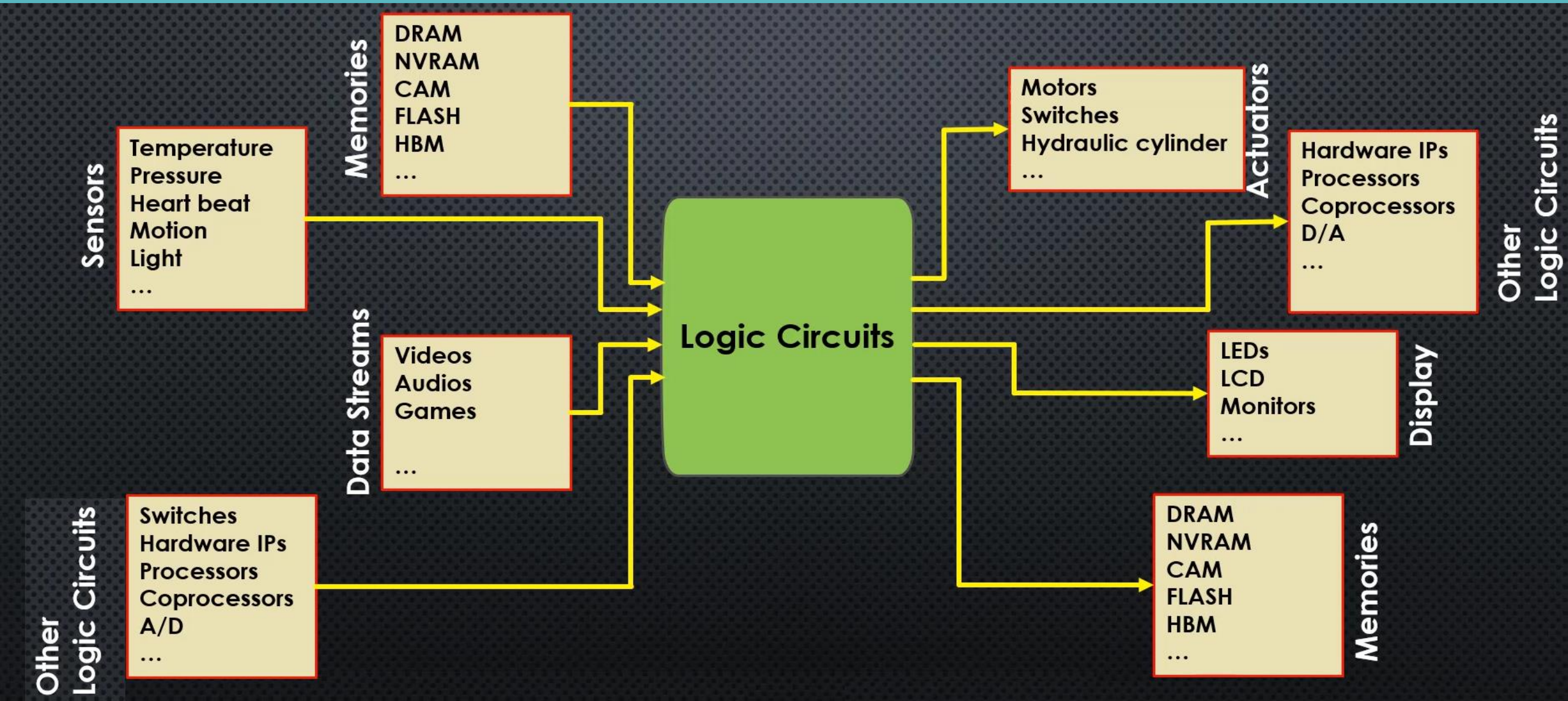
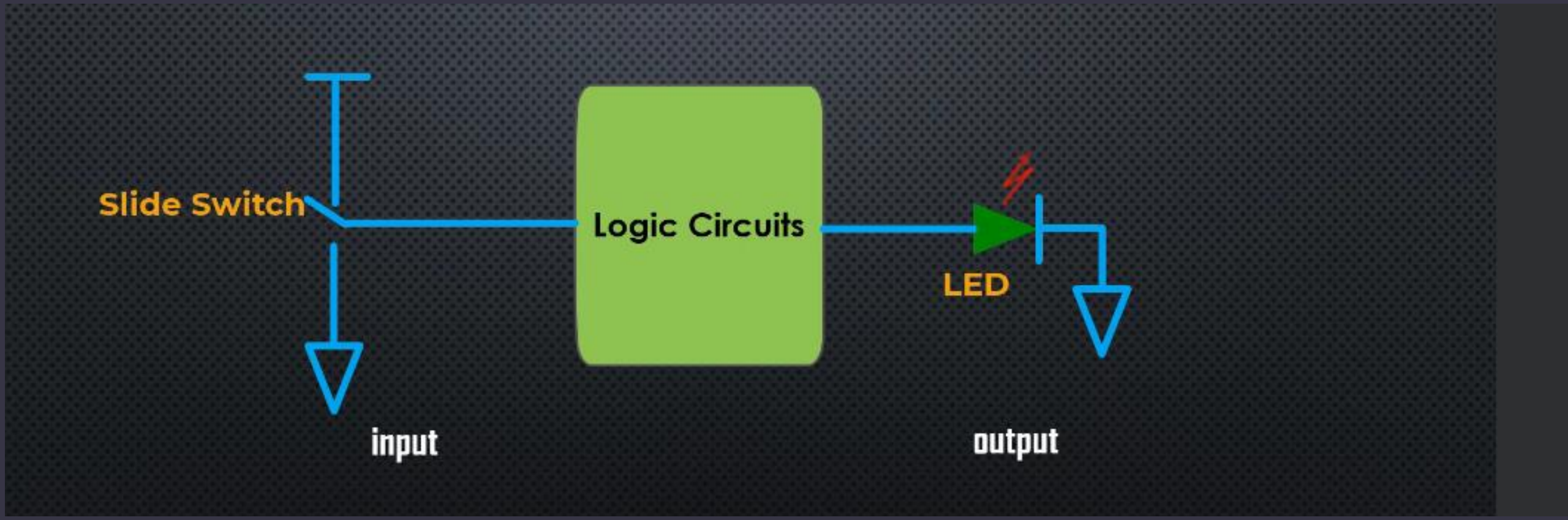
BASIC INPUT/OUTPUT

Lecture - 3

LOGIC CIRCUIT

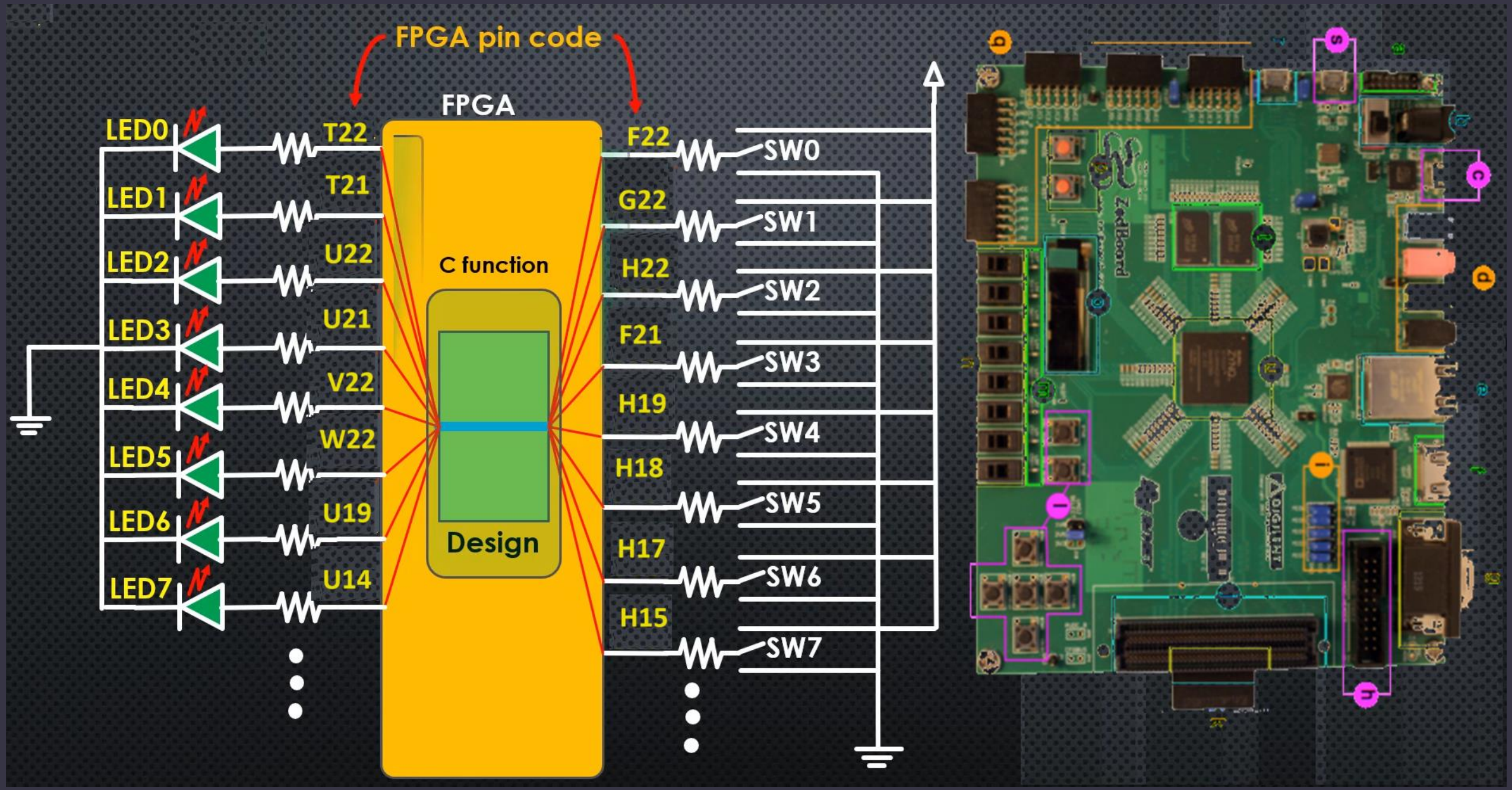


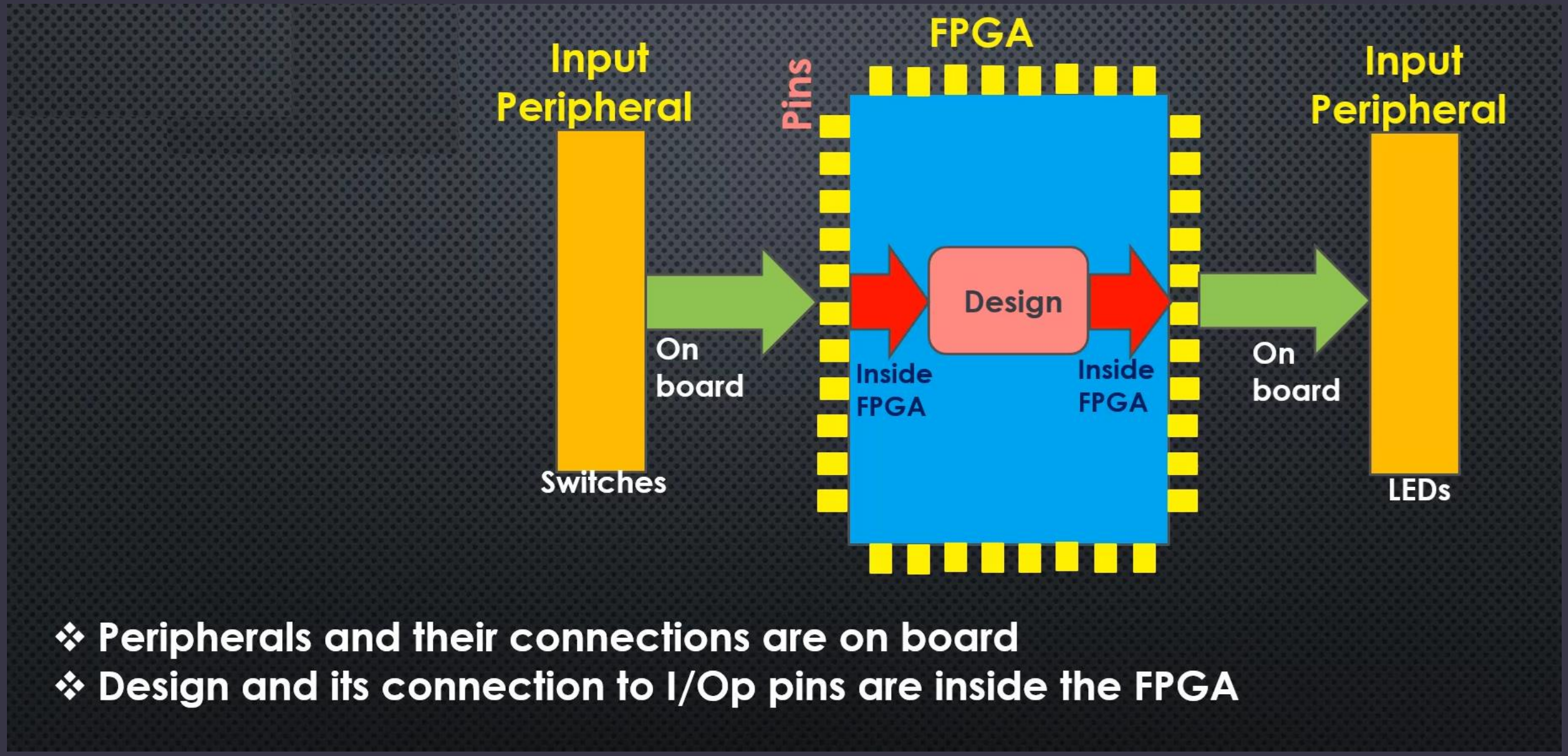
SWITCH AND LED CIRCUIT



IN THIS EXPERIMENT, WE WILL LEARN

- How to receive and send data in a logic circuit
- How to use HLS design flow to describe such a design
- Use the Xilinx Vivado to generate bitstream for design with input and outputs
- Program the Zed board and set the input data and check the outputs


