Introduction to FPGAs
Lecture 1: FPGAs, Logic Gates, Programming an FPGA

Adam Duncan Ph.D.

Class Introduction

About Me: Electrical/Computer Engineer at NSWC Crane

Class Goals: 1) Learn about FPGAs. 2) Program FPGAs to do fun things.

Class SW Materials: https://github.com/adamdunc/whs

Field-Programmable Gate Arrays (FPGAs) are "**Programmable Hardware**" devices

CPU

- ~1 inch^2 with ~10 billion transistors
- Fast (~3GHz)
- Hardware does one thing really well (execute code)

FPGA

- ~1 inch^2 with ~10 billion transistors
- Not as Fast (~500MHz)
- Hardware can be programmed to do anything

FPGA hardware behavior configured by "**Bitstream**" file

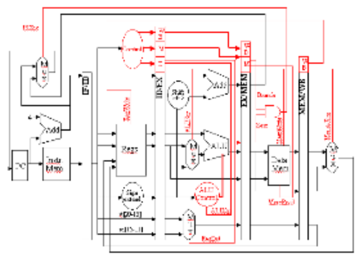
Custom Sensor Bitstream

Neutral Network Bitstream

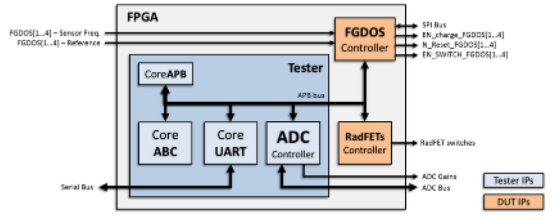
CPU Bitstream

CPU

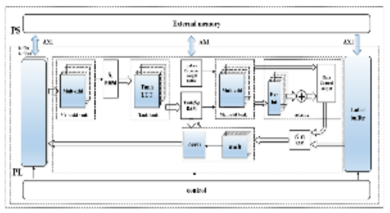
FPGA



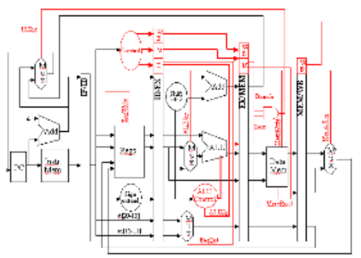
CPU is always CPU Datapath



FPGA is now customized sensor driver/sampler



FPGA is now Neural Network Datapath



FPGA is now CPU Datapath

FPGA benefits

- **HW ultimately drives performance** (SW optimizations limited by HW)
- **Parallelization** (i.e. 35 parallel ALUs executing at same time)
- **Unique needs** (read non-standard sensors at deterministic time intervals)
- **Consolidates PCB circuitry** (CPU + sensor logic + display driving)
- **Prototype Custom Integrated Circuits** (wafers >$10M)
- **Update hardware in field** (security updates, new features, bug fixes)
- **Replace obsolete hardware** (configure FPGA to “emulate” obsolete component”

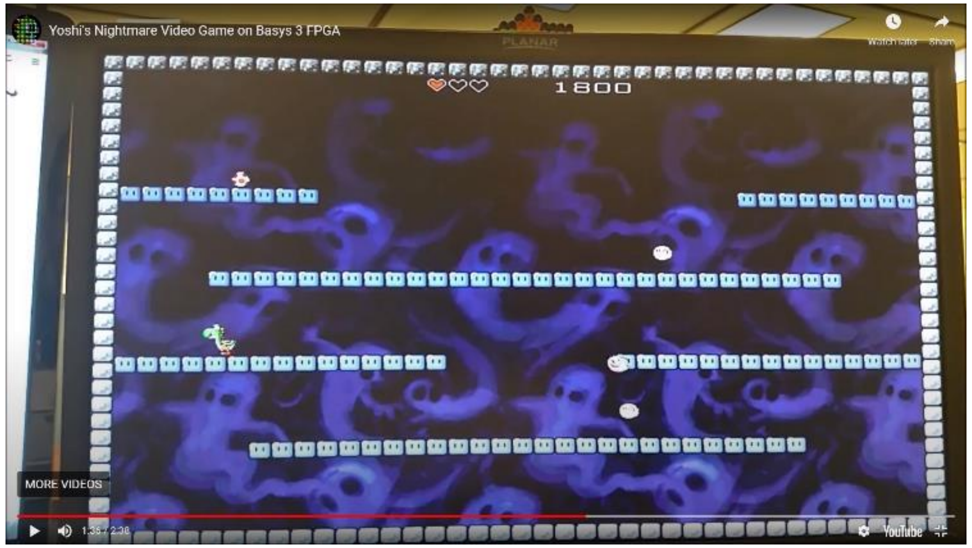
FPGAs in the Wild

Ease of use:
- New HW designs in ~1 day
- Design portability

Cost:
- $1 for low-cost FPGAs
- $100K for rugged/large FPGAs

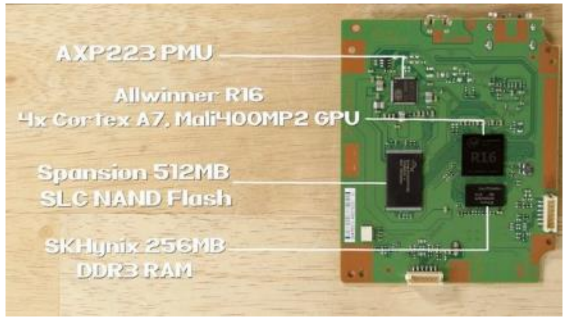
Products with FPGAs:
- Consumer electronics
- DoD Programs
- Automotive
- Networking equipment
- Test/Measurement equipment

FPGA Demo: Yoshi's Nightmare on our class FPGA

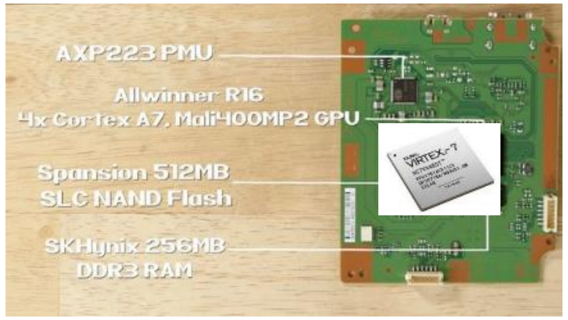


We can configure the FPGA to emulate video game system driving a display

NES Classic Circuit Board

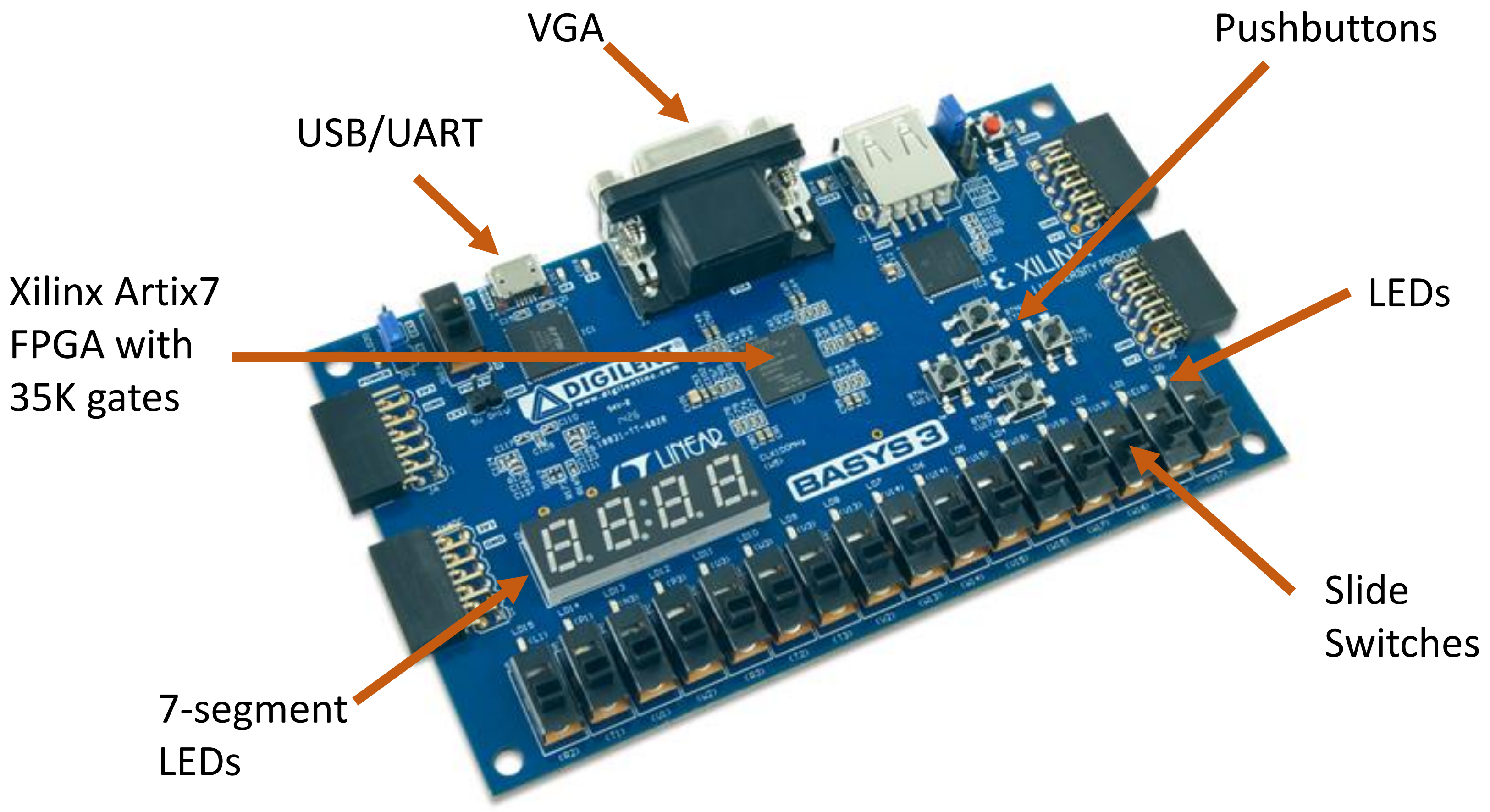


FPGA replacing CPU/GPU, RAM, Flash



https://github.com/jconenna/Yoshis-Nightmare

Meet Our FPGA: Xilinx Artix7 on a BASYS3 PCB



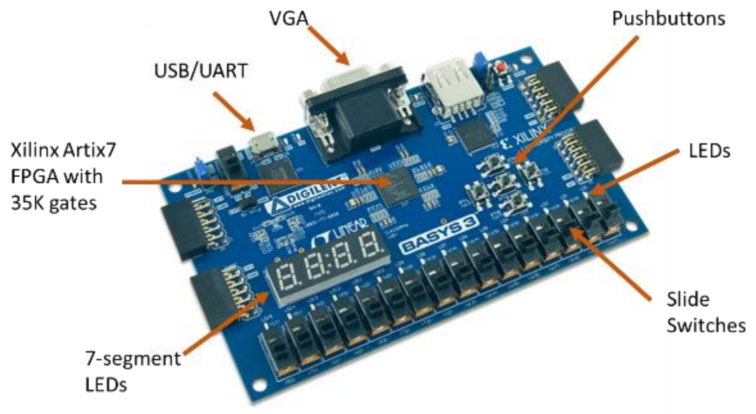
https://reference.digilentinc.com/basys3/refmanual

This week we design/simulate/program an FPGA!

