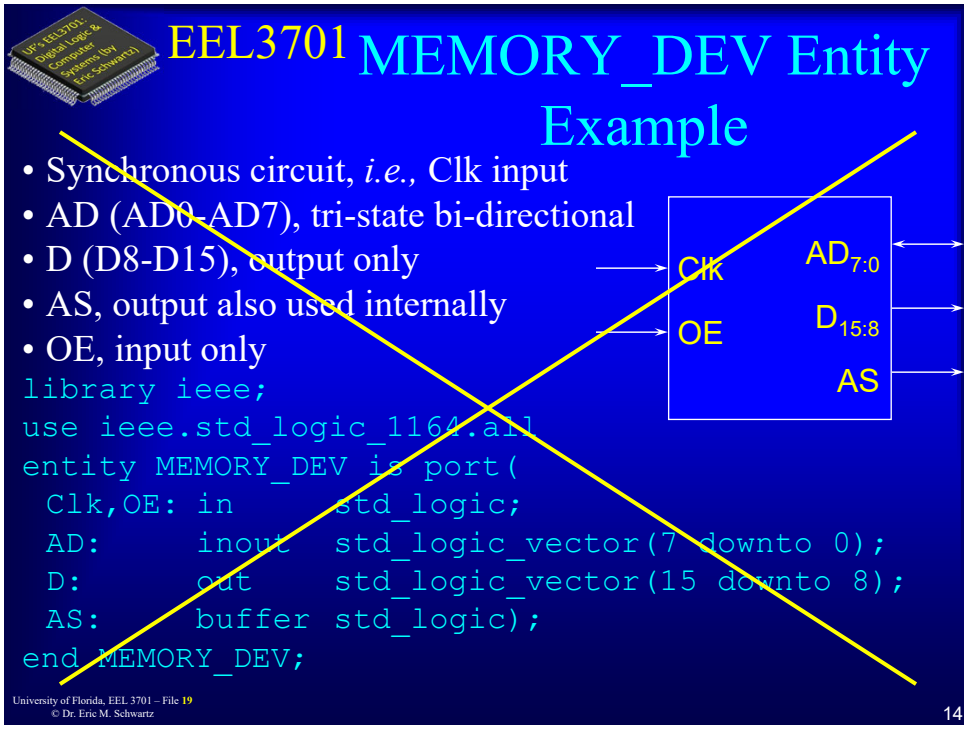
11


EEL3701
VHDL: IEEE 1164 TYPE
• IEEE 1164 defines the following type:
type std_ulogic is('U', -- Uninitialized
'X', -- Forcing unknown
'0', -- Forcing 0
'1', -- Forcing 1
'Z', -- High impedance
'W', -- Weak unknown
'L', -- Weak 0
'H', -- Weak 1
'-', -- Don't care
);
same types as std_logic
Only these allowed in VHDL simulator
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
12

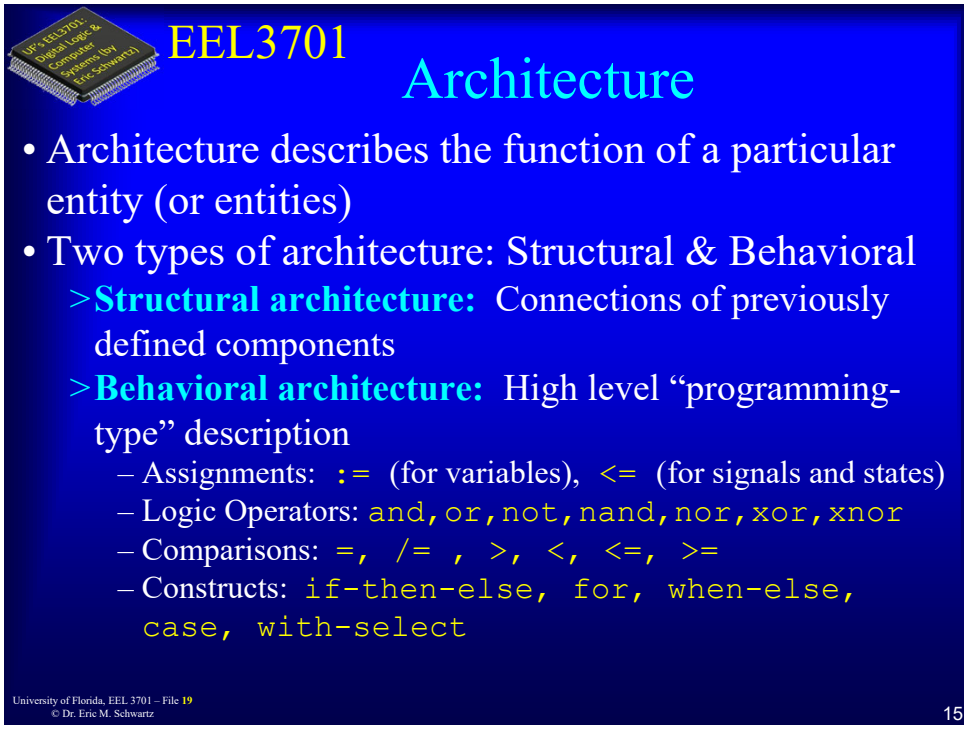
12


EEL3701
VHDL: IEEE 1164 TYPE
• 'L' (weak '0') means floats 'L'; but an input can easily drive it to high; think "*pull-down*"
• 'H' (weak '1') means floats 'H'; but an input can easily drive it to low; think "*pull-up*"
• The type **std_logic** is a "resolved version" of **std_ulogic**, *i.e.,* a decision is made when a signal is driven from multiple sources
• Do **not** use **std_ulogic** and **std_ulogic_vector**
• **std_logic_vector** is a grouping of std_logic (just as bit_vector is a grouping of bit)
• Must use **std_logic** and **std_logic_vector** if need more than the two values (0 and 1 = low and high, respectively) available with **bit** and **bit_vector**
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
13

