

Text Mode v. Graphic Editor

**STEP BY STEP PROJECT CREATION
THROUGH BOTH VHDL AND SCHEMATICS**



CANDACE ROSS

EECE494 COMPUTER BUS AND SOC INTERFACING

ELECTRICAL AND COMPUTER ENGINEERING

HOWARD UNIVERSITY

INSTRUCTOR: DR. CHARLES KIM

Pull out your laptops and follow along!



Outline



- I. What is text mode versus graphic editor?
- II. Text Mode Step by Step
- III. Schematic Step by Step
- IV. Implementation of VHDL and Schematic on Board
 - I. Pin planner
- V. Debugging Tips

What is the Difference?



- **Text mode:**
 - Written code to implement design
 - Better choice for more complex projects
- **Graphic mode:**
 - Designing circuits to implement design
 - Better choice for simpler projects
 - Also good when its easy to visualize are a logic circuit

Background on the Example We'll Follow



- Create a project called PushLeds
 - 5 inputs: 4 switches, 1 push button
 - 4 outputs: LEDs
- Each switch turns on to light a corresponding LED. Whenever the user holds the push button, the LEDs that should be on will light.

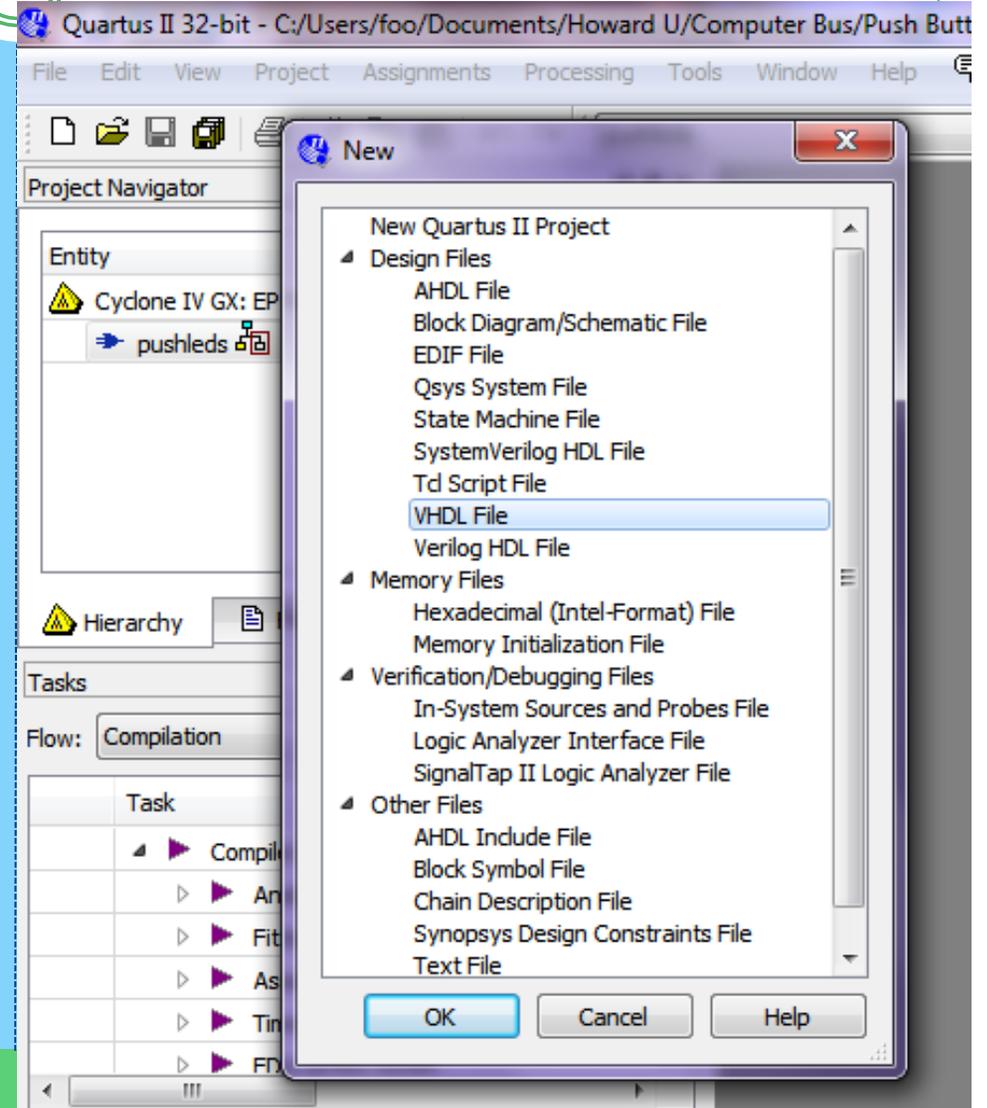
VHDL File



- There are three parts to a VHDL file
 - 1) Library
 - 2) Entity
 - 3) Architecture
- Let's first go through a blank file and then the example mentioned earlier

Text Mode (VHDL) Steps

- Create a new VHDL file
- Go to:
 - New > File > VHDL File



Blank VHDL File Example



```
1 library IEEE;  
2 use ieee.std_logic_1164.all;  
3  
4  
5  
6 entity pushleds is  
7     port ();  
8 end push leds;  
9  
10  
11  
12 architecture arch of pushleds is  
13 begin  
14  
15     end arch;
```