FPGA pin code
FPGA
C function
Design
LED0
LED1
LED2
LED3
LED4
LED5
LED6
LED7
T22
T21
U22
U21
V22
W22
U19
U14
F22
G22
H22
F21
H19
H18
H17
H15
SW0
SW1
SW2
SW3
SW4
SW5
SW6
SW7



- Peripherals and their connections are on board
- Design and its connection to I/Op pins are inside the FPGA

ASSIGNMENT Q

Can we describe a circuit design and the connection between its ports and the FPGA I/O pins in HLS C functions?

HLS PORT

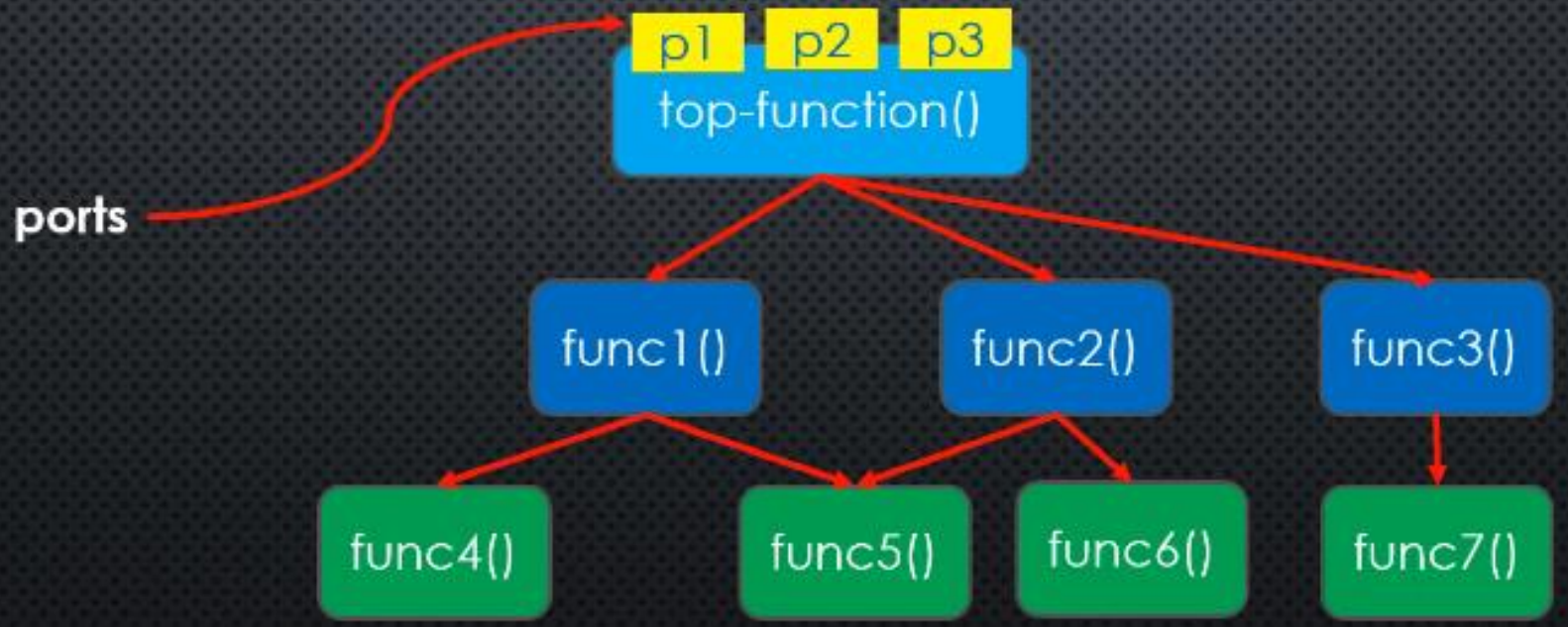
Basic input/output

INPUT/OUTPUT EXAMPLE

```
#define data_type unsigned char

void basic_input_output(
                data_type    input
                data_type  *output) {

  *output = input;
}
```

TOP-LEVEL FUNCTION

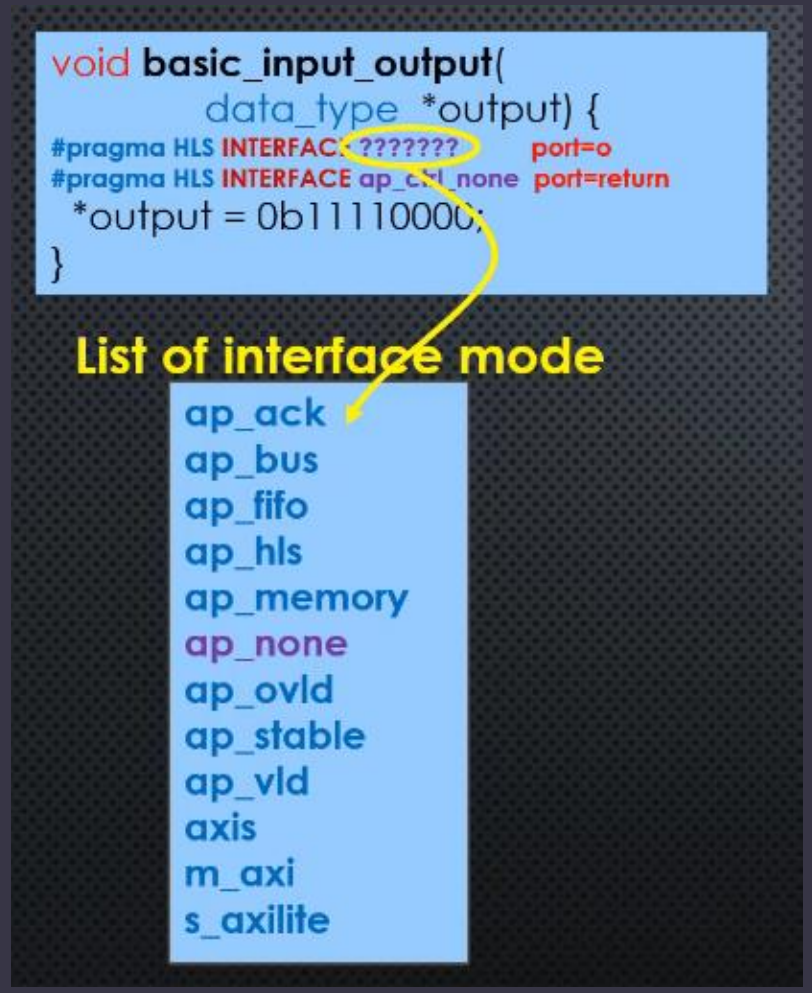
When the top-level function is synthesized, the arguments (or parameters) to the function are synthesized into RTL ports. This process is called ***interface synthesis***.



PORT INTERFACES

```
void basic_input_output(
        data_type  *output) {
#pragma HLS INTERFACE ???????      port=o
#pragma HLS INTERFACE ap_ctrl_none  port=return
  *output = 0b11110000;
}
```

PORT INTERFACES

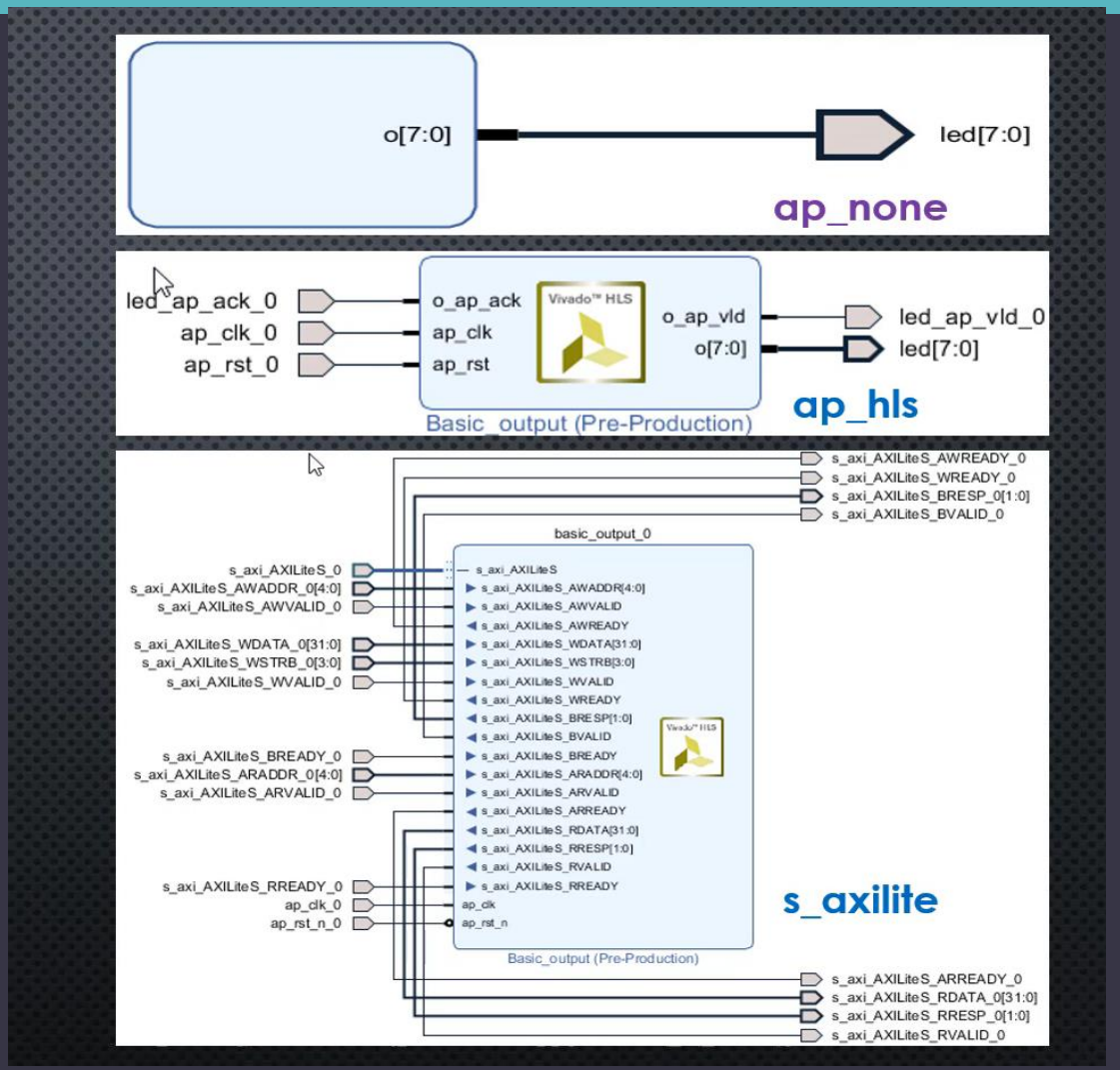


PORT INTERFACES

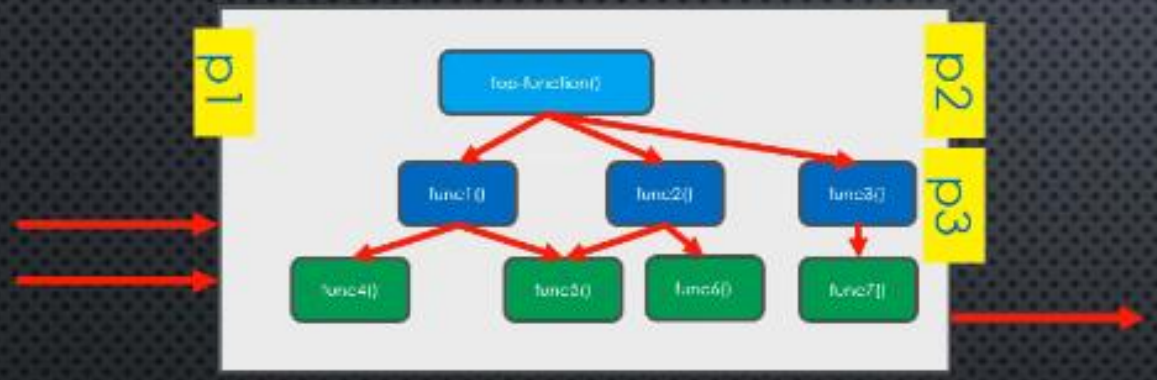
```
void basic_input_output(
        data_type *output) {
#pragma HLS INTERFACE ??????? port=o
#pragma HLS INTERFACE ap_ctrl_none port=return
  *output = 0b11110000;
}
```

