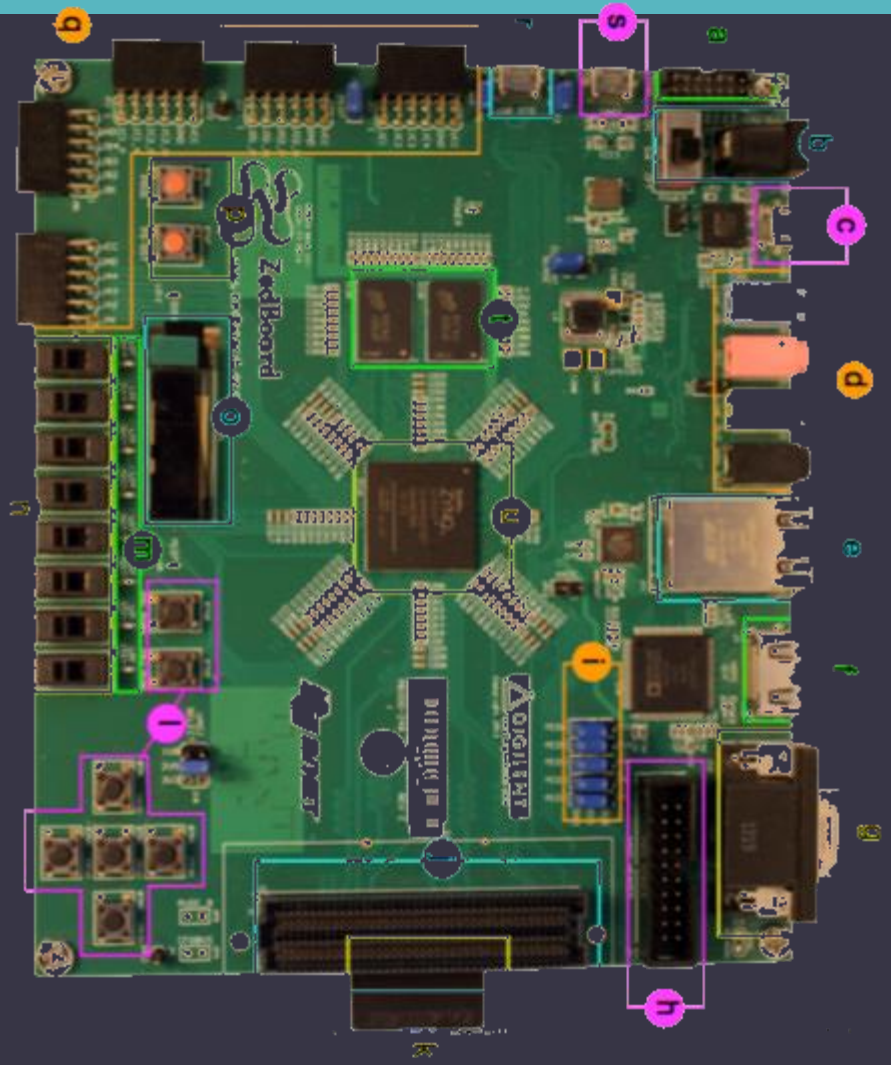
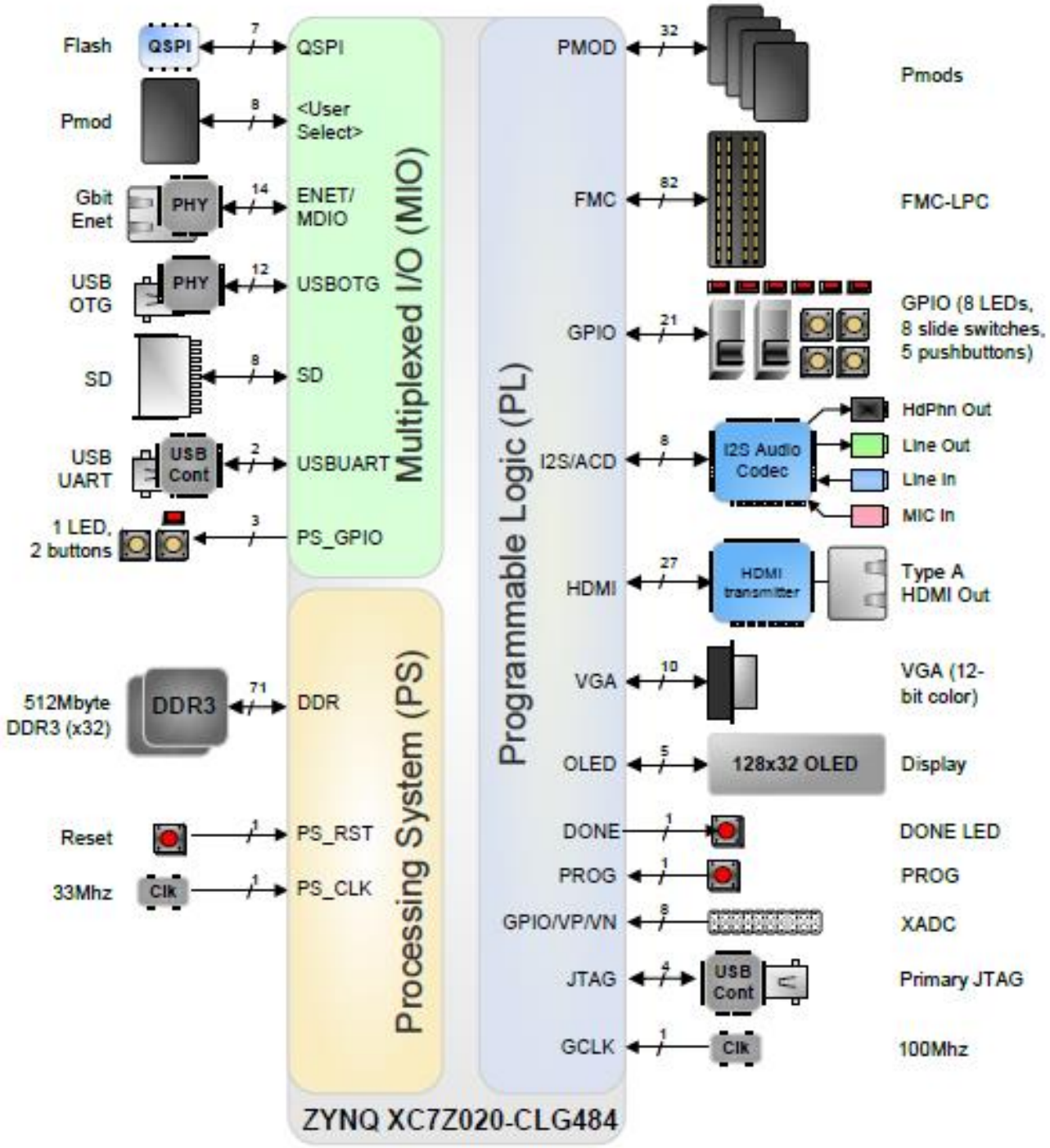
BASIC OUTPUT - CONFIGURATION