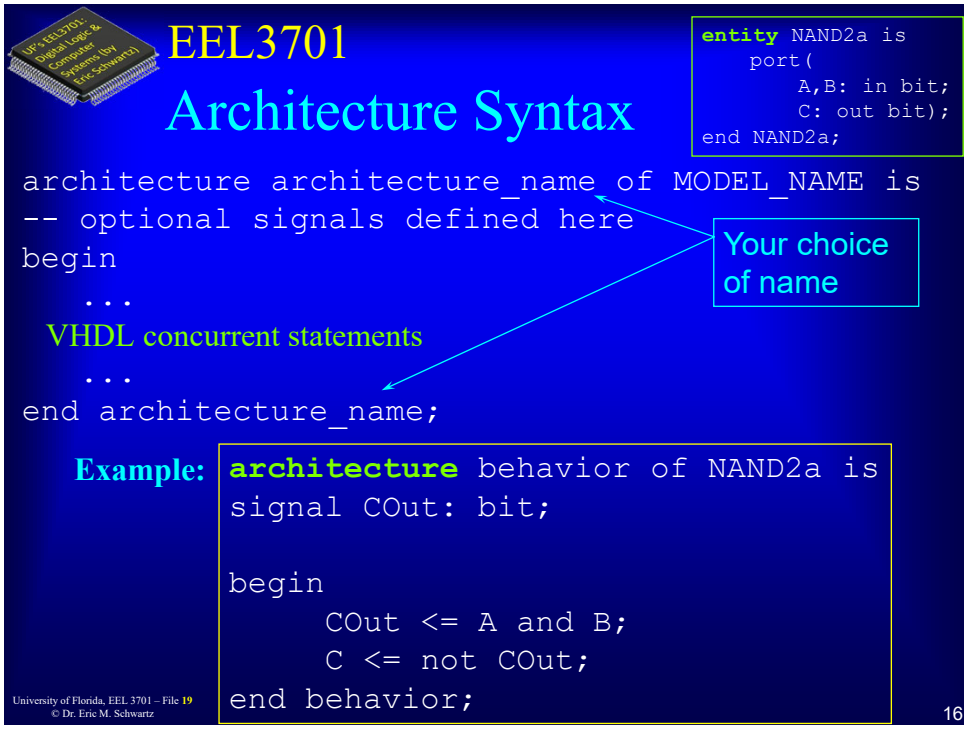
13


EEL3701
MEMORY_DEV Entity Example
• Synchronous circuit, *i.e.,* Clk input
• AD (AD0-AD7), tri-state bi-directional
• D (D8-D15), output only
• AS, output also used internally
• OE, input only
Clk
OE
AD7:0
D15:8
AS
library ieee;
use ieee.std_logic_1164.all;
entity MEMORY_DEV is port(
Clk,OE: in std_logic;
AD: inout std_logic_vector(7 downto 0);
D: out std_logic_vector(15 downto 8);
AS: buffer std_logic);
end MEMORY_DEV;
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
14

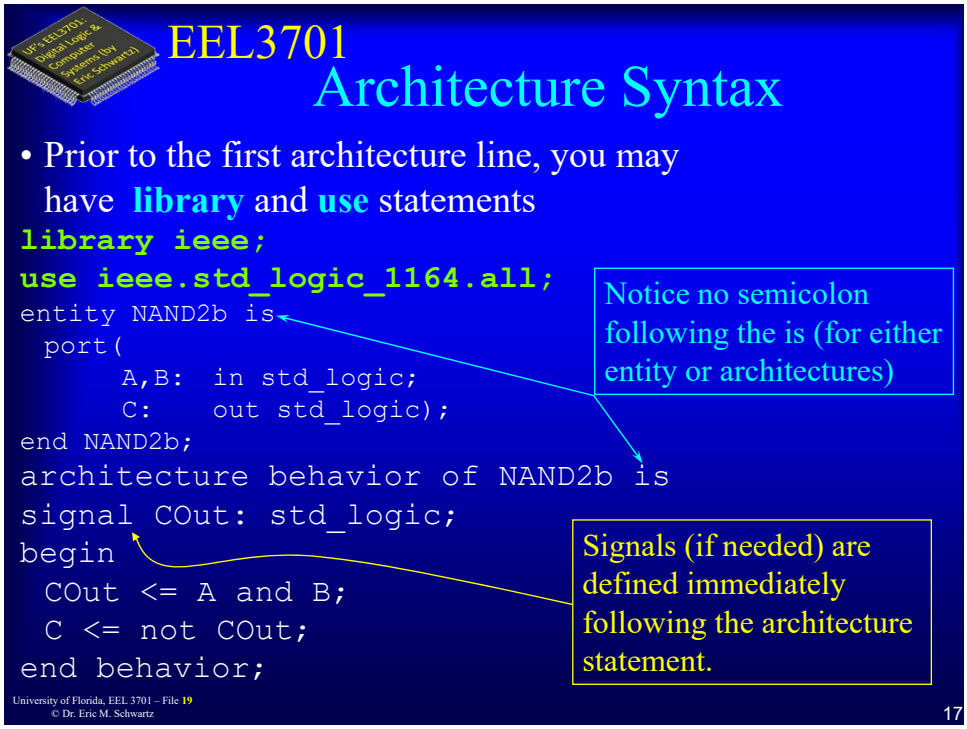
14


EEL3701
Architecture
• Architecture describes the function of a particular entity (or entities)
• Two types of architecture: Structural & Behavioral
> Structural architecture: Connections of previously defined components
> Behavioral architecture: High level "programming-type" description
– Assignments: := (for variables), <= (for signals and states)
– Logic Operators: and,or,not,nand,nor,xor,xnor
– Comparisons: =, /= , >, <, <=, >=
– Constructs: if-then-else, for, when-else, case, with-select
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
15