List of interface mode

- ap_ack
- ap_bus
- ap_fifo
- ap_hls
- ap_memory
- ap_none
- ap_ovld
- ap_stable
- ap_vld
- axis
- m_axi
- s_axilite



TOP-LEVEL FUNCTION CONTROL SIGNAL



Block level interface mode

- ap_ctrl_chain
- ap_ctrl_hs
- ap_ctrl_none
- a_axilite

none
start
done
clock
....

ap_ctrl_none

INPUT/OUTPUT EXAMPLE

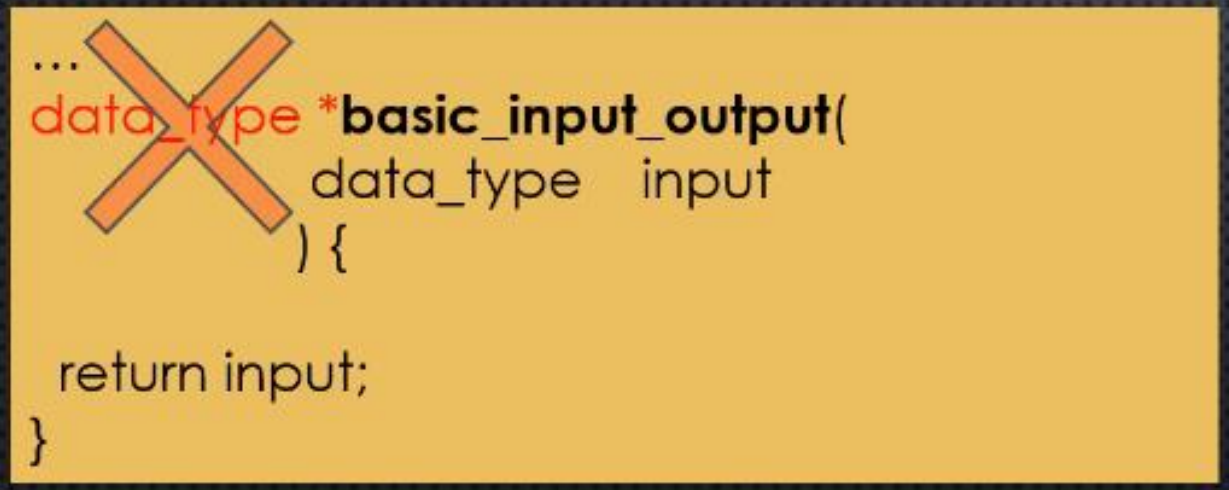
```
typedef unsigned char data_type;

void basic_input_output(
                data_type    input,
                data_type  *output) {
#pragma HLS INTERFACE ap_none        port=input
#pragma HLS INTERFACE ap_none        port=output
#pragma HLS INTERFACE ap_ctrl_none   port=return

  *output = input;
}
```

USING FUNCTION RETURN

```
typedef unsigned char data_type;

data_type basic_input_output(
                data_type    input
               ) {
#pragma HLS INTERFACE ap_none          port=input
#pragma HLS INTERFACE ap_ctrl_none  port=return

  return input;
}
```

NOTE

The return value to the top-level function cannot be a pointer.



```
...
data_type *basic_input_output(
            data_type    input
           ) {

  return input;
}
```

SUMMARY

- Each HLS hardware description can have only one top-function
- Only arguments in the top function require an associated interface
- There is also a block-level interface assigned to the top-function

ASSIGNMENT Q

Add interfaces pragmas to this top-level function

```
#define data_type unsigned char

void basic_input_output(
                data_type    input,
                data_type  &output) {

  output = input;
}
```

HLS LAB

Basic input/output

SUMMARY

- Two groups of interfaces should be added to the top-function in an HLS design
  - Argument interfaces
  - And top-function block interfaces
- If you don’t add any of these interfaces, the Vivado-HLS considers the corresponding default interface

ASSIGNMENT @

Use the following C function as the LED controller that returns the LED values , then generate the corresponding IP using the Vivado-HLS.

```
unsigned char led_crtl(char sw_input) {
  return sw_input;
}
```

VIVADO LAB

Basic input/output

ASSIGNMENT Q

A simple hardware circuit connects five lower LEDs (i.e., LD0 to LD4) on the Basys3 board to the five lower slide switches.

a- Describe the hardware in Vivado-HLS and generate the RTL-IP.

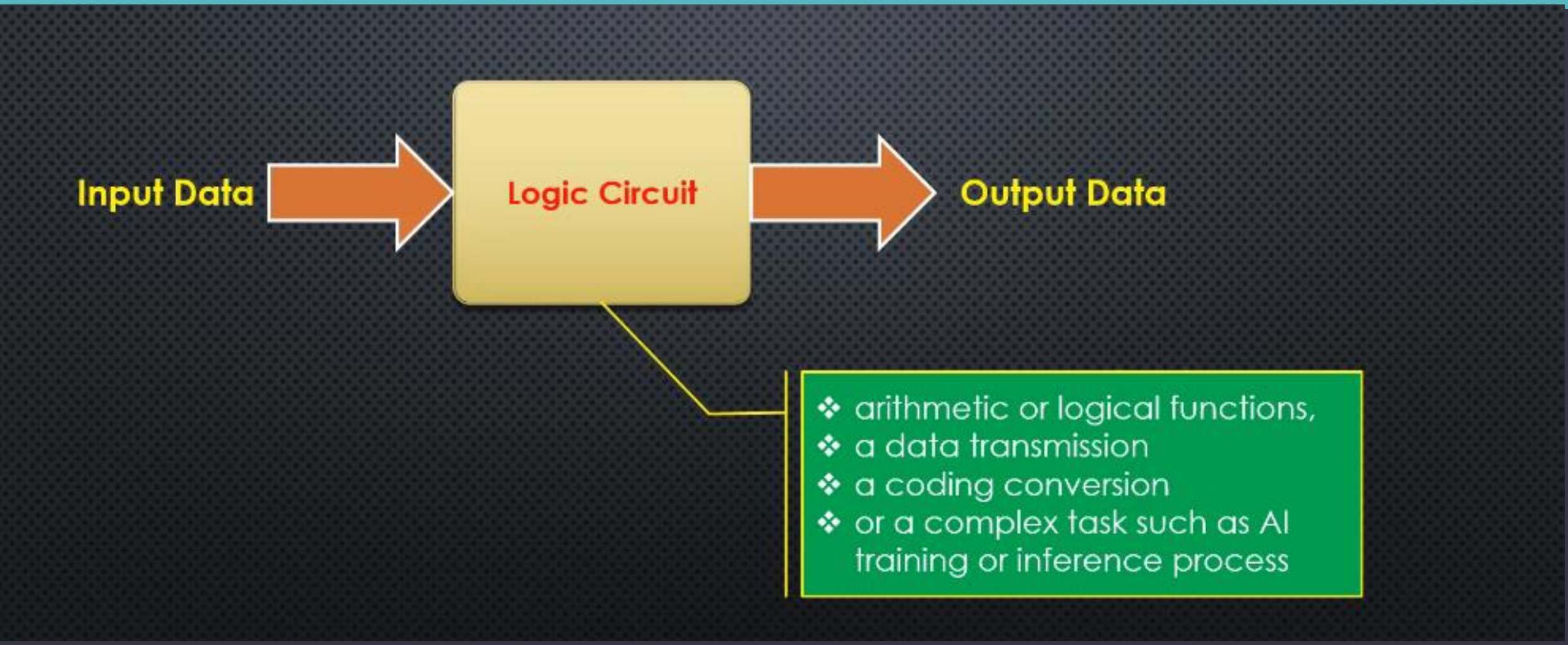
b- Import the design into the Xilinx Vivado suite and generate the FPAG bitstream.

c- Program the Basys3 board and check the output.

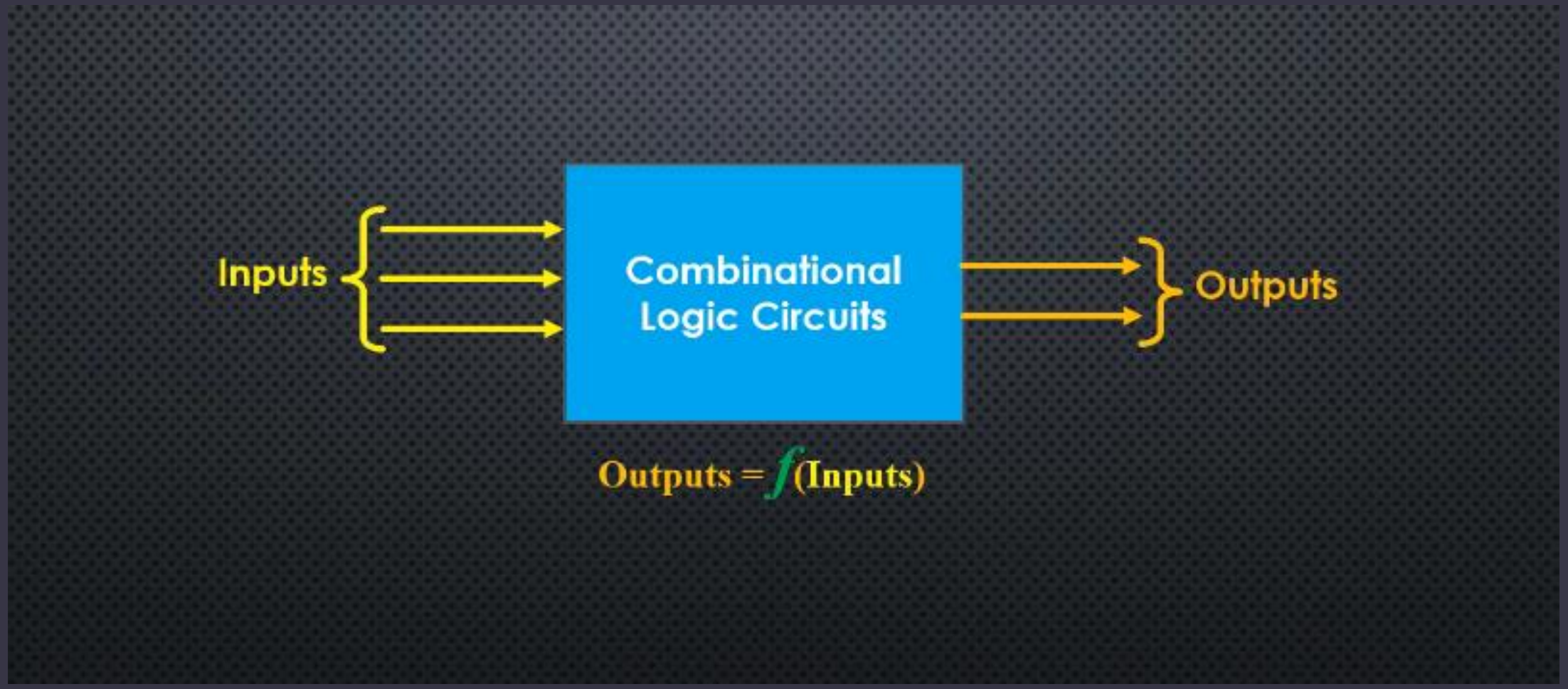
COMBINATIONAL CIRCUIT

Lecture 3

LOGIC CIRCUIT



COMBINATIONAL LOGIC CIRCUITS



DEFINITION

Combinational circuit

DEFINITION

A circuit whose outputs dependents only on the state of its inputs.