Lecture - 2

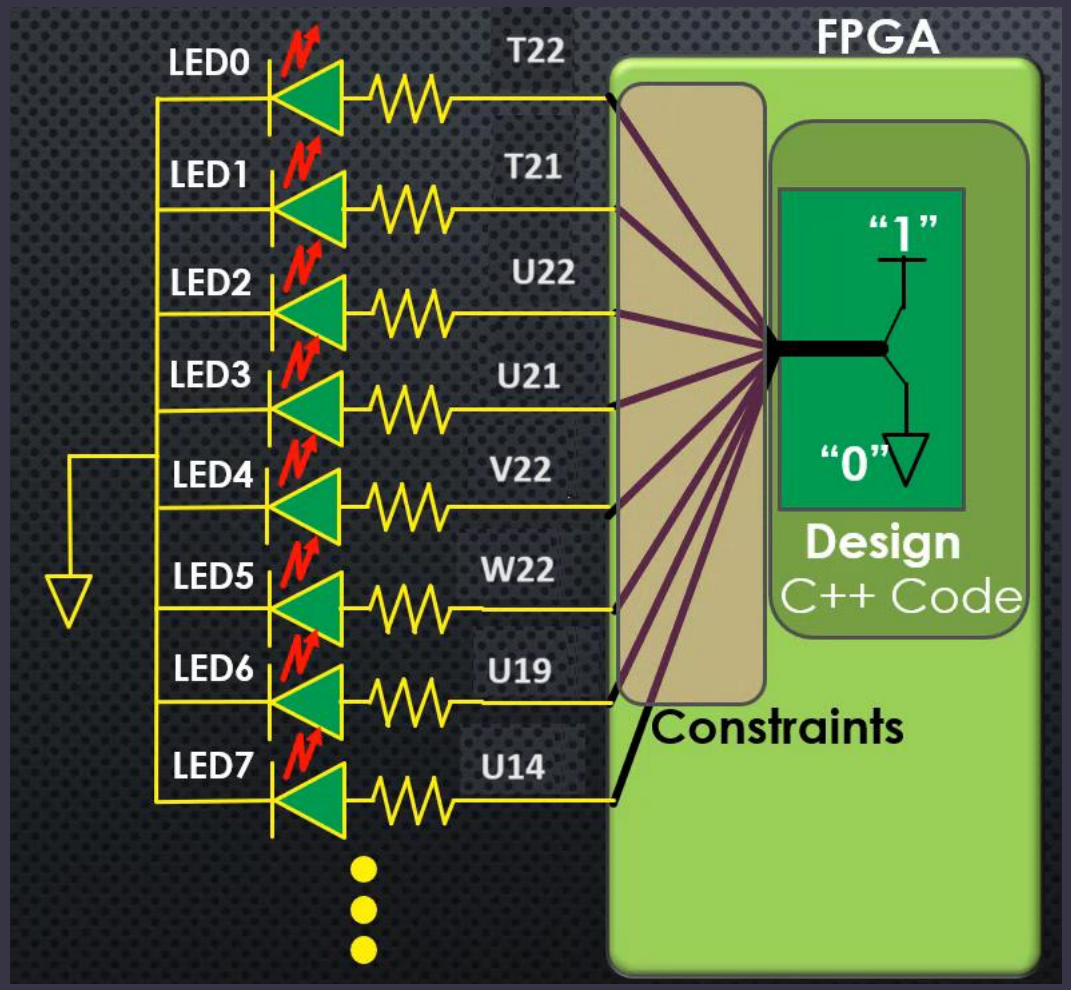
FPGA LED STRUCTURE

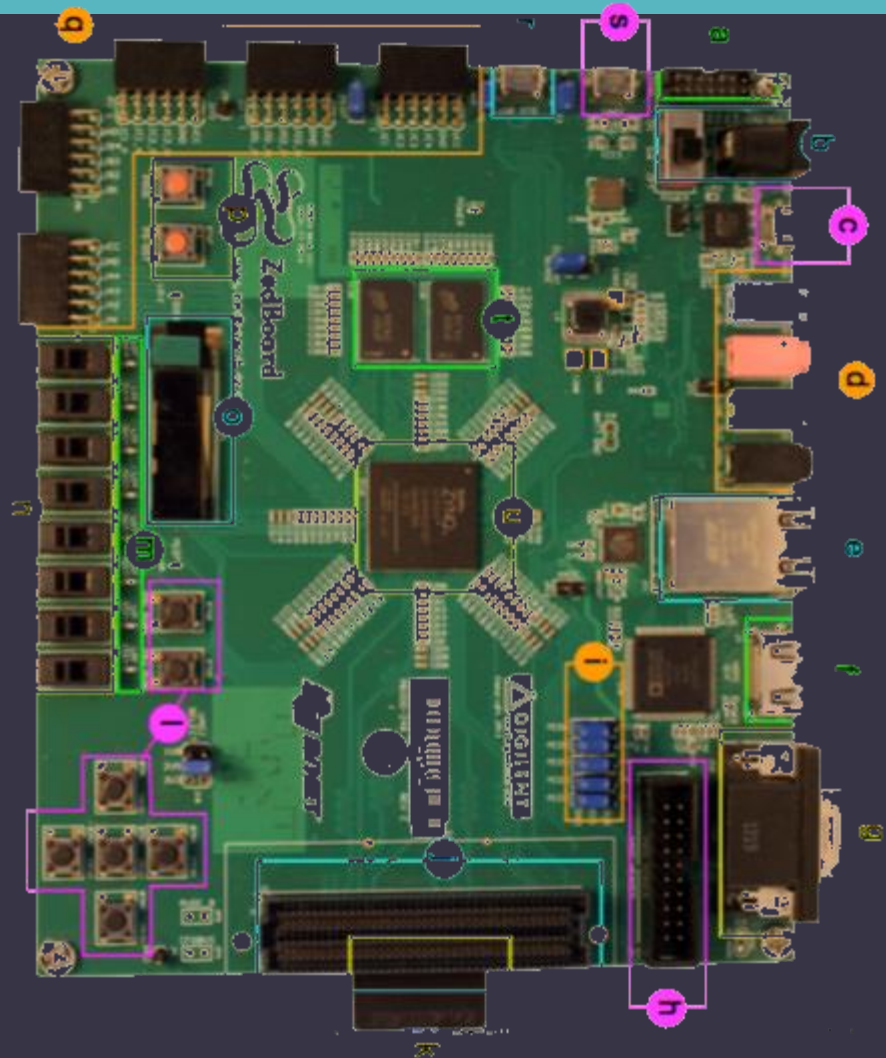
Signal Name	Subsection	Zynq pin
LD0	PL	T22
LD1	PL	T21
LD2	PL	U22
LD3	PL	U21
LD4	PL	V22
LD5	PL	W22
LD6	PL	U19
LD7	PL	U14
LD9	PS	D5 (MIO7)



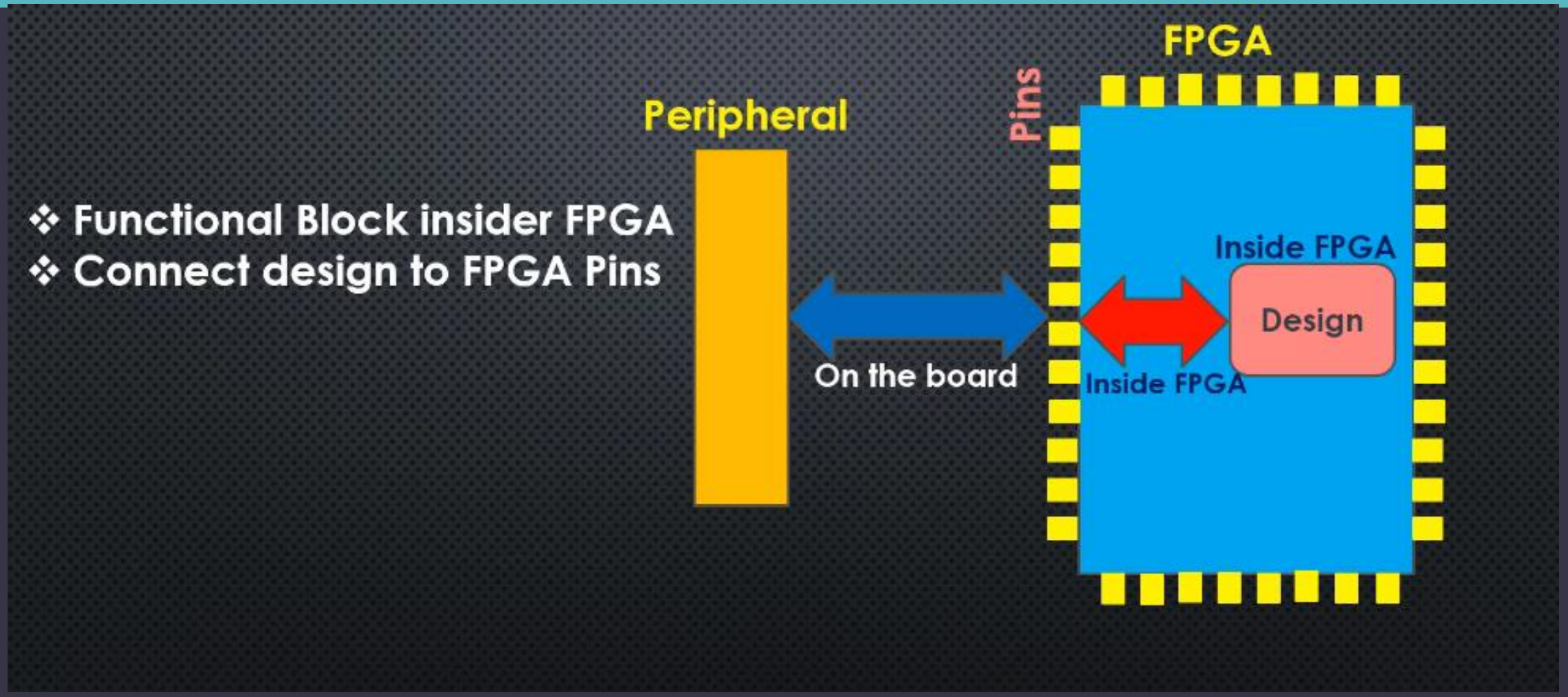

Flash
QSPI
7
QSPI
Pmod
8
<User Select>
Gbit Enet
PHY
14
ENET/ MDIO
USB OTG
PHY
12
USBOTG
SD
8
SD
USB UART
USB Cont
2
USBUART
1 LED, 2 buttons
3
PS_GPIO
Multiplexed I/O (MIO)
512Mbyte DDR3 (x32)
DDR3
71
DDR
Reset
1
PS_RST
33Mhz
Clk
1
PS_CLK
Processing System (PS)
ZYNQ XC7Z020-CLG484
Programmable Logic (PL)
PMOD
32
Pmods
FMC
82
FMC-LPC
GPIO
21
GPIO (8 LEDs, 8 slide switches, 5 pushbuttons)
I2S/ACD
8
I2S Audio Codec
HdPhn Out
Line Out
Line In
MIC In
HDMI
27
HDMI transmitter
Type A HDMI Out
VGA
10
VGA (12-bit color)
OLED
5
128x32 OLED
Display
DONE
1
DONE LED
PROG
1
PROG
GPIO/VP/VN
8
XADC
JTAG
4
USB Cont
Primary JTAG
GCLK
1
Clk
100Mhz

CONCEPT


FPGA
LED0
LED1
LED2
LED3
LED4
LED5
LED6
LED7
T22
T21
U22
U21
V22
W22
U19
U14
"1"
"0"
Design
C++ Code
Constraints


ZedBoard

SUMMARY


❖ Functional Block insider FPGA
❖ Connect design to FPGA Pins
Peripheral
On the board
FPGA
Pins
Inside FPGA
Design
Inside FPGA

ASSIGNMENT Q-1

Which part can be implemented with an HLS C/C++ function?

