

Programmable Electronics using FPGA

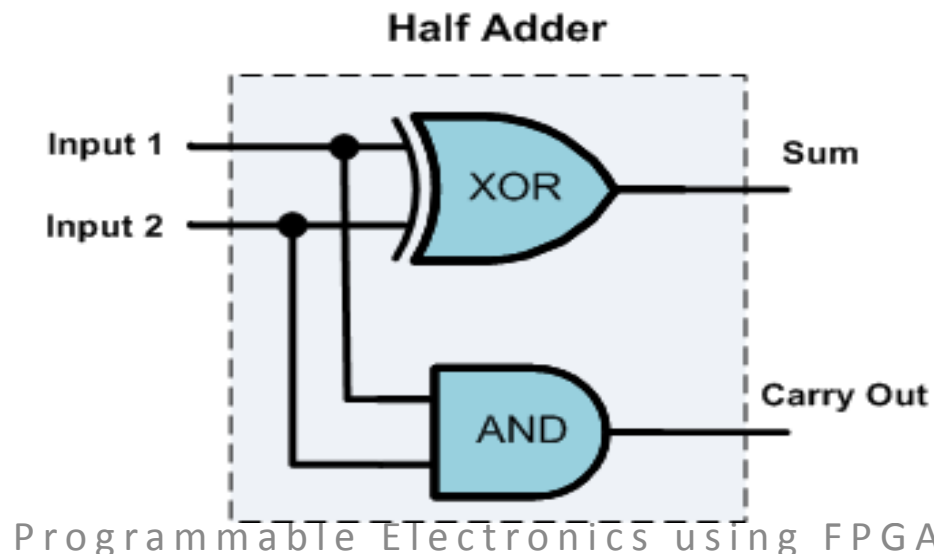
This week

- Combinational and sequential circuits
- The importance of clocks
- An introduction to VHDL
- Entities and architectures
- Example: AND gate design
- Abstraction levels – Data flow modelling
- Lab work: Design a Half Adder

Combinational Logic

- Arrangement of logic gates with a set of input and outputs.
- There is no feedback among inputs and outputs. Here outputs are a function of inputs only.

Example : A Half Adder circuit where Inputs (*input1*, *input2*) are directly mapped into outputs (*Sum*, *CarryOut*).

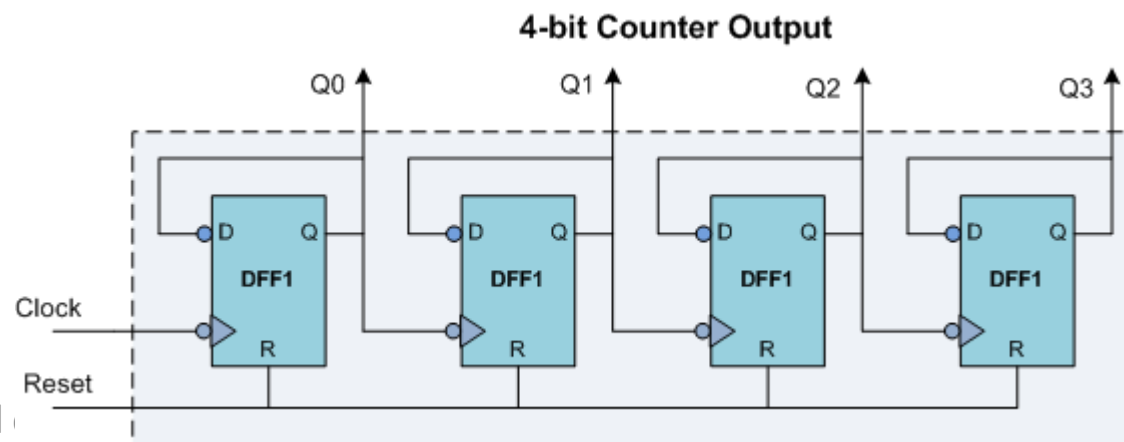


Sequential Logic

- Interconnection of Flip-Flops and Gates
- Outputs depend on inputs and entire/some history of execution.
- It is based on:
 - Current state of input signal
 - Current state (stored in memory element) of the system
 - Next state of the system

