CSU22022 Computer Architecture I

Second Lecture

Michael Manzke

2023-2024

Trinity College Dublin

NAND Gate implementation

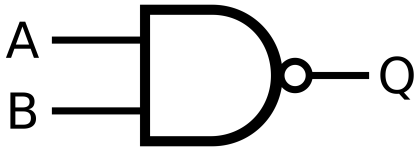

Figure 1: NAND Gate

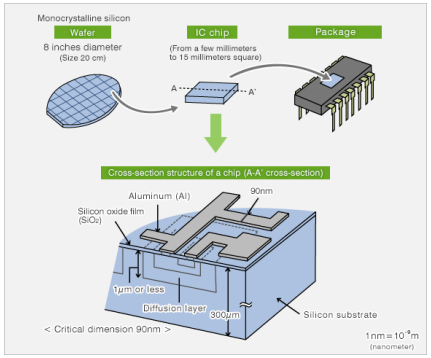

Figure 2: Integrated Circuit

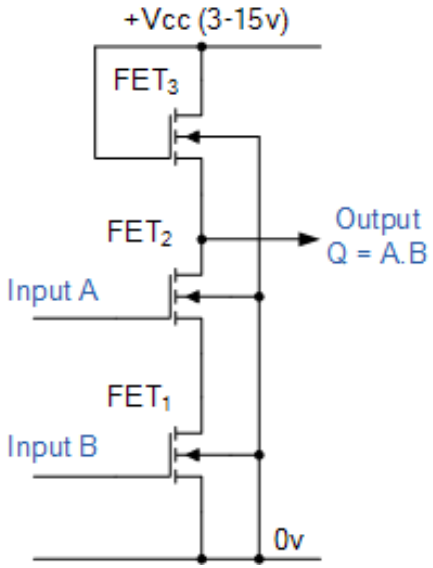

Figure 3: NAND CMOS gate

NAND Gate Behaviour

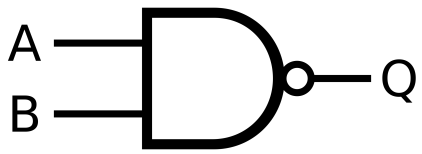

Figure 4: NAND Gate

A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

Table 1: NAND Truth Table

The NAND truth table in table 1 specifies the behaviour of a NAND gate. We are interested in its logical behaviour, but not in its physical implementation, shown on the previous slide. We will assume a certain propagation delay for every component.