1) Logic circuit functionality inside the FPGA

2) Connections between the logic circuit boards and the FPGA pins

3) Connections between the FPGA pins and the LEDs on the board

HLS DESIGN OVERVIEW

Lecture - 2

HLS DESIGN

Vivado HLS

High-Level Synthesis

- Functionality in C/C++
- Port Interfaces
- Optimizations
- Simulation
- Debugging
- Co-Simulation
- IP generator

Vivado

Logic Synthesis

- Integrate the HLS IP into system Design
- Add Constraints
- Synthesis
- Implementation
- Bitstream Generation

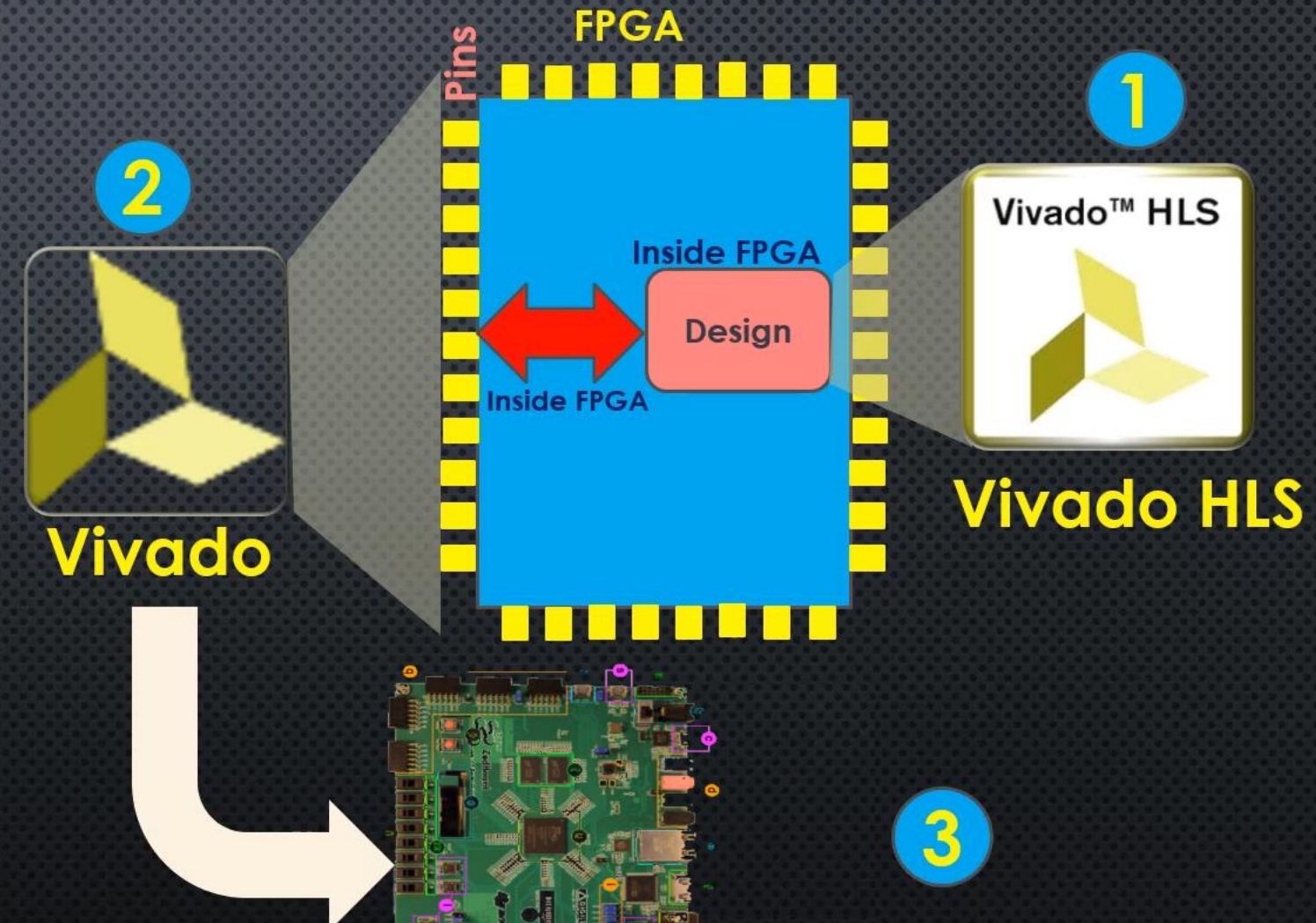
Hardware test

- Program the board
- Provide the inputs
- Investigate the Output

SUMMARY

Xilinx HLS design steps

- using Vivado-HLS to develop the design in C++
- Using Vivado to add constraints and perform logic synthesis
- Programming the board