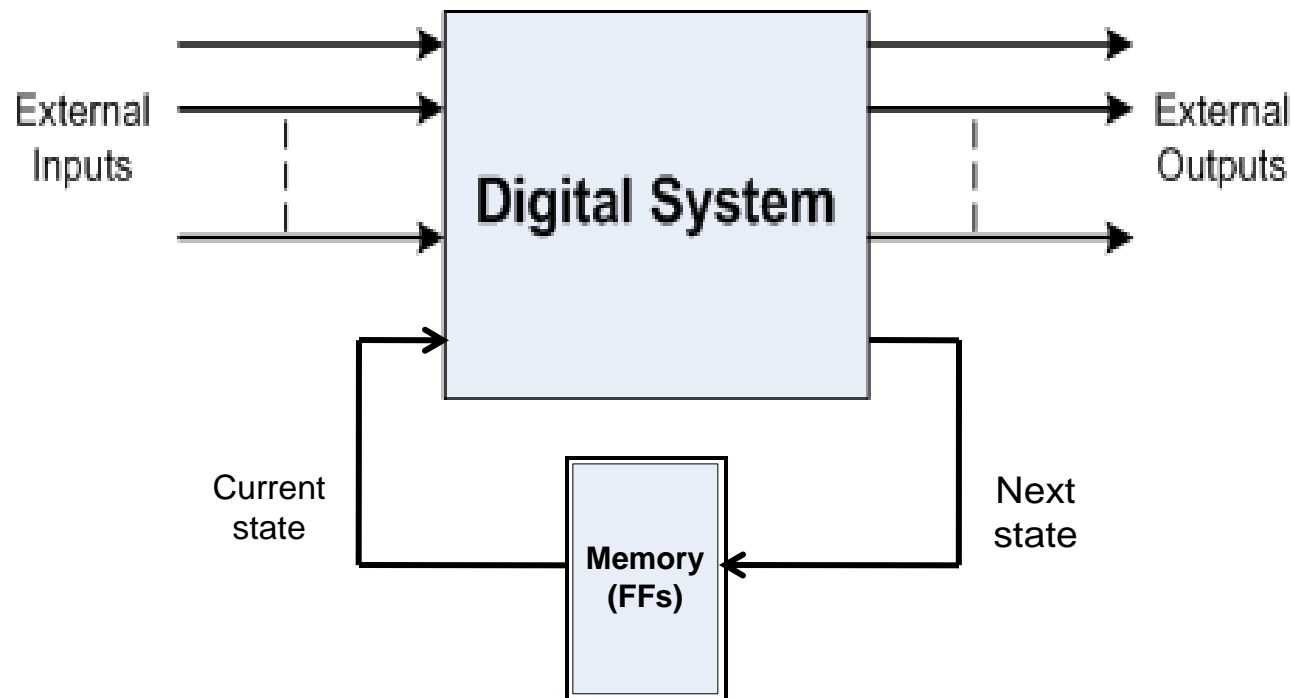
Example:
A ripple carry counter

Programmable



Digital Systems

- Consists of combinational and sequential circuits.
- Presence of feedback distinguishes between sequential and combinational networks.



Examples

➤ Combinational Circuits

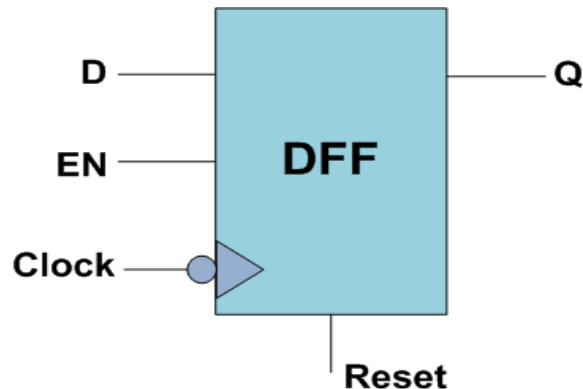
- Multiplexers
- Encoders / Decoders
- Comparators
- Adders / Subtractors
- Multipliers