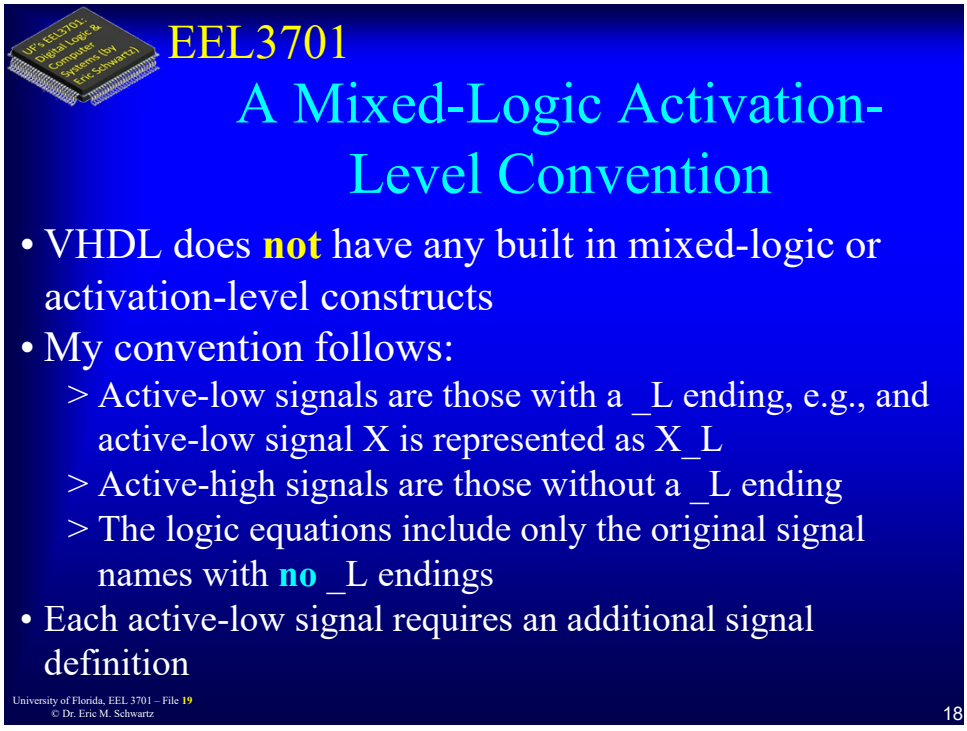
15


EEL3701
Architecture Syntax
entity NAND2a is
port(
A,B: in bit;
C: out bit);
end NAND2a;
architecture architecture_name of MODEL_NAME is
-- optional signals defined here
begin
...
VHDL concurrent statements
...
end architecture_name;
Your choice of name
Example:
architecture behavior of NAND2a is
signal COut: bit;
begin
COut <= A and B;
C <= not COut;
end behavior;
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
16

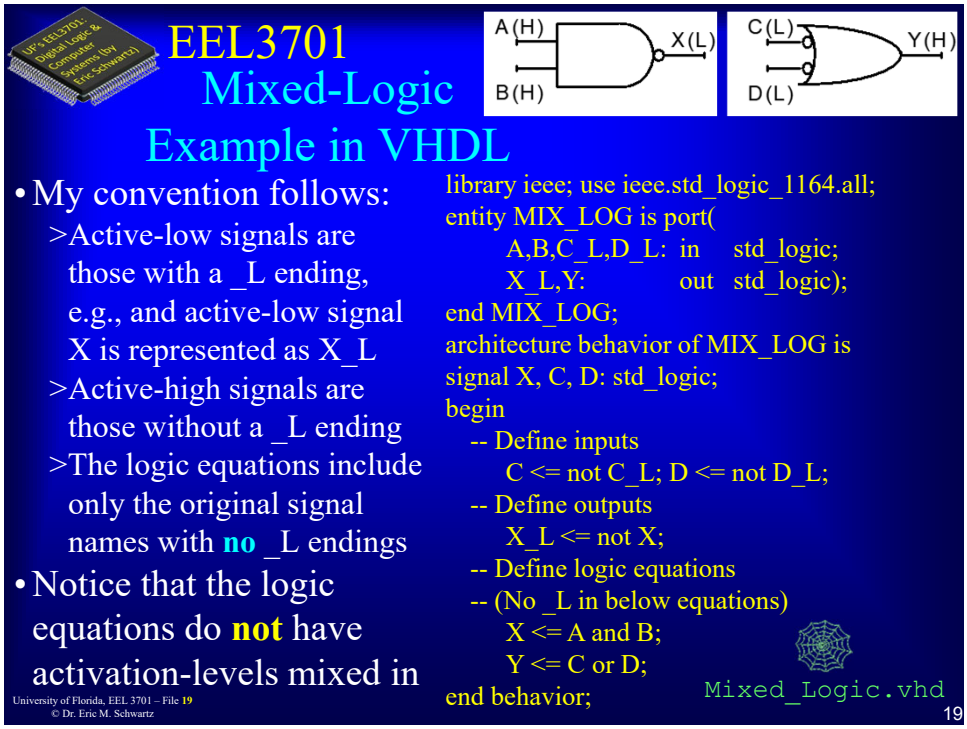
16


EEL3701
Architecture Syntax
• Prior to the first architecture line, you may have library and use statements
library ieee;
use ieee.std_logic_1164.all;
entity NAND2b is
port(
A,B: in std_logic;
C: out std_logic);
end NAND2b;
architecture behavior of NAND2b is
signal COut: std_logic;
begin
COut <= A and B;
C <= not COut;
end behavior;
Notice no semicolon following the is (for either entity or architectures)
Signals (if needed) are defined immediately following the architecture statement.
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
17