Library: this declaration is always the same

Blank VHDL File Example



```
1 library IEEE;
2 use ieee.std_logic_1164.all;
3
4
5
6 entity pushleds is
7     port ();
8 end push led;
9
10
11
12 architecture arch of pushleds is
13 begin
14
15     end arch;|
```

Entity: this is where the inputs and outputs used on the board are declared

Blank VHDL File Example



```
1 library IEEE;  
2 use ieee.std_logic_1164.all;  
3  
4  
5  
6 entity pushleds is  
7     port ();  
8 end push led; -- Note: 'led' is misspelled as 'leds' in the original image  
9  
10  
11  
12 architecture arch of pushleds is  
13 begin  
14  
15 end arch;
```

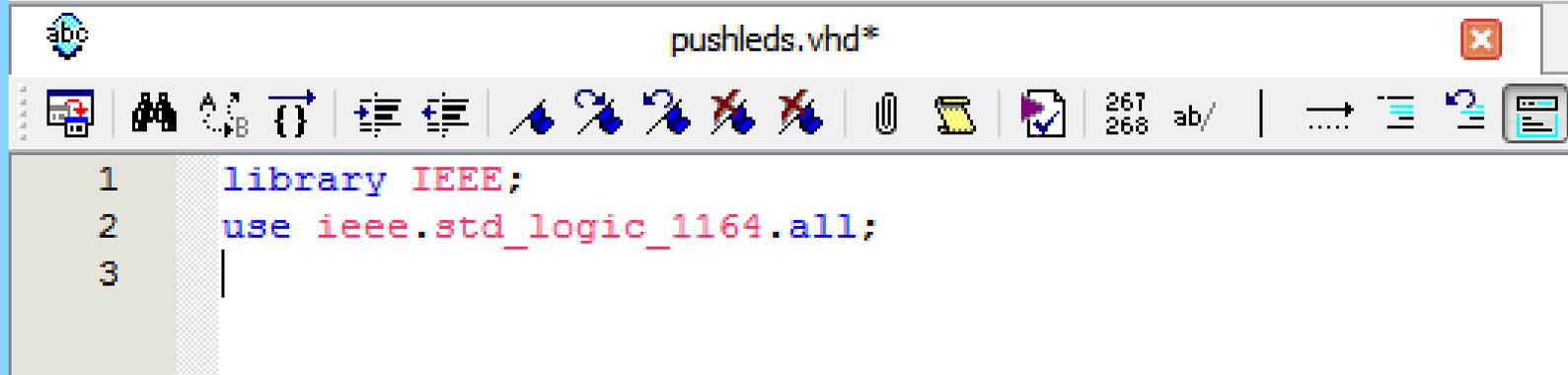
The screenshot shows a text editor window with a toolbar and a file name 'pushleds.vhd'. The code is color-coded: keywords like 'library', 'use', 'entity', 'end', 'architecture', and 'begin' are in blue, while identifiers like 'IEEE', 'std_logic_1164', 'pushleds', and 'arch' are in green. The architecture section (lines 12-15) is highlighted with a light green background. An arrow points from the text 'Architecture: this is where the actual coding begins!' to the start of the architecture block.

Architecture: this is where the actual coding begins!

Our VHDL File: Step by Step



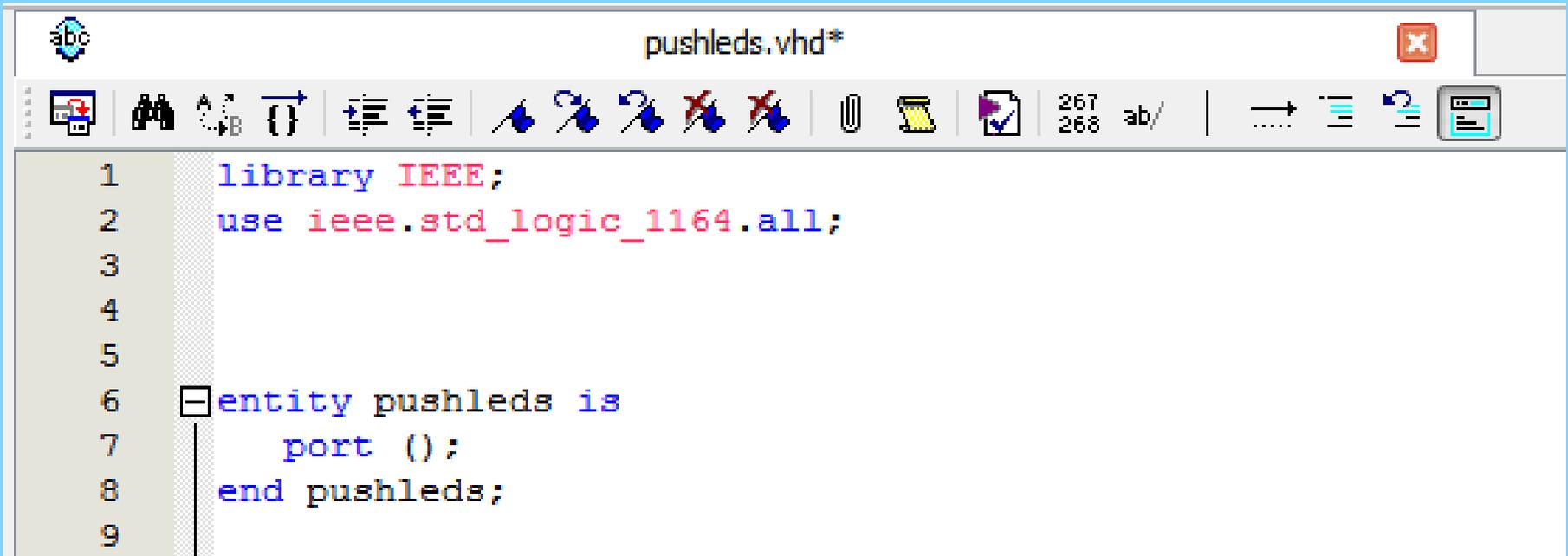
- Let's begin with the library declaration