- **Monday:**
  - (morning) Project 0: Program your FPGA with a pre-made file to play “Yoshi’s Nightmare”
  - (morning) Project 1: Build a half adder circuit with logic gates on a breadboard
  - (afternoon) Project 2: Design/Program your FPGA to behave like a half adder
  - (afternoon) Project 3: Simulate your half adder project
- **Tuesday:**
  - Project 4: Design/Simulate/Program your FPGA to behave as a calculator
  - Project 5: Design/(Simulate)/Program your FPGA to behave as a UART to communicate with your computer.
- **Wednesday:**
  - Project 6.0: Design/(Simulate)/Program your FPGA to play basic Pong
  - Project 6.1: Add Pong features: scoring, keyboard control, sounds, etc
- **Thursday:**
  - Project 6.1 continued: Add more Pong features: scoring, keyboard control, sounds, etc
- **Friday**
  - Project Brief Outs

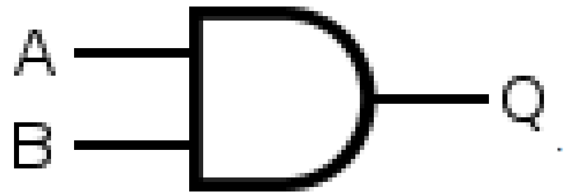
Project 0: Program FPGA to Play Yoshi’s Nightmare in 7 Easy Steps!

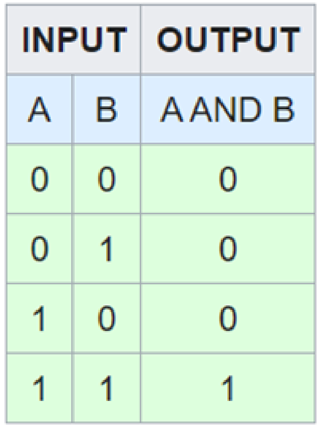
1. Open Vivado on laptop
2. Connect BASYS3 to laptop with micro-usb cable
3. Connect BASYS3 to monitor with VGA cable
4. Open Hardware Manager in Vivado
    1. Click “autoconnect”
    2. Select Program File “nightmare.bit”
    3. Click “Program”
5. Use center pushbutton to start game
6. Use left/right/up pushbutton to move Yoshi
7. Try to score more than 620 points



Project 0 Goals:
1) Run Vivado FPGA EDA software
2) Program FPGA to do something

FPGA hardware behavior controlled by “logic gates” that implement Boolean equations





INPUT		OUTPUT
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

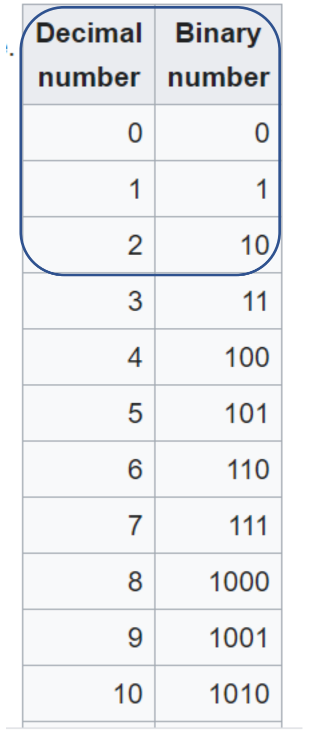
“AND” logic gate

FPGAs contain up to 20 million logic gates!

https://en.wikipedia.org/wiki/Logic_gate

Project 1: Add two binary numbers (half adder) with logic gates on a breadboard

Decimal to binary conversion



Decimal number	Binary number
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111
8	1000
9	1001
10	1010

Half Adder (Decimal)

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 2

Half Adder (Binary)

0 + 0 = 00
0 + 1 = 01
1 + 0 = 01
1 + 1 = 10

Project 1 Goals:

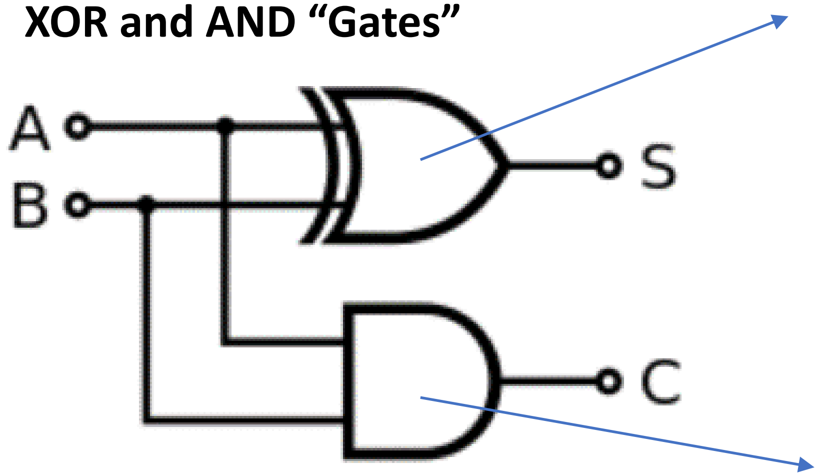
1) Learn about logic gates
2) Build a physical circuit with logic gates

We can build a half-adder circuit with two logic gates

Half Adder "Truth Table"

Inputs		Outputs	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Half Adder "Circuit" with XOR and AND "Gates"



INPUT		OUTPUT
A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

INPUT		OUTPUT
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

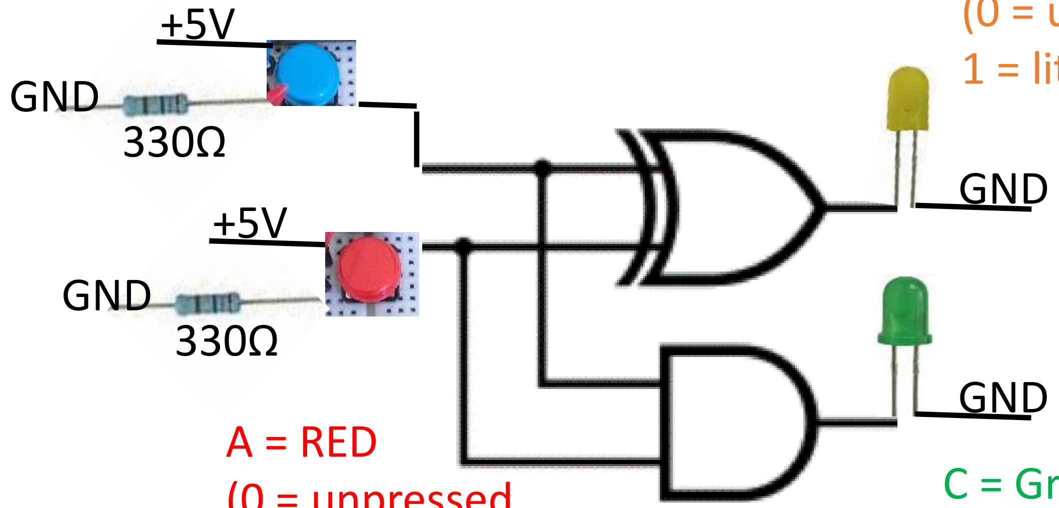
- Note that our FPGA (Artix7 A35T has **35000** gates inside it!)

Let’s build our 5V half-adder with logic gates on a breadboard

Half Adder “Truth Table”

Inputs		Outputs	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Half Adder “Circuit” with XOR and AND “Gates”

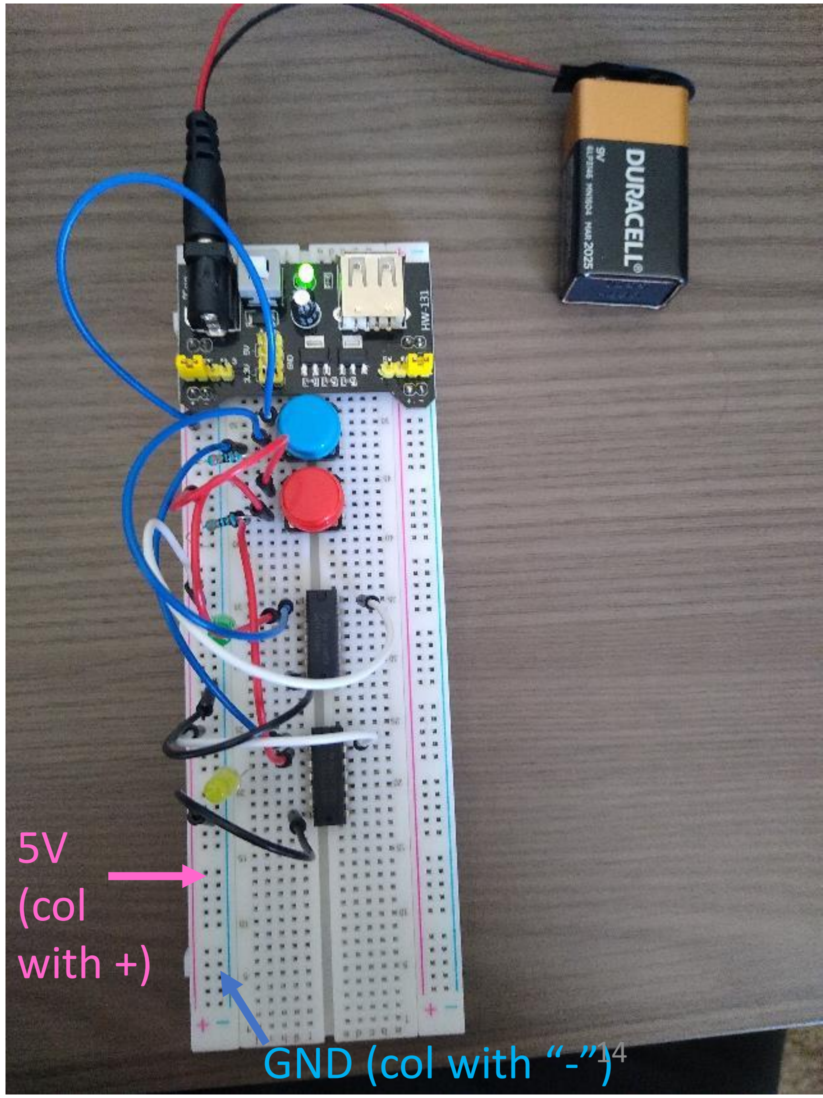


B = Blue
(0 = unpressed,
1 = pressed)