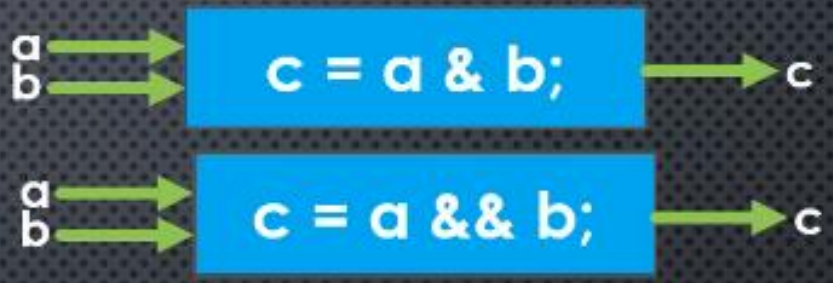
A combinational circuit does not use any types of memories or storage devices.

Combinational Circuit

EXAMPLES

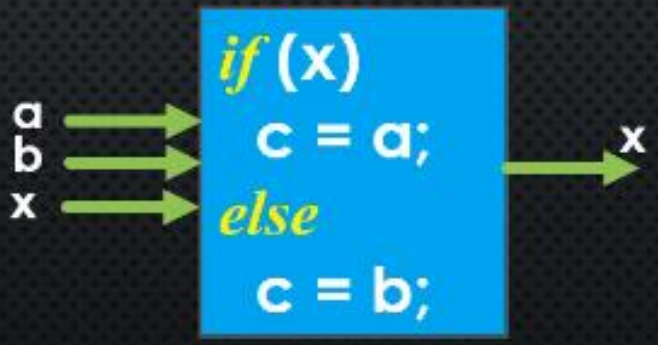
Combinational Circuit

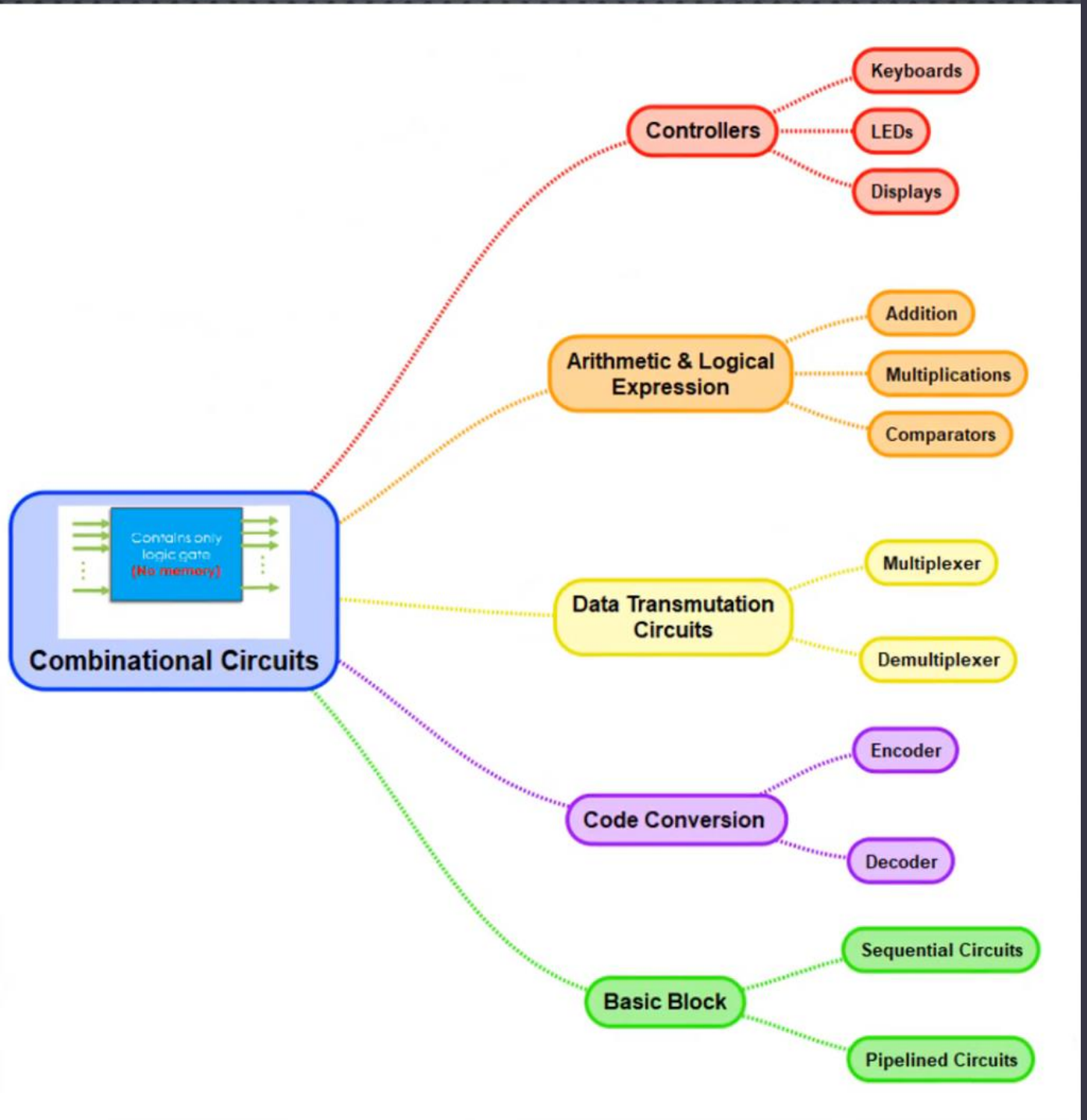
Logical and bitwise expressions:



Simple arithmetic expressions:

Simple programming structures:




Contains only logic gate (No memory)
Combinational Circuits
Controllers
Keyboards
LEDs
Displays
Arithmetic & Logical Expression
Addition
Multiplications
Comparators
Data Transmutation Circuits
Multiplexer
Demultiplexer
Code Conversion
Encoder
Decoder
Basic Block
Sequential Circuits
Pipelined Circuits

ASSIGNMENT Q

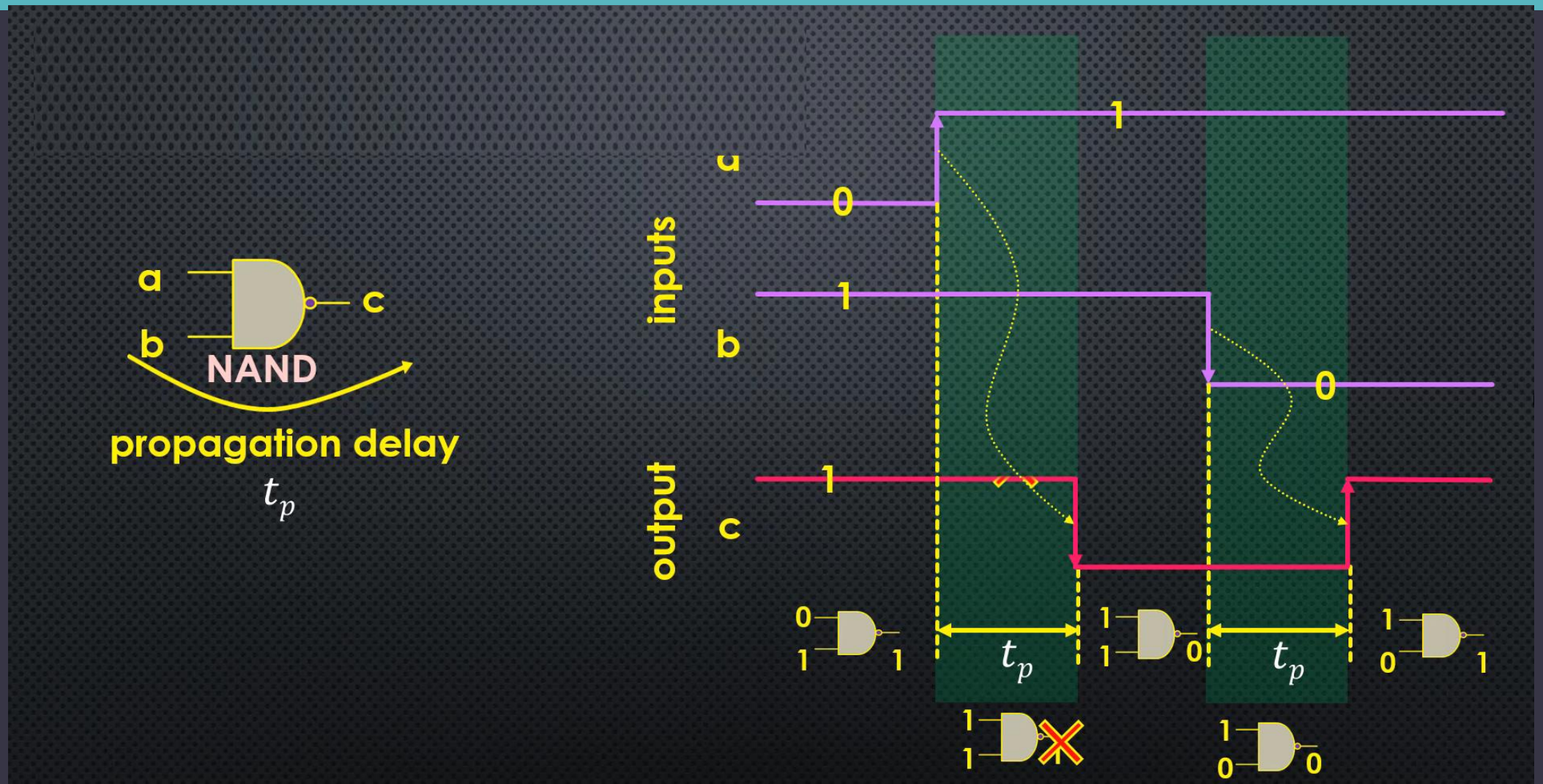
Which circuit can be combinational:

1. Traffic light controller
2. 4-bit integer multiplier
3. Binary counter
4. A three-layer perceptron neural network
5. A recurrent neural network