ASSIGNMENT Q

Which two tasks can be performed in Vivado-HLS

1. Bitstream generation
2. Adding physical constraints
3. Co-simulation
4. Debugging
5. Logic Synthesis

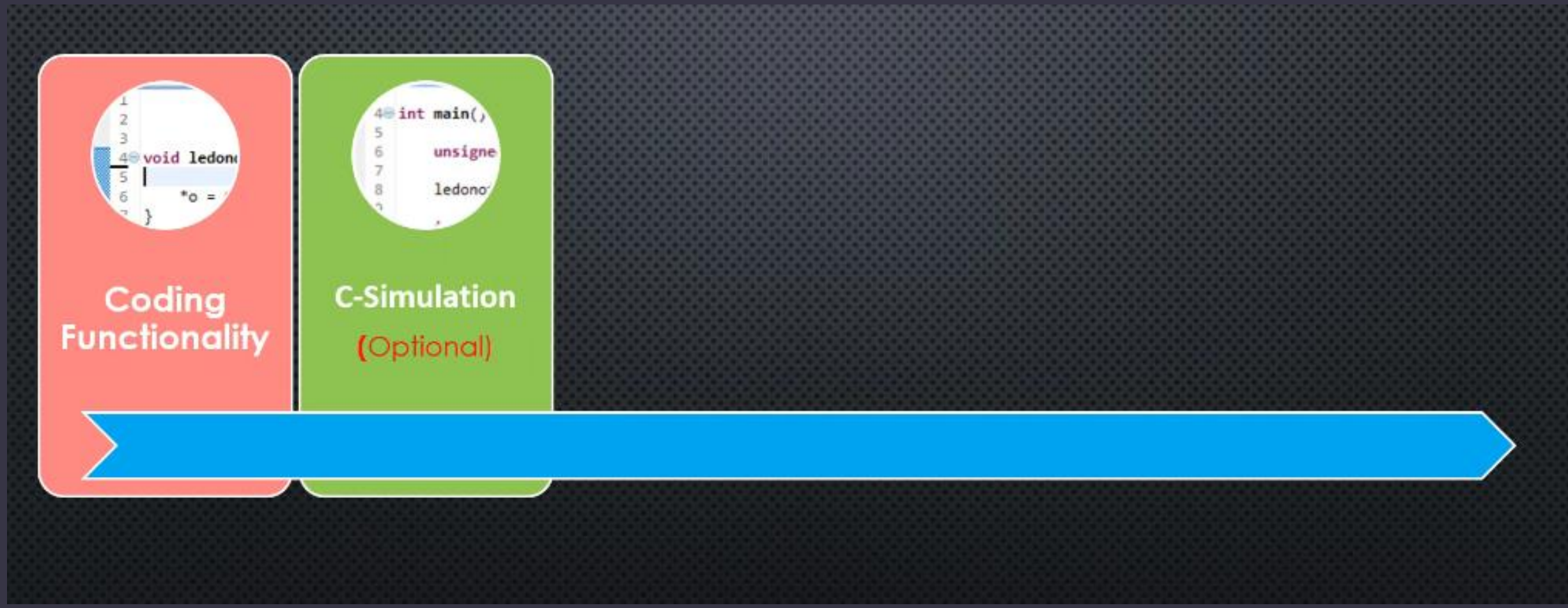
HLS DESIGN FLOW

Lecture - 2

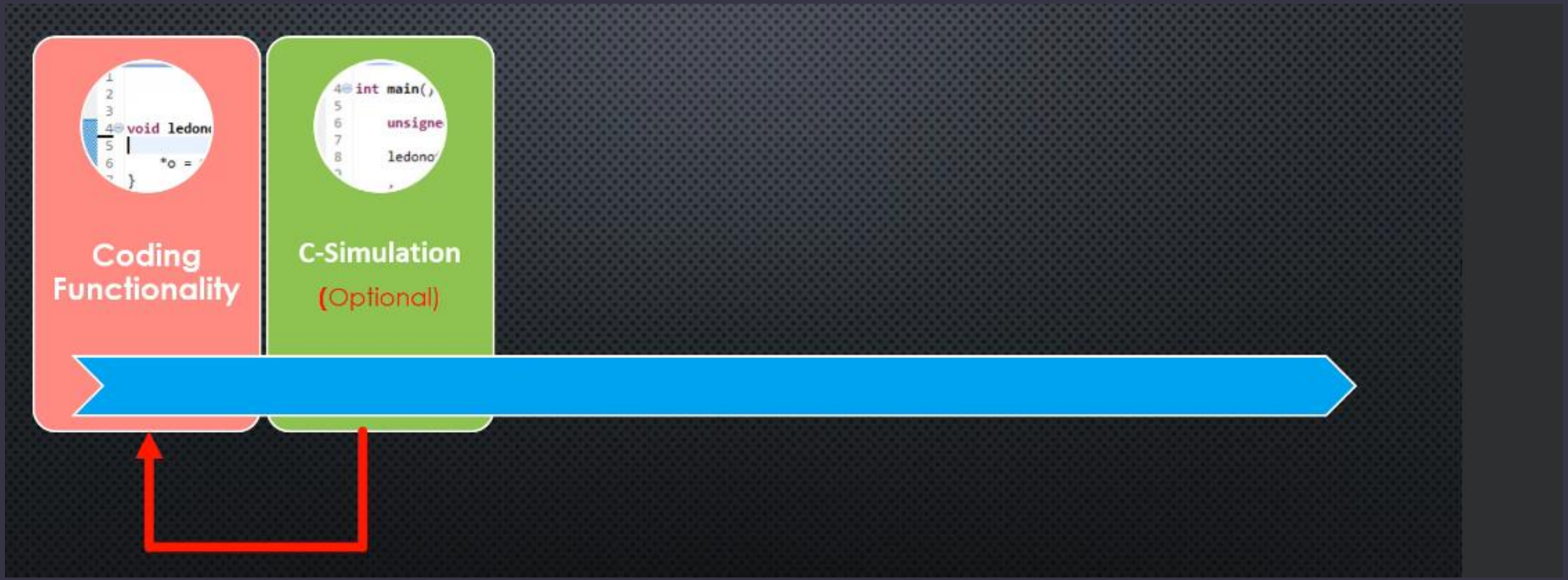
DESIGN FLOW IN VIVADO-HLS

Coding Functionality

DESIGN FLOW IN VIVADO-HLS

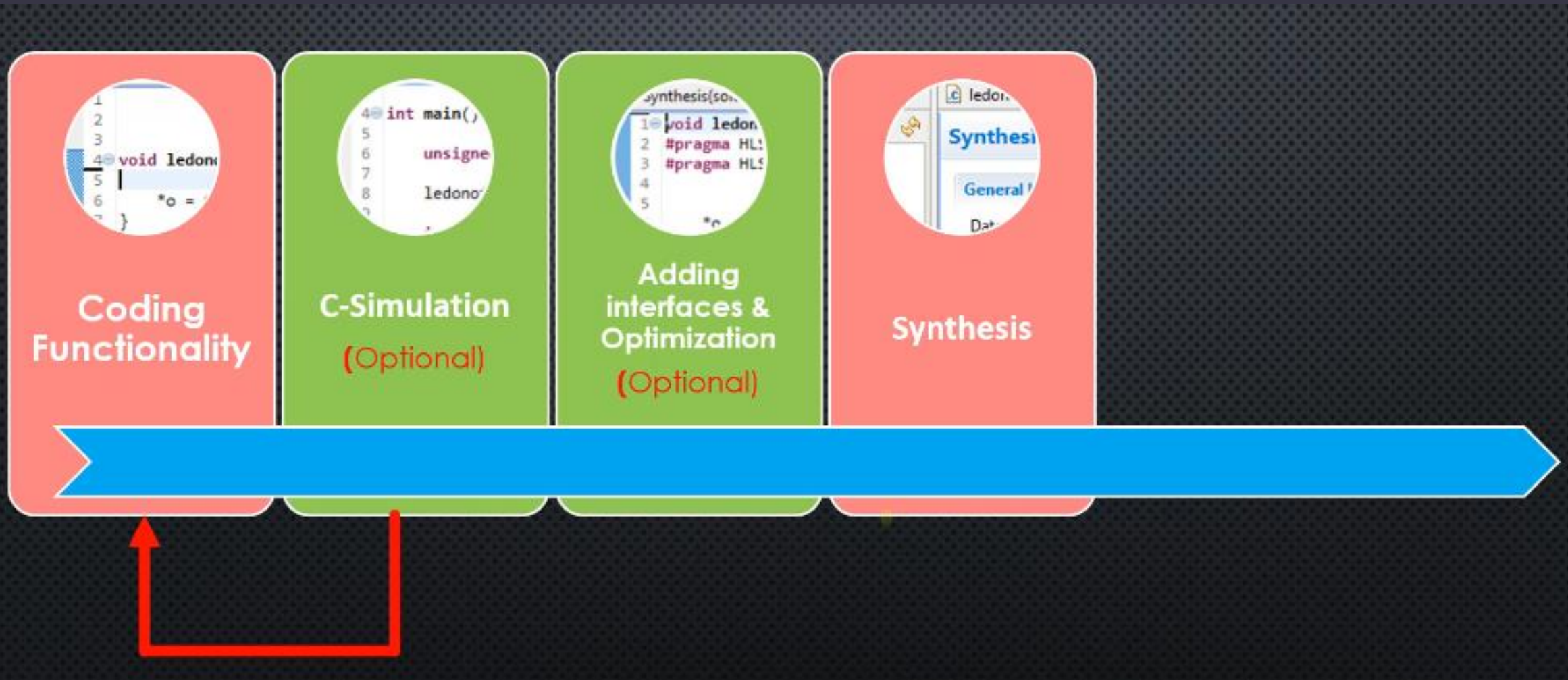


