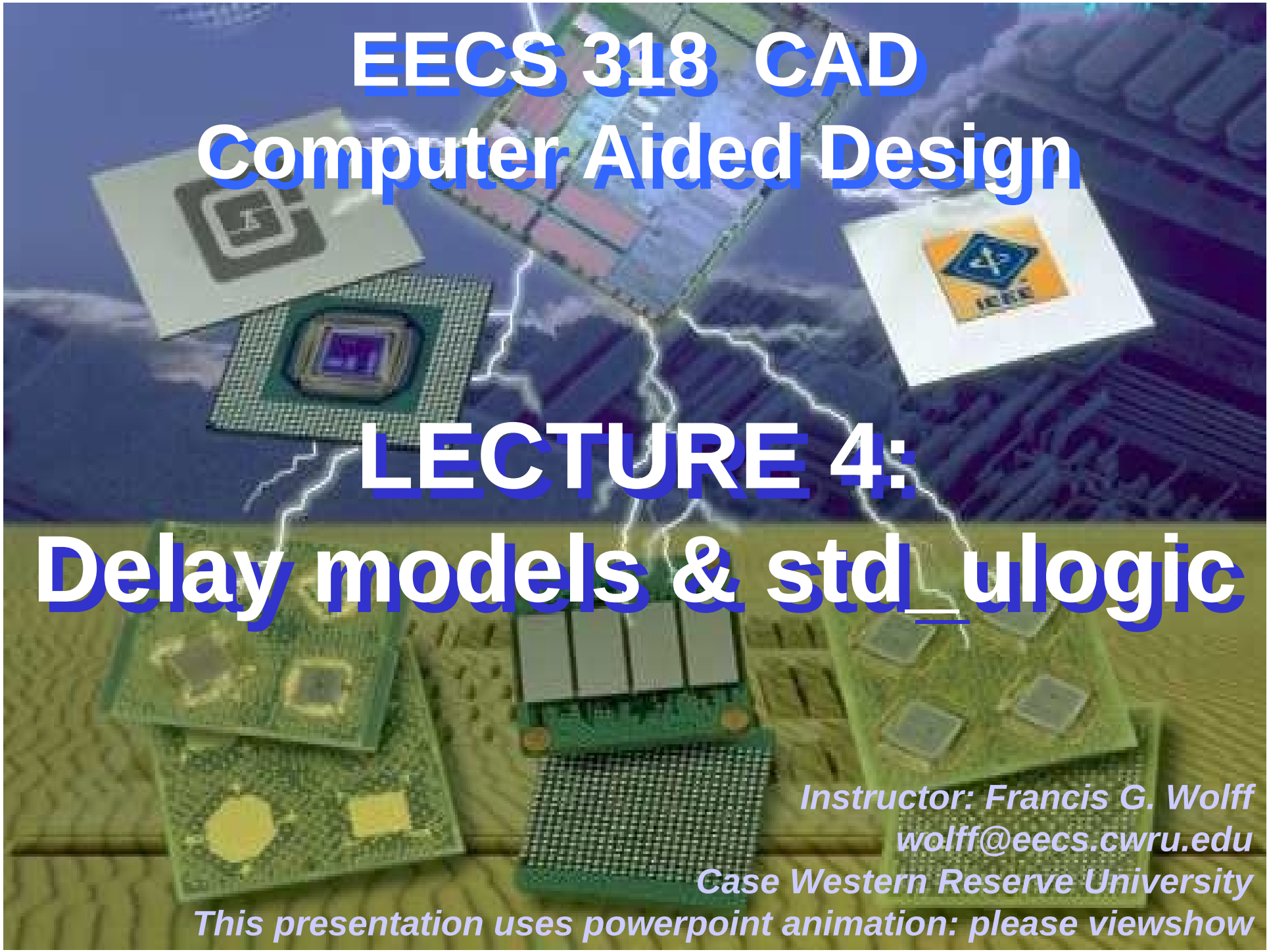


EECS 318 CAD
Computer Aided Design

LECTURE 4:
Delay models & std_ulogic

Instructor: Francis G. Wolff
wolff@eecs.cwru.edu
Case Western Reserve University

This presentation uses powerpoint animation: please viewshow

Delta Delay

- Default signal assignment propagation delay if no delay is explicitly prescribed
  - VHDL signal assignments do not take place immediately
  - Delta is an infinitesimal VHDL time unit so that all signal assignments can result in signals assuming their values at a future time
  - E.g.

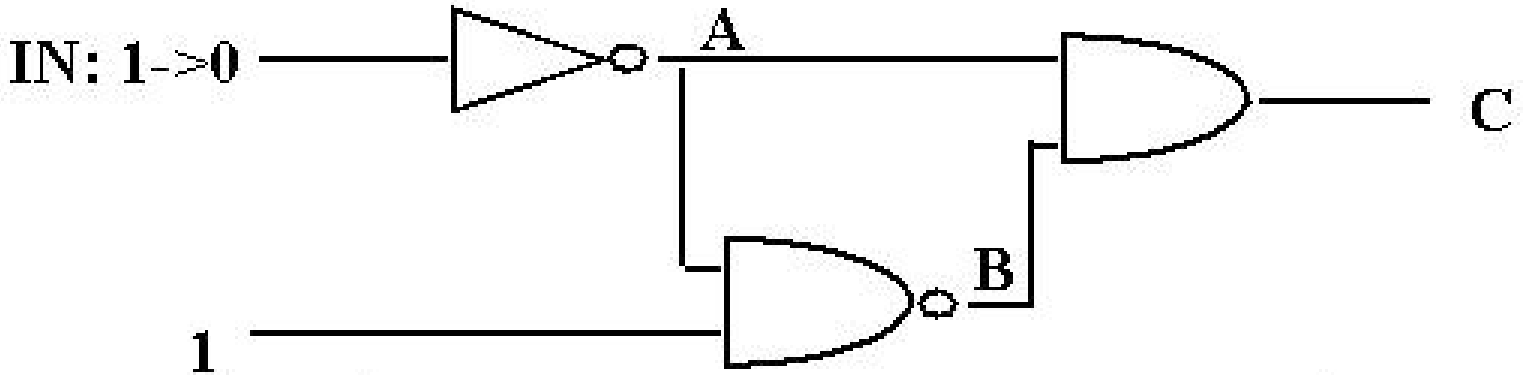
```
Output <= NOT Input;
-- Output assumes new value in one delta cycle
```

- Supports a model of concurrent VHDL process execution
  - Order in which processes are executed by simulator does not affect simulation output

Delta Delay

An Example with Delta Delay

- What is the behavior of C?