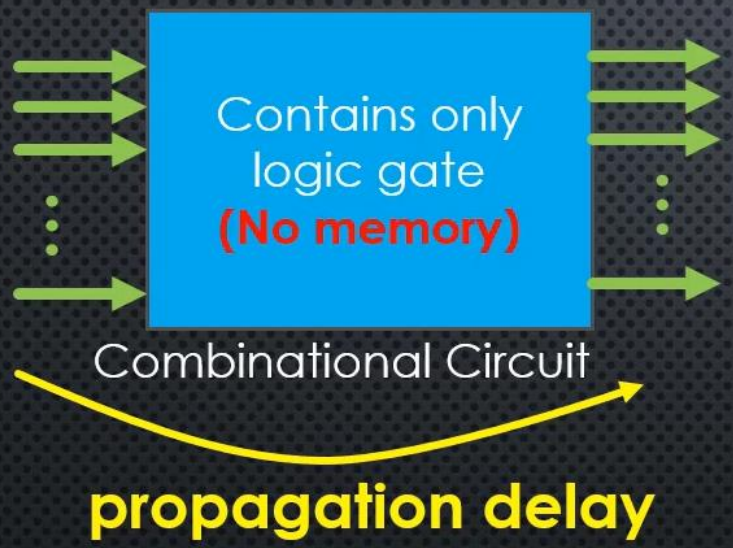
PROPAGATION DELAY

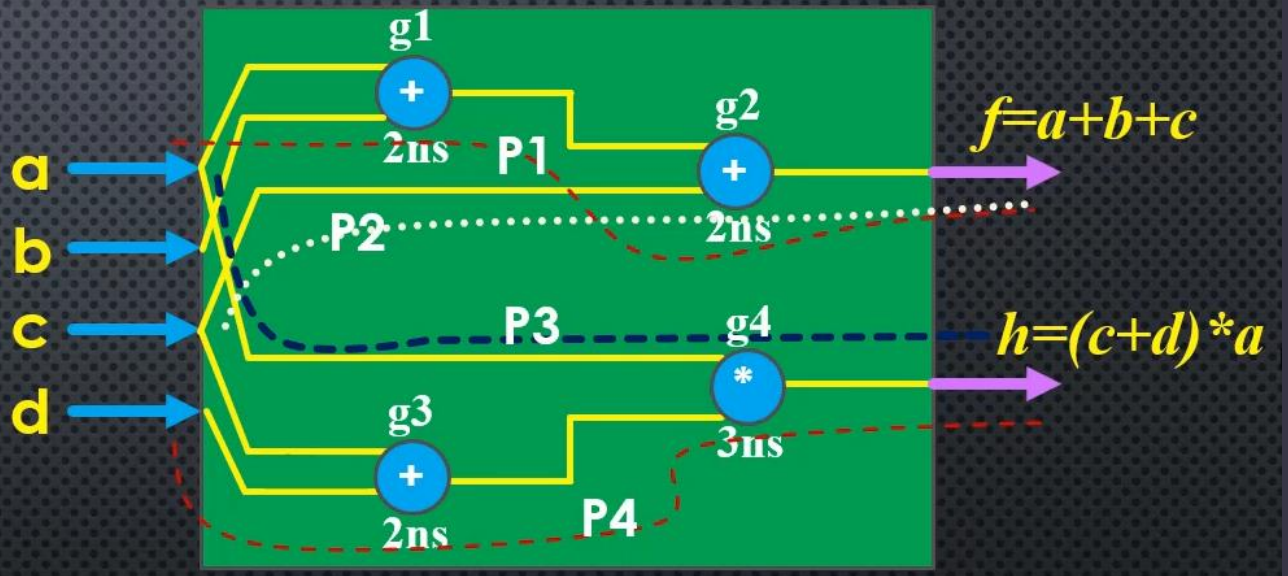
Combinational circuit

GATE PROPAGATION DELAY



GATE PROPAGATION DELAY





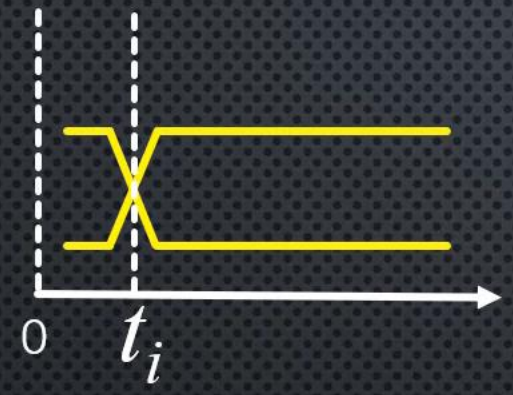
P1(*a or b → g1 → g2 → f*).....4ns

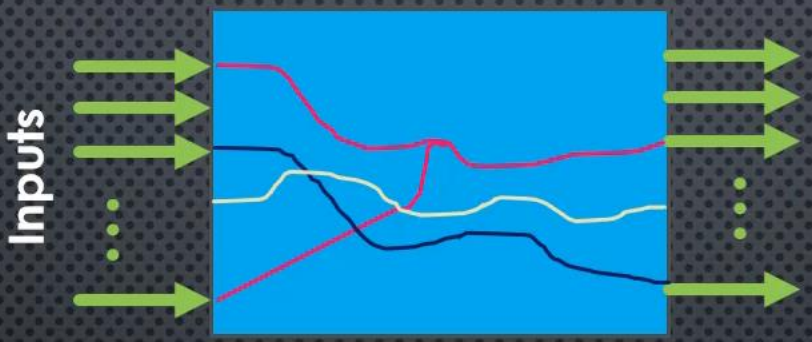
P2(*c → g2 → f*).....2ns

P3(*a → g4 → h*).....3ns

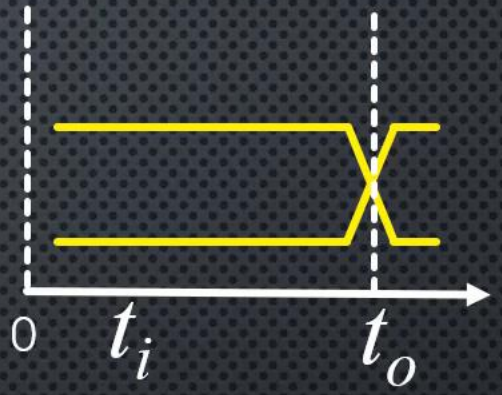
P4(*c or d → g3 → g4 → h*).....5ns

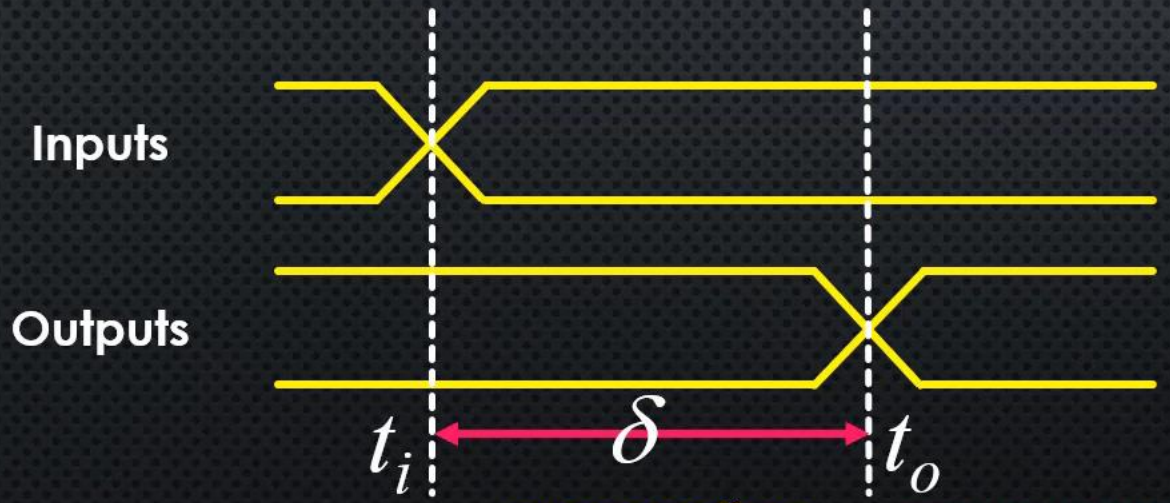
PROPAGATION DELAY





Combinational Circuit





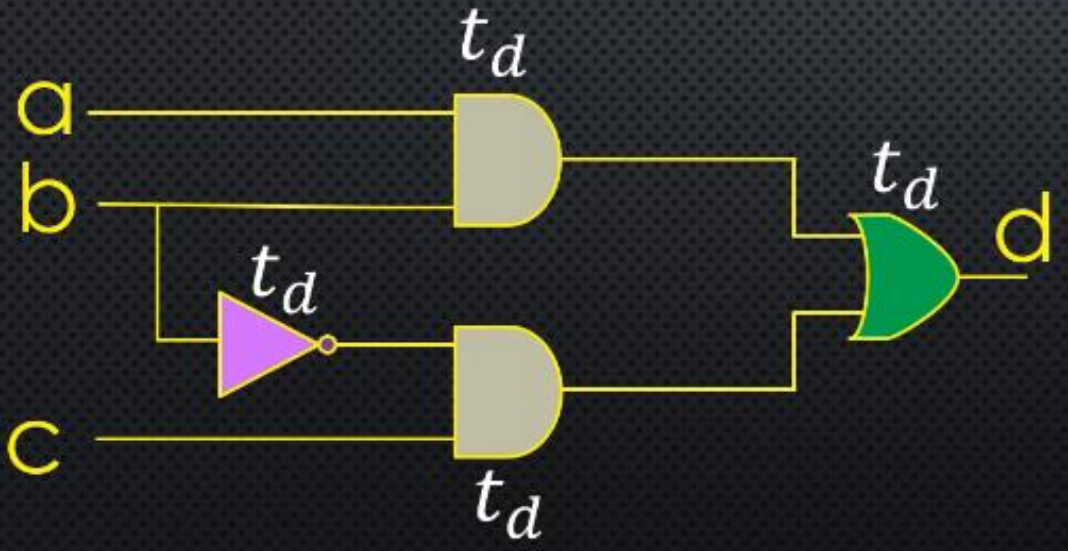
$$\delta = t_o - t_i$$

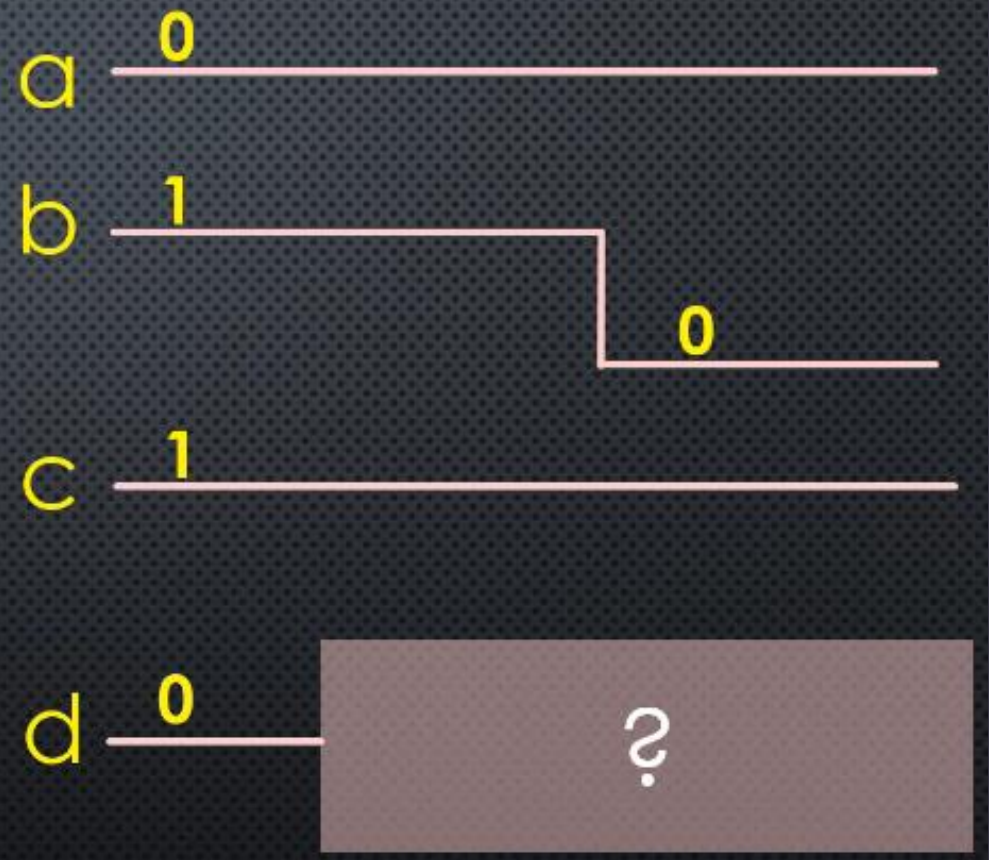
Critical Path defines this delay

ASSIGNMENT Q

Let's assume the following circuit in which all gates have the same propagation delay denoted by t_d.

Find the output signal diagram for these input values.