DESIGN FLOW IN VIVADO-HLS

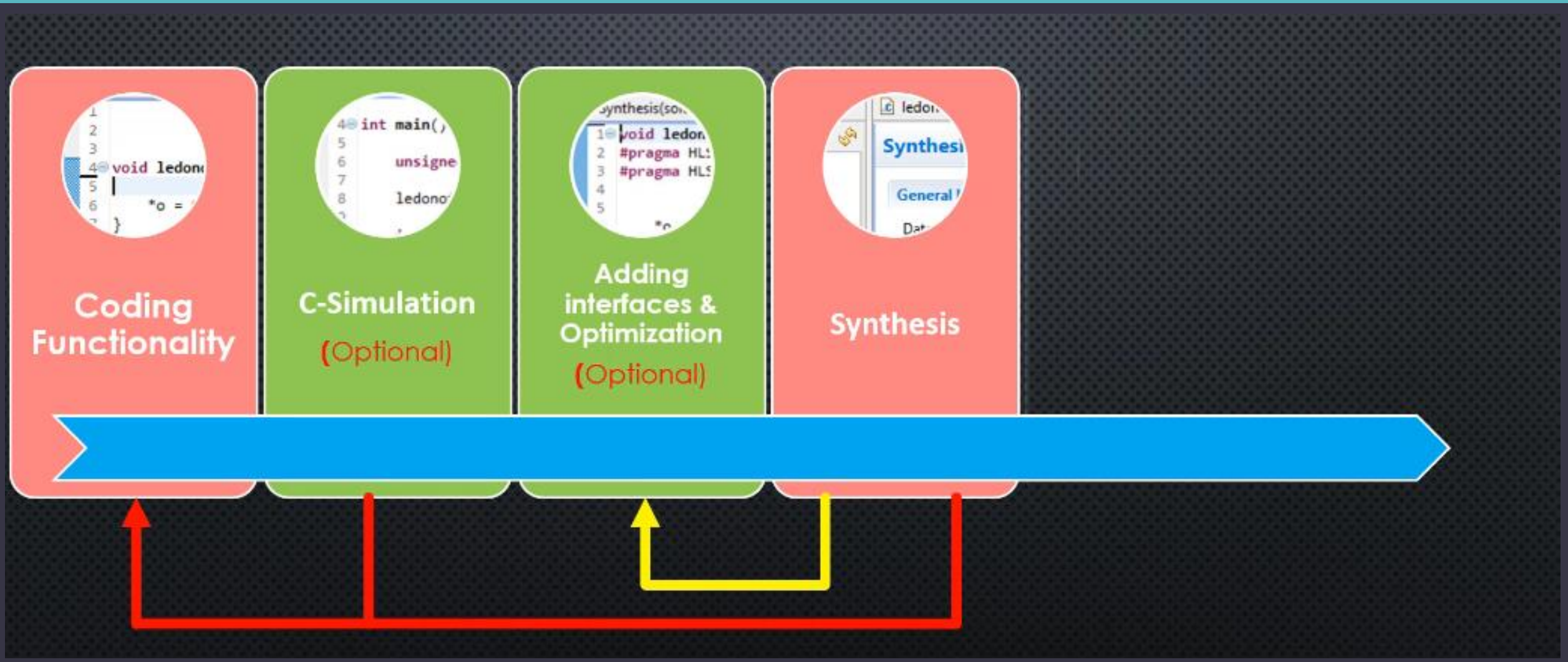


DESIGN FLOW IN VIVADO-HLS

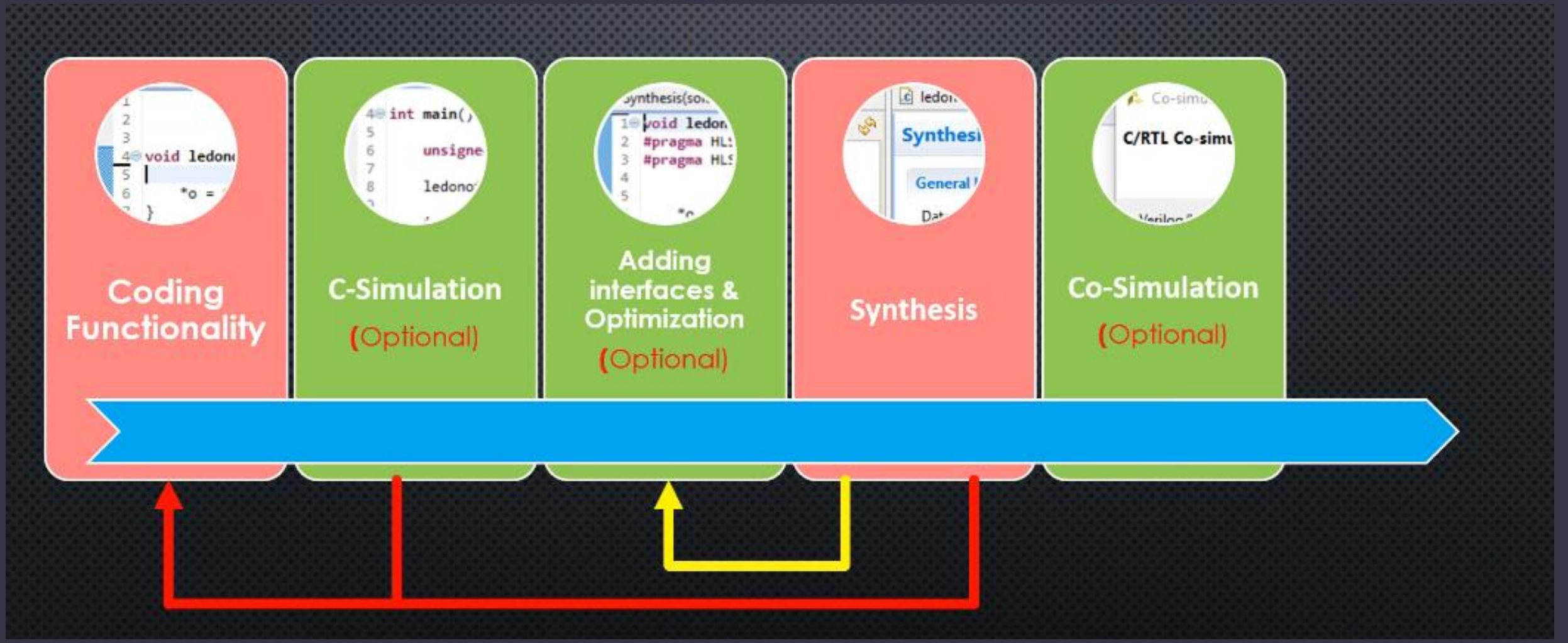
DESIGN FLOW IN VIVADO-HLS



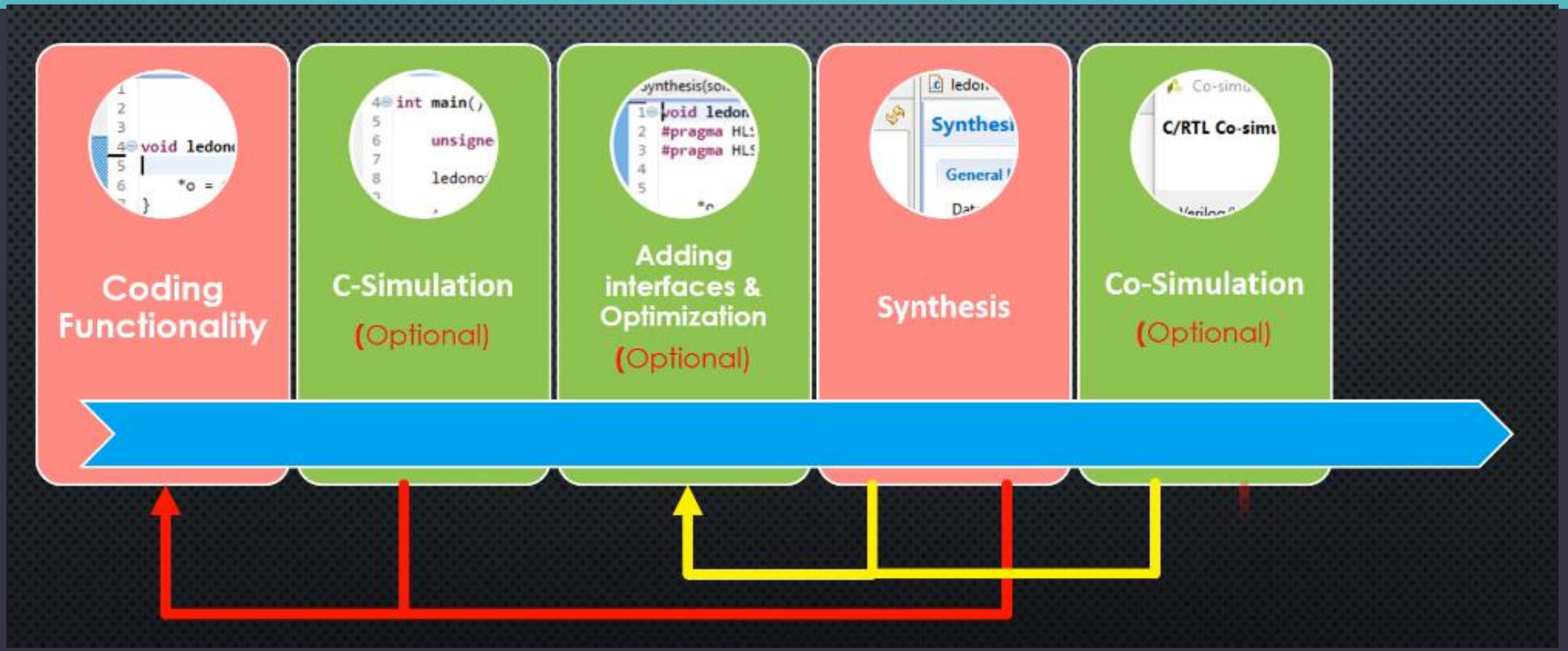
DESIGN FLOW IN VIVADO-HLS



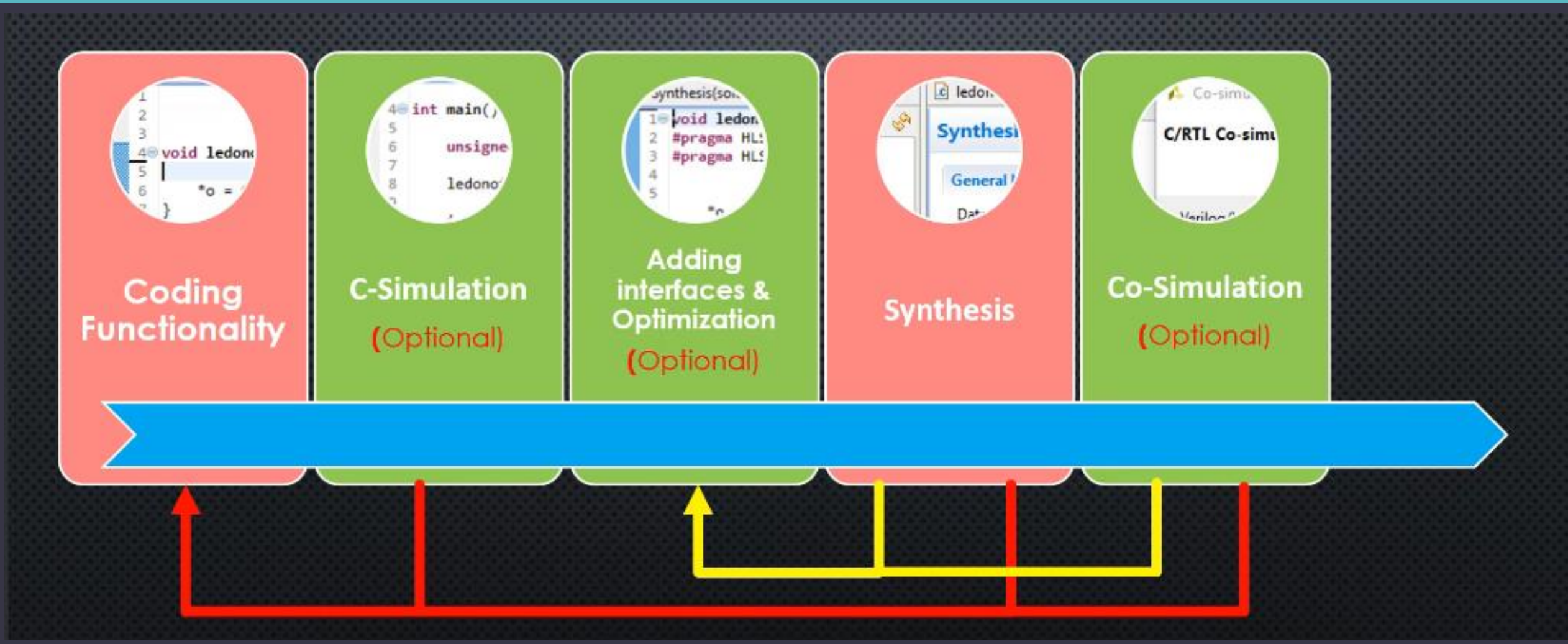
DESIGN FLOW IN VIVADO-HLS



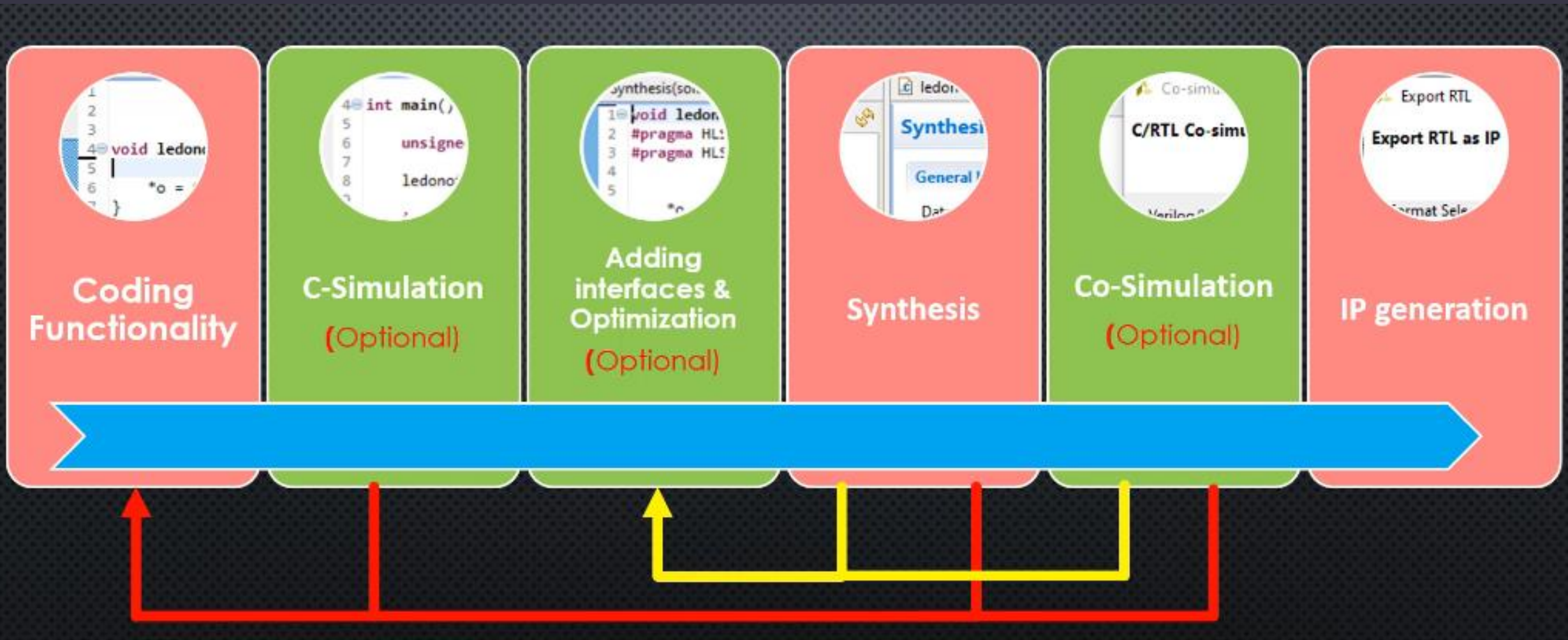
DESIGN FLOW IN VIVADO-HLS