Using delta delay scheduling

Time	Delta	Event
0 ns	1	IN: 1->0 eval INVERTER
	2	A: 0->1 eval NAND, AND
	3	B: 1->0 C: 0->1 eval AND
	4	C: 1->0
1 ns		

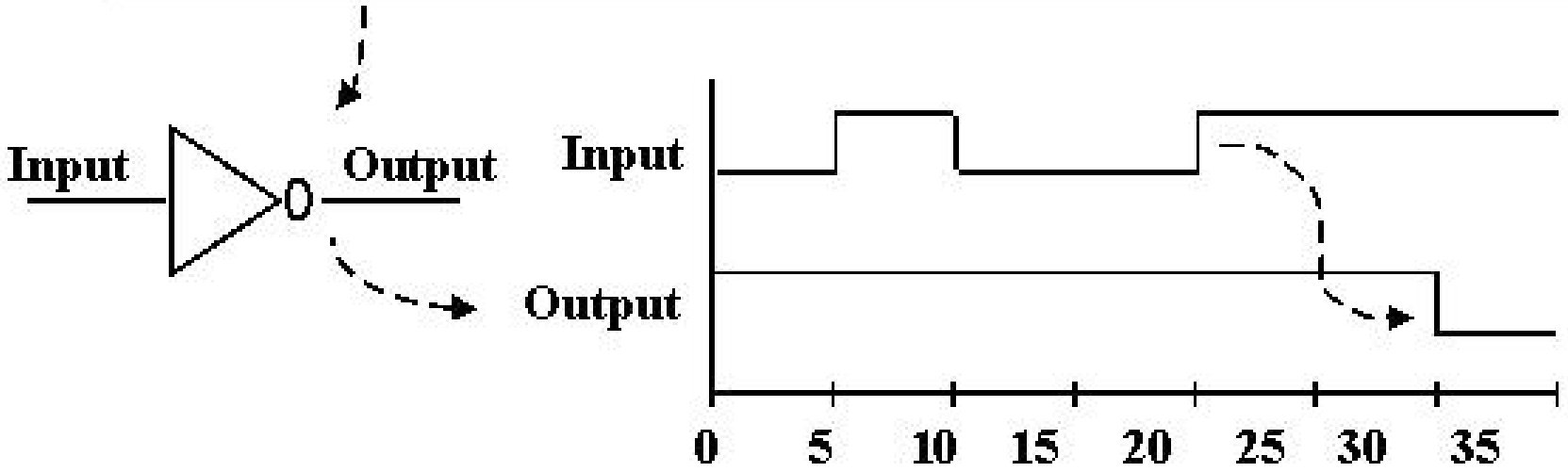
Inertial Delay

- **Provides for specification propagation delay and input pulse width, i.e. 'inertia' of output:**

```
target <= [REJECT time_expression] INERTIAL waveform;
```

- **Inertial delay is default and REJECT is optional :**

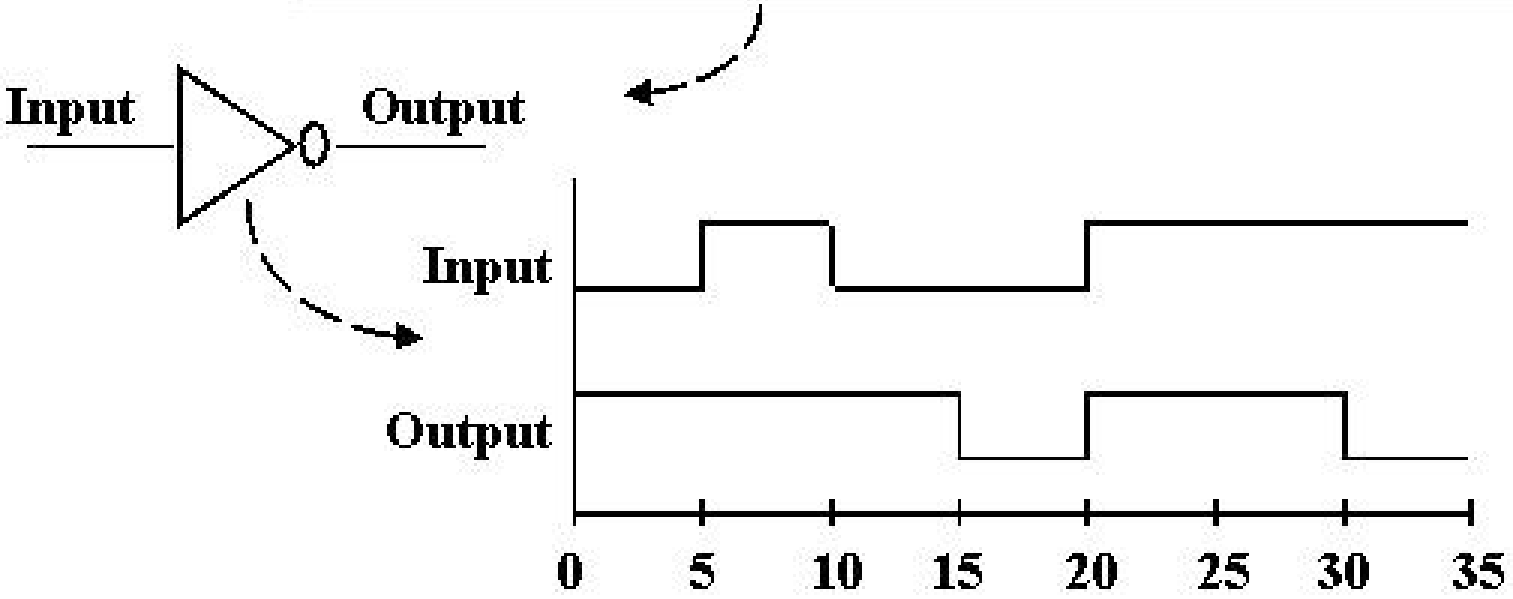
```
Output <= NOT Input AFTER 10 ns;
-- Propagation delay and minimum pulse width are 10ns
```