➤ Sequential Circuits

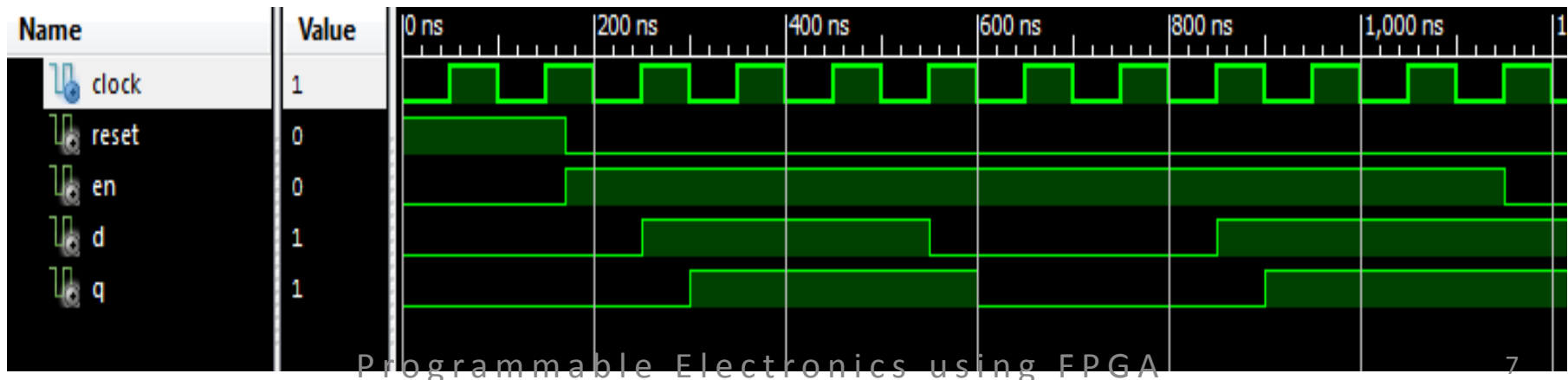
- Flip Flops
- Counters
- Registers
- RAMs

Recap: Clock enabled flip-flops

- D flip-flop is a memory element which operates at the clock edge.
 - Output follows the input signal.
 - Holds the last saved value.



CLK	Reset	EN	D	Q	
-	1	-	-	0	Output will be zero
	0	1	1	1	Q=D
	0	1	0	0	Q=D
	0	0	-	Q	No change