S = Yellow
(0 = unlit,
1 = lit)

A = RED
(0 = unpressed,
1 = pressed)

C = Green
(0 = unlit,
1 = lit)



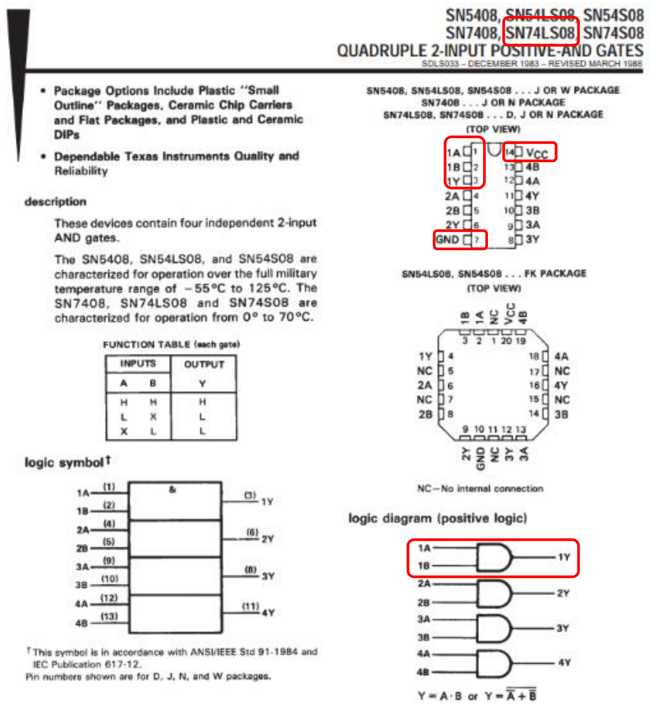
Datasheets show pin connections for the "gates" in our half adder

AND Gate

Datasheet

XOR Gate

Datasheet



SN5408, SN54LS08, SN54S08
SN7408, SN74LS08, SN74S08
QUADRUPLE 2-INPUT POSITIVE-AND GATES
SDLS033 – DECEMBER 1983 – REVISED MARCH 1988

- Package Options Include Plastic "Small Outline" Packages, Ceramic Chip Carriers and Flat Packages, and Plastic and Ceramic DIPs
- Dependable Texas Instruments Quality and Reliability

description

These devices contain four independent 2-input AND gates.

The SN5408, SN54LS08, and SN54S08 are characterized for operation over the full military temperature range of −55°C to 125°C. The SN7408, SN74LS08 and SN74S08 are characterized for operation from 0° to 70°C.

FUNCTION TABLE (each gate)

INPUTS		OUTPUT
A	B	Y
H	H	H
L	X	L
X	L	L

logic symbol†

† This symbol is in accordance with ANSI/IEEE Std 91-1984 and IEC Publication 617-12.
Pin numbers shown are for D, J, N, and W packages.

SN5408, SN54LS08, SN54S08 . . . J OR W PACKAGE
SN7408 . . . J OR N PACKAGE
SN74LS08, SN74S08 . . . D, J OR N PACKAGE
(TOP VIEW)

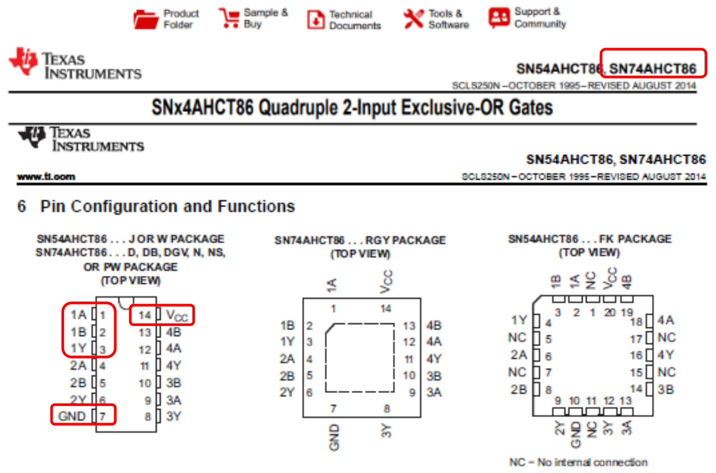
SN54LS08, SN54S08 . . . FK PACKAGE
(TOP VIEW)

NC – No internal connection

logic diagram (positive logic)

$Y = A \cdot B$ or $Y = \overline{\overline{A} + \overline{B}}$

1A 1 · 1B 2 · 1Y 3 · 2A 4 · 2B 5 · 2Y 6 · GND 7 · 3Y 8 · 3A 9 · 3B 10 · 4Y 11 · 4A 12 · 4B 13 · 14 VCC
1B 3 · 1A 2 · NC 1 · VCC 20 · 4B 19 · 1Y 4 · NC 5 · 2A 6 · NC 7 · 2B 8 · 4A 18 · NC 17 · 4Y 16 · NC 15 · 3B 14 · 2Y 9 · GND 10 · NC 11 · 3Y 12 · 3A 13
&
1A (1) · 1B (2) · 2A (4) · 2B (5) · 3A (9) · 3B (10) · 4A (12) · 4B (13) · (3) 1Y · (6) 2Y · (8) 3Y · (11) 4Y
1A · 1B · 1Y · 2A · 2B · 2Y · 3A · 3B · 3Y · 4A · 4B · 4Y



Product Folder · Sample & Buy · Technical Documents · Tools & Software · Support & Community

TEXAS INSTRUMENTS

SN54AHCT86, SN74AHCT86
SCLS250N – OCTOBER 1995 – REVISED AUGUST 2014

SNx4AHCT86 Quadruple 2-Input Exclusive-OR Gates

TEXAS INSTRUMENTS

www.ti.com

SN54AHCT86, SN74AHCT86
SCLS250N – OCTOBER 1995 – REVISED AUGUST 2014

6 Pin Configuration and Functions

SN54AHCT86 . . . J OR W PACKAGE
SN74AHCT86 . . . D, DB, DGV, N, NS, OR PW PACKAGE
(TOP VIEW)

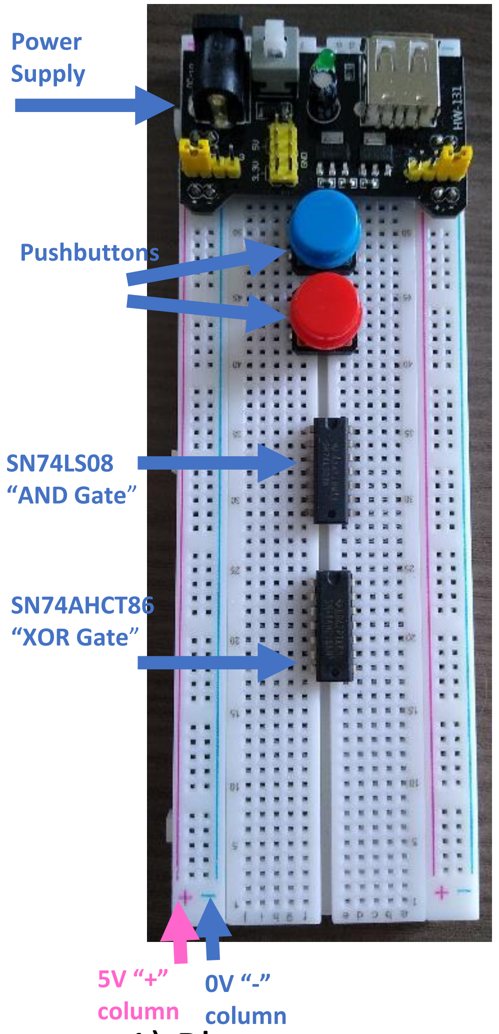
SN74AHCT86 . . . RGY PACKAGE
(TOP VIEW)

SN54AHCT86 . . . FK PACKAGE
(TOP VIEW)