DESIGN FLOW IN VIVADO-HLS



DESIGN FLOW IN VIVADO-HLS

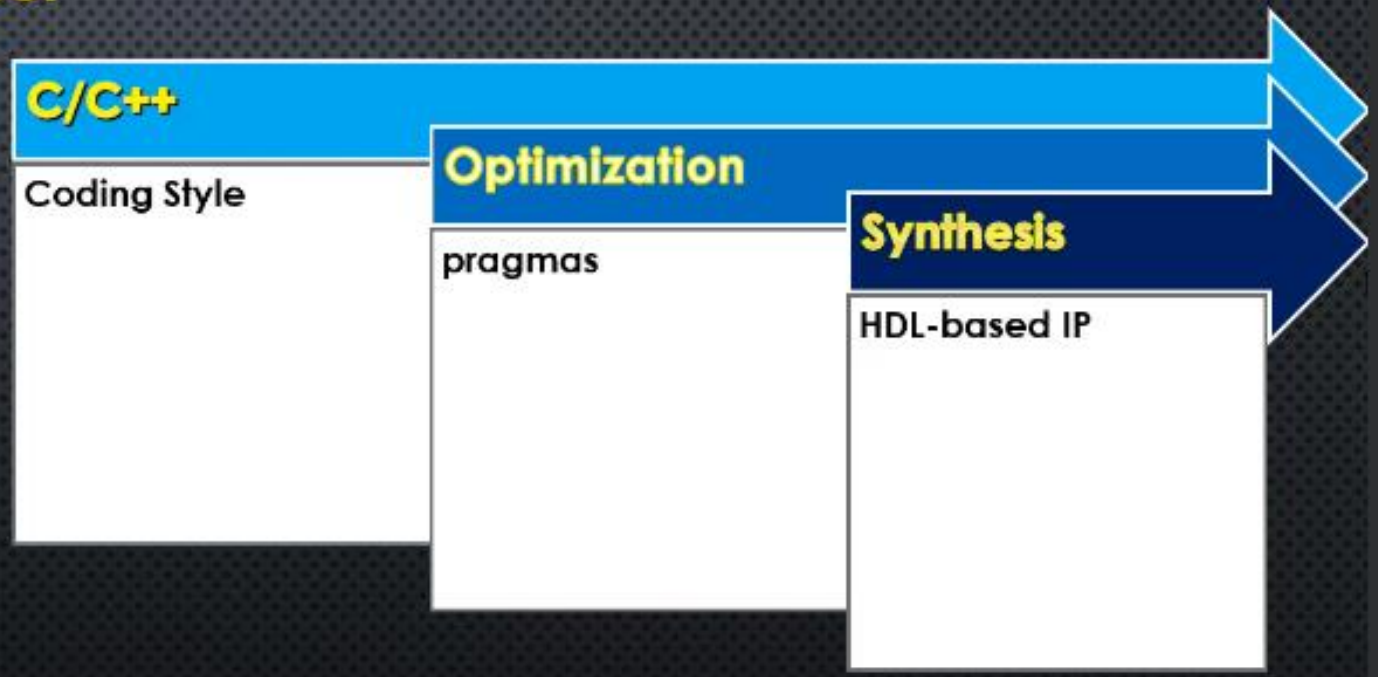


SUMMARY

1- Develop the hardware consider a proper C/C++ coding style

2- Add hardware optimization pragma to the code

3- Perform HLS synthesis to generate HDL based IP



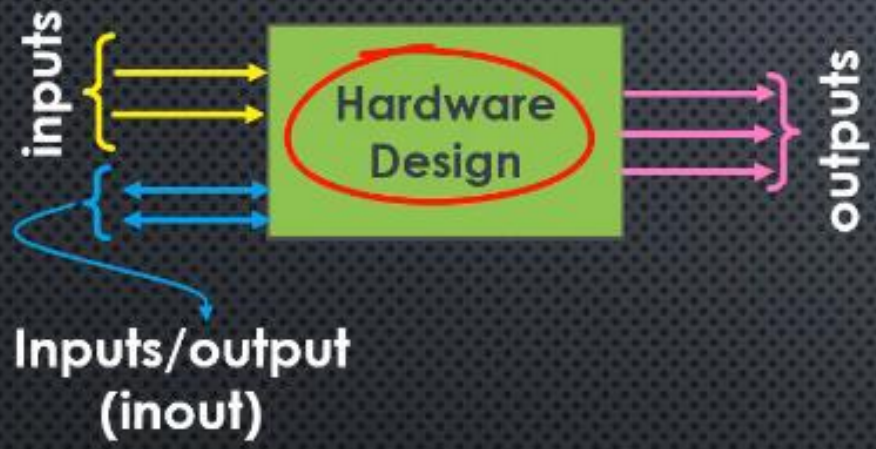
ASSIGNMENT Q

What is the difference between C-Simulation and RTL-Co-Simulation in Vivado-HLS?