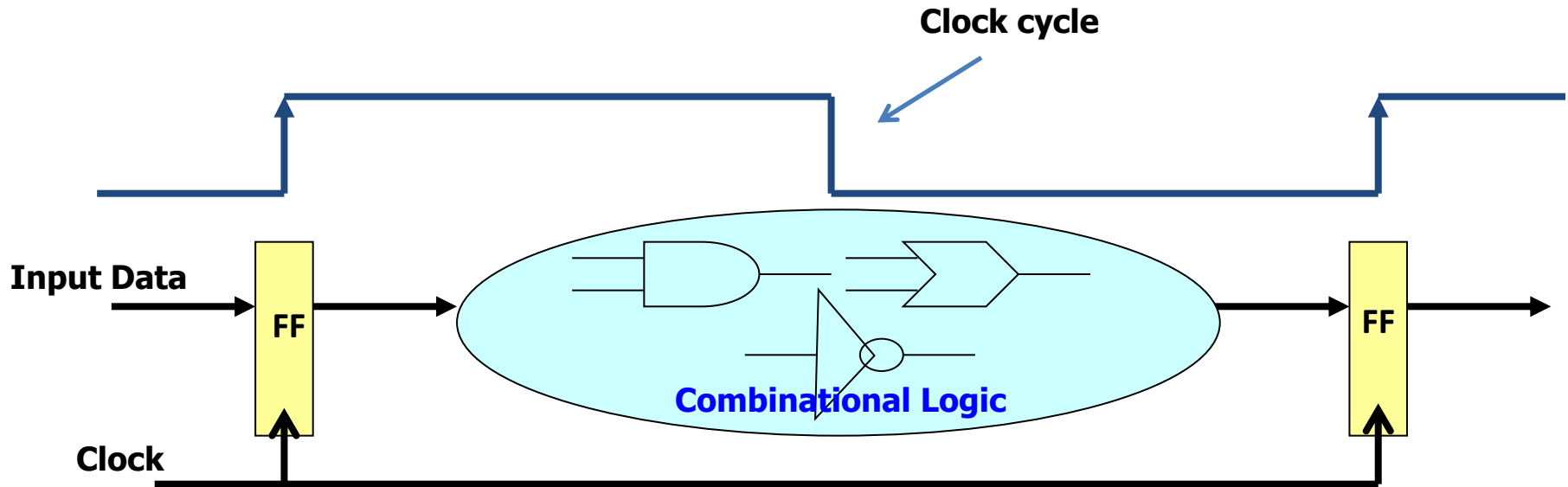
The importance of clocks

- FPGAs are most suitable for synchronous designs where all storage elements are updated by a single clock.
- We often need to consider aspects of our design in order to achieve desired clock speeds.
- This is because of issues such as propagation delay.

Question: How can we estimate the operating frequency of our digital circuit?

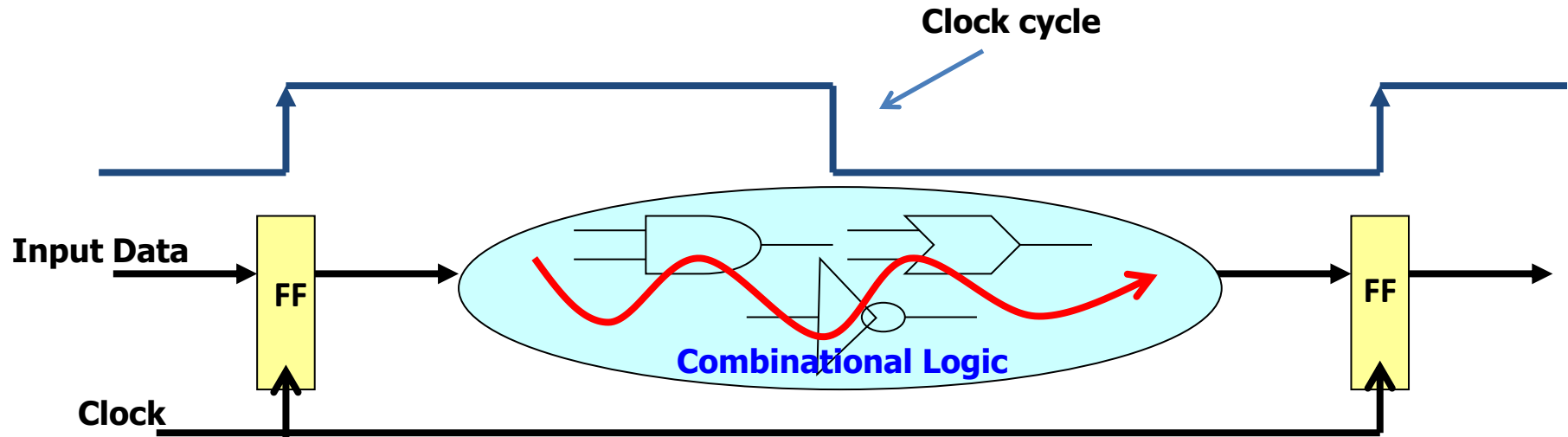
Answer: The longest logic path in the circuit defines the maximum operating frequency of the circuit.

The importance of clocks



- All storage elements are clocked by the same clock edge.
- The combination logic blocks:
 - Inputs are updated at each clock tick
 - All outputs must be stable before the next clock tick

Critical Path & Cycle Time



Question: What is meant by critical path in a circuit?