NAND VHDL Code

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity NANDGate is
5      Port ( A : in STD_LOGIC;
6             B : in STD_LOGIC;
7             Q : out STD_LOGIC);
8  end NANDGate;
9
10 architecture Behavioral of NANDGate is
11
12 begin
13
14 Q <= A NAND B after 3ns;
15
16 end Behavioral;
```

Listing 1: This VHDL code defines an entity that implements the behaviour of a NAND gate

NAND Gate Simulation

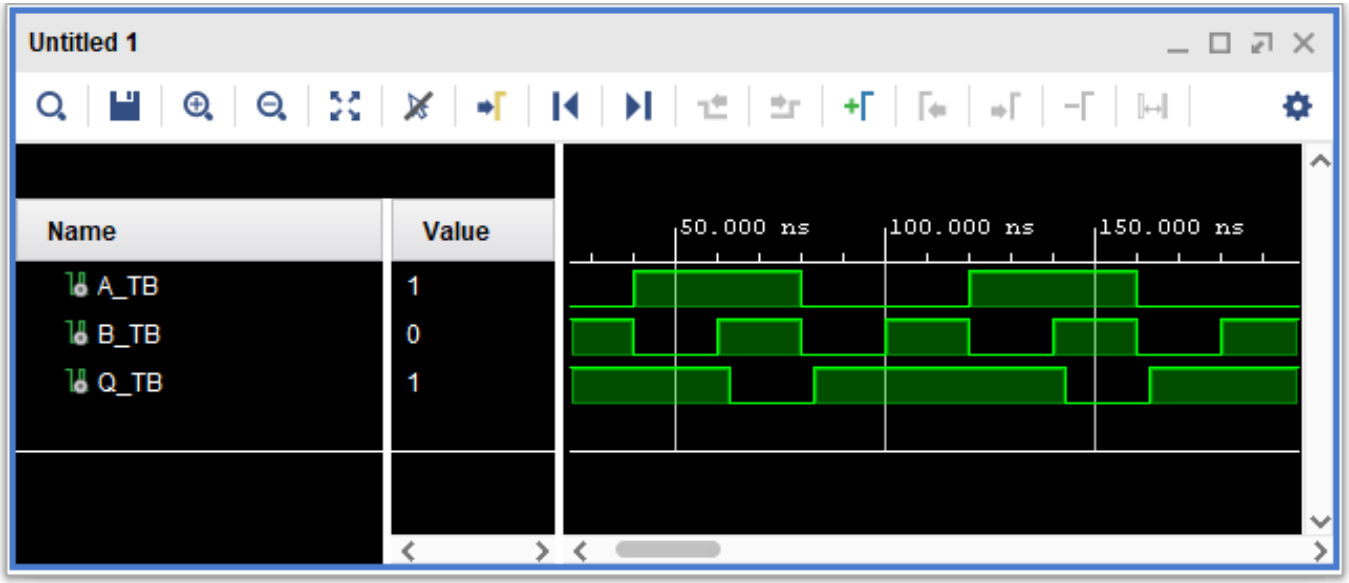

Figure 5: NAND Timing Diagram

The timing diagram in figure 5 shows the simulation result. The simulation changes the input values for A and B over time according to the NAND gate truth table. At the same time, we can observe the output Q.

Event, Propagation Delays and Concurrency

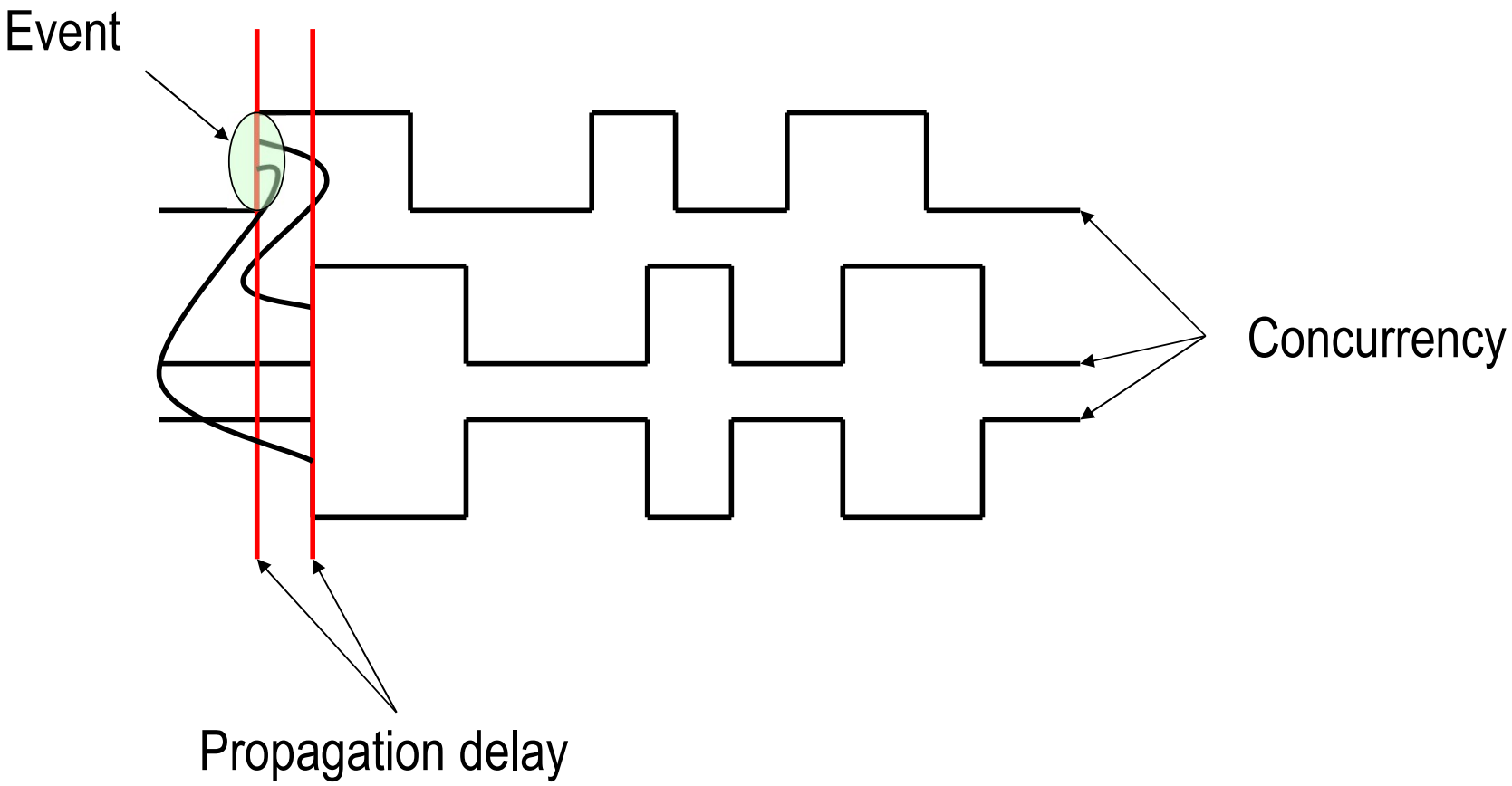