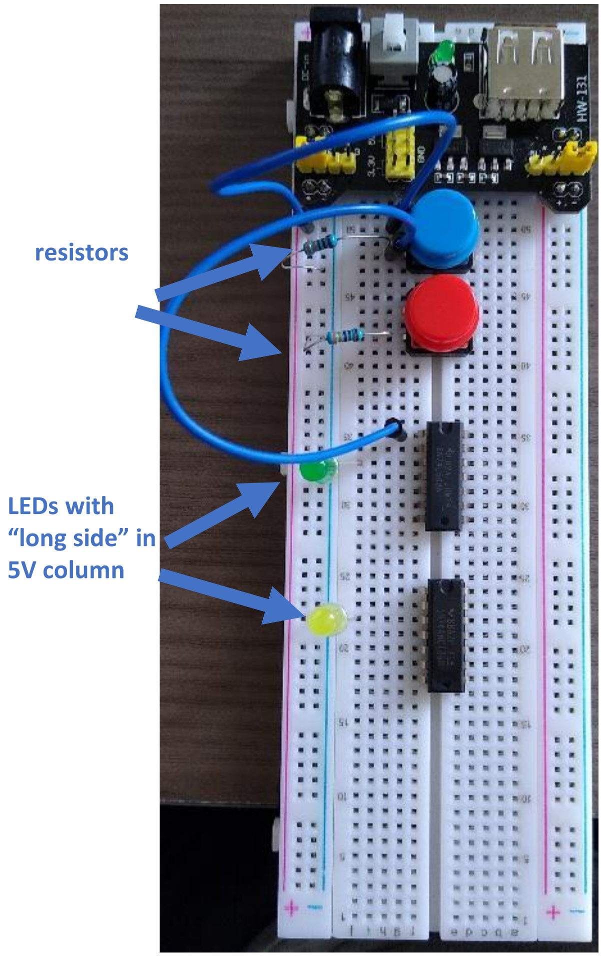
NC – No internal connection

1A 1 · 1B 2 · 1Y 3 · 2A 4 · 2B 5 · 2Y 6 · GND 7 · 3Y 8 · 3A 9 · 3B 10 · 4Y 11 · 4A 12 · 4B 13 · 14 VCC
1A 1 · VCC 14 · 1B 2 · 1Y 3 · 2A 4 · 2B 5 · 2Y 6 · 4B 13 · 4A 12 · 4Y 11 · 3B 10 · 3A 9 · GND 7 · 3Y 8
1B 3 · 1A 2 · NC 1 · VCC 20 · 4B 19 · 1Y 4 · NC 5 · 2A 6 · NC 7 · 2B 8 · 4A 18 · NC 17 · 4Y 16 · NC 15 · 3B 14 · 2Y 9 · GND 10 · NC 11 · 3Y 12 · 3A 13

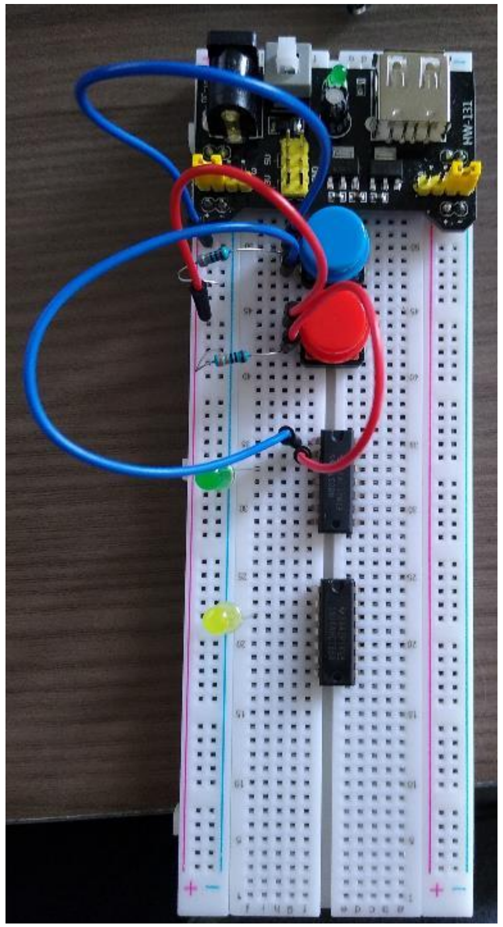
Project 1: Assemble breadboard



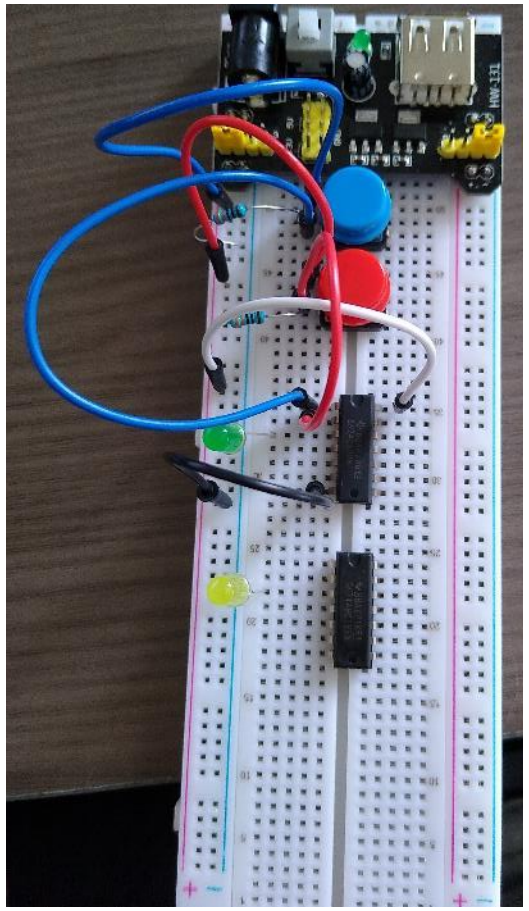
1) Place components on breadboard



2) Add blue wires, LEDs, and 330 ohm resistors

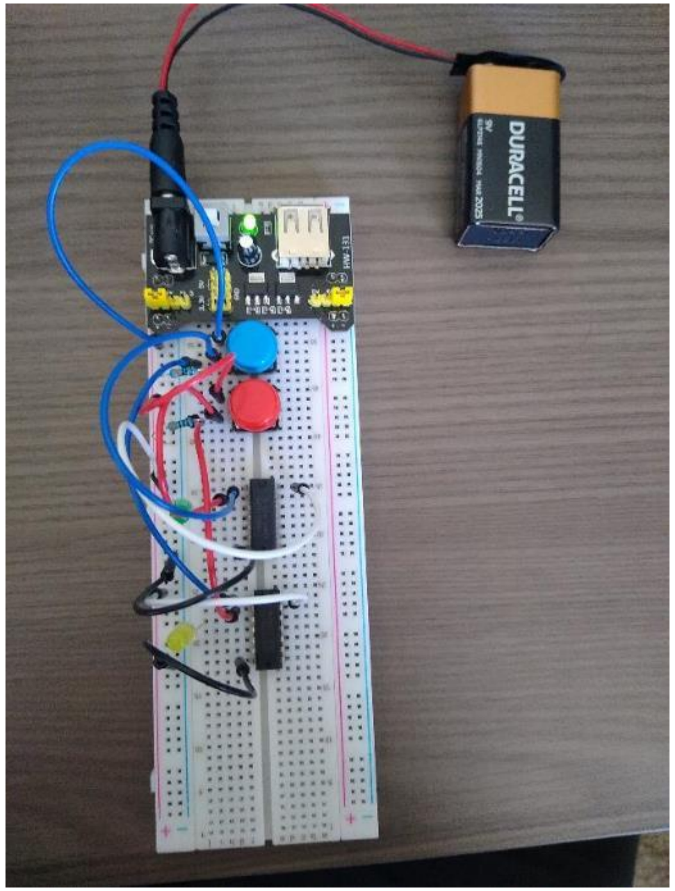


3) Add red wires to blue button and 330Ω resistor



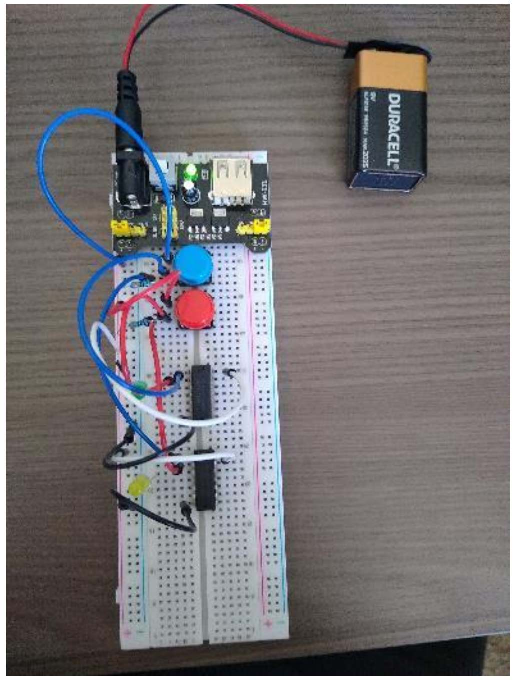
4) Add black wire to "0V" and white wire to "5V"