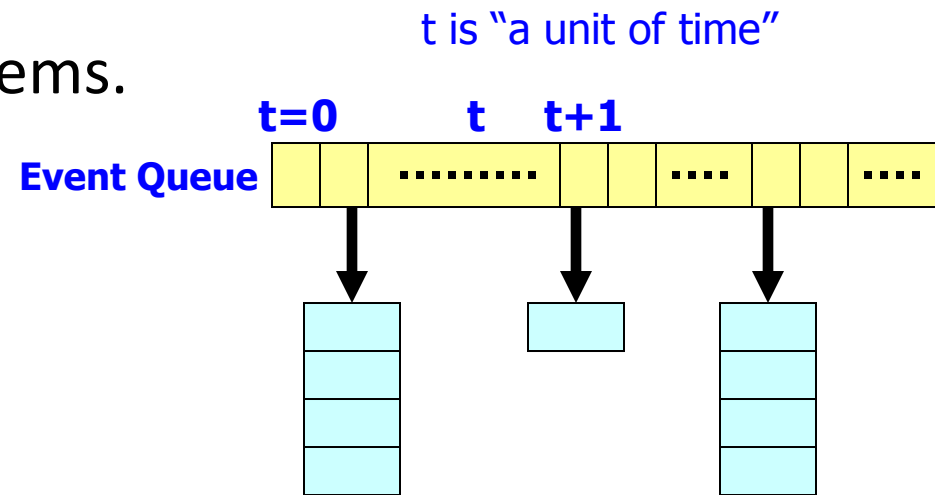
- Critical path: Longest (combinational) path between two consecutive memory elements.
- The cycle time is a function of the critical path and it must be greater than:
 - Longest path through the combination logic + setup + ...

How does simulation work?

Event Driven Simulation

➤ Modeling event driven systems.

- Event : change in state



➤ Simulation starts at $t = 0$;

- Processing events generates new events.
- When all events at time t have been processed simulation time advances to $t+1$.
- Simulation stops when there are no more events in the queue.

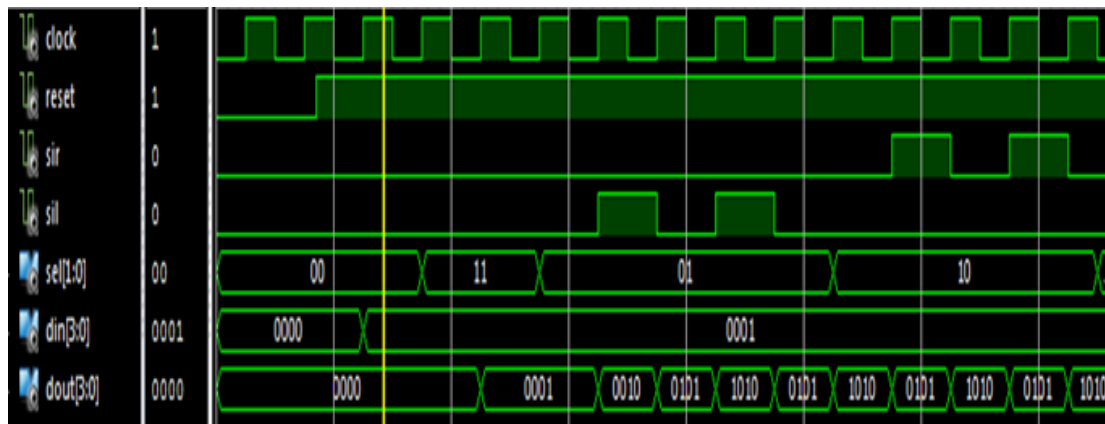
VHDL: Main Language Concepts

Procedural
statements



```
process (A, B, C, D, S)
begin
  if S = "00" then
    Out <= A;
  elsif S = "01" then
    Out <= B;
  elsif S = "10" then
    Out <= C;
  else
    Out <= D;
  end if;
end process
```