A screenshot of a text editor window titled 'pushleds.vhd*'. The window has a toolbar with various icons for editing and file management. The code is displayed on three lines:

```
1 library IEEE;  
2 use ieee.std_logic_1164.all;  
3 |
```

Our VHDL File: Step by Step

- Now move on to the entity
 - The entity name must match the VHDL file name



The screenshot shows a text editor window titled 'pushleds.vhd*'. The editor contains the following VHDL code:

```
1  library IEEE;
2  use ieee.std_logic_1164.all;
3
4
5
6  entity pushleds is
7      port ();
8  end pushleds;
9
```

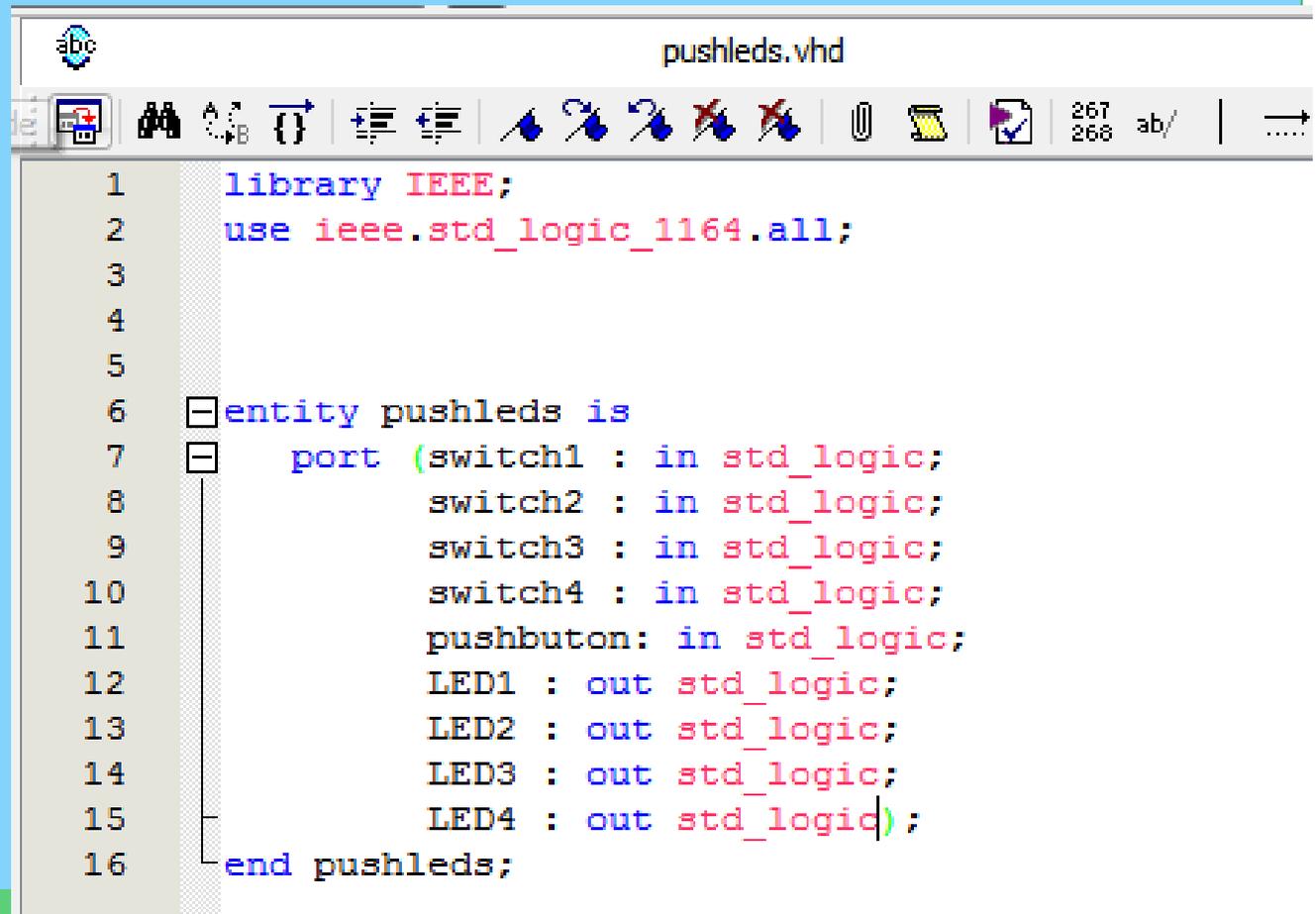
Our VHDL File: Step by Step



- In the entity, we need to declare our inputs and outputs
- They are:
 - 4 switches
 - 1 push button
 - 4 LEDs