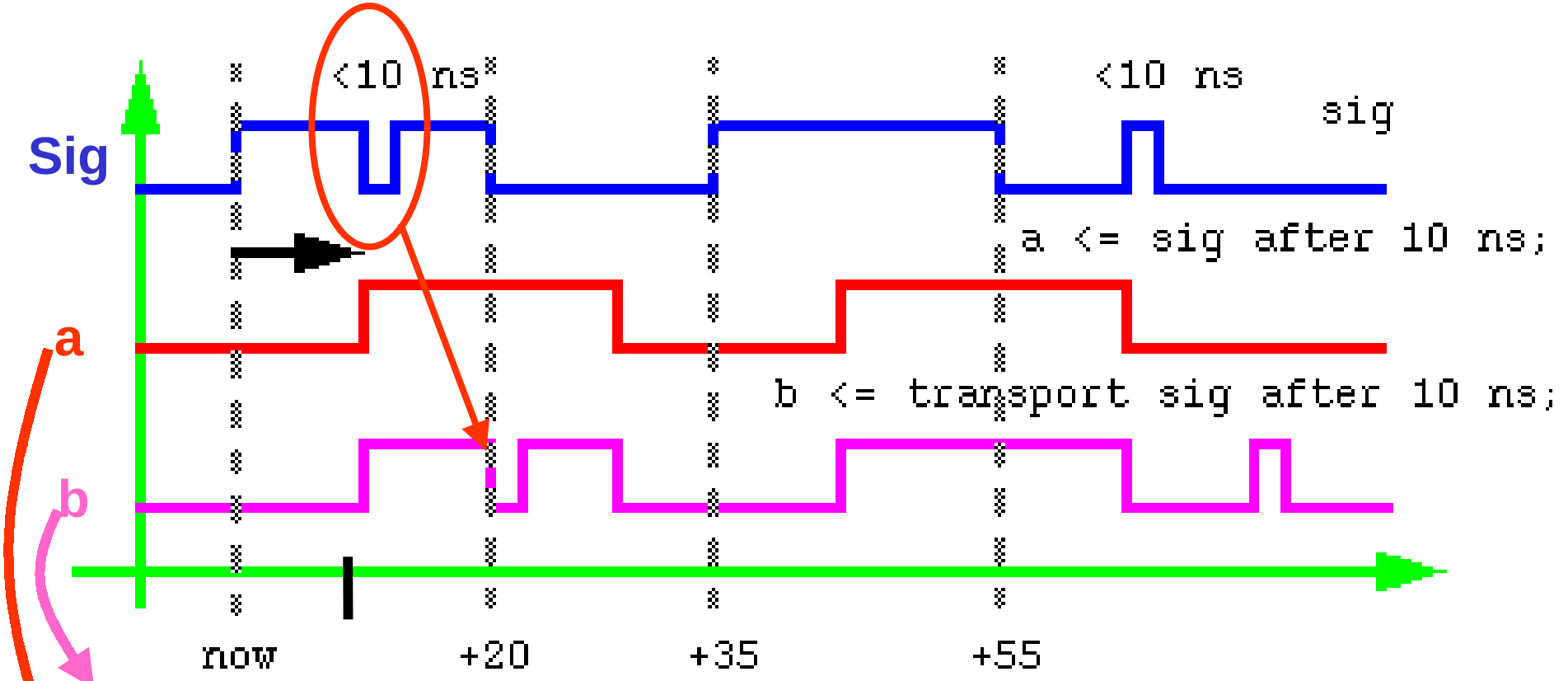
Transport Delay

- Transport delay must be explicitly specified
  - I.e. keyword "TRANSPORT" must be used
- Signal will assume its new value after specified delay

```
-- TRANSPORT delay example
Output <= TRANSPORT NOT Input AFTER 10 ns;
```



Inertial and Transport Delay



Transport Delay is useful for modeling data buses, networks

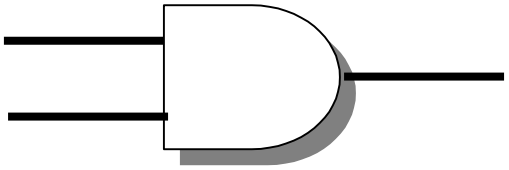
Inertial Delay is useful for modeling logic gates

Combinatorial Logic Operators

#Transistors		
2	**NOT**	**z <= NOT (x); z<= NOT x;**
2+2i	**AND**	**z <= x AND y;**
2i	**NAND**	**z <= NOT (x AND y);**
2+2i	**OR**	**z <= x OR y;**
2i	**NOR**	**z <= NOT (x OR Y);**
10	**XOR**	**z <= (x and NOT y) OR (NOT x AND y);** *z <= (x AND y) NOR (x NOR y); --AOI*
10	**XNOR**	**z <= (x and y) OR (NOT x AND NOT y);** *z <= (x NAND y) NAND (x OR y); --OAI*