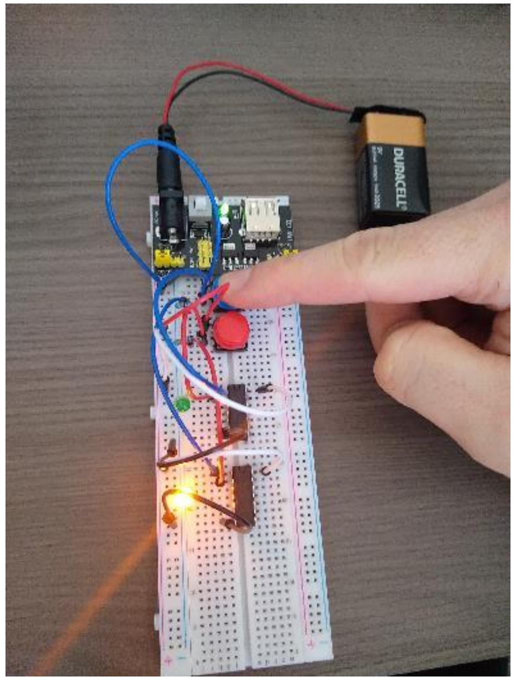
Project 1: Assemble breadboard

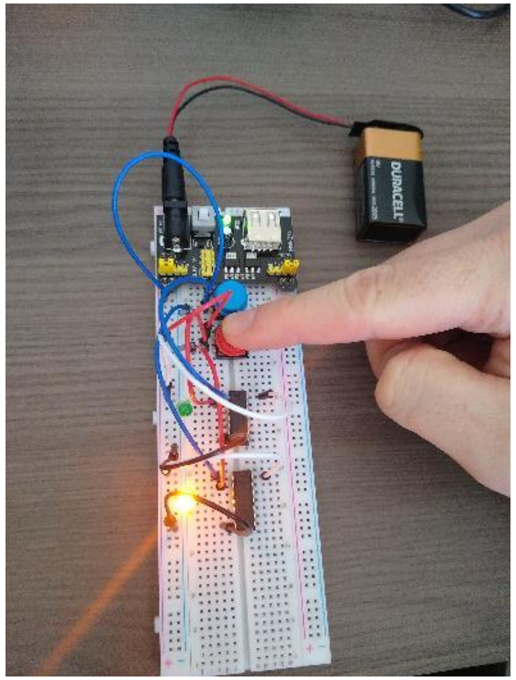


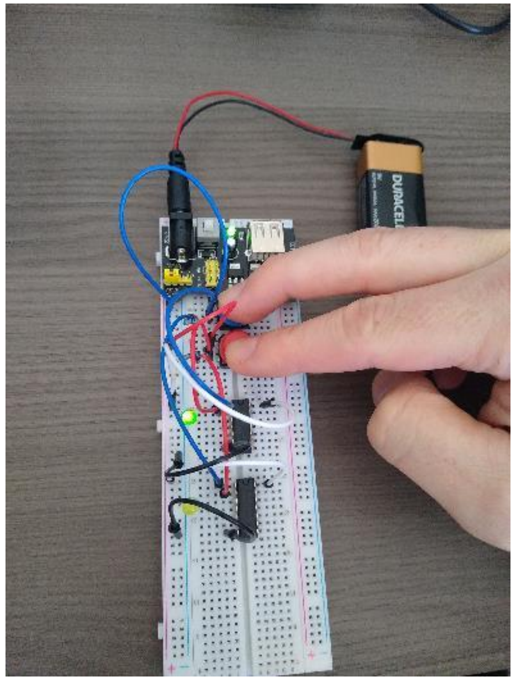
5) Connect blue, red, white, black wires to XOR gate. Connect 9V battery to power supply

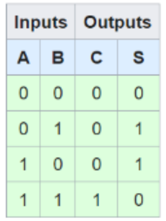
Let’s test our half-adder breadboard solution


Inputs: A=0,B=0
outputs: C=0,S=0


Inputs: A=1,B=0
outputs: C=0,S=1


Inputs: A=0,B=1
outputs: C=0,S=1


Inputs: A=1,B=1
outputs: C=1,S=0



Inputs		Outputs	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Half Adder “Truth Table”

Introduction to FPGAs Lecture 2: Design/Simulate FPGA Design with Vivado

Adam Duncan

Vivado and FPGA Design 101

Xilinx **Vivado** a software program used to design, program, simulate your FPGA.

We write a code in a hardware description language (HDL) like **Verilog** to specify the intended hardware behavior of the FPGA.

Vivado compiles the Verilog and automatically places logic gates inside the FPGA to realize the intended FPGA functionality.

Vivado produces a **bitstream** file that is loaded into the physical FPGA to configure the FPGA behavior.

Verilog 101: Verilog, module, assign

A Verilog **module** is a hierarchal block in Verilog within input/output ports

An **assign** statement performs assignments and arithmetic operations on ports and wires within a module.

```
module top(
   output LED
   );
assign LED = 1;
endmodule
```

Vivado 101: XDC files

Xilinx **Vivado** a software program that turns Verilog into a bitstream file that can configures the hardware behavior of an FPGA.

Xilinx design constraints (**XDC**) files map FPGA physical pins to Verilog top-level module ports.

```
set_property PACKAGE_PIN U16 [get_ports {LED}]
set_property IOSTANDARD LVCMOS33 [get_ports {LED}]
```

Project 2: Create half adder Vivado project from scratch

0) Open Vivado and create new project with default options

1) New Project
- Click "Do not specify sources at this time"

2) Select Part "xc7a35ticpg236-1L"

Follow steps outlined in below link if needed:
https://users.wpi.edu/~rjduck/Basys3%20Vivado%20Decoder%20Tutorial.pdf

Project 2 Goals:
1) Use Vivado to create an FPGA design
2) Use Vivado to program your FPGA design

Project 2: Create half adder Vivado project from scratch

3) create design file “top.v”

```
module top(
   input A, // input port
   input B, // input port
   output SUM, // output port
   output CARRY // output port
   );


  assign SUM   = A ^ B;  // bitwise xor
  assign CARRY = A & B;  // bitwise and


endmodule
```

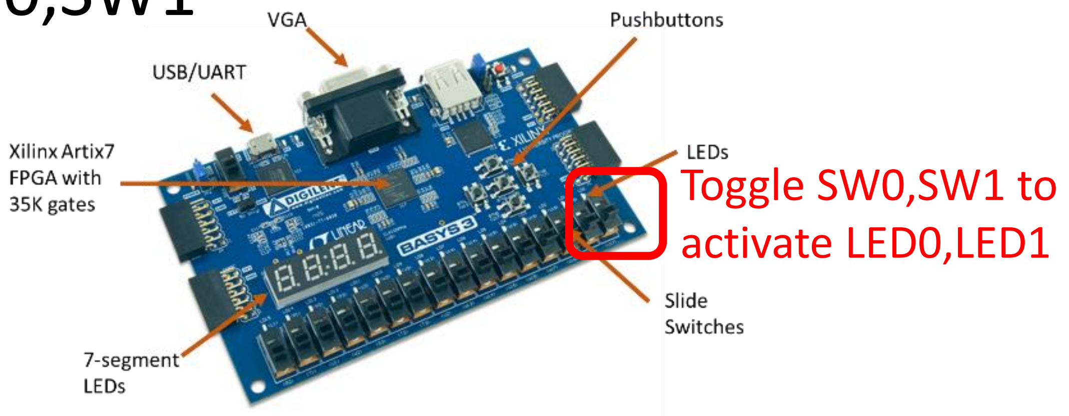
4) create constraints file “top.xdc”

```
set_property PACKAGE_PIN V17 [get_ports A]
set_property IOSTANDARD LVCMOS33 [get_ports A]
set_property PACKAGE_PIN V16 [get_ports B]
set_property IOSTANDARD LVCMOS33 [get_ports B]
set_property PACKAGE_PIN U16 [get_ports SUM]
set_property IOSTANDARD LVCMOS33 [get_ports SUM]
set_property PACKAGE_PIN E19 [get_ports CARRY]
set_property IOSTANDARD LVCMOS33 [get_ports CARRY]
```

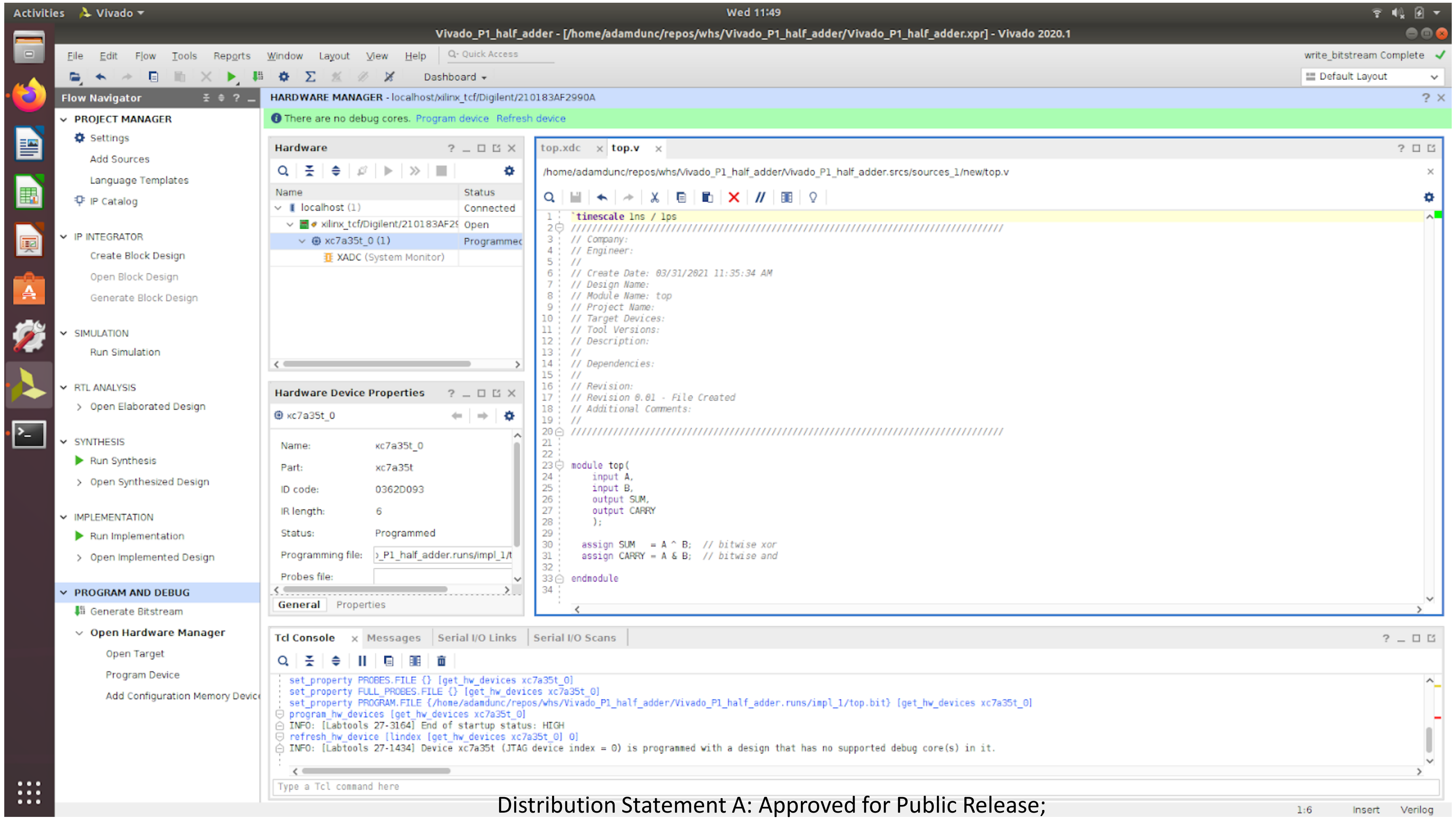
Follow steps outlined in below link as needed
https://users.wpi.edu/~rjduck/Basys3%20Vivado%20Decoder%20Tutorial.pdf