Our VHDL File: Step by Step

- In the entity, we need to declare our inputs and outputs
- They are:
 - 4 switches
 - 1 push button
 - 4 LEDs



```
1 library IEEE;
2 use ieee.std_logic_1164.all;
3
4
5
6 entity pushleds is
7     port (switch1 : in std_logic;
8           switch2 : in std_logic;
9           switch3 : in std_logic;
10          switch4 : in std_logic;
11          pushbuton: in std_logic;
12          LED1  : out std_logic;
13          LED2  : out std_logic;
14          LED3  : out std_logic;
15          LED4  : out std_logic);
16 end pushleds;
```

Our VHDL File: Step by Step



- Architecture: We will actually apply what we want our board to do using code
- First, observe the format of the empty architecture

```
16   end pushleds;  
17  
18  
19   architecture arch of pushleds is  
20   begin  
21   L  
22   end architecture;
```

Our VHDL File: Step by Step



- Our goal is for each LED to turn on when its switch is on; let's begin there...

```
18  |
19  |  architecture arch of pushleds is
20  |  begin
21  |     LED1 <= switch1;
22  |     LED2 <= switch2;
23  |     LED3 <= switch3;
24  |     LED4 <= switch4;
25  | end architecture;
```

Our VHDL: Step by Step



- We also only want the LEDs to light when the push button is pressed

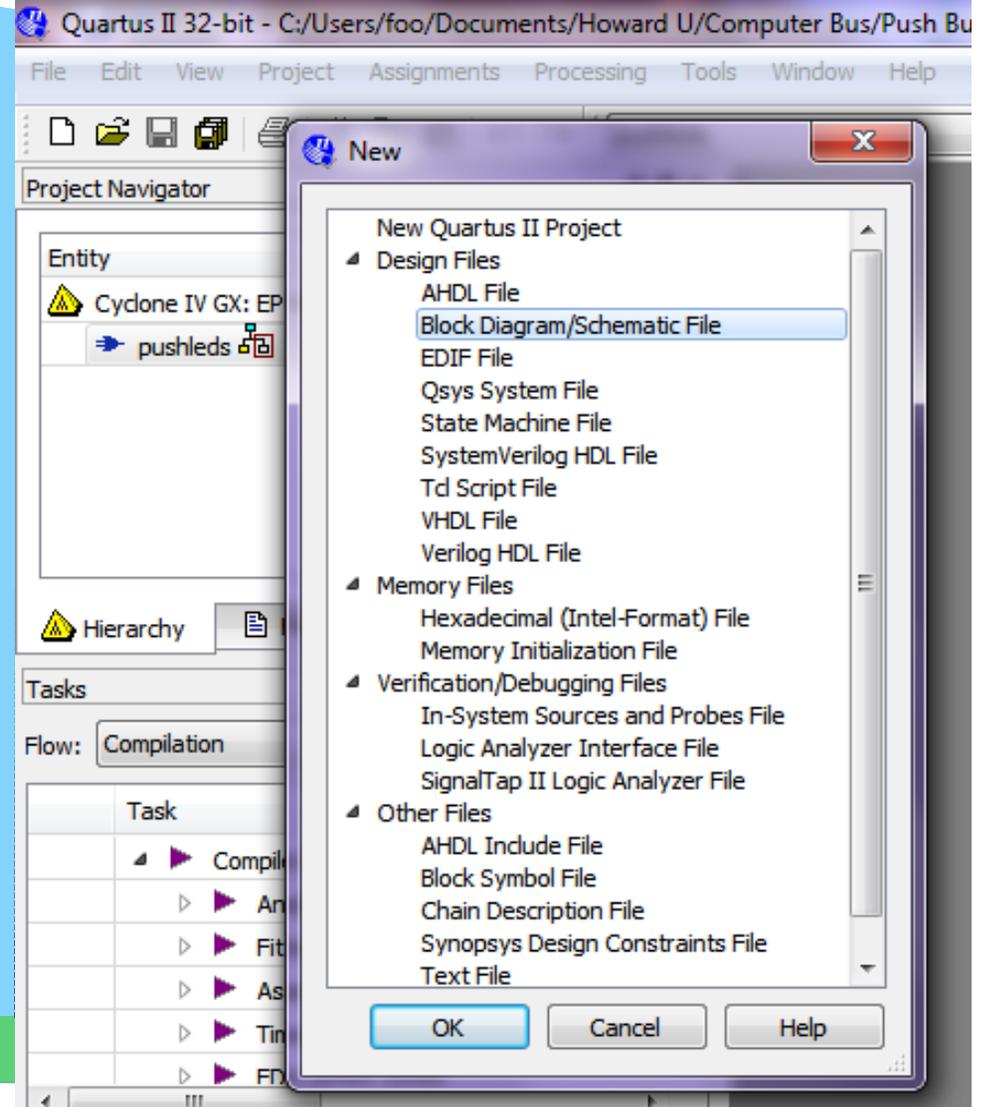
```
18 |
19 | architecture arch of pushleds is
20 | begin
21 |     LED1 <= switch1 AND pushbutton;
22 |     LED2 <= switch2 AND pushbutton;
23 |     LED3 <= switch3 AND pushbutton;
24 |     LED4 <= switch4 AND pushbutton;
25 | end architecture;
```

