Design of Digital Circuits

Lab 2 Supplement:
Mapping Your Circuit to FPGA

Prof. Onur Mutlu
ETH Zurich

Spring 2019

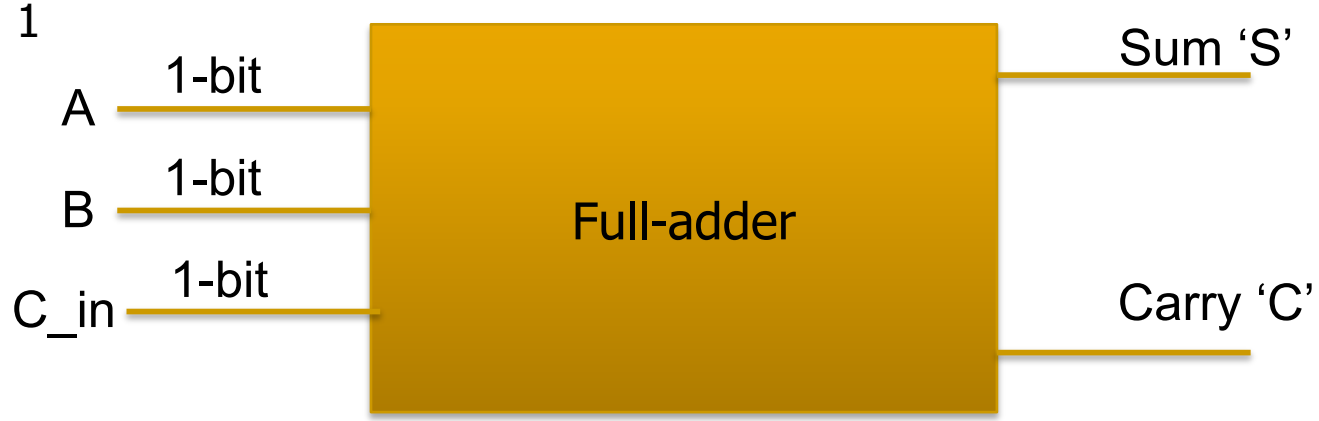
12 March 2019

What Will We Learn?

- In Lab 2, you will learn how to map your circuits to an FPGA.
- Design a 4-bit adder.
  - Design a 1-bit full-adder.
  - Use full-adders to design a 4-bit adder.
- Program your FPGA using Vivado software for HDL design.
- Work with your FPGA board and see the results of your designs on the FPGA output (in this case LEDs).

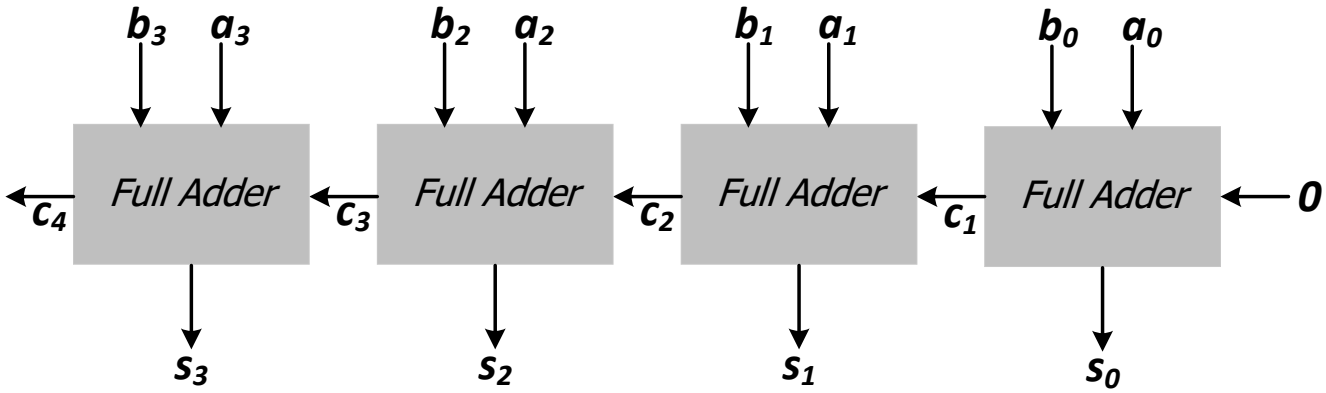
Design an Adder (I)