Program half adder bitstream FPGA on BASYS3

- Open “Hardware Manager” in Vivado
- Click “Program Device”
- Select “top.bit”from “~/WHS/Projects/student_p1.runs/impl_1“ or equivalent directory
- After FPGA is programmed toggle SW0,SW1
  - You should see LED0,LED1 toggle


VGA
Pushbuttons
USB/UART
Xilinx Artix7 FPGA with 35K gates
LEDs
Toggle SW0,SW1 to activate LED0,LED1
Slide Switches
7-segment LEDs

Half adder Vivado Project View


Vivado_P1_half_adder - [/home/adamdunc/repos/whs/Vivado_P1_half_adder/Vivado_P1_half_adder.xpr] - Vivado 2020.1

write_bitstream Complete ✓

HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210183AF2990A

There are no debug cores. Program device Refresh device

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210183AF2	Open
xc7a35t_0 (1)	Programmed
XADC (System Monitor)	

Hardware Device Properties: xc7a35t_0

Name:	xc7a35t_0
Part:	xc7a35t
ID code:	0362D093
IR length:	6
Status:	Programmed
Programming file:	_P1_half_adder.runs/impl_1/t
Probes file:	

/home/adamdunc/repos/whs/Vivado_P1_half_adder/Vivado_P1_half_adder.srcs/sources_1/new/top.v

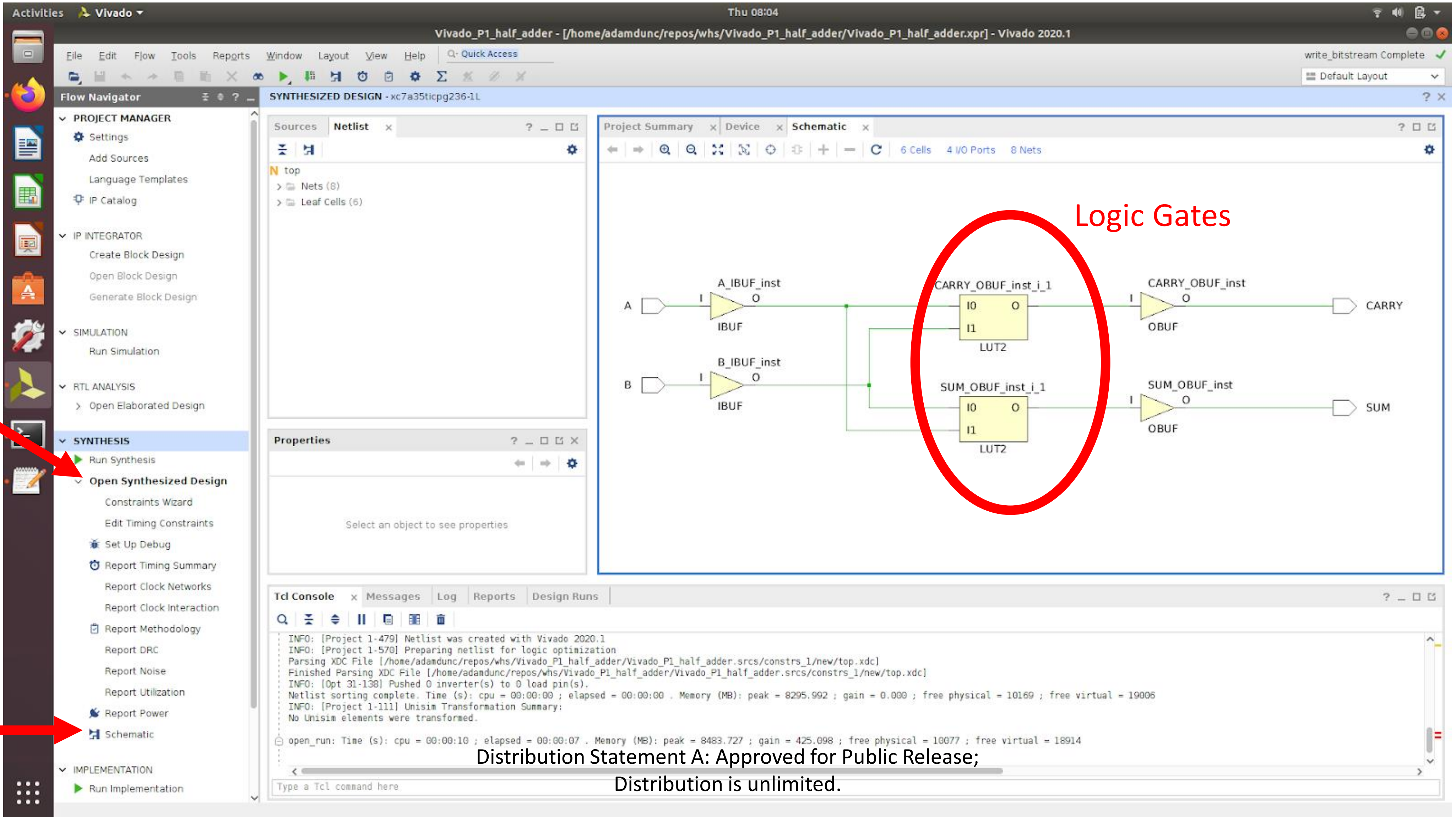
```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/31/2021 11:35:34 AM
// Design Name:
// Module Name: top
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module top(
    input A,
    input B,
    output SUM,
    output CARRY
    );

  assign SUM   = A ^ B;  // bitwise xor
  assign CARRY = A & B;  // bitwise and

endmodule
```

Tcl Console

```
set_property PROBES.FILE {} [get_hw_devices xc7a35t_0]
set_property FULL_PROBES.FILE {} [get_hw_devices xc7a35t_0]
set_property PROGRAM.FILE {/home/adamdunc/repos/whs/Vivado_P1_half_adder/Vivado_P1_half_adder.runs/impl_1/top.bit} [get_hw_devices xc7a35t_0]
program_hw_devices [get_hw_devices xc7a35t_0]
INFO: [Labtools 27-3164] End of startup status: HIGH
refresh_hw_device [lindex [get_hw_devices xc7a35t_0] 0]
INFO: [Labtools 27-1434] Device xc7a35t (JTAG device index = 0) is programmed with a design that has no supported debug core(s) in it.
```

Distribution Statement A: Approved for Public Release; Distribution is unlimited.

Open Synthesized Design to see “logic gate” usage



```
INFO: [Project 1-479] Netlist was created with Vivado 2020.1
INFO: [Project 1-570] Preparing netlist for logic optimization
Parsing XDC File [/home/adamdunc/repos/whs/Vivado_P1_half_adder/Vivado_P1_half_adder.srcs/constrs_1/new/top.xdc]
Finished Parsing XDC File [/home/adamdunc/repos/whs/Vivado_P1_half_adder/Vivado_P1_half_adder.srcs/constrs_1/new/top.xdc]
INFO: [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
Netlist sorting complete. Time (s): cpu = 00:00:00 ; elapsed = 00:00:00 . Memory (MB): peak = 8295.992 ; gain = 0.000 ; free physical = 10169 ; free virtual = 19006
INFO: [Project 1-111] Unisim Transformation Summary:
No Unisim elements were transformed.

open_run: Time (s): cpu = 00:00:10 ; elapsed = 00:00:07 . Memory (MB): peak = 8483.727 ; gain = 425.098 ; free physical = 10077 ; free virtual = 18914
```

FPGA 101: Simulations

FPGA **simulations** allow you to control and observe the behavior of your Verilog code.

Simulations are useful for assessing performance and debugging.

Simulations allow you to control/observe signals internal to the circuit running on the FPGA.

Verilog 101: Testbenches

A Verilog **testbench** is an HDL file that instantiates a Verilog module to simulate its behavior.

Testbench uses Verilog code to stimulate the module being tested.

Project 3: Simulate your half adder

0) Open your half adder Vivado project

1) Create simulation source “tb_top.v”

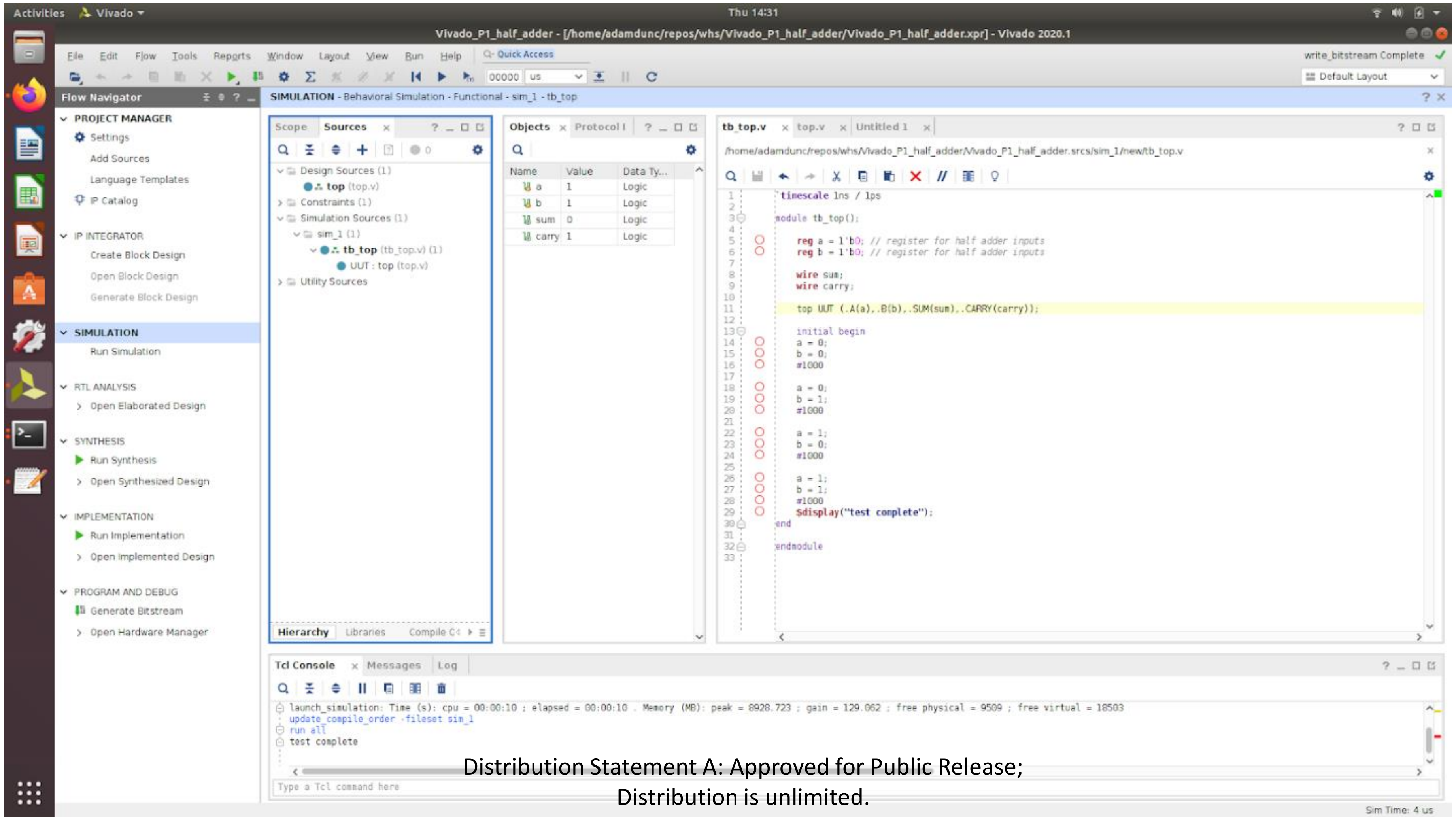
2) Add code as directed in the Project 3 README