Figure 6: Propagation Delay

VHDL Signal

- May be 0, 1, or Z
- Equivalent to wires in digital circuits
- May be assigned values
- Signals are associated with time values
- Sequences of values determines the waveform

Shared Signals

Hardware description languages must be expressive enough to describe signal that may be driven by one or more sources. This is the case if we implement a Bus.

Design Entity – Gate Level Example

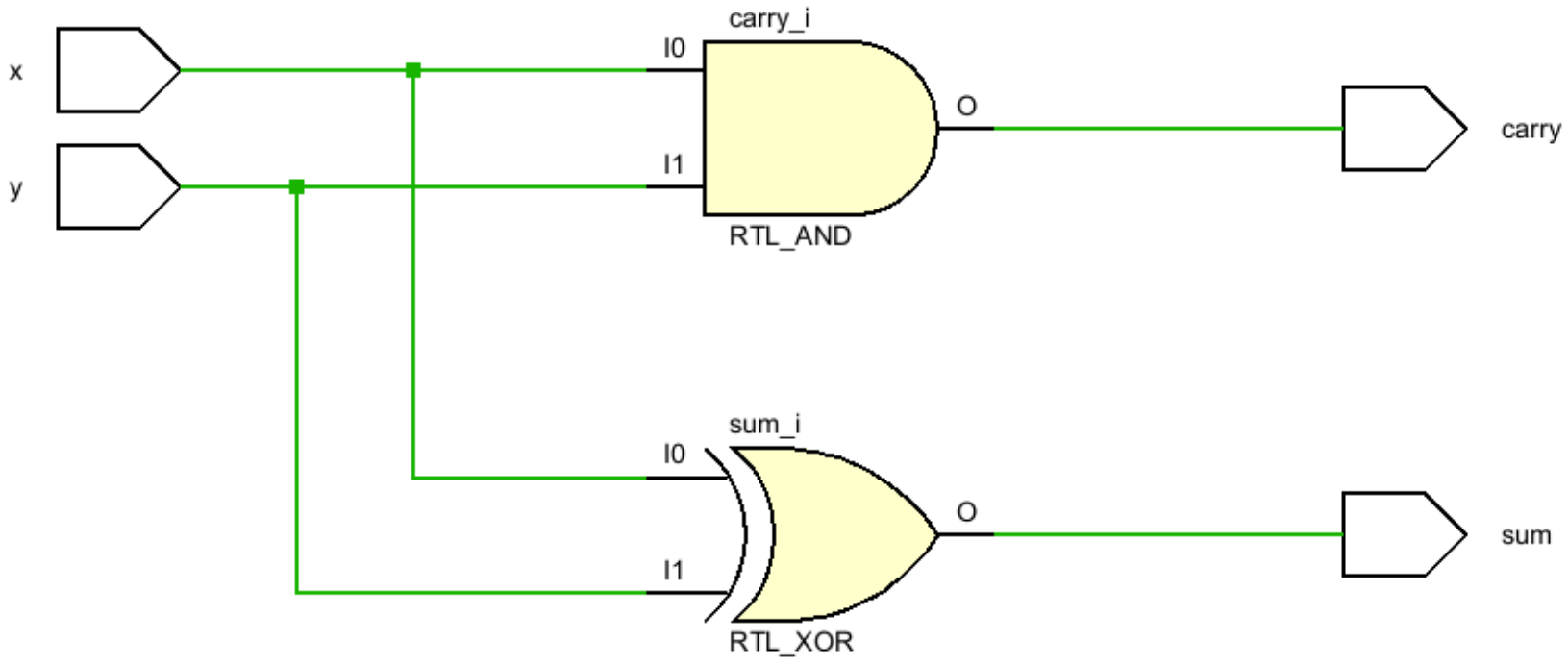

Figure 7: Half-adder

- Input signals: x, y
- Output signals: sum, carry

Design Entity – Description

- Input signals
- Output signals
- Behaviour
  - Truth table
  - Boolean equation
  - Wires between gates
- There are two components in the design entity description
  - The interface
  - Internal behaviour

Entity Declaration