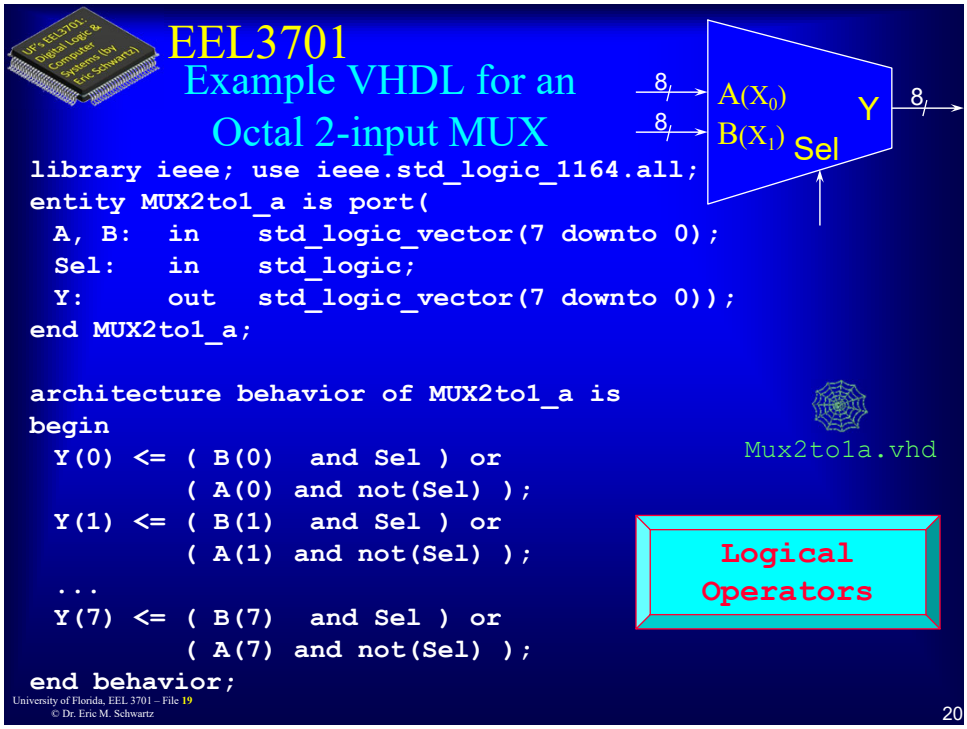
17


EEL3701
A Mixed-Logic Activation-Level Convention
• VHDL does not have any built in mixed-logic or activation-level constructs
• My convention follows:
> Active-low signals are those with a _L ending, e.g., and active-low signal X is represented as X_L
> Active-high signals are those without a _L ending
> The logic equations include only the original signal names with no _L endings
• Each active-low signal requires an additional signal definition
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
18

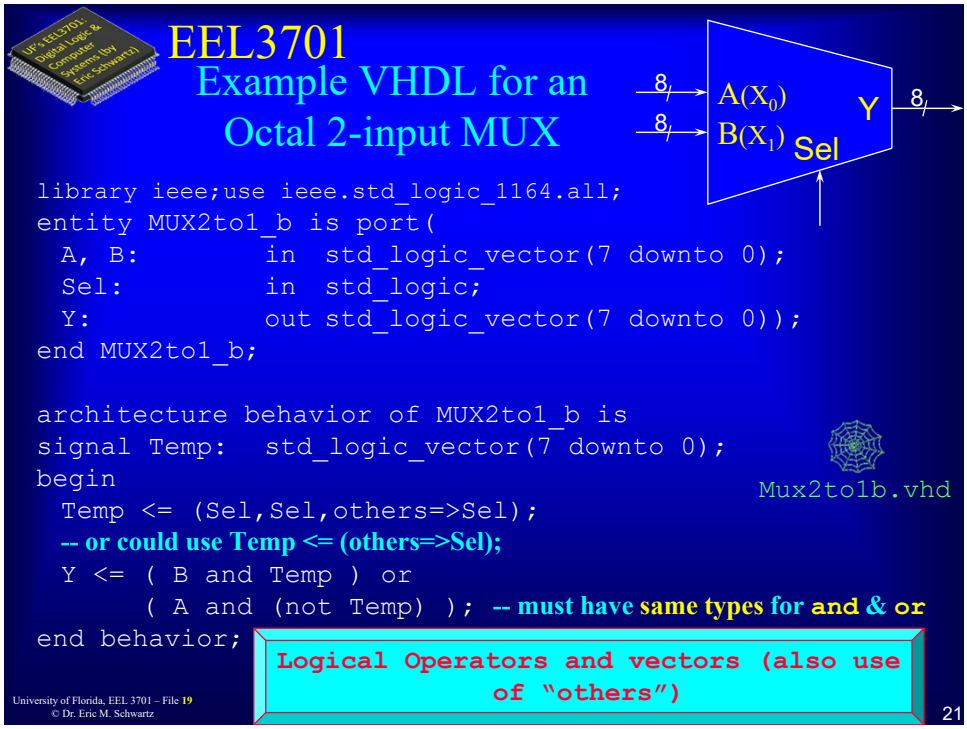
18


EEL3701
Mixed-Logic
Example in VHDL
A(H)
B(H)
X(L)
C(L)
D(L)
Y(H)
• My convention follows:
>Active-low signals are those with a _L ending, e.g., and active-low signal X is represented as X_L
>Active-high signals are those without a _L ending
>The logic equations include only the original signal names with no _L endings
• Notice that the logic equations do not have activation-levels mixed in
library ieee; use ieee.std_logic_1164.all;
entity MIX_LOG is port(
A,B,C_L,D_L: in std_logic;
X_L,Y: out std_logic);
end MIX_LOG;
architecture behavior of MIX_LOG is
signal X, C, D: std_logic;
begin
-- Define inputs
C <= not C_L; D <= not D_L;
-- Define outputs
X_L <= not X;
-- Define logic equations
-- (No _L in below equations)
X <= A and B;
Y <= C or D;
end behavior;
Mixed_Logic.vhd
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
19