Execution
flow



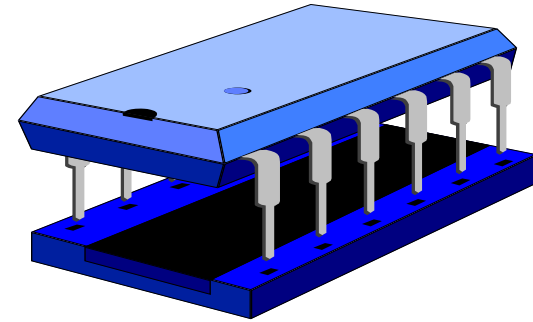
Explore timings

User Identifiers – Signal names

- Formed from { [A-Z], [a-z], [0-9], _ }, but ..
- .. can't begin with _ or [0-9].
- .. Can't have special character (#) or hyphen (-)
 - myidentifier
 - m_y_identifier
 - my_identifier3
 - 3my_identifier (X)
 - my-identifier (X)
 - _my_identifier (X)
 - _myidentifier# (X)
- VHDL is not a case sensitivity language.
 - myid = Myid

Entities and Architectures

- How can we physically describe a chip?
- Every chip can be treated as a Black Box which has pins for external communication.
- A black box is known as an Entity and the logic inside is known as the Architecture.
- Every VHDL design consists of at least one entity / architecture pair or one entity with multiple architectures.



Entities and architectures

➤ Entities are the basic building block.

➤ Entities are:

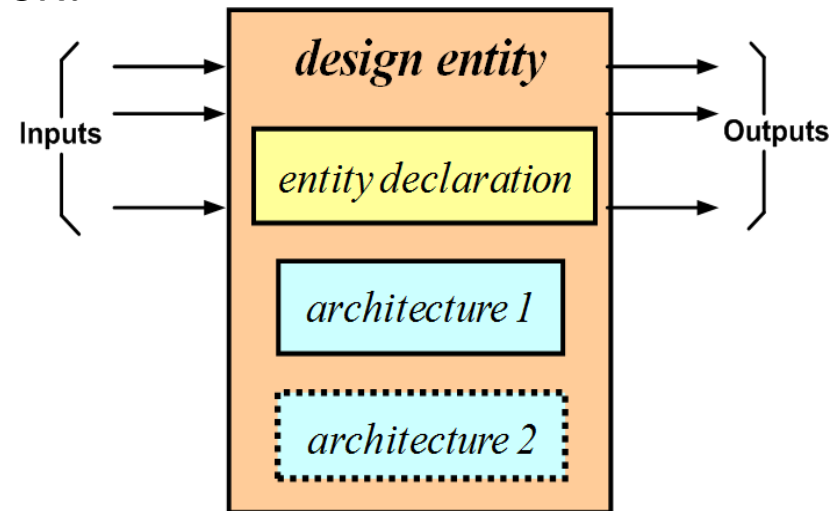
- Defined
- Instantiated

➤ Defined: describe what it does

➤ Instantiate: create a specific instance

➤ Similar in principle to defining variable types and variable instantiations in software

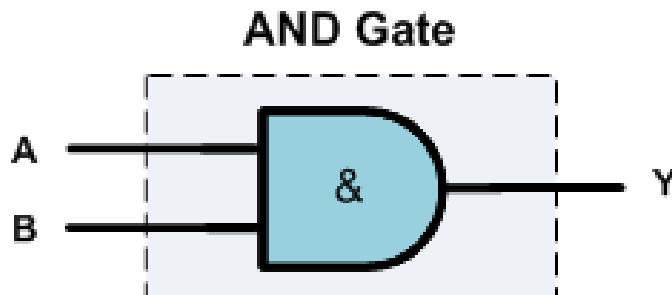
➤ There are ports (wires) for interconnection with other entities for building bigger systems.



Entity Declaration

- An entity declaration describes the interface of the component, i.e. input and output ports.

Example: simple AND gate design.