Follow steps outlined in below link if needed:
https://users.wpi.edu/~rjduck/Basys3%20Vivado%20Decoder%20Tutorial.pdf

Project 3 Goals:

1) Use Vivado to simulate an FPGA design

Project 3: Create tb_top.v testbench file and simulate



Vivado_P1_half_adder - [/home/adamdunc/repos/whs/Vivado_P1_half_adder/Vivado_P1_half_adder.xpr] - Vivado 2020.1

SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_top

Name	Value	Data Type
a	1	Logic
b	1	Logic
sum	0	Logic
carry	1	Logic

/home/adamdunc/repos/whs/Vivado_P1_half_adder/Vivado_P1_half_adder.srcs/sim_1/new/tb_top.v

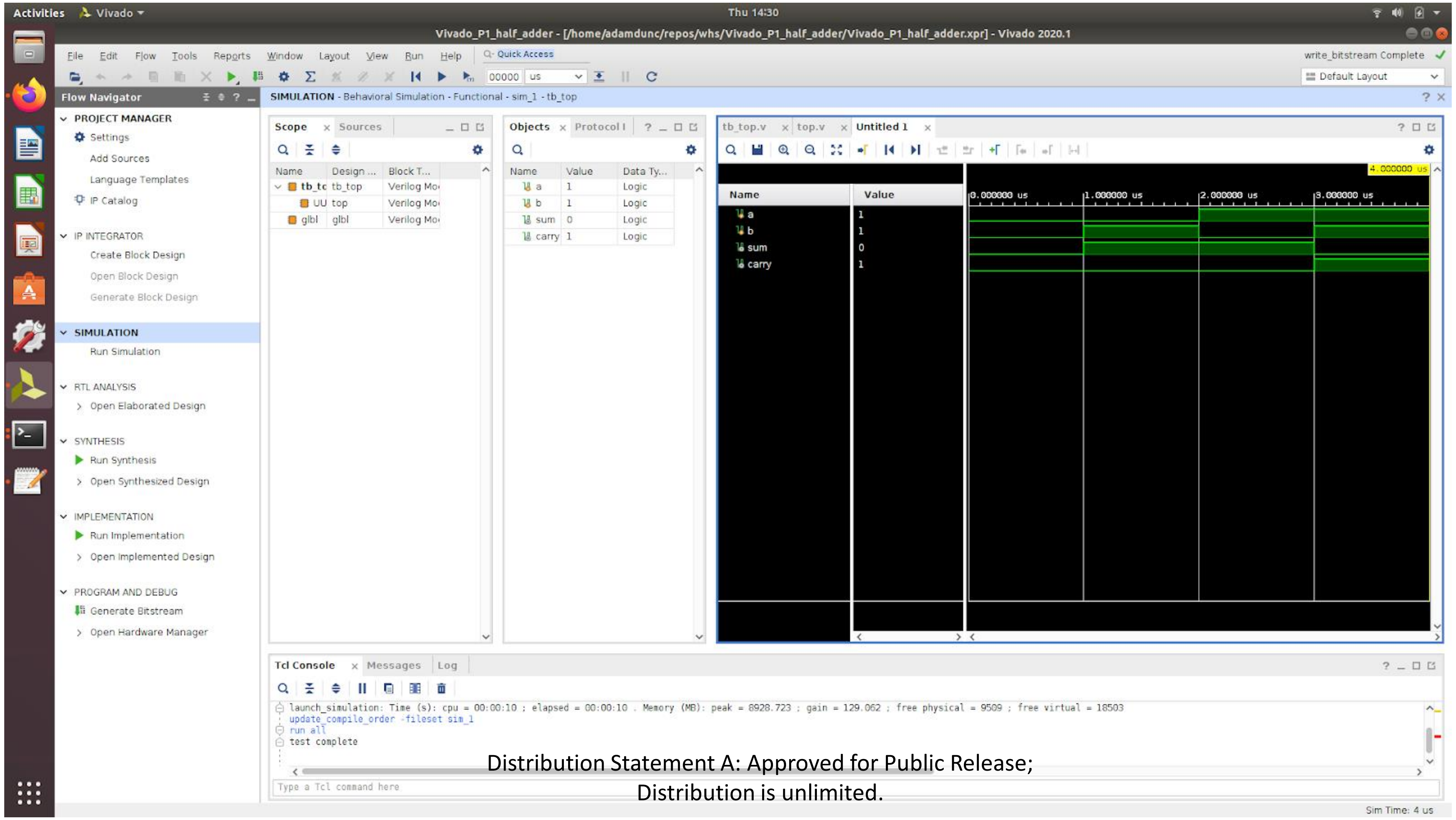
```
`timescale 1ns / 1ps

module tb_top();

    reg a = 1'b0; // register for half adder inputs
    reg b = 1'b0; // register for half adder inputs

    wire sum;
    wire carry;

    top UUT (.A(a),.B(b),.SUM(sum),.CARRY(carry));

    initial begin
    a = 0;
    b = 0;
    #1000

    a = 0;
    b = 1;
    #1000

    a = 1;
    b = 0;
    #1000

    a = 1;
    b = 1;
    #1000
    $display("test complete");
end

endmodule
```

```
launch_simulation: Time (s): cpu = 00:00:10 ; elapsed = 00:00:10 . Memory (MB): peak = 8928.723 ; gain = 129.062 ; free physical = 9509 ; free virtual = 18503
update_compile_order -fileset sim_1
run all
test complete
```

Sim Time: 4 us

Distribution Statement A: Approved for Public Release;
Distribution is unlimited.

Run Vivado “Simulate”



Vivado_P1_half_adder - [/home/adamdunc/repos/whs/Vivado_P1_half_adder/Vivado_P1_half_adder.xpr] - Vivado 2020.1

write_bitstream Complete

SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_top

Name	Design ...	Block T...
tb_tc	tb_top	Verilog Mo
UU	top	Verilog Mo
glbl	glbl	Verilog Mo

Name	Value	Data Ty...
a	1	Logic
b	1	Logic
sum	0	Logic
carry	1	Logic

```
launch_simulation: Time (s): cpu = 00:00:10 ; elapsed = 00:00:10 . Memory (MB): peak = 8928.723 ; gain = 129.062 ; free physical = 9509 ; free virtual = 18503
update_compile_order -fileset sim_1
run all
test complete
```

Sim Time: 4 us

Distribution Statement A: Approved for Public Release;
Distribution is unlimited.