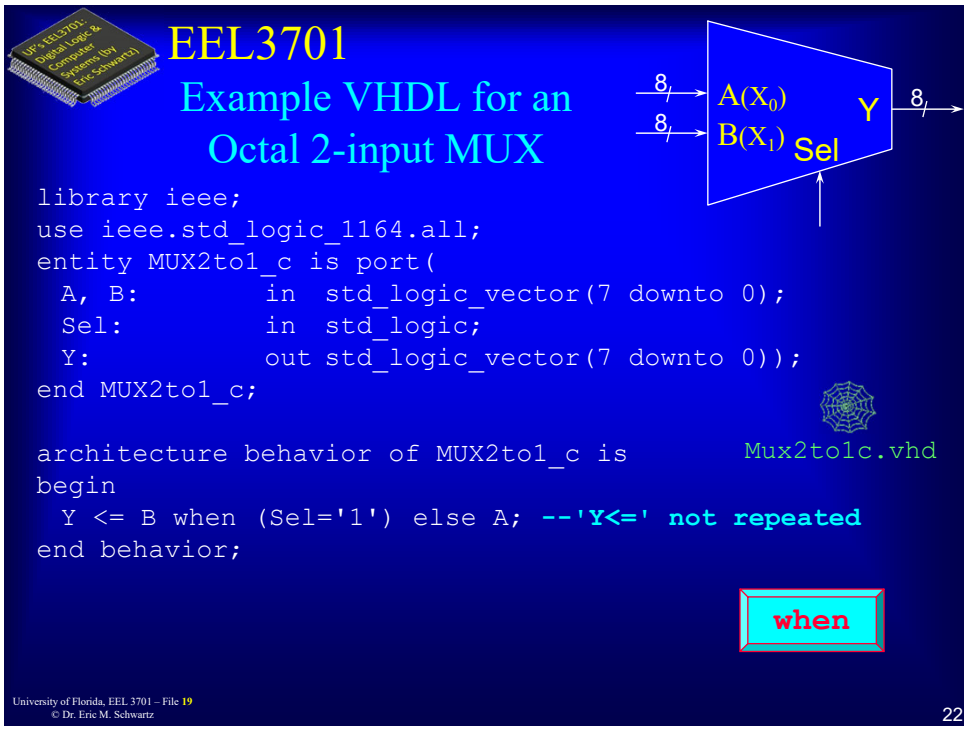
19


EEL3701
Example VHDL for an
Octal 2-input MUX
8
8
A(X0)
B(X1)
Sel
Y
8
library ieee; use ieee.std_logic_1164.all;
entity MUX2to1_a is port(
A, B: in std_logic_vector(7 downto 0);
Sel: in std_logic;
Y: out std_logic_vector(7 downto 0));
end MUX2to1_a;
architecture behavior of MUX2to1_a is
begin
Y(0) <= (B(0) and Sel) or
(A(0) and not(Sel));
Y(1) <= (B(1) and Sel) or
(A(1) and not(Sel));
...
Y(7) <= (B(7) and Sel) or
(A(7) and not(Sel));
end behavior;
Mux2to1a.vhd
Logical Operators
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
20