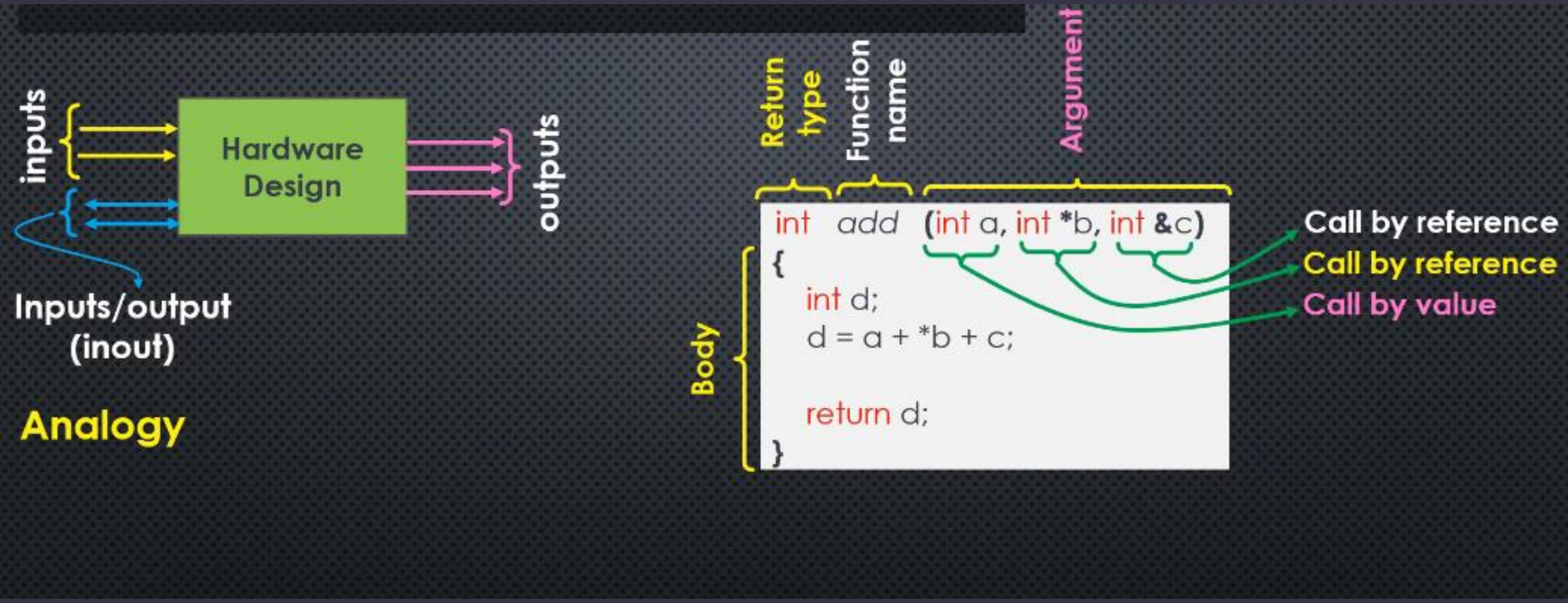
HLS C/C++

Lecture - 2

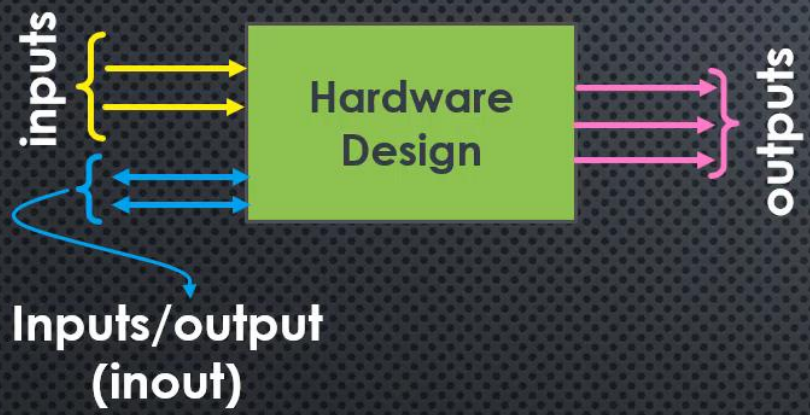
DESIGN (C/C++) IN VIVADO HLS

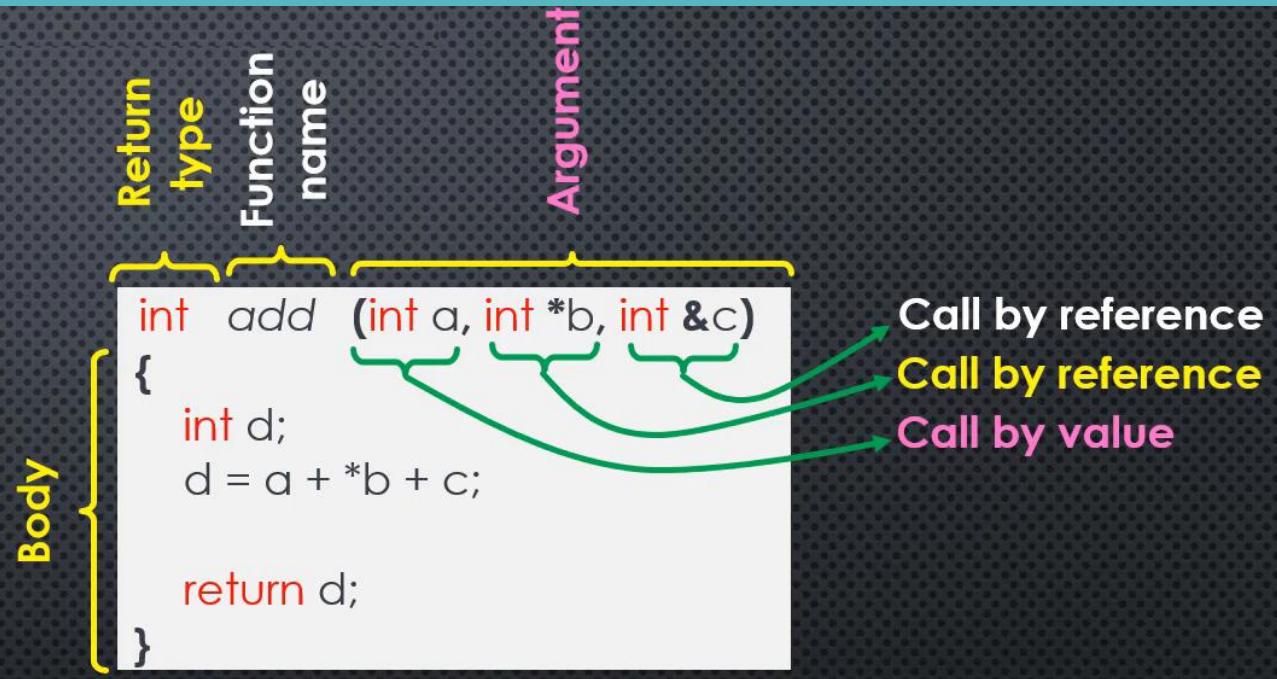


DESIGN (C/C++) IN VIVADO HLS



DESIGN (C/C++) IN VIVADO HLS





Analogy

Hardware	Software
Outputs	Function return & Call by reference arguments
Inputs	Call by value arguments
Inouts	Call by reference arguments
Design	Function body

LED CONTROLLER IN VIVADO HLS

WE NEED A C/C++ FUNCTION TO RETURN A VALUE.
THERE ARE THREE WAYS TO RETURN A VALUE

- USING THE ***RETURN*** STATEMENT IN THE FUNCTION
- USING A **POINTER** ARGUMENT
- USING A **REFERENCE** ARGUMENT (C++ FEATURE)

LED CONTROLLER FUNCTION

Return statement

```
unsigned char ledonoff() {
  return 0b11110000;
}
```

Pointer argument (Call by reference)

```
void ledonoff( unsigned char *o) {
  *o = 0b11110000;
}
```

Reference argument (Call by reference)

```
void ledonoff(unsigned char &o) {
  o = 0b11110000;
}
```

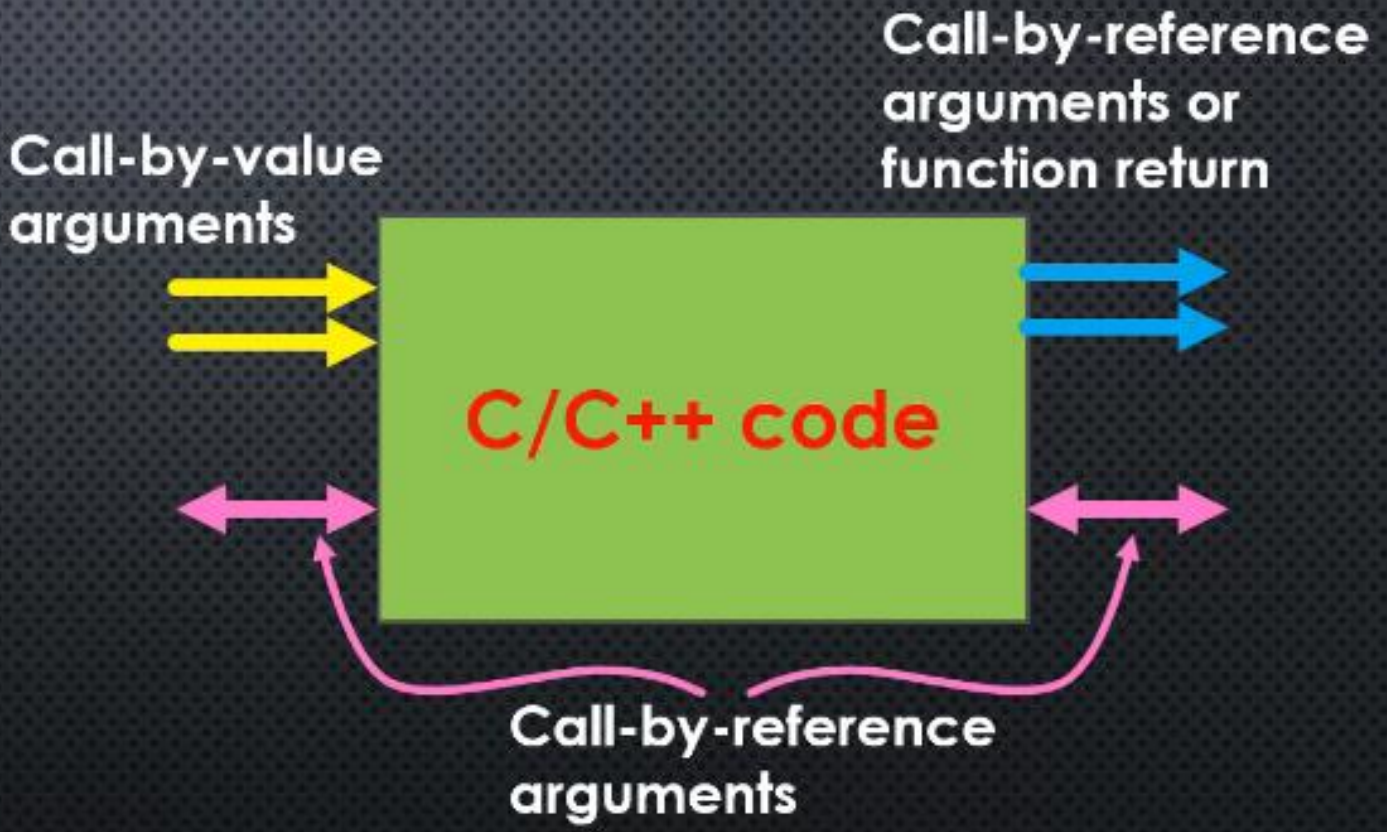
A COMMON MISTAKE

```
void ledoff(unsigned char o) {
  o = 0b11110000;
}
```

Call by value argument

SUMMARY

- A software function can describe a hardware module
- The function arguments model the hardware ports
- The function body implements the hardware behaviour



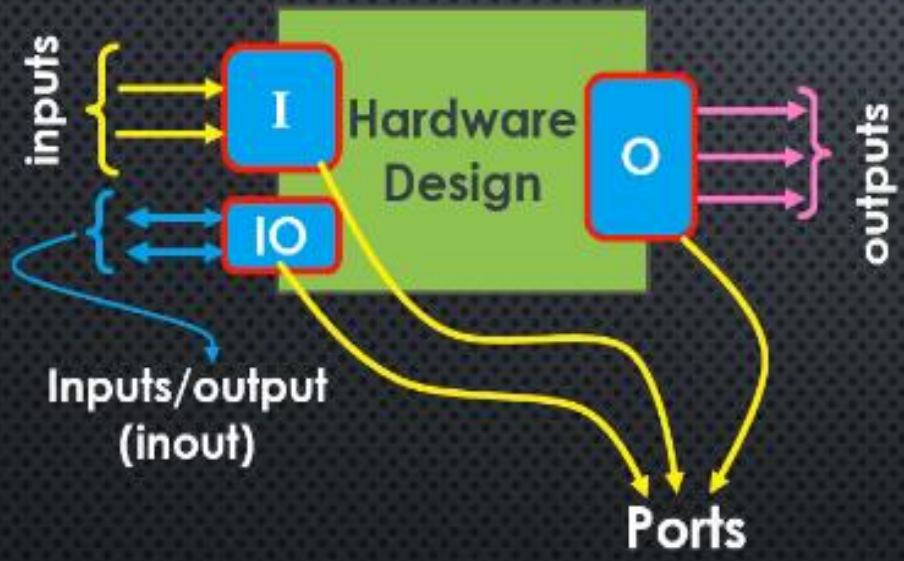
ASSIGNMENT Q

- How to describe an 8-led controller in HLS that turns on the lower 4 LEDs and keeps the rest off.