Footnote: (i=#inputs) *We are only referring to CMOS static transistor ASIC gate designs*
Exotic XOR designs can be done in 6 (J. W. Wang, IEEE J. Solid State Circuits, 29, July 1994)

Std_logic AND: Un-initialized value



NOT	0	1	U
	1	0	U

AND	0	1	U
0	0	0	0
1	0	1	U
U	0	U	U

OR	0	1	U
0	0	1	U
1	1	1	1
U	U	1	U

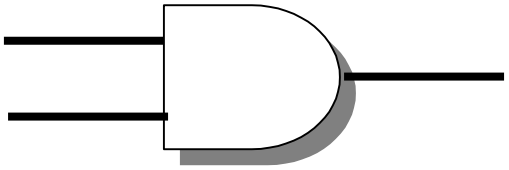
0 AND <anything> is 0

0 NAND <anything> is 1

1 OR <anything> is 1

1 NOR <anything> is 0

Std_logic AND: X Forcing Unknown Value



NOT	0	X	1	U
	1	X	0	U

AND	0	X	1	U
0	0	0	0	0
X	0	X	X	U
1	0	X	1	U
U	0	U	U	U

OR	0	X	1	U
0	0	X	1	U
X	X	X	1	U
1	1	1	1	1
U	U	U	1	U

0 AND <anything> is 0

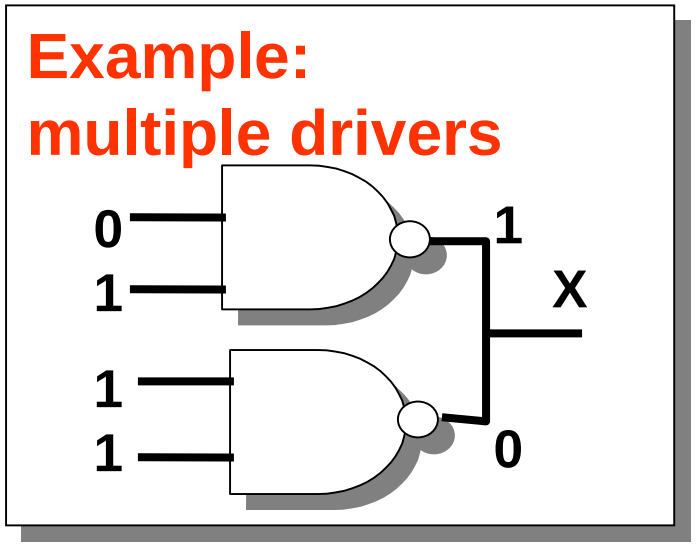
0 NAND <anything> is 1

1 OR <anything> is 0

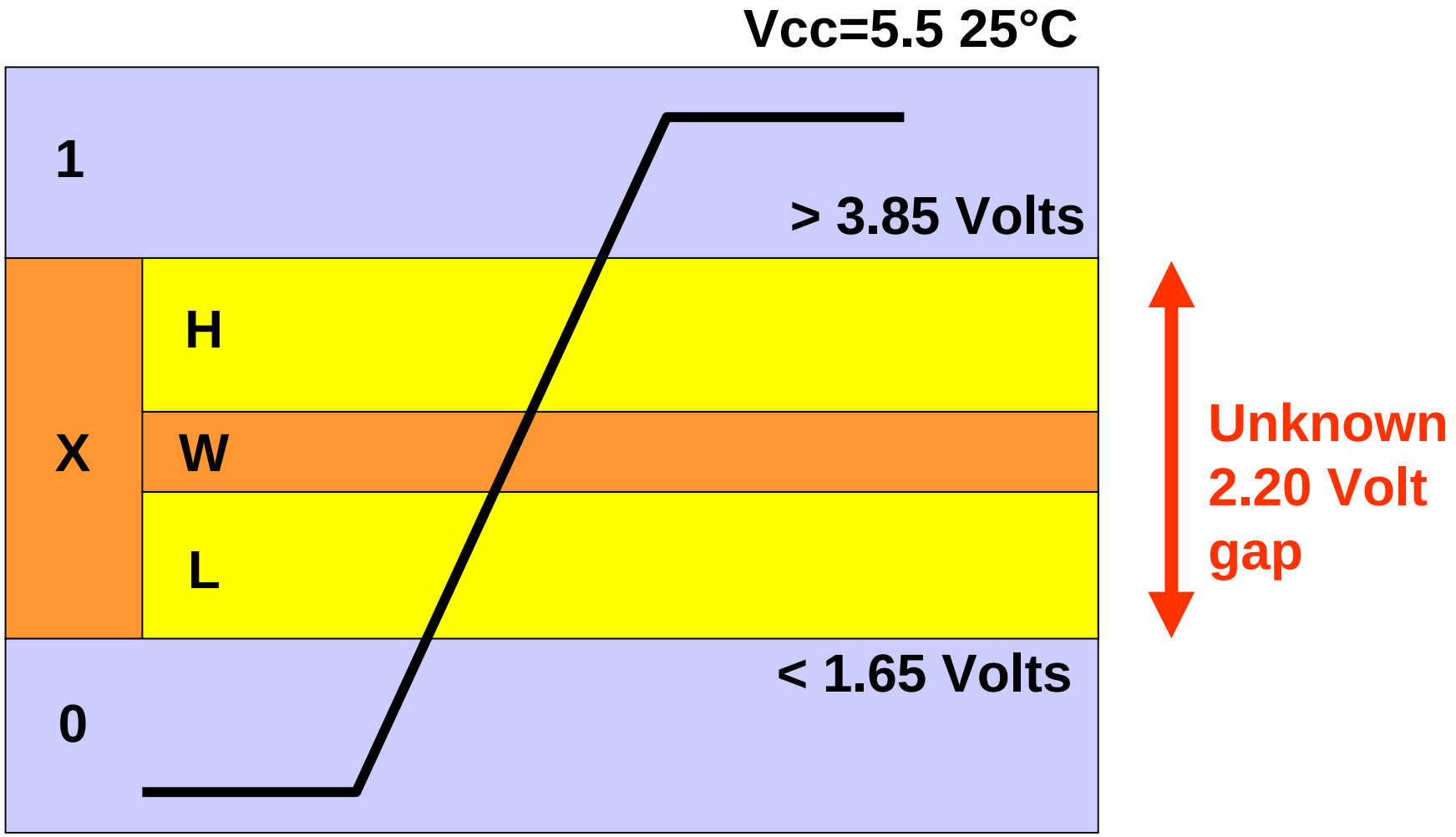
0 NOR <anything> is 1

Modeling logic gate values: std_ulogic