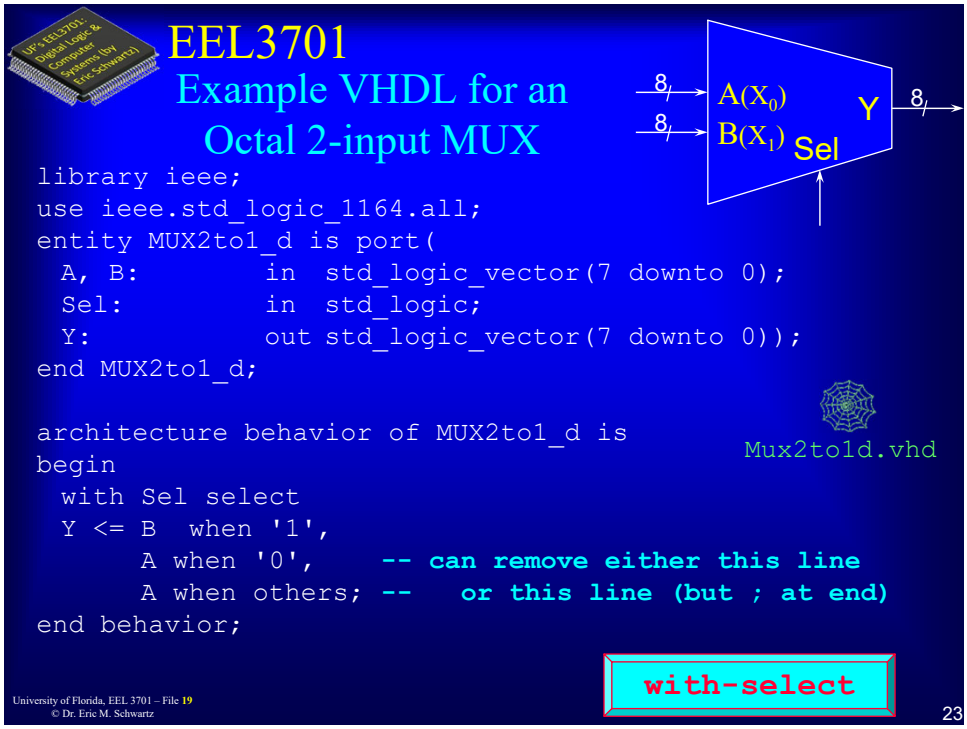
20


EEL3701
Example VHDL for an
Octal 2-input MUX
8
8
A(X0)
B(X1)
Sel
Y
8
library ieee;use ieee.std_logic_1164.all;
entity MUX2to1_b is port(
A, B: in std_logic_vector(7 downto 0);
Sel: in std_logic;
Y: out std_logic_vector(7 downto 0));
end MUX2to1_b;
architecture behavior of MUX2to1_b is
signal Temp: std_logic_vector(7 downto 0);
begin
Temp <= (Sel,Sel,others=>Sel);
-- or could use Temp <= (others=>Sel);
Y <= (B and Temp) or
(A and (not Temp)); -- must have same types for and & or
end behavior;
Mux2to1b.vhd
Logical Operators and vectors (also use of "others")
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
21

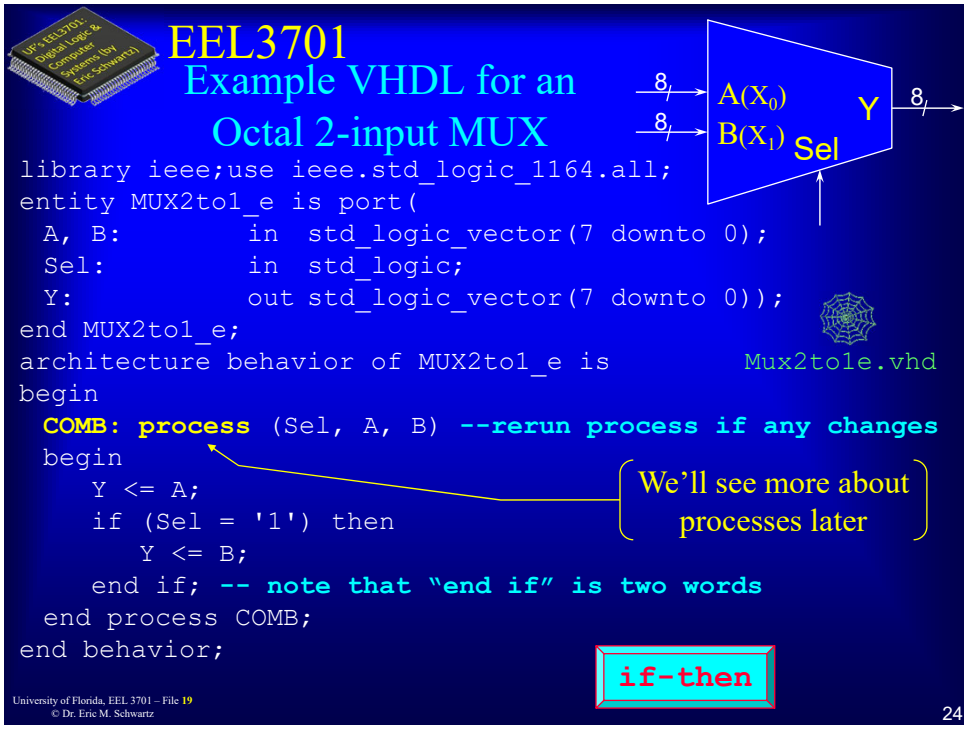
21


EEL3701
Example VHDL for an
Octal 2-input MUX
8
8
A(X0)
B(X1)
Sel
Y
8
library ieee;
use ieee.std_logic_1164.all;
entity MUX2to1_c is port(
A, B: in std_logic_vector(7 downto 0);
Sel: in std_logic;
Y: out std_logic_vector(7 downto 0));
end MUX2to1_c;
architecture behavior of MUX2to1_c is
begin
Y <= B when (Sel='1') else A; --'Y<=' not repeated
end behavior;
Mux2to1c.vhd
when
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
22

22


EEL3701
Example VHDL for an
Octal 2-input MUX
8
8
A(X0)
B(X1)
Sel
Y
8
library ieee;
use ieee.std_logic_1164.all;
entity MUX2to1_d is port(
A, B: in std_logic_vector(7 downto 0);
Sel: in std_logic;
Y: out std_logic_vector(7 downto 0));
end MUX2to1_d;
architecture behavior of MUX2to1_d is
begin
with Sel select
Y <= B when '1',
A when '0', -- can remove either this line
A when others; -- or this line (but ; at end)
end behavior;
Mux2to1d.vhd
with-select
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
23