**ANY QUESTIONS ON
VHDL CODE?**



Graphic Mode: Step by Step

- Instead of creating a VHD file, create a schematic
- Go to:
File > New > Block Diagram/ Schematic File



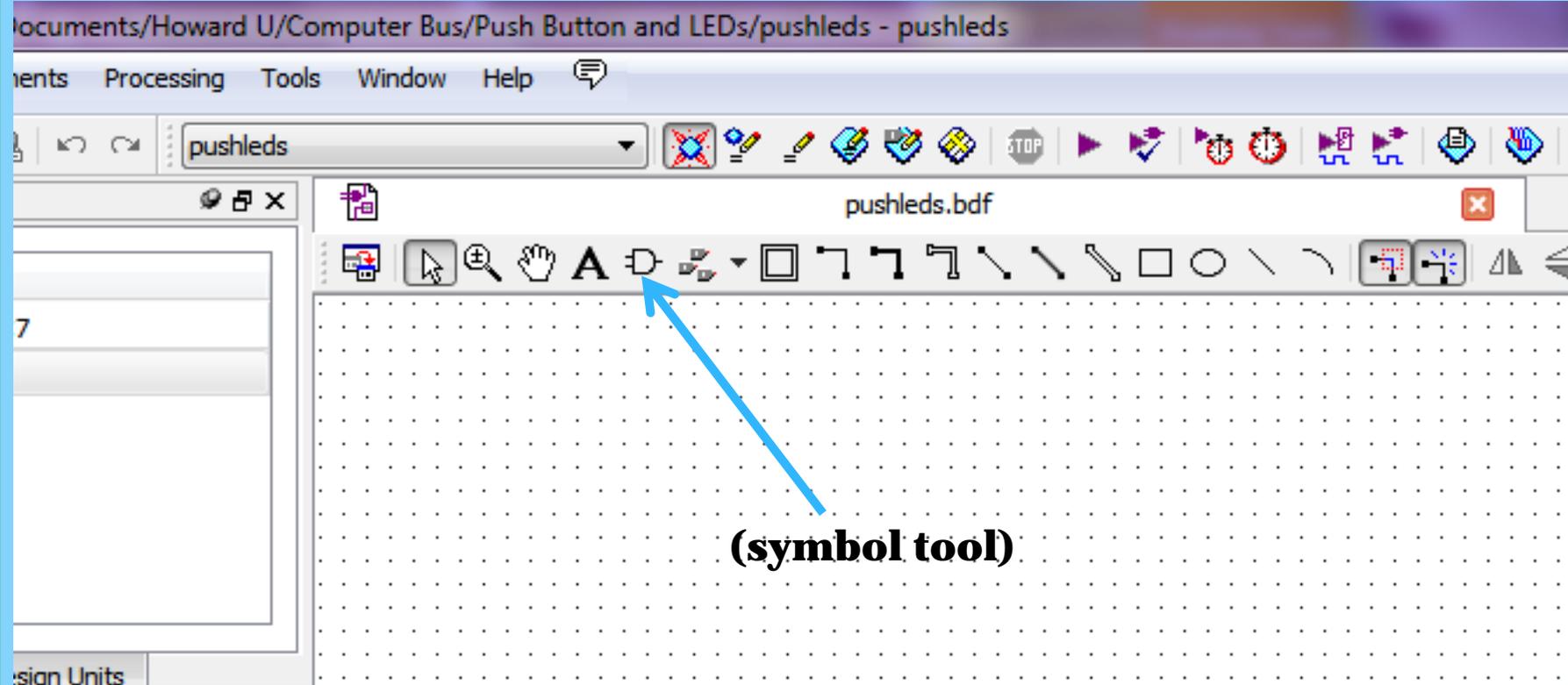
Graphic Mode: Step by Step



- Similarly to the VHDL file, we will create the project using a circuit
- There are still 5 inputs and 4 outputs
- Use digital logic to create the circuit

Graphic Mode: Step by Step

- To include any inputs, outputs or logic gates, go here:



Graphic Mode: Step by Step

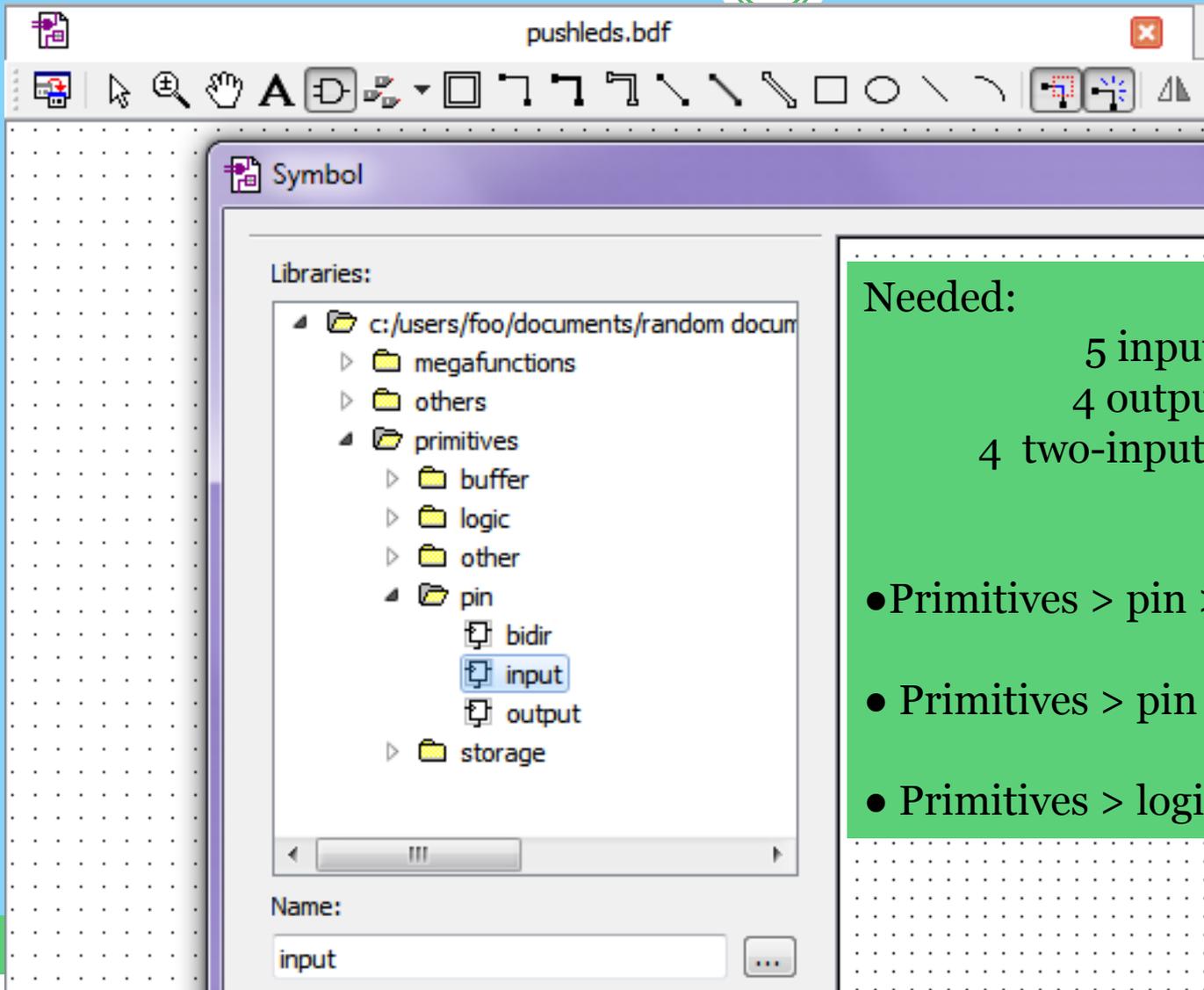


The screenshot shows a software interface with two windows. The top window is titled 'pushleds.vhd' and 'pushleds.bdf'. Below it is a toolbar with various icons for drawing and editing. The bottom window is titled 'Symbol' and shows a file explorer view of a library. The library path is 'c:/users/foo/documents/random docum'. The library contains three folders: 'megafunctions', 'others', and 'primitives'. Below the library view is a 'Name:' label.

Needed: 5 inputs, 4 outputs, 4 two-input AND gates

- Primitives > pin > input
- Primitives > pin > output
- Primitives > logic > and2