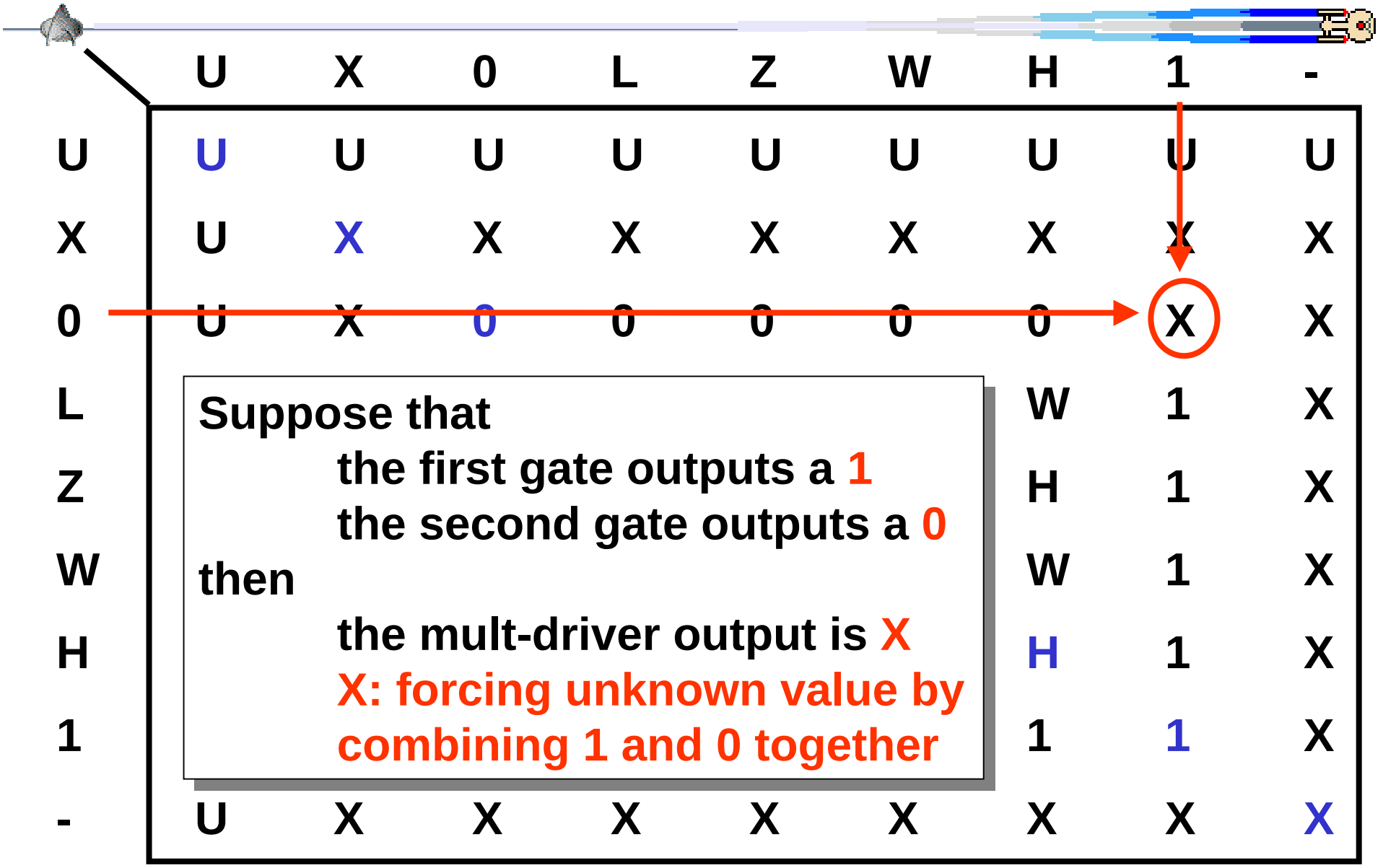
```
TYPE std_ulogic IS ( -- Unresolved LOGIC
        'Z',    -- High Impedance (Tri-State)
        '1',    -- Forcing 1
        'H',    -- Weak 1
        'X',    -- Forcing Unknown: i.e. combining 0 and 1
        'W',    -- Weak Unknown: i.e. combining H and L
        'L',    -- Weak 0
        '0',    -- Forcing 0
        'U',    -- Un-initialized
        '-',    -- Don't care
);
```



The rising transition signal



Multiple output drivers: Resolution Function



	U	X	0	L	Z	W	H	1	-
U	U	U	U	U	U	U	U	U	U
X	U	X	X	X	X	X	X	X	X
0	U	X	0	0	0	0	0	X	X
L							W	1	X
Z							H	1	X
W							W	1	X
H							H	1	X
1							1	1	X
-	U	X	X	X	X	X	X	X	X

Suppose that
the first gate outputs a 1
the second gate outputs a 0
then
the mult-driver output is X
X: forcing unknown value by combining 1 and 0 together