- Design a full-adder: Receives two 1-bit numbers A and B and a 1-bit input carry (C_in), and returns outputs S and C as sum and carry of the operation, respectively.
  - Example: A = 1, B =1, C_in=1
    - S = 1
    - C = 1



Design an Adder (II)

- Design a 4-bit adder: Receives two 1-bit numbers A and B and a 1-bit input carry (C_in), and returns outputs S and C as sum and carry of the operation, respectively.
  - Example: A = 1110, B =0001, C_in=1
    - S = 0000
    - C = 1



- Hint: Use four full-adders to design a 4-bit adder.

Design an Adder (Overview)

1. You will use truth tables to derive the Boolean equation of the adder.
2. You will design the schematic of the circuit using logic gates.
3. In the next step, you will use Vivado to write your design in Verilog.
4. In the end, you will use Vivado to program the FPGA.

Vivado

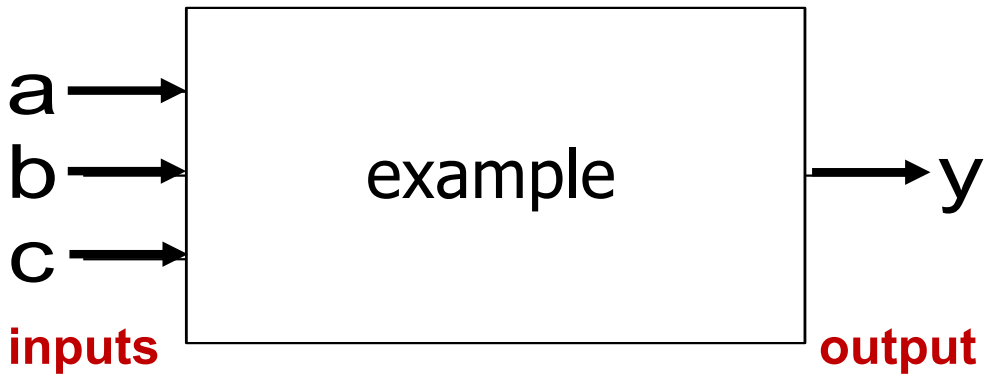
- For this course, we use the software Vivado for FPGA programming.
- The computers in the lab rooms are already installed with the necessary software.
- If you wish to use your own computer, you can refer to the following instructions:
  - https://reference.digilentinc.com/learn/programmable-logic/tutorials/basys-3-getting-started/start

Verilog

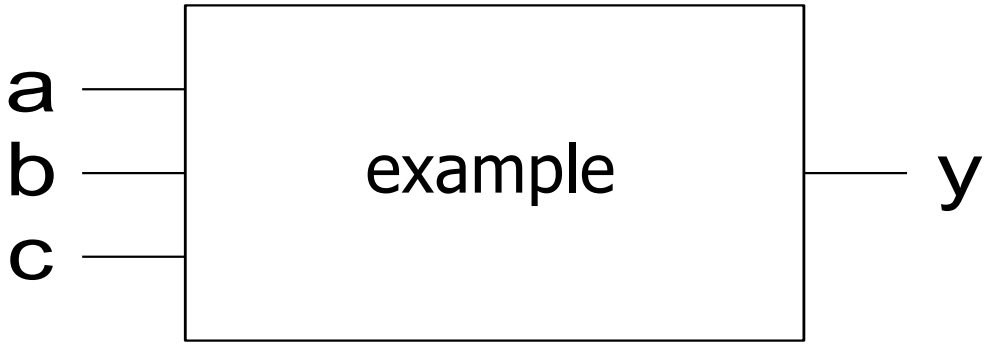
Defining a Module in Verilog

- A module is the main building block in Verilog.