Entity name *Port names* *Port type*

```

ENTITY and_gate IS
    PORT (
        A    : IN  STD_LOGIC;
        B    : IN  STD_LOGIC;
        Y    : OUT STD_LOGIC
    );
END and_gate;

```

Port modes (data flow directions)

Architecture Declaration

- The architecture describes what the circuit actually does.
 - The implementation of the associated entity.

Example: simple AND gate design.

The diagram shows a VHDL code snippet for an Entity-Architecture declaration. The code is as follows:

```
ARCHITECTURE dataflow of and_gate IS
begin
    Y <= A AND B;
end dataflow;
```

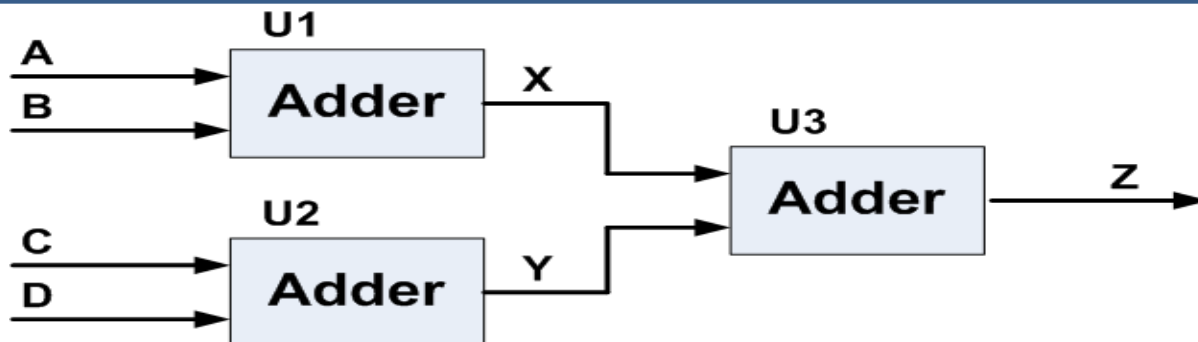
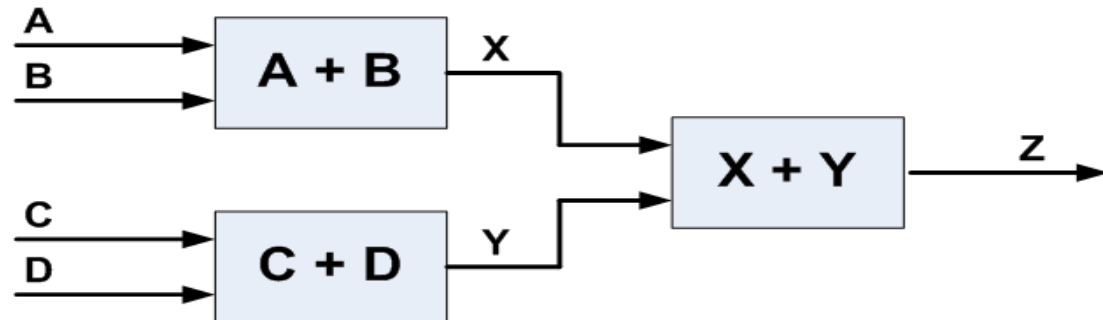
Annotations with red arrows point to specific parts of the code:

- Architecture name**: Points to the word `dataflow`.
- Entity name**: Points to the word `and_gate`.
- Dataflow statement**: Points to the assignment statement `Y <= A AND B;`.
- End architecture**: Points to the word `end`.

Example: adders for concurrency

$$\begin{aligned}
 &A + B + C + D = Z \\
 &\downarrow \\
 &(A + B) + (C + D) = Z \\
 &\downarrow \\
 &X + Y = Z
 \end{aligned}$$

Concurrency



Structure

Example: AND Gate Design in VHDL

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY and_gate IS
PORT (
    A,B : IN STD_LOGIC;
    Y : OUT STD_LOGIC
);
END and_gate;