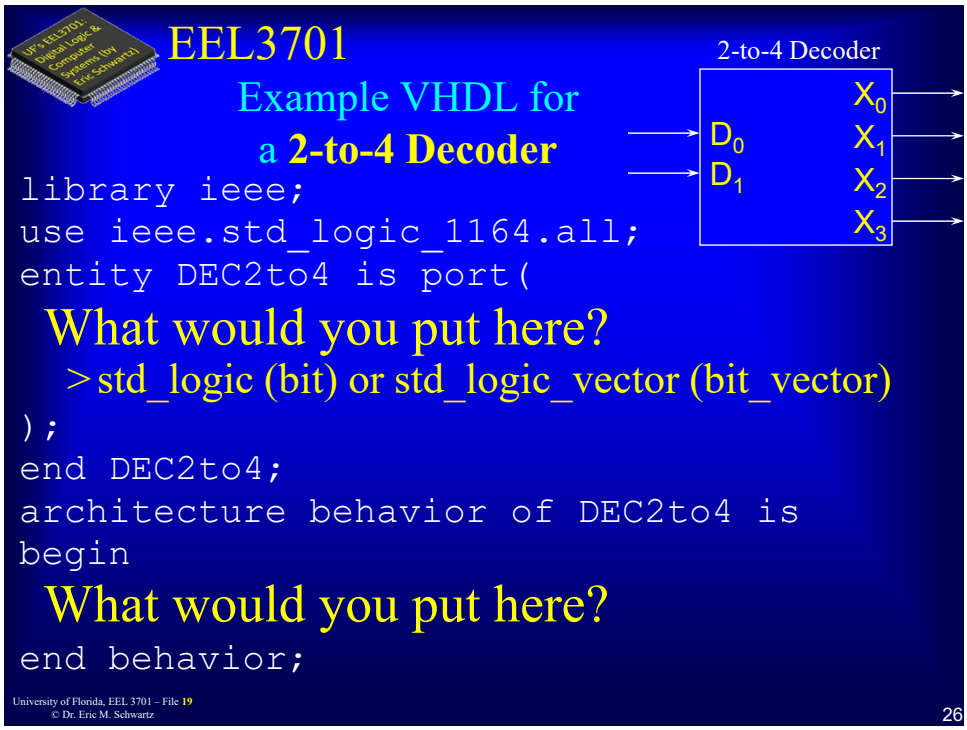
23


EEL3701
Example VHDL for an
Octal 2-input MUX
8
8
A(X0)
B(X1)
Sel
Y
8
library ieee;use ieee.std_logic_1164.all;
entity MUX2to1_e is port(
A, B: in std_logic_vector(7 downto 0);
Sel: in std_logic;
Y: out std_logic_vector(7 downto 0));
end MUX2to1_e;
architecture behavior of MUX2to1_e is
begin
COMB: process (Sel, A, B) --rerun process if any changes
begin
Y <= A;
if (Sel = '1') then
Y <= B;
end if; -- note that "end if" is two words
end process COMB;
end behavior;
Mux2to1e.vhd
We'll see more about processes later
if-then
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
24

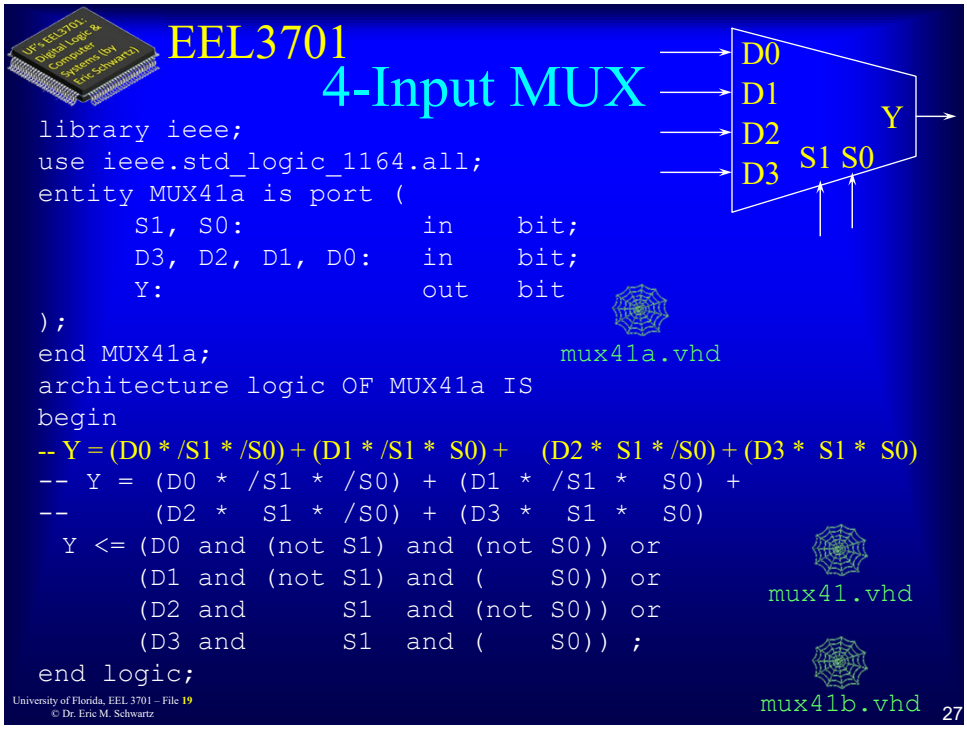
24


EEL3701
Example VHDL for an
Octal 2-input MUX
8
8
A(X0)
B(X1)
Sel
Y
8
library ieee;use ieee.std_logic_1164.all;
entity MUX2to1_e is port(
A, B: in std_logic_vector(7 downto 0);
Sel: in std_logic;
Y: out std_logic_vector(7 downto 0));
end MUX2to1_e;
architecture behavior of MUX2to1_e is
begin
COMB: process (Sel, A, B) --rerun process if any changes
begin
if (Sel = '1') then
Y <= B;
else
Y <= A;
end if; -- note that "end if" is two words
end process COMB;
end behavior;
Mux2to1f.vhd
if-then-else
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
25