- We first need to define:
  - Name of the module
  - Directions of its ports (e.g., input, output)
  - Names of its ports
- Then:
  - Describe the functionality of the module.



Implementing a Module in Verilog



name of module

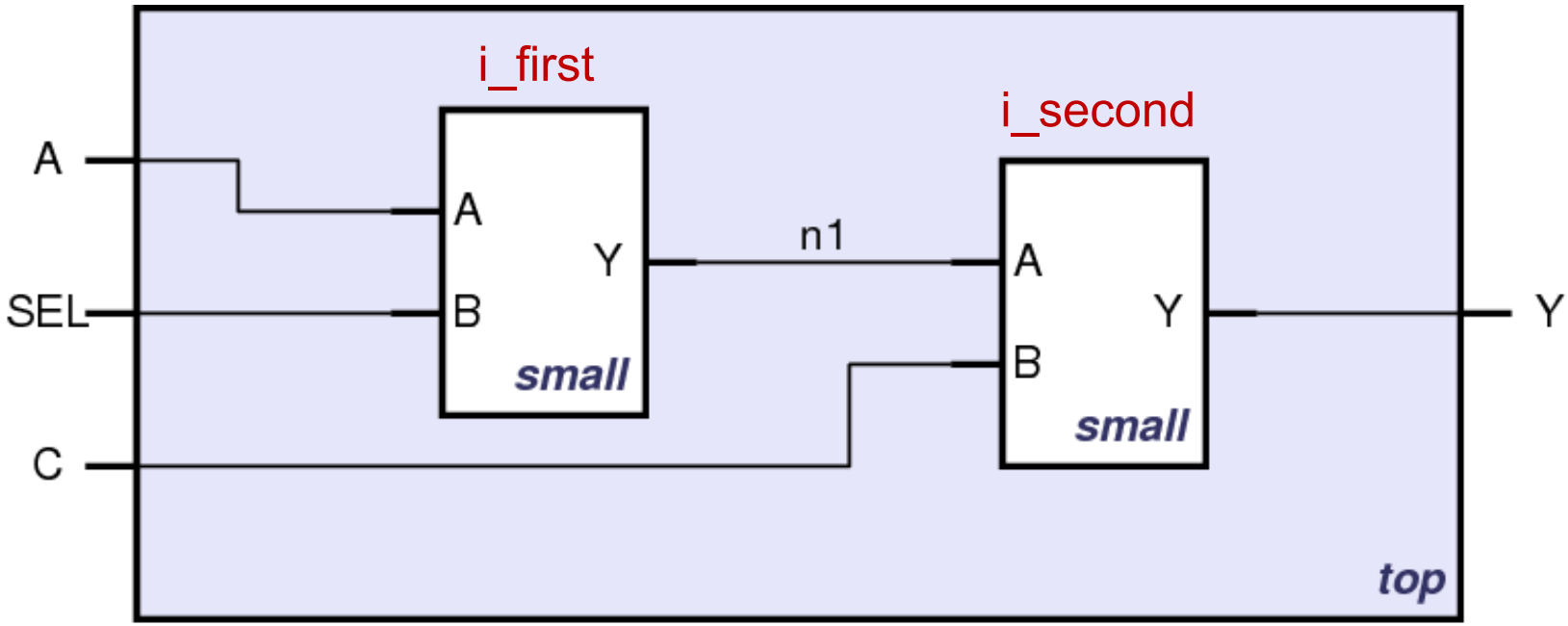
Port list (inputs and outputs)

```
module example (a, b, c, y);
        input a;
        input b;
        input c;
        output y;

// here comes the circuit description

endmodule
```