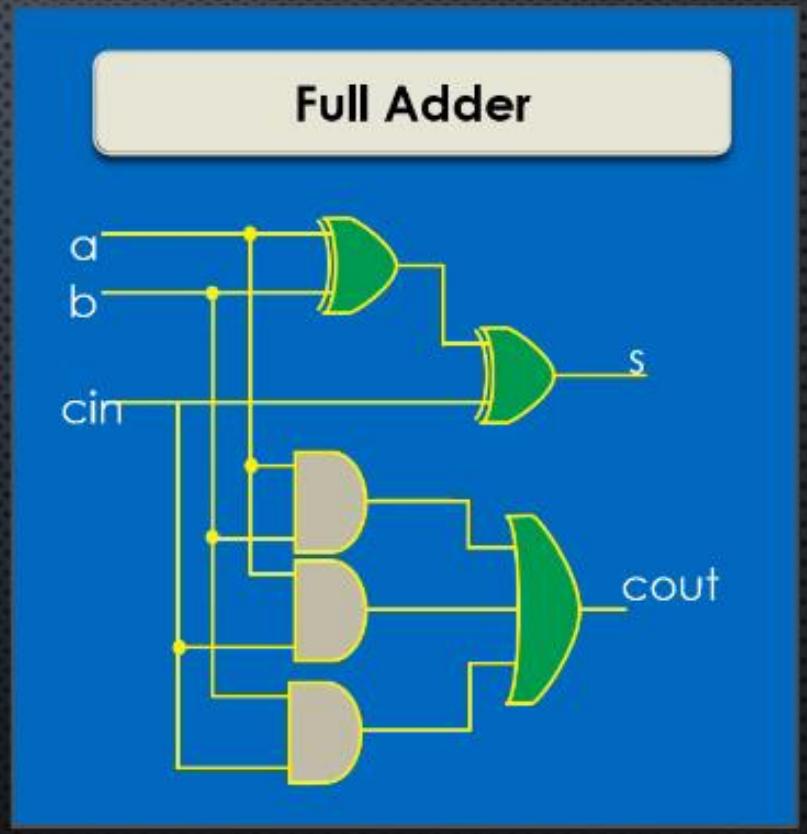




ASSIGNMENT Q

Let's assume the full-adder circuit in which all gates have the same propagation delay denoted by t_d.

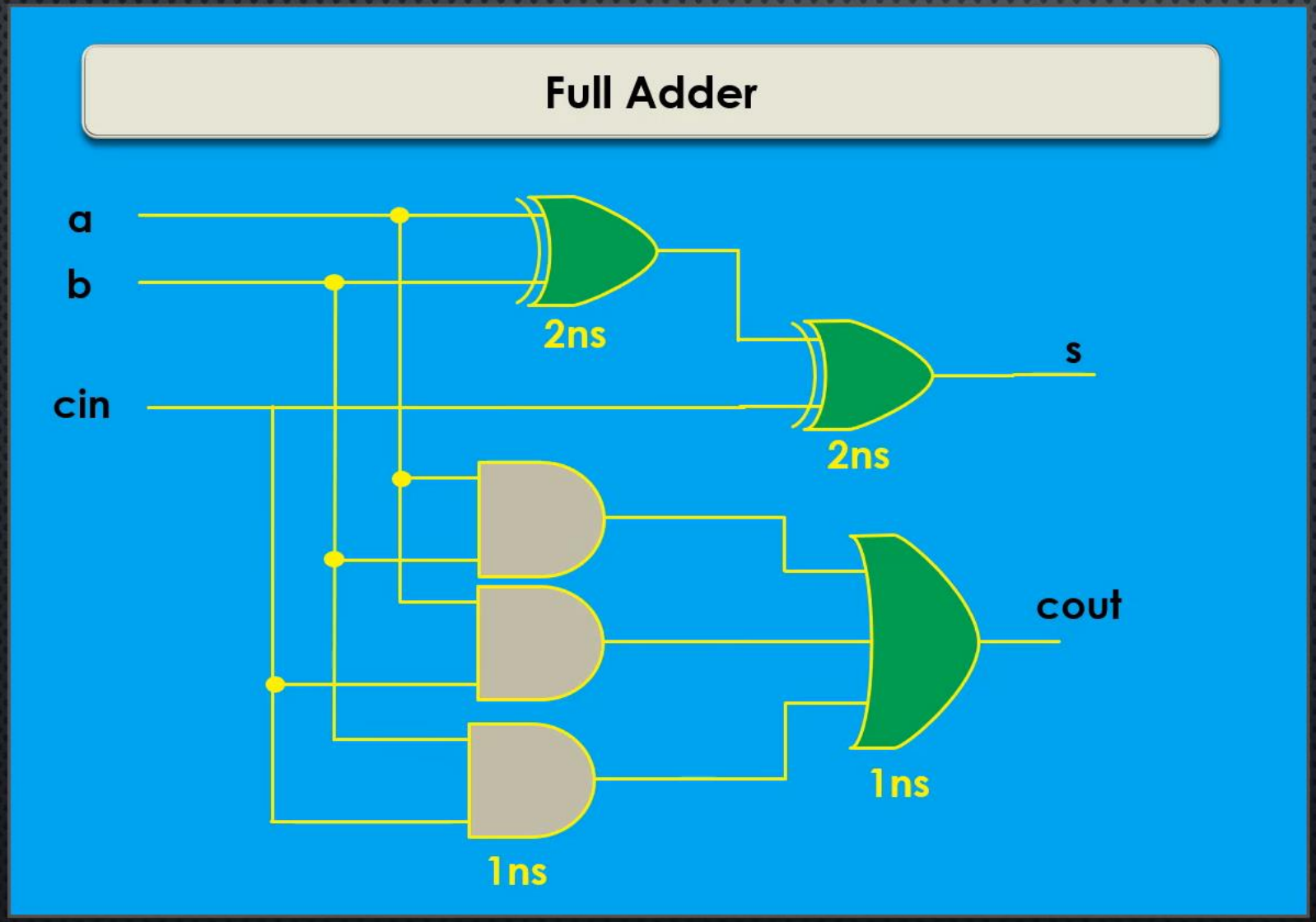
Find the propagation delay of this circuit.