Graphic Mode: Step by Step



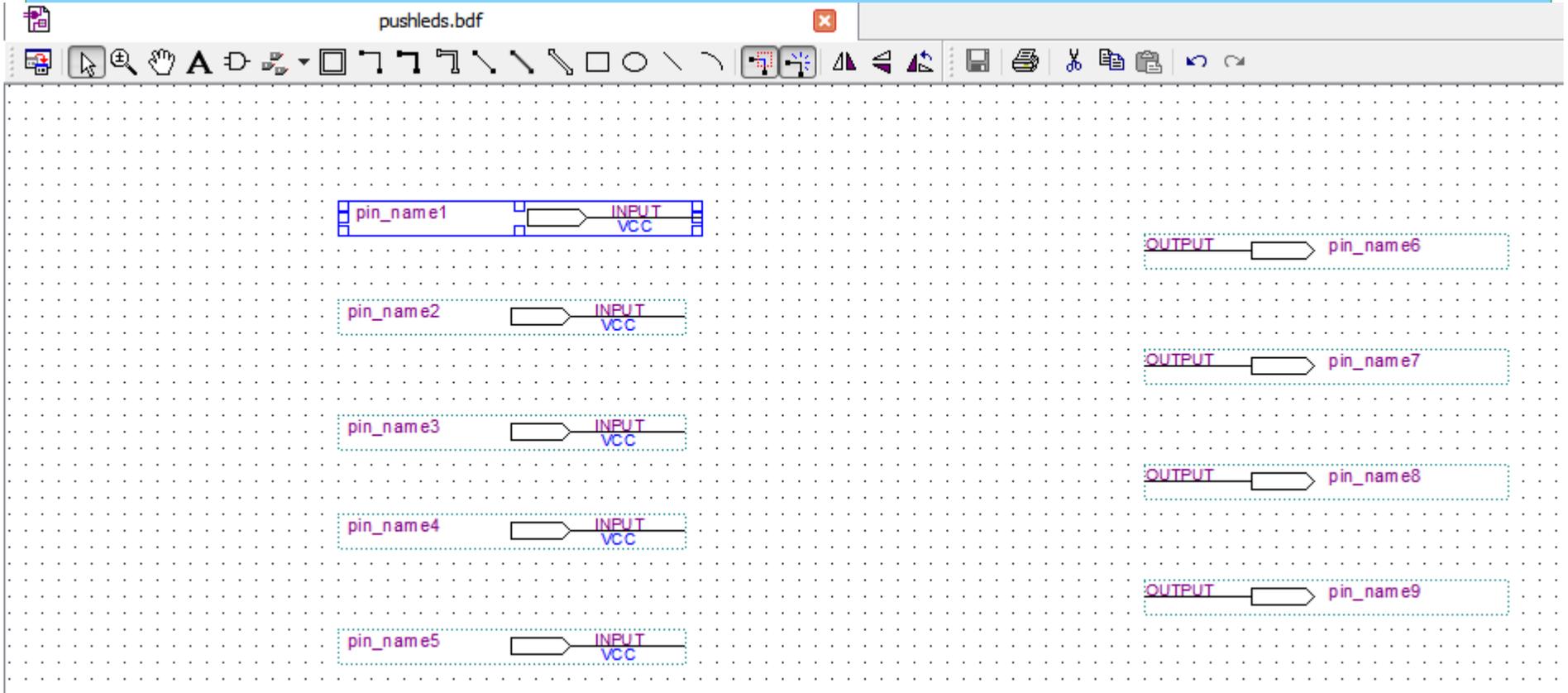
Needed:

5 input pins
4 output pins
4 two-input AND gates

- Primitives > pin > input
- Primitives > pin > output
- Primitives > logic > and2