ports have a declared type

a module definition

Structural HDL: Instantiating a Module

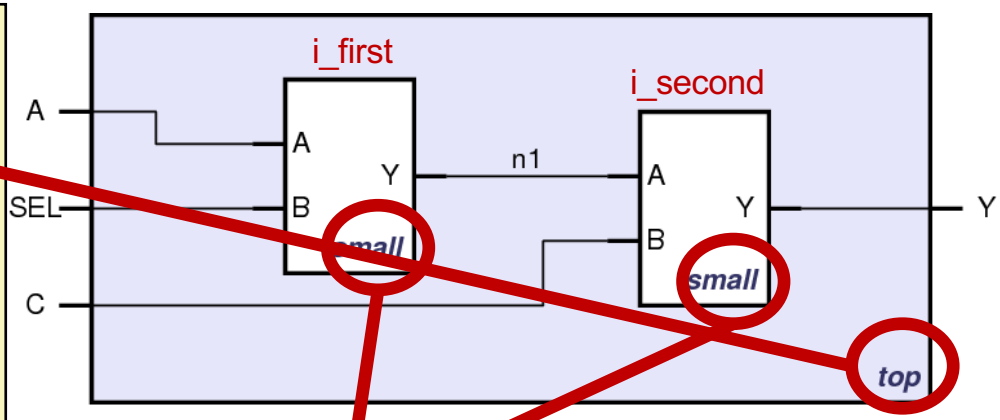


Schematic of module "top" that is built from two instances of module "small"

Structural HDL Example (1)

- **Module Definitions in Verilog**

```
module top (A, SEL, C, Y);
  input A, SEL, C;
  output Y;
  wire n1;









endmodule
```

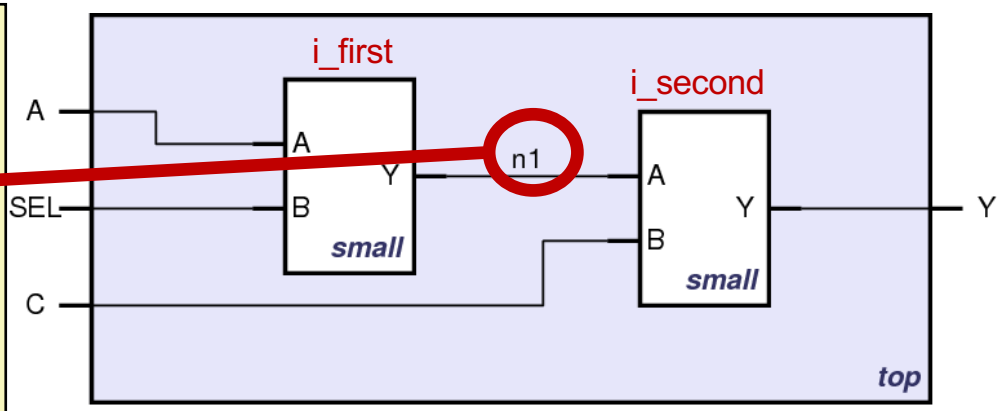



```
module small (A, B, Y);
  input A;
  input B;
  output Y;

// description of small

endmodule
```

Structural HDL Example (2)

- **Defining wires (module interconnections)**

```
module top (A, SEL, C, Y);
  input A, SEL, C;
  output Y;
  wire n1;




endmodule
```