```
1 entity HalfAdder is
2     Port ( x : in bit;
3            y : in bit;
4            sum : out bit;
5            carry : out bit);
6 end HalfAdder;
```

Listing 2: Entity declaration

- Keywords
  - Purple
- Entity Name
  - HalfAdder
- Port
  - Inputs
  - Outputs

Entity Declaration Details

- Purple type denotes VHDL reserved keywords
  - entity, is, Port, in, out, and end
- VHDL is not case sensitive
  - Half-adder = HALF-ADDER
- Ports define the input and output of the design entity
- Port signals must declare their types

Port Declaration

- Signal types defined in the VHDL language
- bit
  - Represents a single-bit signal
- bit_vector
  - Represents a vector of signal of type bit
- bit and bit_vector are only two out of several other VHDL data types

IEEE 1164 Signals Values

U	Uninitialised
X	Forcing Unknown
0	Forcing 0
1	Forcing 1
Z	High Impedance
W	Weak Unknown
L	Weak 0
H	Weak 1
-	Don't Care

Table 2: IEEE 1164 standard defines nine-value signals

IEEE Library

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity HalfAdder is
5     Port ( x : in bit;
6            y : in bit;
7            sum : out bit;
8            carry : out bit);
9 end HalfAdder;
```

Listing 3: The following modifications are required to make the previous entity declaration IEEE compliant

Signal Mode

```
1 entity HalfAdder is
2     Port ( x : in bit;
3            .
4            .
5            .
```

- Port declaration distinguishes between:
  - in input signal
  - out output signal
  - inout bidirectional signal

4 to 1 8-bit Multiplexer

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity MUX_4to1_8bit is
5      Port(I0 : in STD_LOGIC_VECTOR (7 downto 0);
6           I1 : in STD_LOGIC_VECTOR (7 downto 0);
7           I2 : in STD_LOGIC_VECTOR (7 downto 0);
8           I3 : in STD_LOGIC_VECTOR (7 downto 0);
9           Sel : in STD_LOGIC_VECTOR (1 downto 0);
10          Z : out STD_LOGIC_VECTOR (7 downto 0));
11 end MUX_4to1_8bit;
```

Listing 4: This example uses std_logic_vector(7 downto 0)

std_logic_vector(7 downto 0)

```
1 entity MUX_4to1_8bit is
2     Port(I0 : in STD_LOGIC_VECTOR (7 downto 0);
3          .
4          .
5          .
6 end MUX_4to1_8bit;
```

- This example refers to 8-bit input vector
  - bit 7 - most significant bit,
  - bit 0 - least significant bit,

Entity's Internal Behaviour

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity HalfAdder is
5      Port ( x : in STD_LOGIC;
6             y : in STD_LOGIC;
7             sum : out STD_LOGIC;
8             carry : out STD_LOGIC);
9  end HalfAdder;
10
11 architecture Behavioral of HalfAdder is
12 -- declaration
13 begin
14 -- description of behaviour
15 end Behavioral;
```

Listing 5: VHDL describes the internal behaviour in the architecture construct

Architecture

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity HalfAdder is
5      Port ( x : in STD_LOGIC;
6             y : in STD_LOGIC;
7             sum : out STD_LOGIC;
8             carry : out STD_LOGIC);
9  end HalfAdder;
10
11 architecture Behavioral of HalfAdder is
12
13 begin
14     sum <= x xor y after 5ns;
15     carry <= x and y after 5ns;
16 end Behavioral;
```

Listing 6: Line 14 and 15 describe the behaviour of the HalfAdder entity