Graphic Mode: Step by Step

- Let's begin by inserting our 5 inputs and 4 outputs



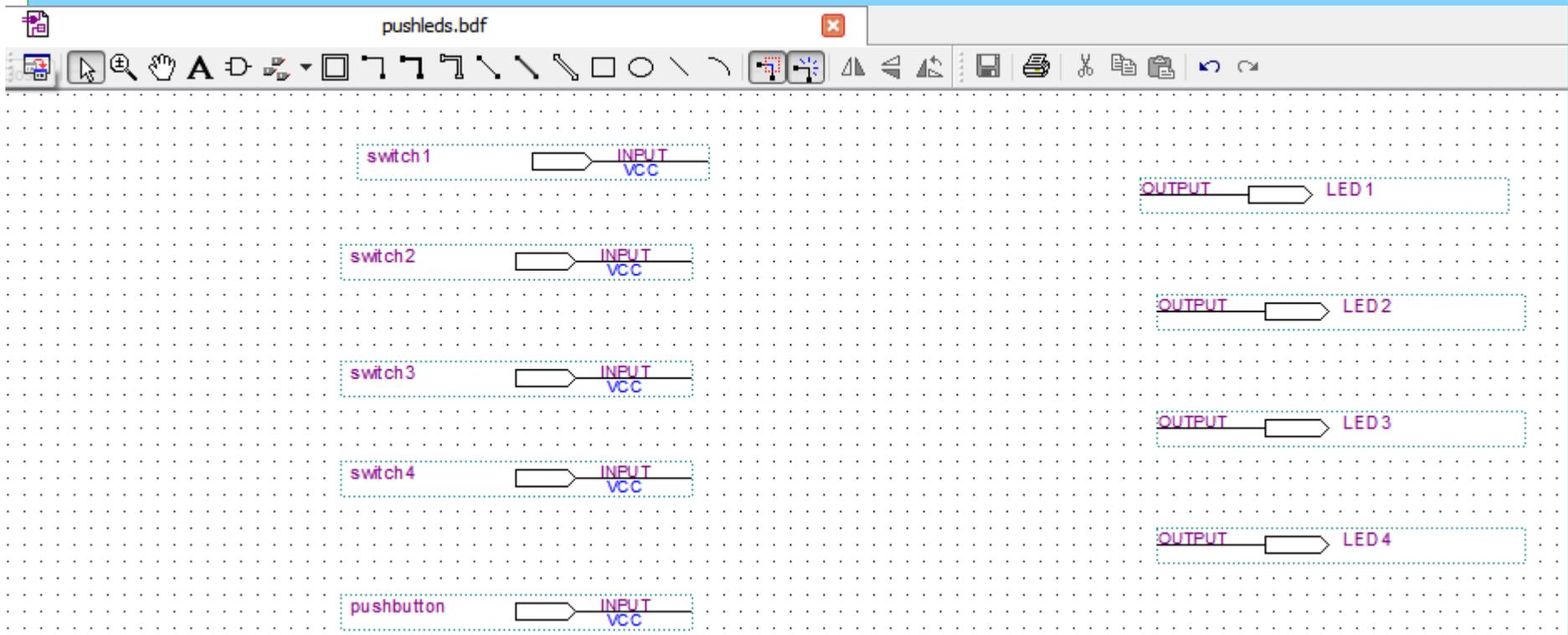
Graphic Mode: Step by Step



- ... and let's name our pins
 - Just double click on the input or output
 - Switch1, switch2, led1, led2...

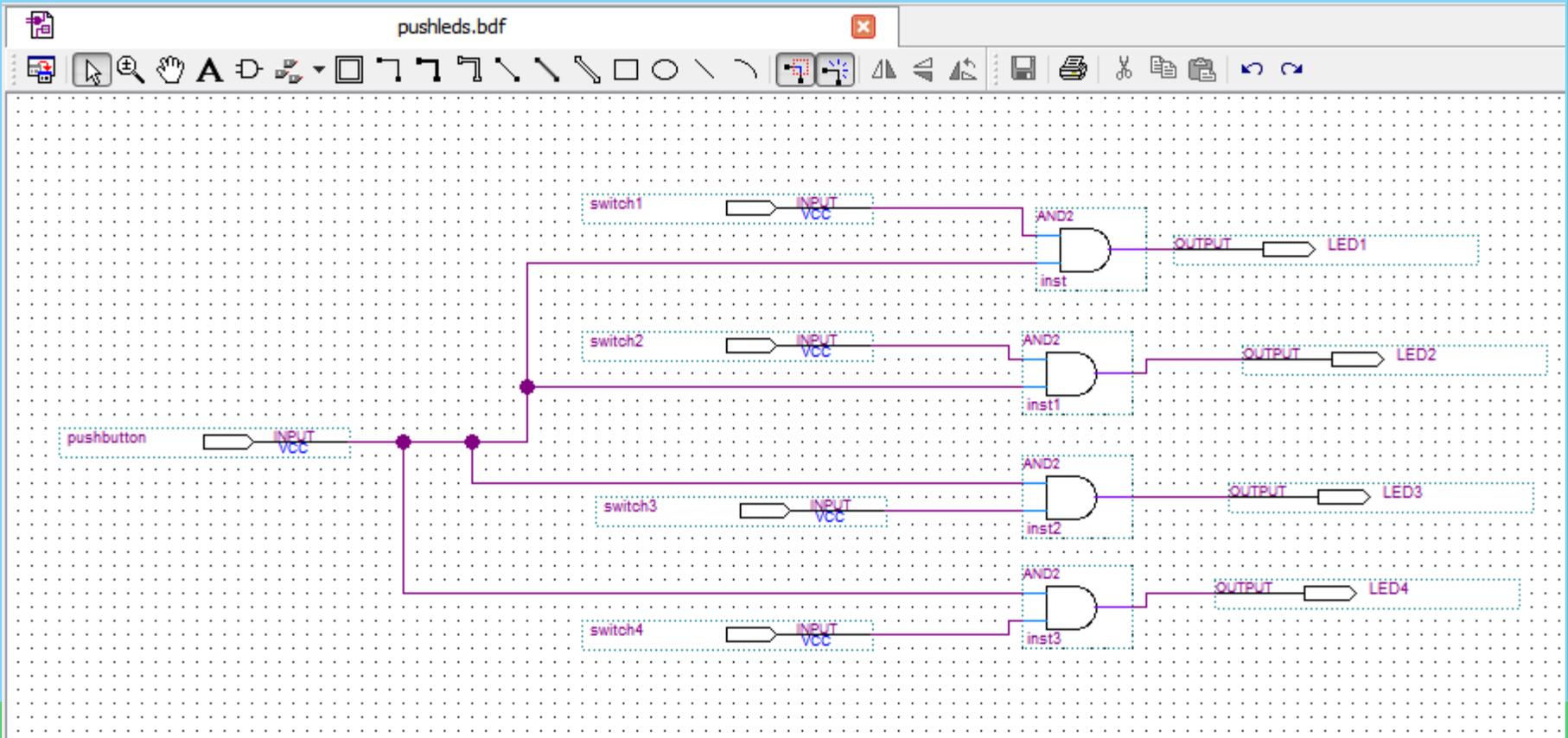
Graphic Mode: Step by Step

- ... and let's name our pins
 - Just double click on the input or output



Graphic Mode: Step by Step

- We can finish by inserting our AND gates
 - Ex: LED1 <= switch1 AND pushbutton



**ANY QUESTIONS ON
THE SCHEMATICS?**



Implementation: the Finishing Touch



- For the sake of time, we will not actually implement in class
- The following slide give details on the on-board implementation
- There are also additional example problems that can be practiced, using both the schematics and VHDL

Helpful Debugging Tips