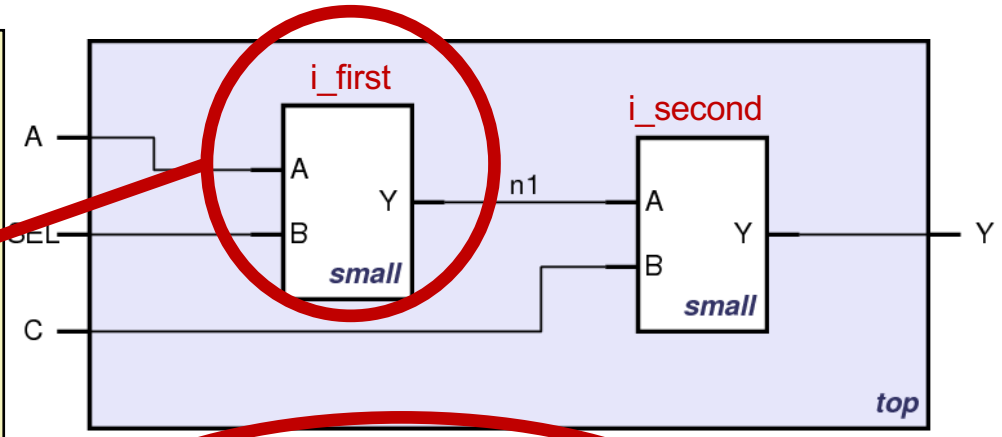



```
module small (A, B, Y);
  input A;
  input B;
  output Y;

// description of small

endmodule
```

Structural HDL Example (3)

- **The first instantiation of the "small" module**

```
module top (A, SEL, C, Y);
  input A, SEL, C;
  output Y;
  wire n1;

// instantiate small once
small i_first ( .A(A),
                .B(SEL),
                .Y(n1)   );




endmodule
```

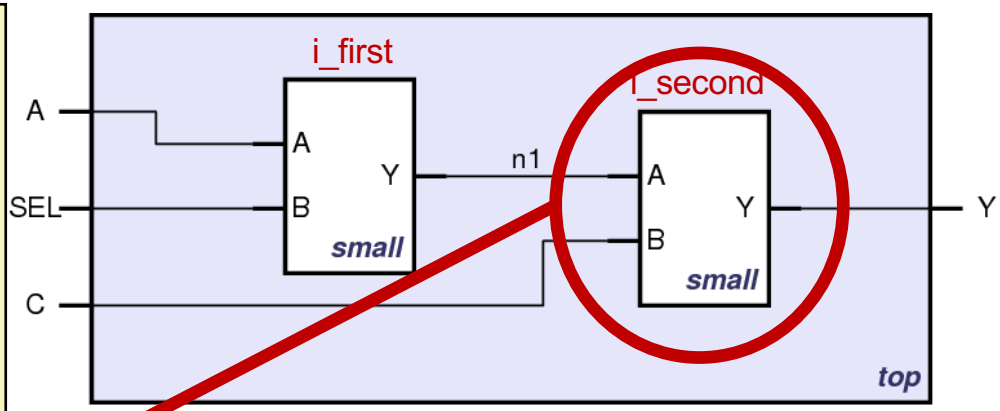



```
module small (A, B, Y);
  input A;
  input B;
  output Y;

// description of small

endmodule
```

Structural HDL Example (4)

- **The second instantiation of the "small" module**

```
module top (A, SEL, C, Y);
  input A, SEL, C;
  output Y;
  wire n1;

// instantiate small once
small i_first ( .A(A),
                .B(SEL),
                .Y(n1)   );

// instantiate small second time
small i_second ( .A(n1),
                 .B(C),
                 .Y(Y) );

endmodule
```