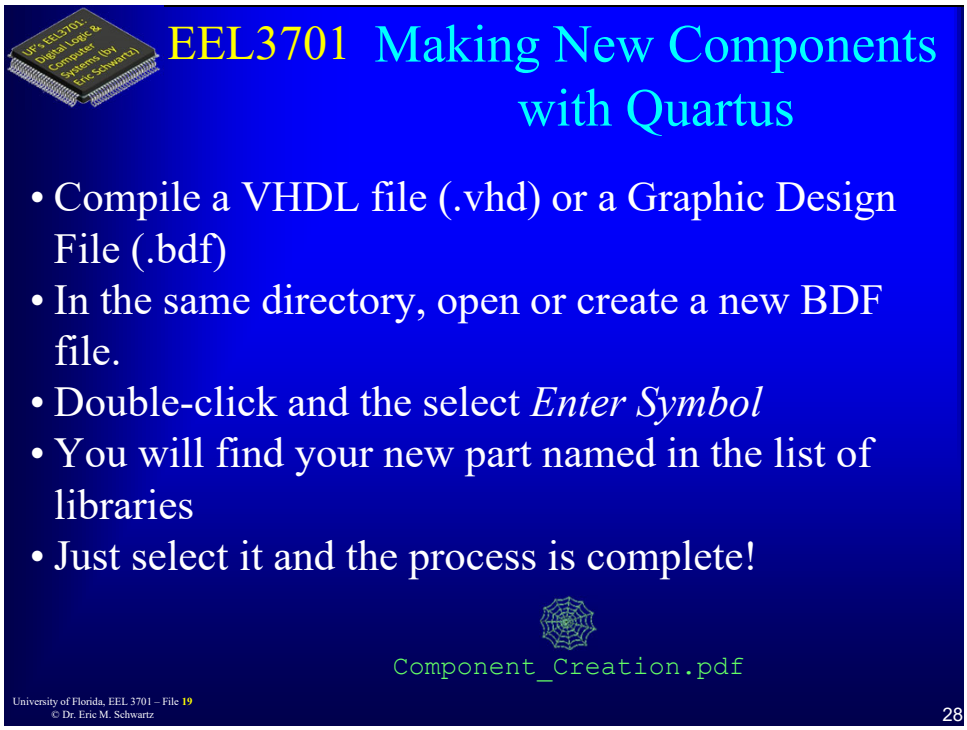
25


EEL3701
Example VHDL for
a 2-to-4 Decoder
2-to-4 Decoder
D0
D1
X0
X1
X2
X3
library ieee;
use ieee.std_logic_1164.all;
entity DEC2to4 is port(
What would you put here?
>std_logic (bit) or std_logic_vector (bit_vector)
);
end DEC2to4;
architecture behavior of DEC2to4 is
begin
What would you put here?
end behavior;
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
26

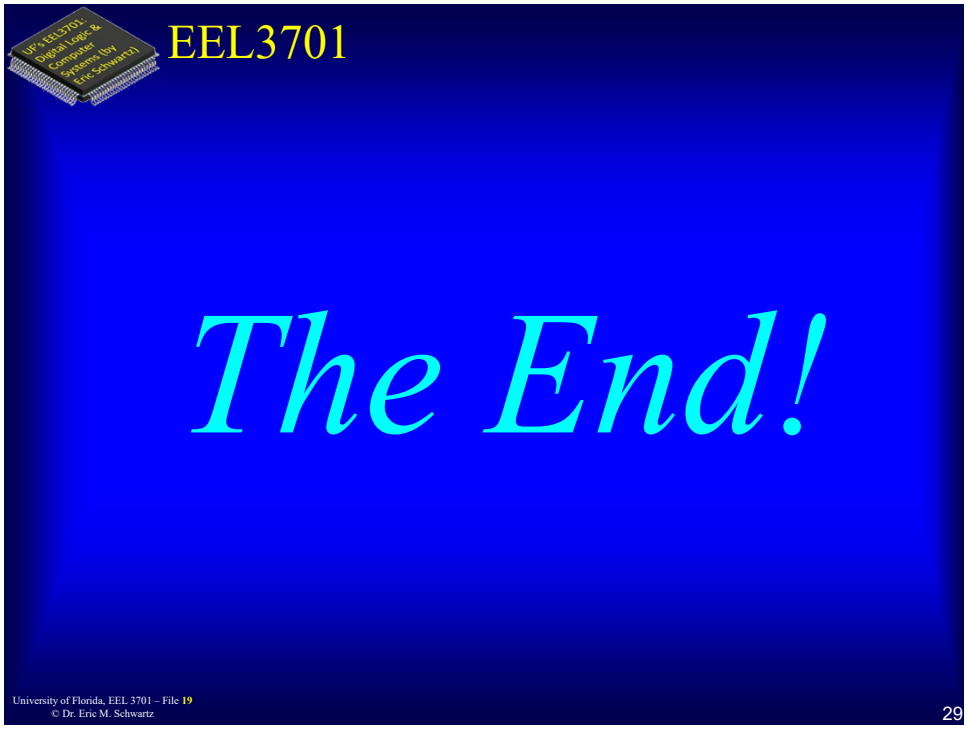
26


EEL3701
4-Input MUX
D0
D1
D2
D3
S1 S0
Y
library ieee;
use ieee.std_logic_1164.all;
entity MUX41a is port (
S1, S0: in bit;
D3, D2, D1, D0: in bit;
Y: out bit
);
end MUX41a;
architecture logic OF MUX41a IS
begin
-- Y = (D0 * /S1 * /S0) + (D1 * /S1 * S0) + (D2 * S1 * /S0) + (D3 * S1 * S0)
-- Y = (D0 * /S1 * /S0) + (D1 * /S1 * S0) +
-- (D2 * S1 * /S0) + (D3 * S1 * S0)
Y <= (D0 and (not S1) and (not S0)) or
(D1 and (not S1) and (S0)) or
(D2 and S1 and (not S0)) or
(D3 and S1 and (S0)) ;
end logic;
mux41a.vhd
mux41.vhd
mux41b.vhd
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
27

27


EEL3701 Making New Components with Quartus
• Compile a VHDL file (.vhd) or a Graphic Design File (.bdf)
• In the same directory, open or create a new BDF file.
• Double-click and the select *Enter Symbol*
• You will find your new part named in the list of libraries
• Just select it and the process is complete!
Component_Creation.pdf
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
28

28


EEL3701
The End!
University of Florida, EEL 3701 – File 19
© Dr. Eric M. Schwartz
29

29