HLS PORT

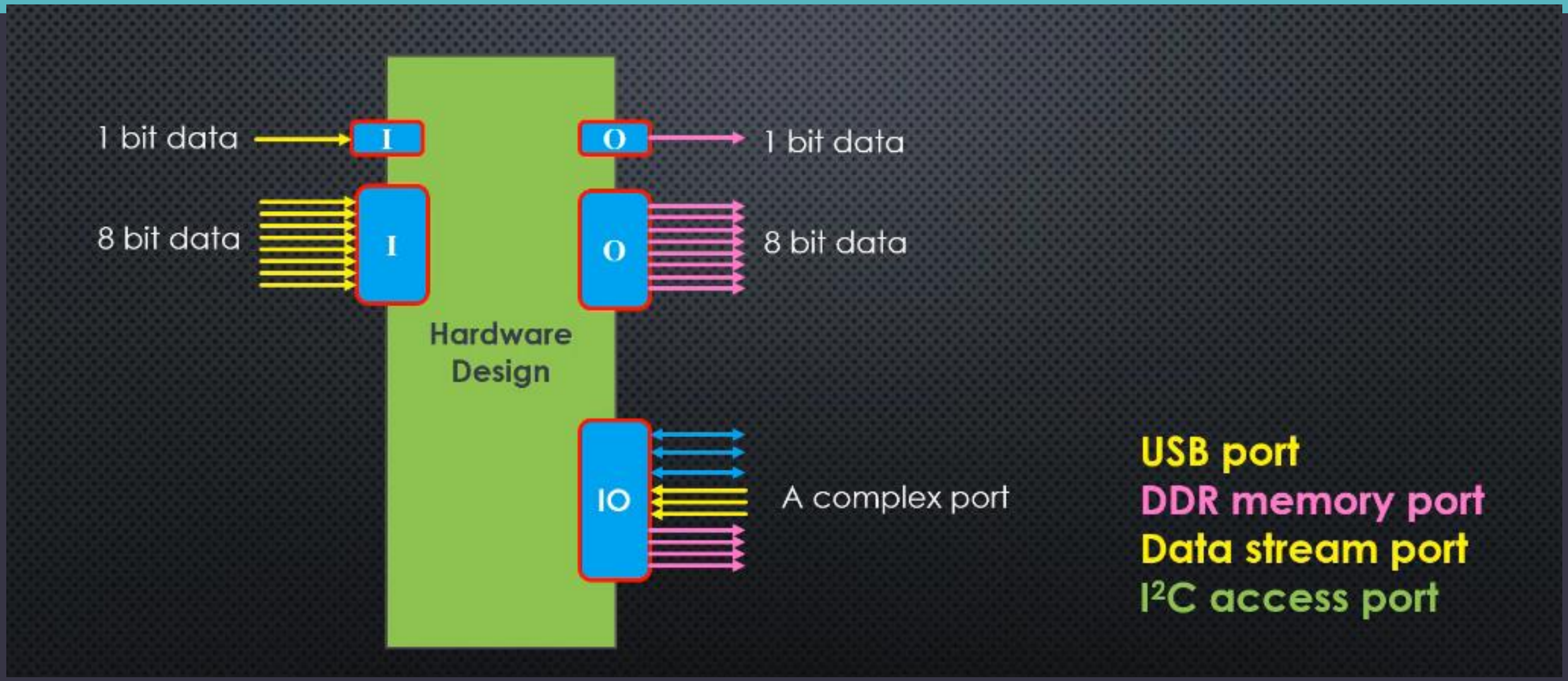
Lecture - 2

HARDWARE PORT

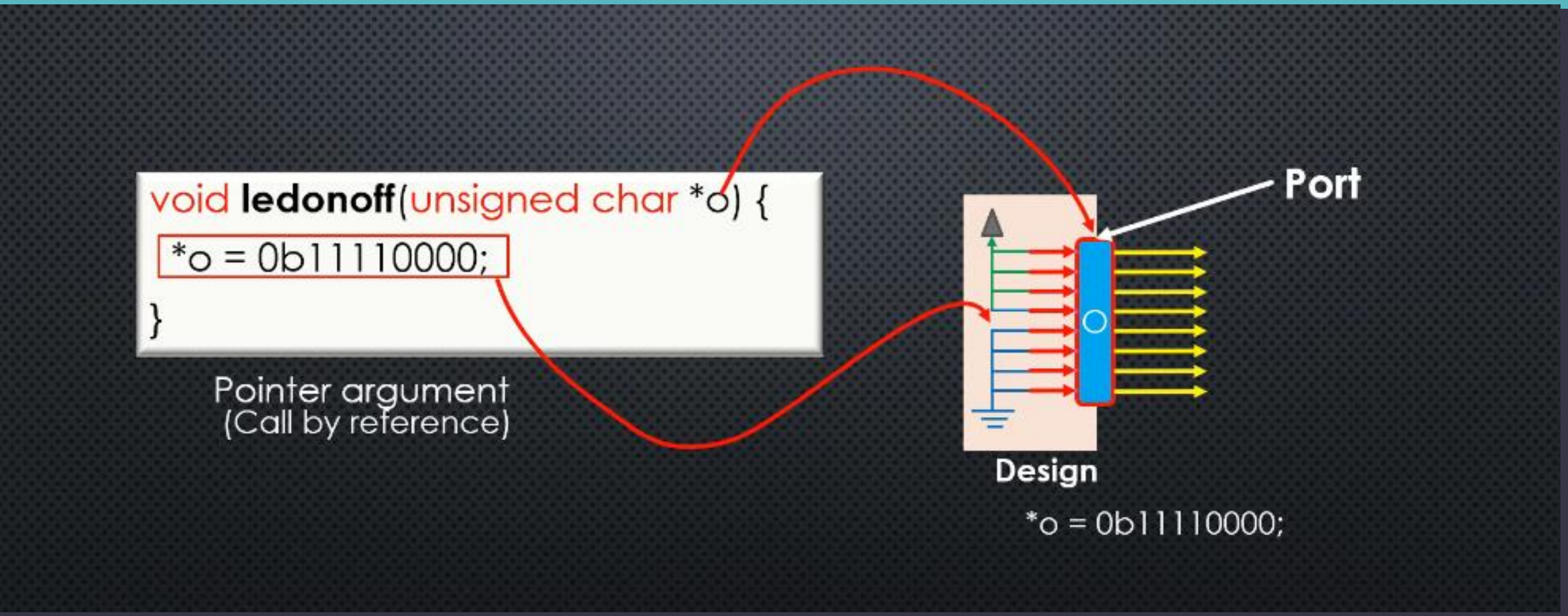


- Hardware designs use ports to communicate with other modules on a system.
- Ports can be input, output, or inout.

HARDWARE PORT EXAMPLES

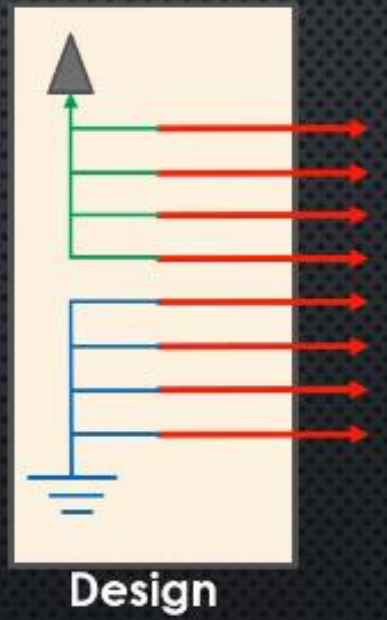


HLS DESIGN PORTS



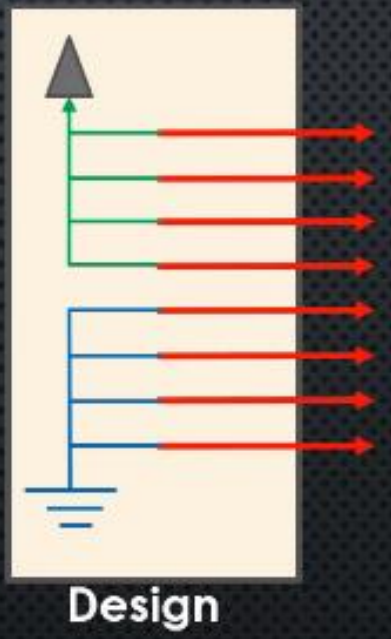
PORT & FUNCTION INTERFACES

```
void ledonoff(unsigned char *o) {



  *o = 0b11110000;
}
```



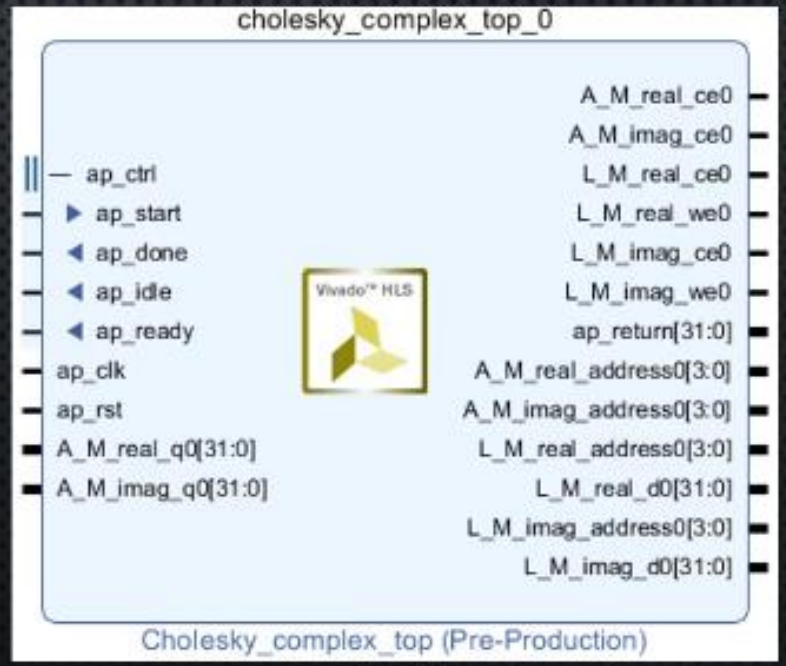
PORT & FUNCTION INTERFACES

```
void ledonoff(unsigned char *o) {

#pragma HLS  INTERFACE   AP_NONE   PORT=o

#pragma HLS  INTERFACE  AP_CTRL_NONE  PORT=return


  *o = 0b11110000;
}
```



SUMMARY

- Compiler directives can be used to assign a hardware interface with a proper protocol to a function description
- The compiler directives can be applied to
  - Function arguments
  - Or the function itself



ASSIGNMENT Q

- **How to describe an 8-led controller in HLS that turns on the lower 4 LEDs and keeps the rest off? You can use the quiz question in the previous lecture and add proper HLS pragmas.**

HLS LAB

Lecture - 2

STEP WISE STEP PROCESS

In Vivado-HLS, we

- Create a project
- Add a source file
- Perform High-Level Synthesis
- Generate IP

SUMMARY

Main tasks for developing a design in Vivado-HLS are

- Creating a project
- Writing the design functionality in C/C++
- Synthesising the design
- Generating the corresponding RTL IP

ASSIGNMENT Q

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 ledonoff() {
  return 0b11110000;
}
```

VIVADO LAB

Lecture - 2

Any Question...

Thank you