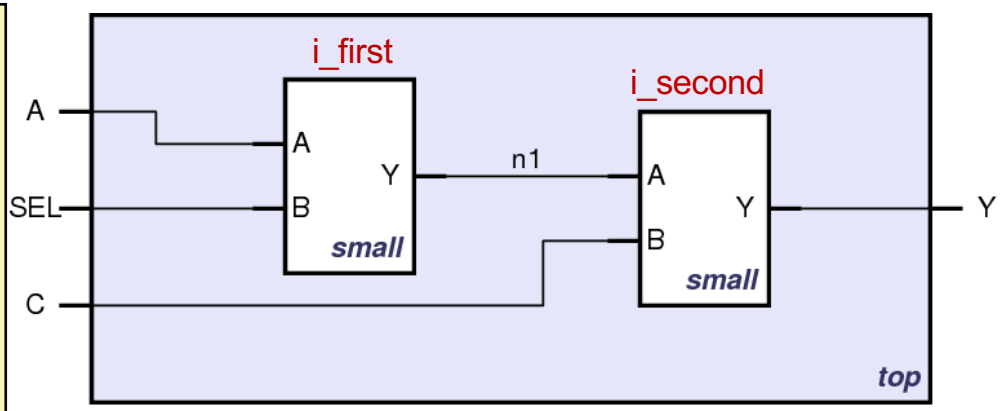



```
module small (A, B, Y);
  input A;
  input B;
  output Y;

// description of small

endmodule
```

Structural HDL Example (5)

- **Short form of module instantiation**

```
module top (A, SEL, C, Y);
  input A, SEL, C;
  output Y;
  wire n1;

// alternative
small i_first ( A, SEL, n1 );

/* Shorter instantiation,
   pin order very important */

// any pin order, safer choice
small i_second ( .B(C),
                 .Y(Y),
                 .A(n1) );

endmodule
```



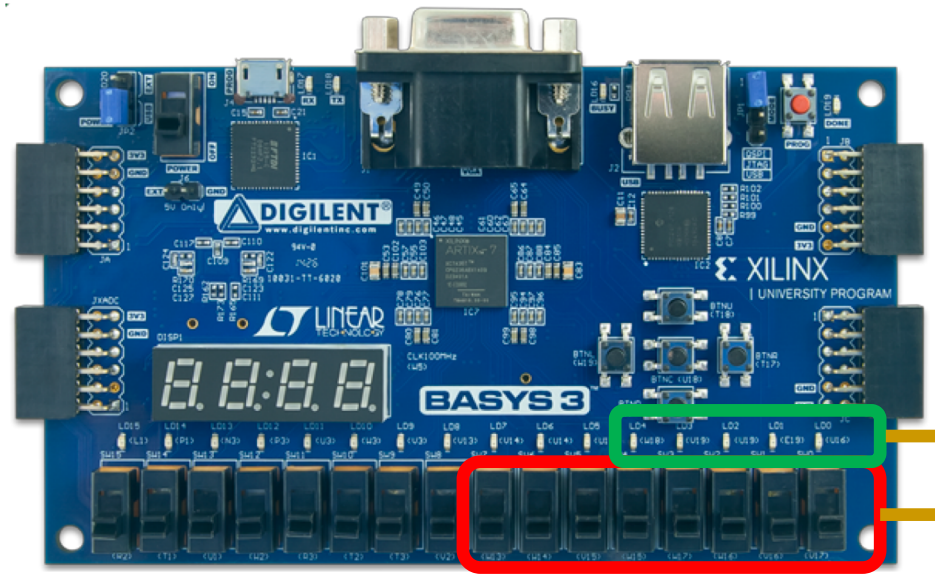

```
module small (A, B, Y);
  input A;
  input B;
  output Y;

// description of small

endmodule
```

Basys 3 FPGA Board

In this course, we will be using the Basys 3 boards from Digilent, as shown below.

You can learn more about the Basys 3 Starter Board from:

http://store.digilentinc.com/basys-3-artix-7-fpga-trainer-board-recommended-for-introductory-users/



For this Lab:

Output LEDs

Input Switches

Last Words

- In this lab, you will map your circuit to an FPGA.
- First you will design a 4-bit adder.
- Then, you will learn how to use Xilinx Vivado for writing Verilog and how to connect to the Basys 3 board.
- Finally, you will program the FPGA and get the circuit running on the FPGA board.
- You will find more exercises in the lab report.

Design of Digital Circuits

Lab 2 Supplement:
Mapping Your Circuit to FPGA

Prof. Onur Mutlu
ETH Zurich

Spring 2019

12 March 2019