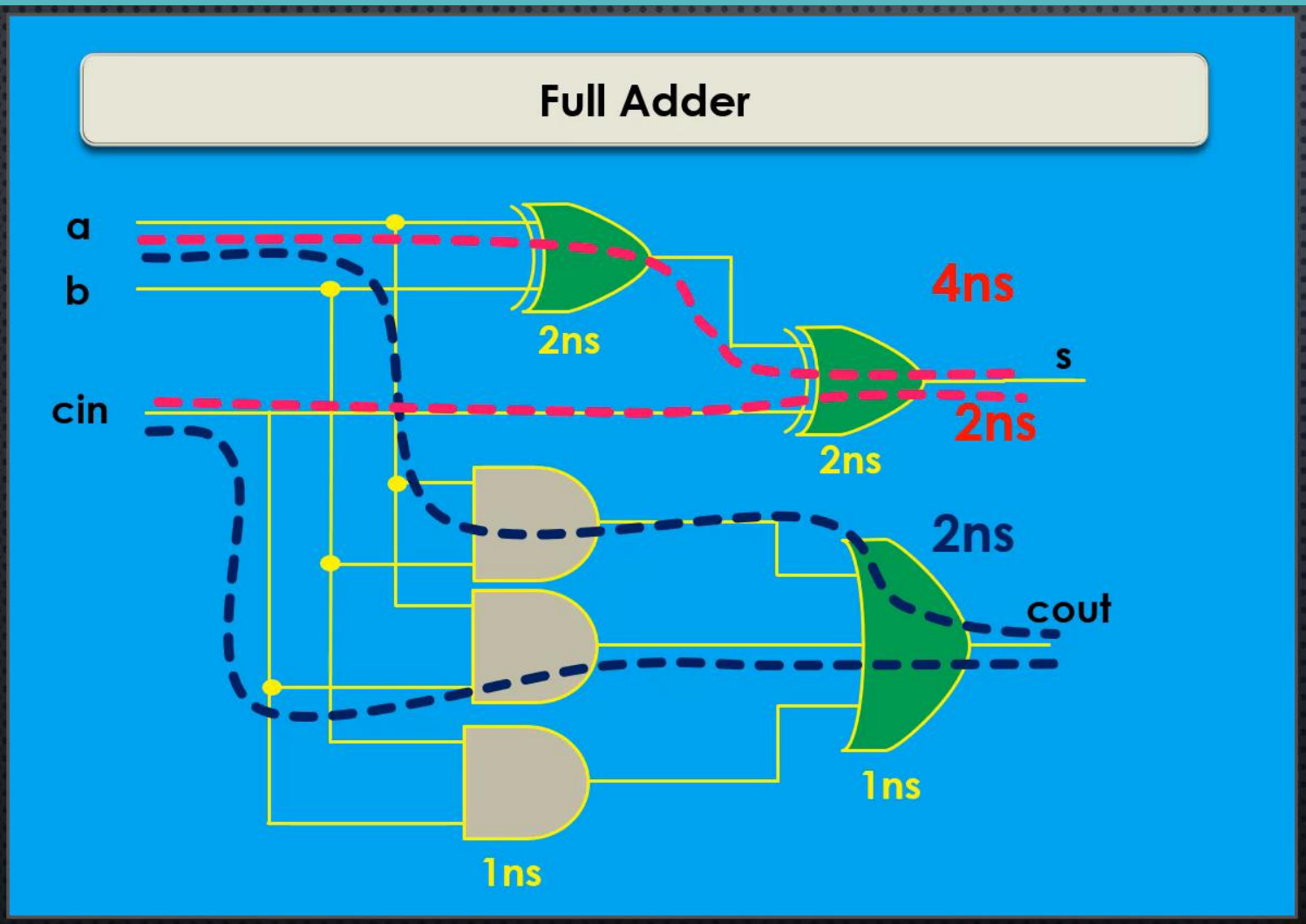
BINARY ADDER DELAY

Combinational circuit

FULL ADDER DELAY



FULL ADDER DELAY



a→s 4ns

b→s 4ns

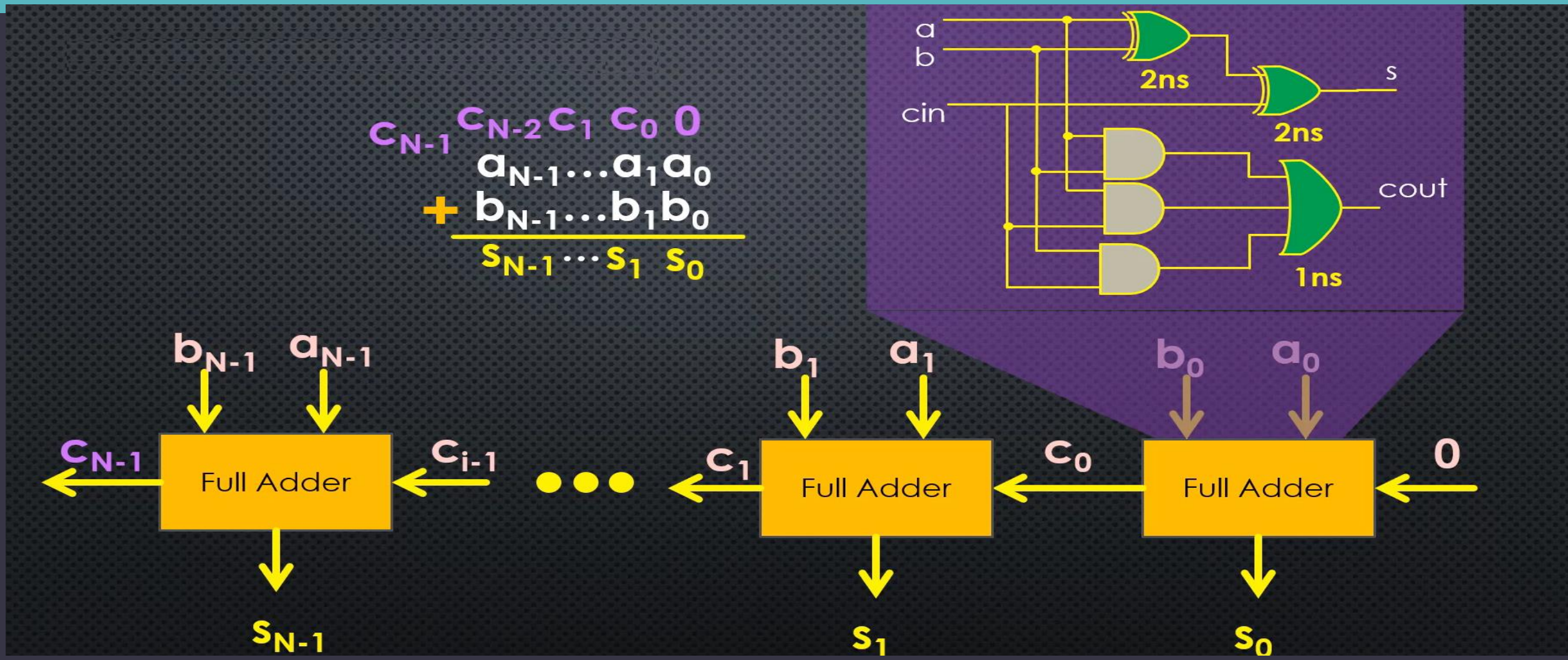
cin→s 2ns

a→cout 2ns

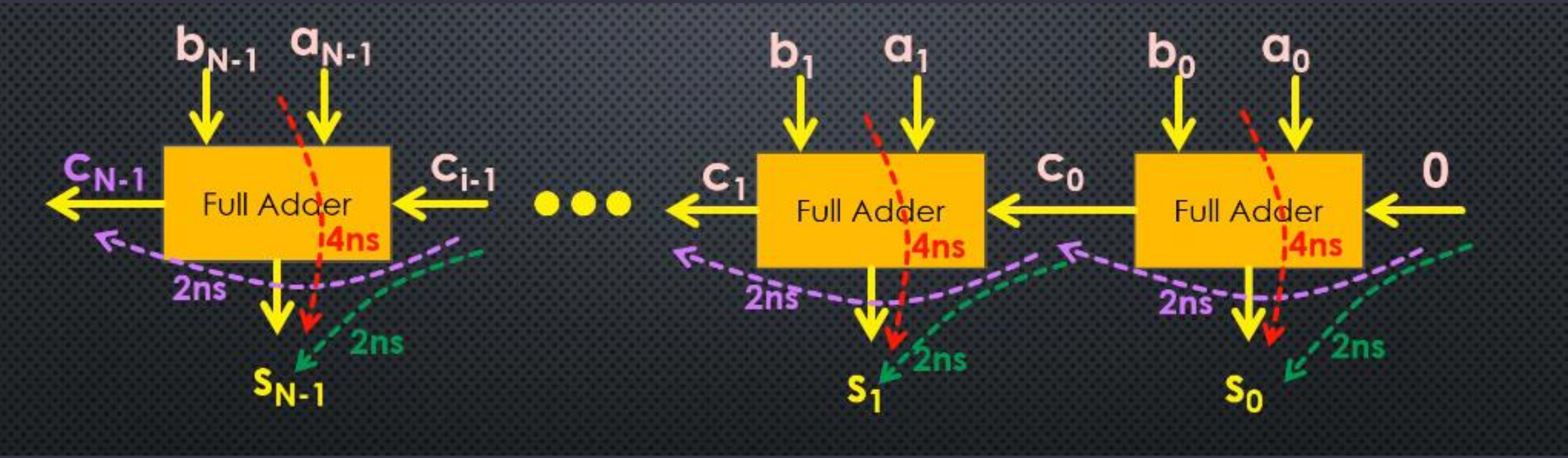
b→cout 2ns

cin→cout 2ns

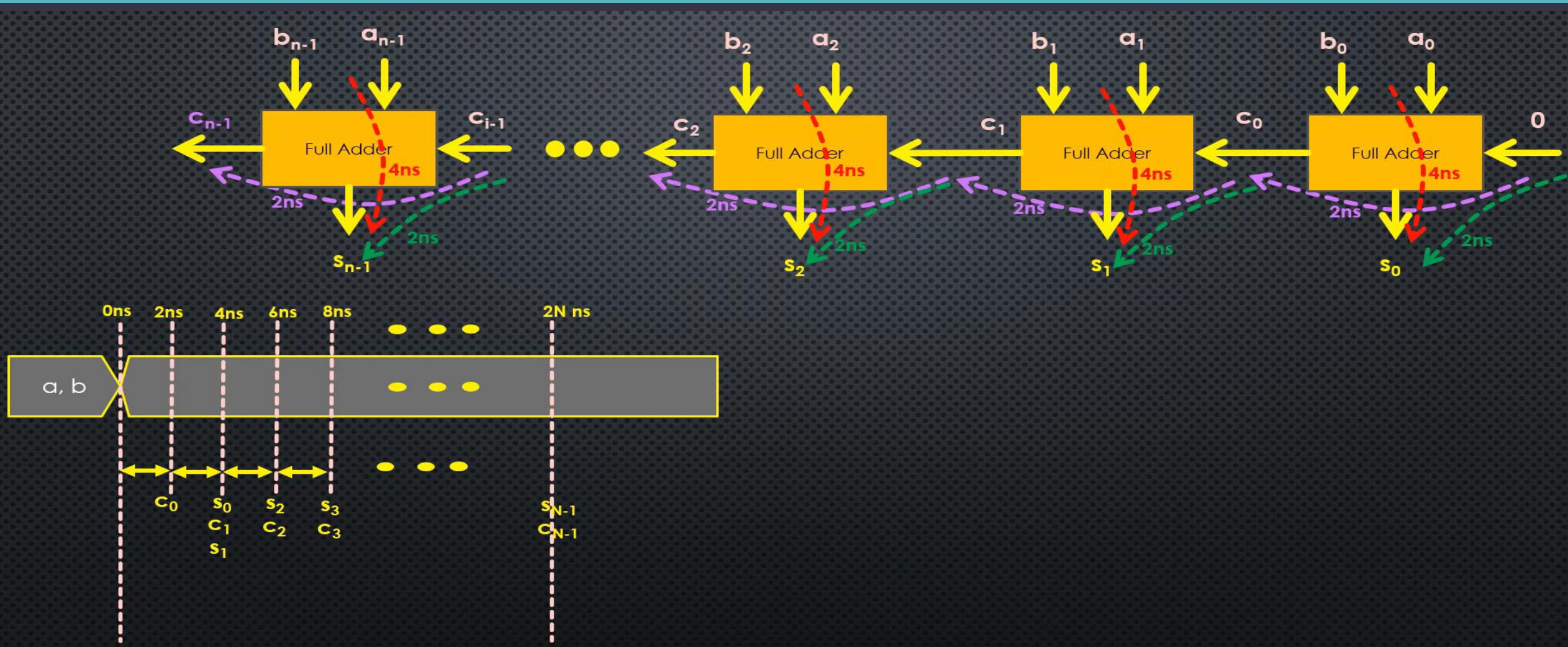
N-BIT BINARY ADDER



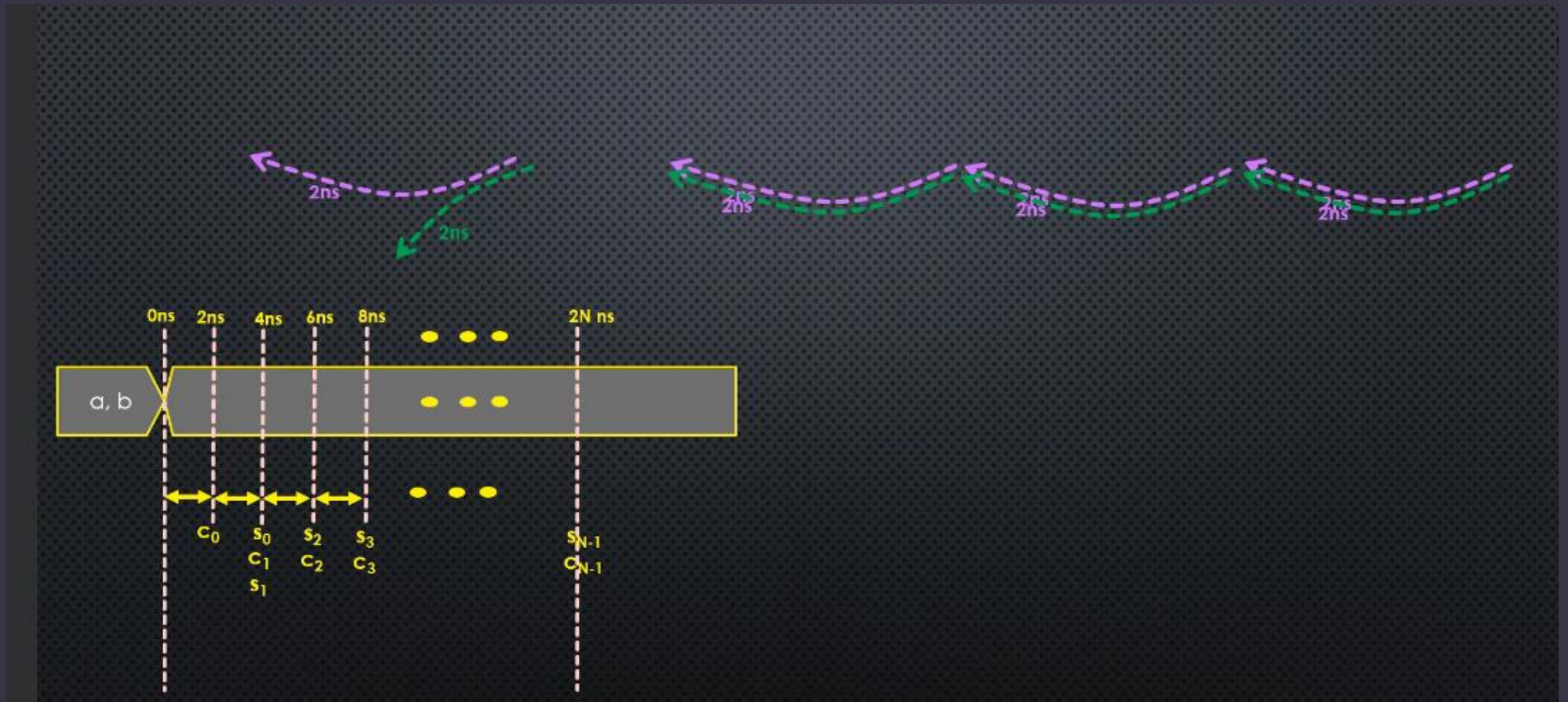
N-BIT BINARY ADDER PATHS



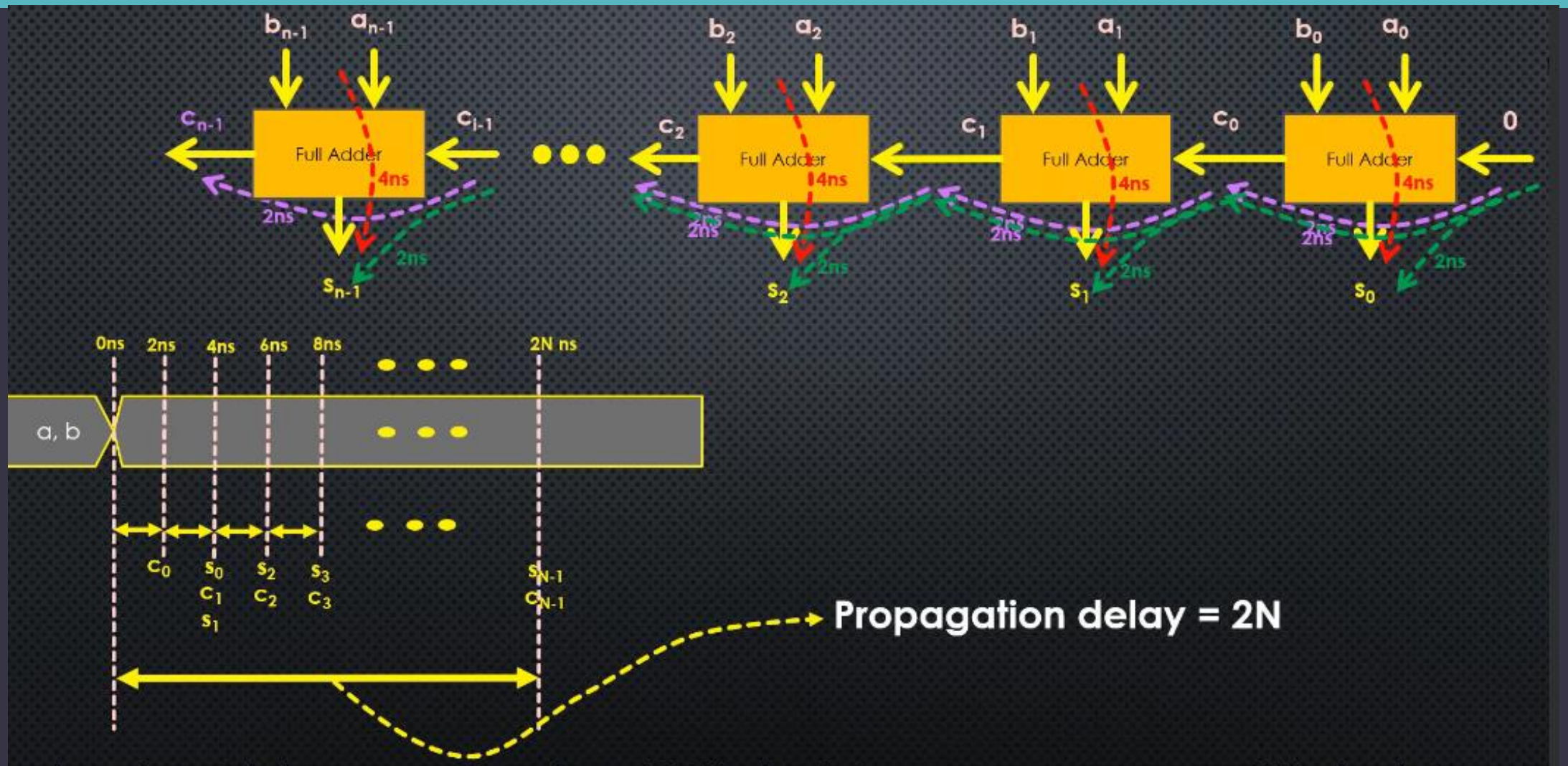
N-BIT ADDER PROPAGATION DELAY



N-BIT ADDER PROPAGATION DELAY



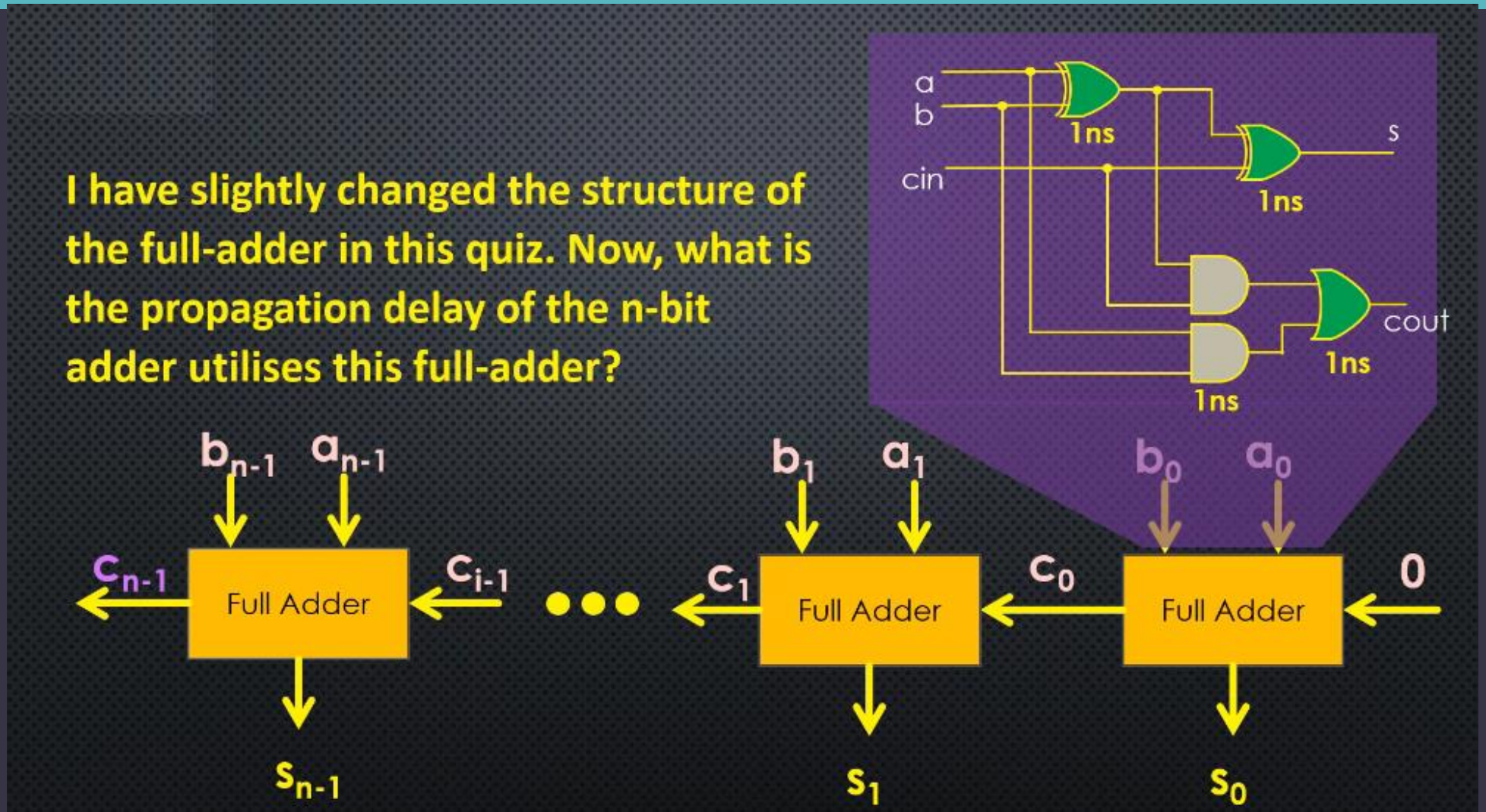
N-BIT ADDER PROPAGATION DELAY



SUMMARY

Serially connected modules in hardware increase the corresponding propagation delay.

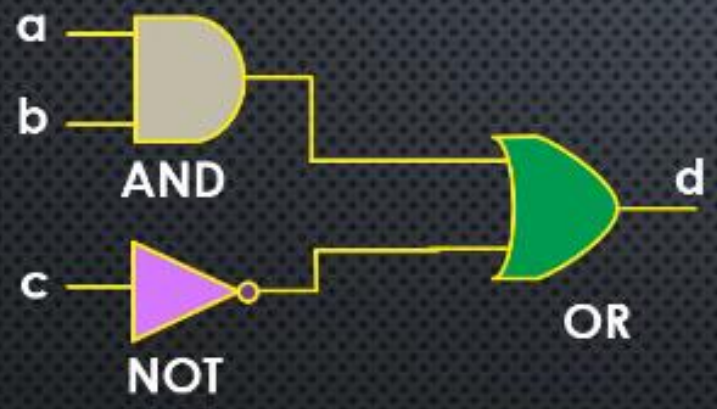
ASSIGNMENT Q



HLS

Combinational circuit

A SIMPLE COMBINATIONAL CIRCUIT



Logic	C/C++ Logical Operators	C/C++ Bitwise Operators
AND	&&	&
OR	||	|
NOT	!	~

$$d = ab + \overline{c}$$

$$d = (a\ AND\ b)\ OR\ (NOT\ c)$$

In C/C++: $d = (a \&\& b) \,||\, !\,c;$

or $d = (a \,\&\, b) \,|\, \sim c;$

A SIMPLE COMBINATIONAL CIRCUIT

```
void simple_combinational_circuit (
                bool    a, bool    b, bool    c,
                bool    *d ) {









  *d = (a && b)  | |  !c;
}
```

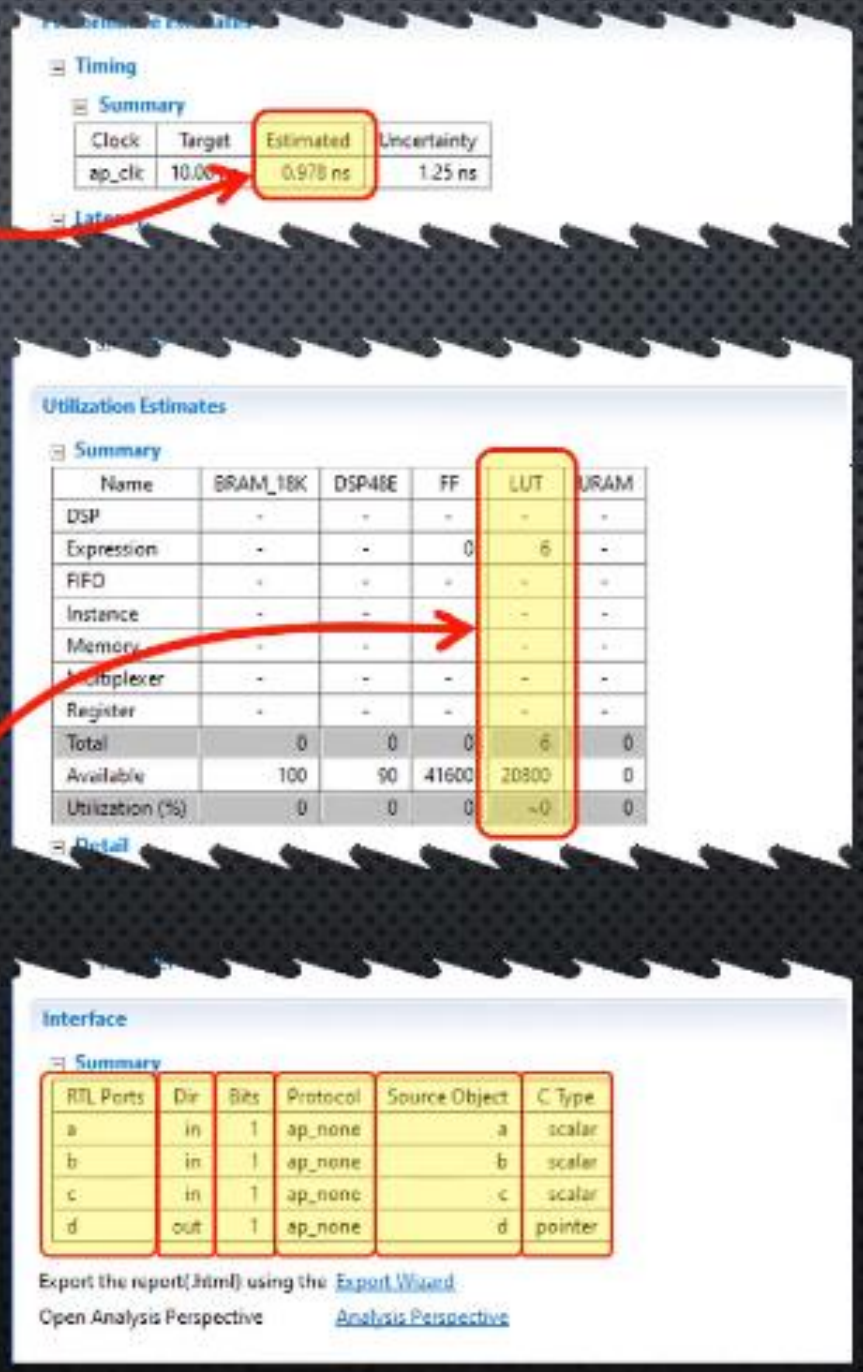
A SIMPLE COMBINATIONAL CIRCUIT

```
void simple_combinational_circuit (
                bool    a, bool    b, bool    c,
                bool    *d ) {
#pragma HLS INTERFACE ap_none        port=a
#pragma HLS INTERFACE ap_none        port=b
#pragma HLS INTERFACE ap_none        port=c
#pragma HLS INTERFACE ap_none        port=d
#pragma HLS INTERFACE ap_ctrl_none   port=return

  *d = (a && b)  | |  !c;
}