Multiple output drivers: Resolution Function

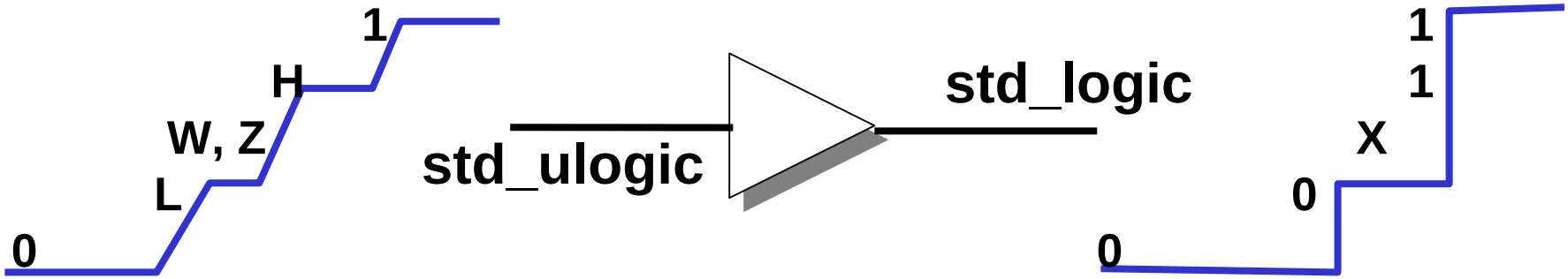
	U	X	0	L	Z	W	H	1	-
U	U	U	U	U	U	U	U	U	U
X		X	X	X	X	X	X	X	X
0			0	0	0	0	0	X	X
L				L	L	W	W	1	X
Z					Z	W	H	1	X
W						W	W	1	X
H							H	1	X
1								1	X
-									X

Observe that 0 pulls down all weak signals to 0

H <driving> L => W

- Note the multi-driver resolution table is symmetrical

Resolution Function: std_logic buffer gate



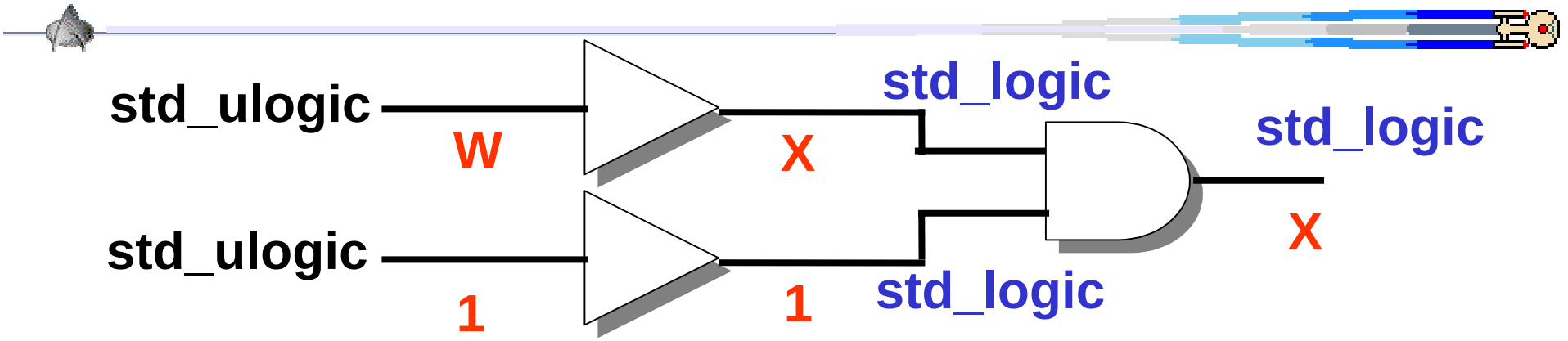
input:	U	0	L	W	X	Z	H	1	-
output:	U	0	0	X	X	X	1	1	X

0 or L becomes 0

H or 1 becomes 1

Transition zone becomes X

Resolving input: std_logic AND GATE



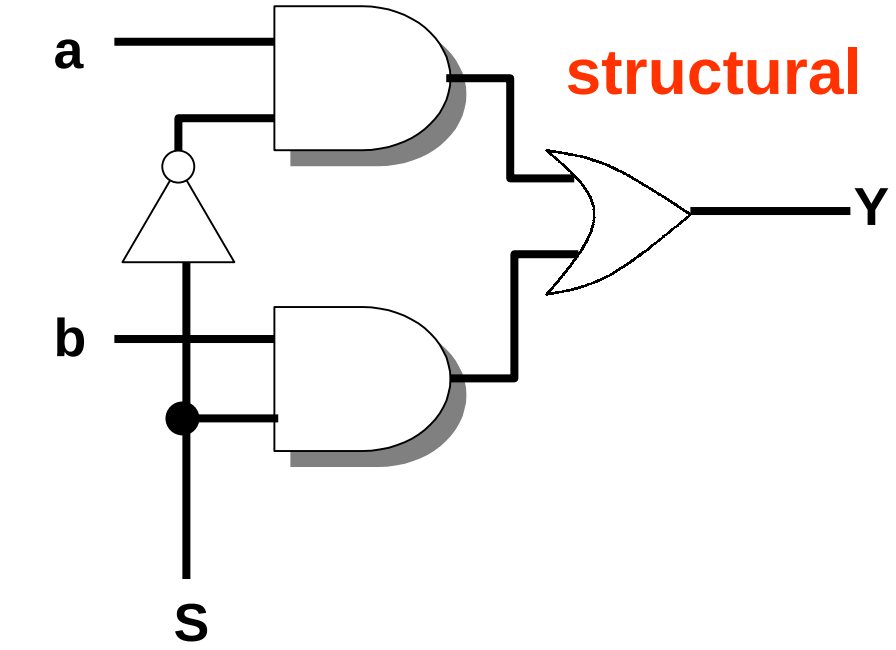
Process each input as an unresolved to resolved buffer.