ARCHITECTURE dataflow of and_gate IS
Begin

    Y <= A AND B;

end dataflow;
```

Have you noticed these lines?



VHDL Libraries

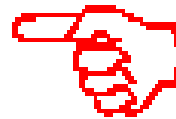
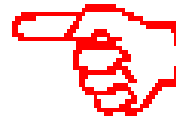
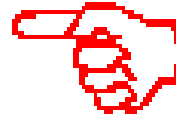
- We often create and use libraries of designs
 - Library IEEE
 - Define the library package used
 - Use IEEE.std_logic_1164.all;
 - Define the library file used
 - For example, STD_LOGIC_1164 defines '1' as logic high and '0' as logic low.
 - `output <= '1';` -- Assign logic high to output
 - Use IEEE.std_logic_arith.all;
 - Use IEEE.std_logic_signed.all;
 - Use IEEE.std_logic_unsigned.all;

Abstraction Levels in VHDL

Dataflow

Behavioral

Structural



We will say more about this later.

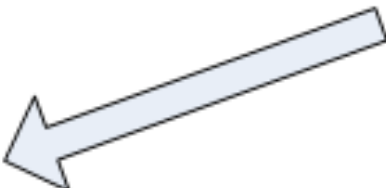
Dataflow Level in VHDL

- Architecture is designed by specifying the data flows between hardware registers.
- Concurrent assignment statements are used for dataflow descriptions.

Example: Concurrent assignment statements.

```
architecture dataflow of gates is  
  signal s1, s2: std_logic;  
begin  
    s1 <= a AND b;  
    s2 <= b AND cin;  
    cout <= s1 OR s2;  
end dataflow;
```

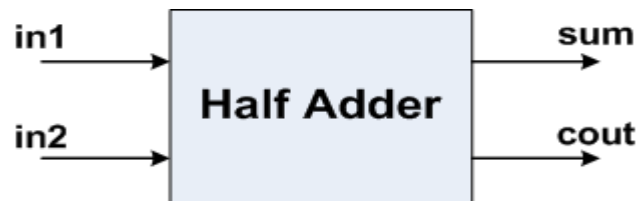
*Concurrent
statements*



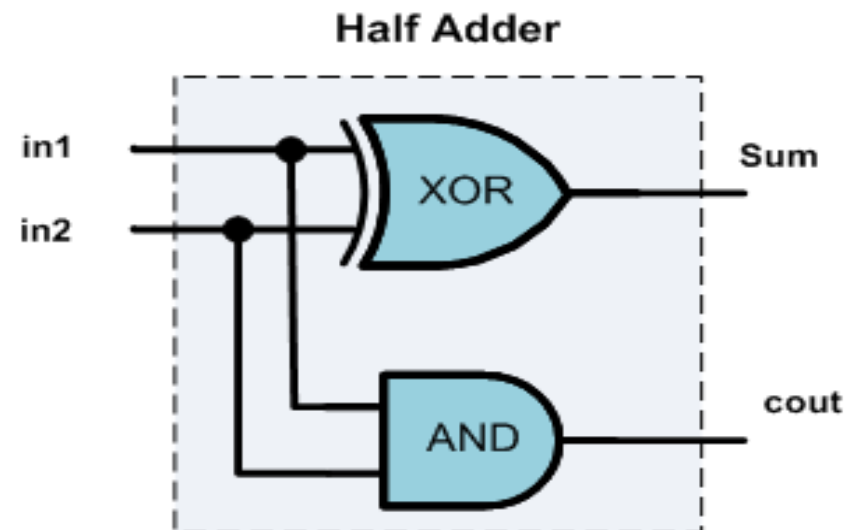
Lab work: Designing a half adder

- A half adder consists of two gates – XOR and AND.
- These gates operate side by side.

Example: Half adder.



in1	in2	sum	cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



Lab work: Designing a half adder

- Dataflow level is best for half adder design.
- We need two concurrent statements in an architecture – one for XOR gate and second for AND gate.
- In this lab, you will implement and test your own half adder.

Hint: Design of XOR and AND gates using dataflow level

For XOR gate:

```
sum <= in1 XOR in2;
```

For AND gate:

```
cout <= in1 AND in2;
```