```

Propagation delay estimation

```
void simple_combinational_circuit (
                bool    a, bool    b, bool    c,
                bool    *d ) {
#pragma HLS INTERFACE ap_none        port=a
#pragma HLS INTERFACE ap_none        port=b
#pragma HLS INTERFACE ap_none        port=c
#pragma HLS INTERFACE ap_none        port=d
#pragma HLS INTERFACE ap_ctrl_none   port=return

  *d = (a && b) || !c;
}
```

Resource utilization



Timing

Summary

Clock	Target	Estimated	Uncertainty
ap_clk	10.00	0.978 ns	1.25 ns

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	6	-
FIFO	-	-	-	-	-
Instance	-	-	-	-	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	-	-
Register	-	-	-	-	-
Total	0	0	0	6	0
Available	100	90	41600	20800	0
Utilization (%)	0	0	0	~0	0

Detail

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
a	in	1	ap_none	a	scalar
b	in	1	ap_none	b	scalar
c	in	1	ap_none	c	scalar
d	out	1	ap_none	d	pointer

Export the report(.html) using the Export Wizard

Open Analysis Perspective Analysis Perspective

TAKEAWAY MESSAGE

- The HLS report of combinational circuit synthesised by vivado-HLS
  - Should not utilise any memory elements
  - Should not have any clock port in the interface part

ASSIGNMENT Q

Complete this code by defining the port interfaces using HLS pragmas.

```
bool simple_combinational_circuit (bool    a, bool    b, bool    c) {



  return (a && b)  | |  !c;
}
```

HLS LAB

Combinational circuit

ASSIGNMENT Q

Generate the RTL-IP corresponding to this simple combinational circuit.

```
bool simple_combinational_circuit (bool    a, bool    b, bool    c) {



  return (a && b)  | |  !c;
}
```

Any Question...

Thank you