Then process the gate as a standard logic gate { 0, X, 1, U }

For example, let's transform z <= 'W' AND '1';

```
z <= 'W' AND '1';    -- convert std_ulogic 'W' to std_logic 'X'
z <= 'X' AND '1';    -- now compute the std_logic AND
z <= 'X';
```

2-to-1 Multiplexor: with-select-when



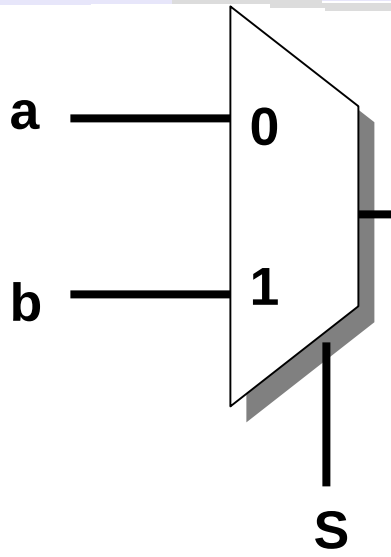
structural

```
Y <= sa OR sb;

sa <= a AND NOT s;

sb <= b AND s;
```

combinatorial logic



behavioral

Only values allowed

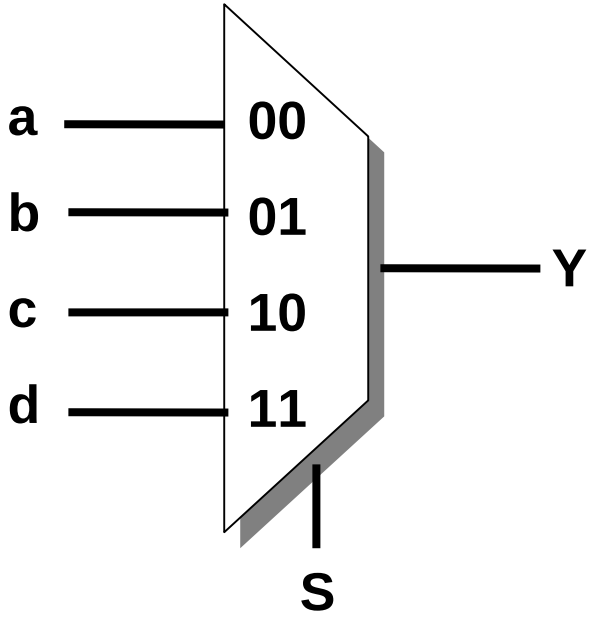
```
WITH s SELECT
     Y <= a WHEN '0',
          b WHEN '1';
```

or alternatively

```
WITH s SELECT
     Y <= a WHEN '0',
          b WHEN OTHERS;
```

4-to-1 Multiplexor: with-select-when

```
Y <= sa OR sb OR sc OR sd;
sa <= a AND ( NOT s(1) AND NOT s(0) );
sb <= b AND ( NOT s(1) AND s(0) );
sc <= c AND ( s(1) AND NOT s(0) );
sd <= d AND ( s(1) AND s(0) );
```



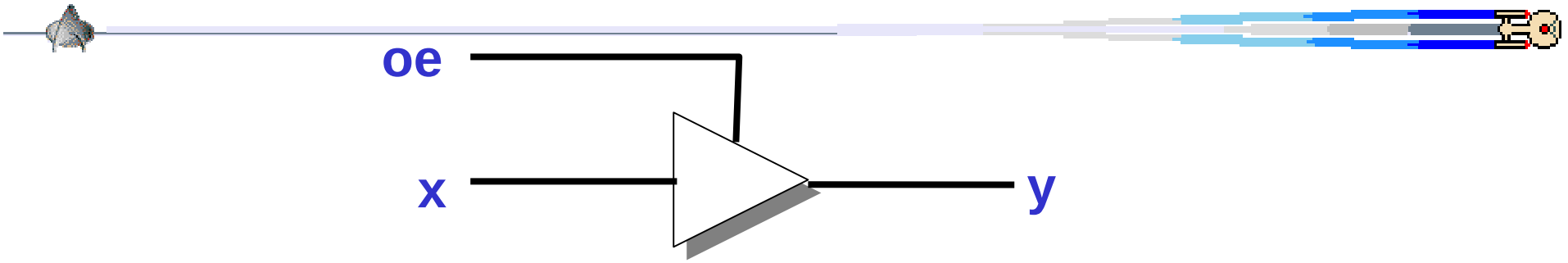
Structural Combinatorial logic

As the complexity of the combinatorial logic grows, the SELECT statement, simplifies logic design but at a loss of structural information

```
WITH s SELECT
  Y <= a WHEN "00",
       b WHEN "01",
       c WHEN "10",
       d WHEN OTHERS;
```

behavioral

Tri-State buffer



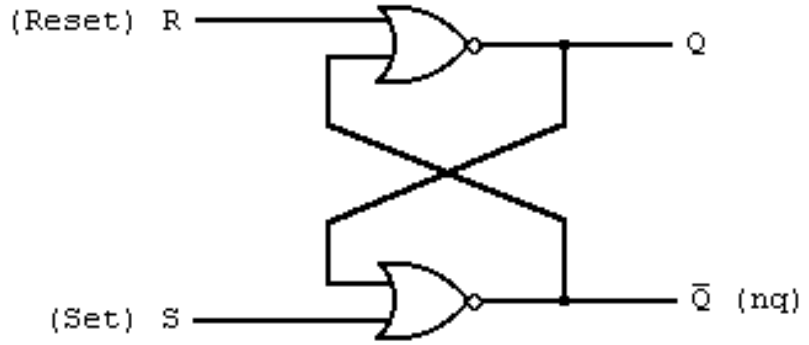

```
ENTITY Buffer_Tri_State IS
        PORT(x:     IN      std_logic;
                y:      OUT     std_logic;
                oe:     IN      std_logic
); END;
```

```
ARCHITECTURE Buffer3 OF Buffer_Tri_State IS
BEGIN

        WITH oe SELECT
        y <=    x WHEN '1',   -- Enabled:  y <= x;
                'Z' WHEN '0'; -- Disabled: output a tri-state

END;
```

Assignment #2 (Part 1 of 3) Due Thurs, 9/14



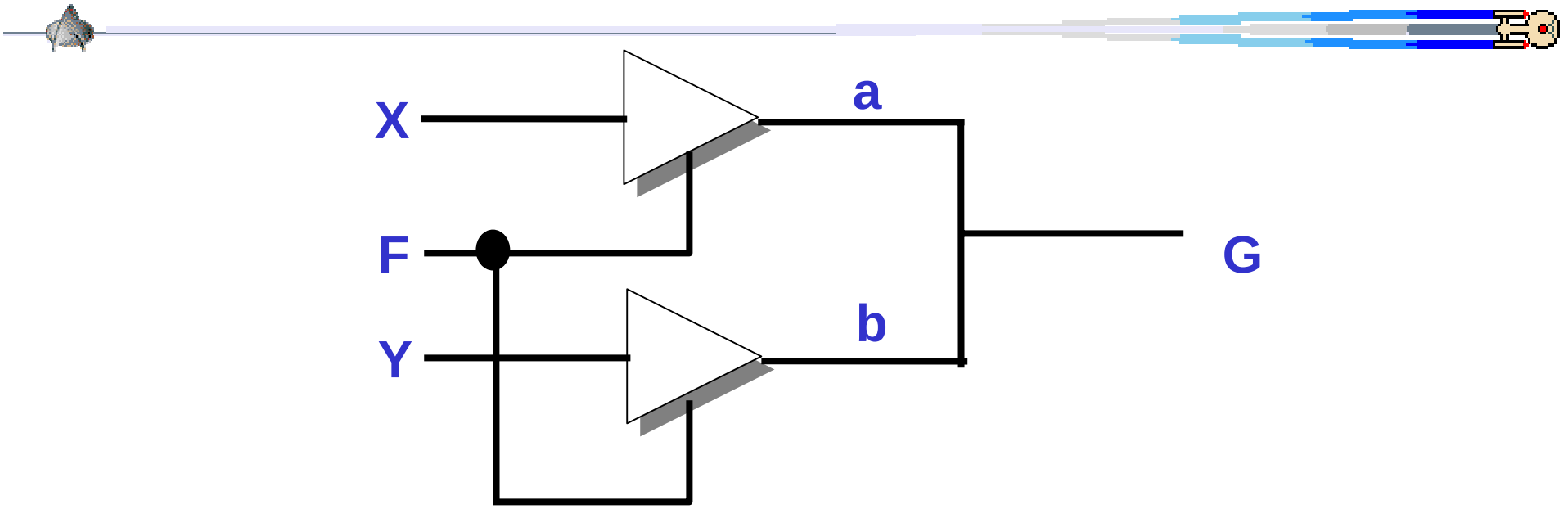
SR Latch

1) Assume each gate is 10 ns delay for the above circuit.

(a) Write entity-architecture for a inertial model
(b) Given the following waveform, draw, R, S, Q, NQ (inertial)
R <= '0', '1' after 25 ns, '0' after 30 ns;
S <= '1', '0' after 20 ns, '1' after 35 ns, '0' after 50 ns;

(c) Write entity-architecture for a transport model
(d) Given the waveform in (b) draw, R, S, Q, NQ (transport)

Assignment #2 (Part 2 of 3)



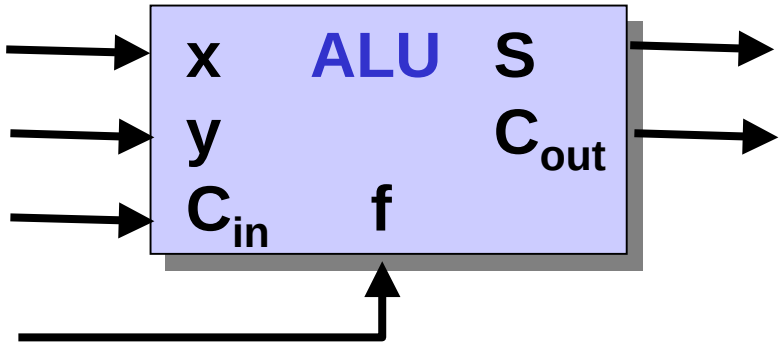
(2) Given the above two tri-state buffers connected together (assume transport model of 5ns per gate), draw X, Y, F, a, b, G for the following input waveforms:

X <= '1', '0' after 10 ns, '1' after 20 ns, 'L' after 30 ns, '1' after 40 ns;
Y <= '0', 'L' after 10 ns, 'W' after 20 ns, 'Z' after 30 ns, 0 after 40 ns;
F <= '0', '1' after 10 ns, '0' after 50 ns;

Assignment #2 (Part 3 of 3)

3a) Write (no programming) a entity-architecture for a 1-bit ALU. The input will consist of x, y, Cin, f and the output will be S and Cout. Use as many sub-components as possible. The input function f will enable the following operations:

function f	ALU bit operation
000	S = 0 Cout = 0
001	S = x
010	S = y
011	S = x AND y
100	S = x OR y
101	S = x XOR y
110	(Cout, S) = x + y + Cin;
111	(Cout, S) = full subtractor



3b) Calculate the number of transistors for the 1-bit ALU

3c) Write a entity-architecture for a N-bit ALU (for-generate)