Project 4: Design/Simulate/Program your FPGA to be a calculator

1) Add code as directed in the Project 4 README

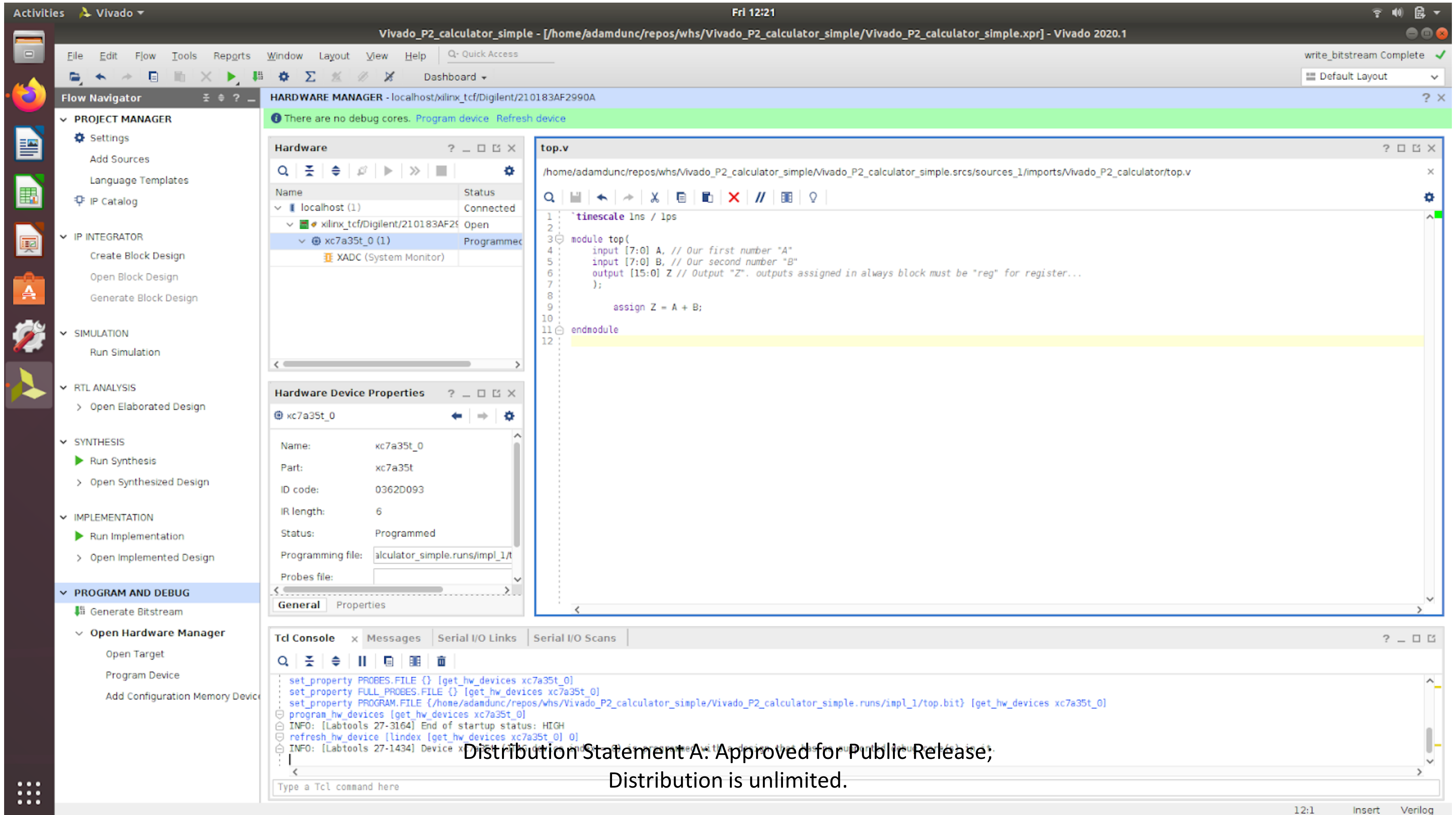
See github/adamdunc/whs for README

Project 3 Goals:

1) Learn useful Verilog syntax and features
2) Utilize 7-segment LEDs

Follow steps outlined in below link if needed: https://users.wpi.edu/~rjduck/Basys3%20Vivado%20Decoder%20Tutorial.pdf

Project 4 Step 1: Design/Simulate Calculator



Vivado_P2_calculator_simple - [/home/adamdunc/repos/whs/Vivado_P2_calculator_simple/Vivado_P2_calculator_simple.xpr] - Vivado 2020.1

write_bitstream Complete ✓

HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210183AF2990A

There are no debug cores. Program device Refresh device

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210183AF2990A	Open
xc7a35t_0 (1)	Programmed
XADC (System Monitor)	

Hardware Device Properties

Name:	xc7a35t_0
Part:	xc7a35t
ID code:	0362D093
IR length:	6
Status:	Programmed
Programming file:	alculator_simple.runs/impl_1/t
Probes file:	

top.v

/home/adamdunc/repos/whs/Vivado_P2_calculator_simple/Vivado_P2_calculator_simple.srcs/sources_1/imports/Vivado_P2_calculator/top.v

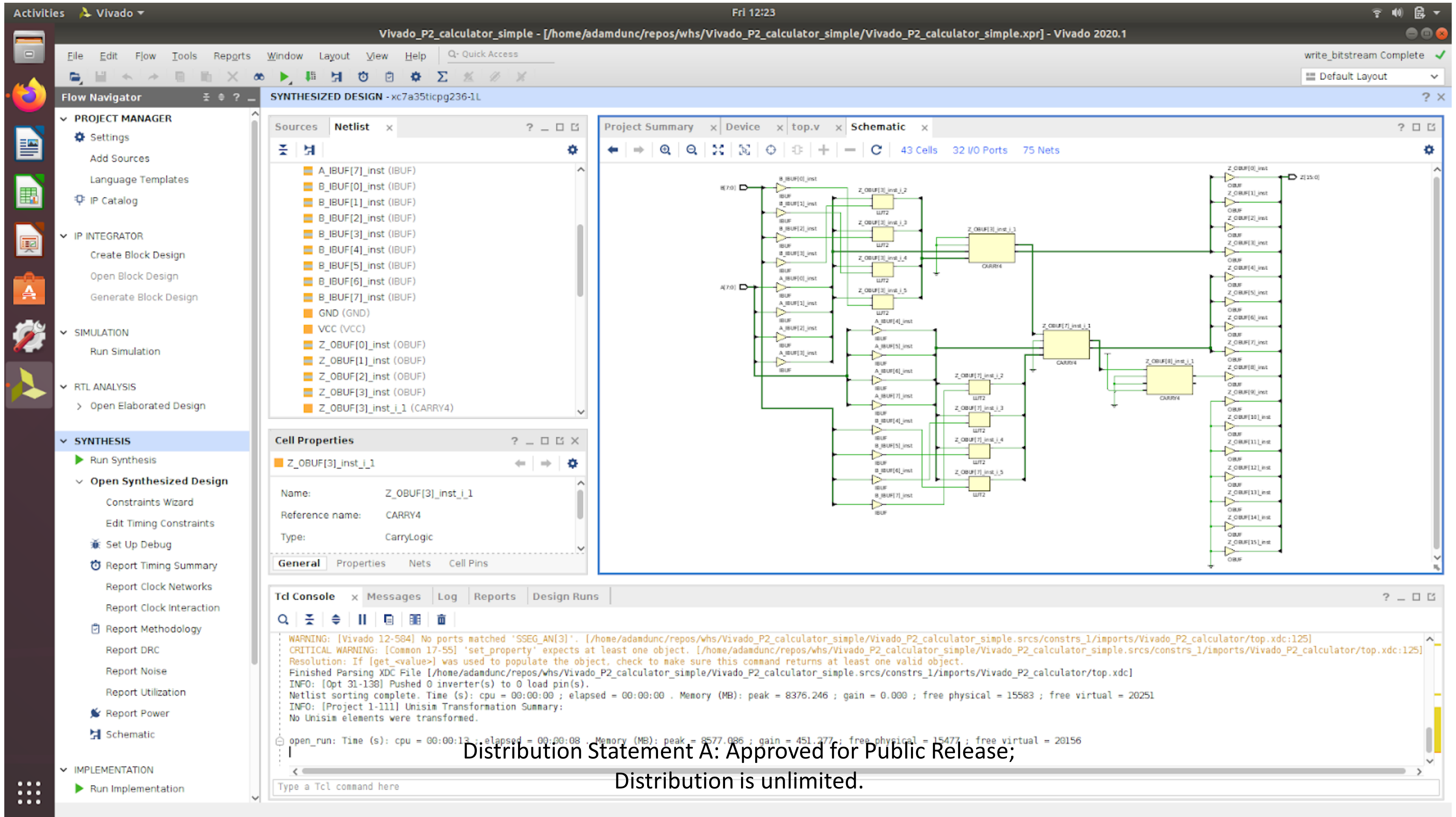
```verilog
`timescale 1ns / 1ps

module top(
    input [7:0] A, // Our first number "A"
    input [7:0] B, // Our second number "B"
    output [15:0] Z // Output "Z". outputs assigned in always block must be "reg" for register...
    );

        assign Z = A + B;

endmodule

```

Tcl Console

```
set_property PROBES.FILE {} [get_hw_devices xc7a35t_0]
set_property FULL_PROBES.FILE {} [get_hw_devices xc7a35t_0]
set_property PROGRAM.FILE {/home/adamdunc/repos/whs/Vivado_P2_calculator_simple/Vivado_P2_calculator_simple.runs/impl_1/top.bit} [get_hw_devices xc7a35t_0]
program_hw_devices [get_hw_devices xc7a35t_0]
INFO: [Labtools 27-3164] End of startup status: HIGH
refresh_hw_device [lindex [get_hw_devices xc7a35t_0] 0]
INFO: [Labtools 27-1434] Device [illegible]
```

Distribution Statement A: Approved for Public Release; Distribution is unlimited.

Project 4 Step 2: Note increased logic gate usage



A_IBUF[7]_inst (IBUF)
B_IBUF[0]_inst (IBUF)
B_IBUF[1]_inst (IBUF)
B_IBUF[2]_inst (IBUF)
B_IBUF[3]_inst (IBUF)
B_IBUF[4]_inst (IBUF)
B_IBUF[5]_inst (IBUF)
B_IBUF[6]_inst (IBUF)
B_IBUF[7]_inst (IBUF)
GND (GND)
VCC (VCC)
Z_OBUF[0]_inst (OBUF)
Z_OBUF[1]_inst (OBUF)
Z_OBUF[2]_inst (OBUF)
Z_OBUF[3]_inst (OBUF)
Z_OBUF[3]_inst_i_1 (CARRY4)

Z_OBUF[3]_inst_i_1

Name:	Z_OBUF[3]_inst_i_1
Reference name:	CARRY4
Type:	CarryLogic

```
WARNING: [Vivado 12-584] No ports matched 'SSEG_AN[3]'. [/home/adamdunc/repos/whs/Vivado_P2_calculator_simple/Vivado_P2_calculator_simple.srcs/constrs_1/imports/Vivado_P2_calculator/top.xdc:125]
CRITICAL WARNING: [Common 17-55] 'set_property' expects at least one object. [/home/adamdunc/repos/whs/Vivado_P2_calculator_simple/Vivado_P2_calculator_simple.srcs/constrs_1/imports/Vivado_P2_calculator/top.xdc:125]
Resolution: If [get_<value>] was used to populate the object, check to make sure this command returns at least one valid object.
Finished Parsing XDC File [/home/adamdunc/repos/whs/Vivado_P2_calculator_simple/Vivado_P2_calculator_simple.srcs/constrs_1/imports/Vivado_P2_calculator/top.xdc]
INFO: [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
Netlist sorting complete. Time (s): cpu = 00:00:00 ; elapsed = 00:00:00 . Memory (MB): peak = 8376.246 ; gain = 0.000 ; free physical = 15583 ; free virtual = 20251
INFO: [Project 1-111] Unisim Transformation Summary:
No Unisim elements were transformed.

open_run: Time (s): cpu = 00:00:13 ; elapsed = 00:00:08 . Memory (MB): peak = 8577.086 ; gain = 451.277 ; free physical = 15477 ; free virtual = 20156
```

Distribution Statement A: Approved for Public Release; Distribution is unlimited.

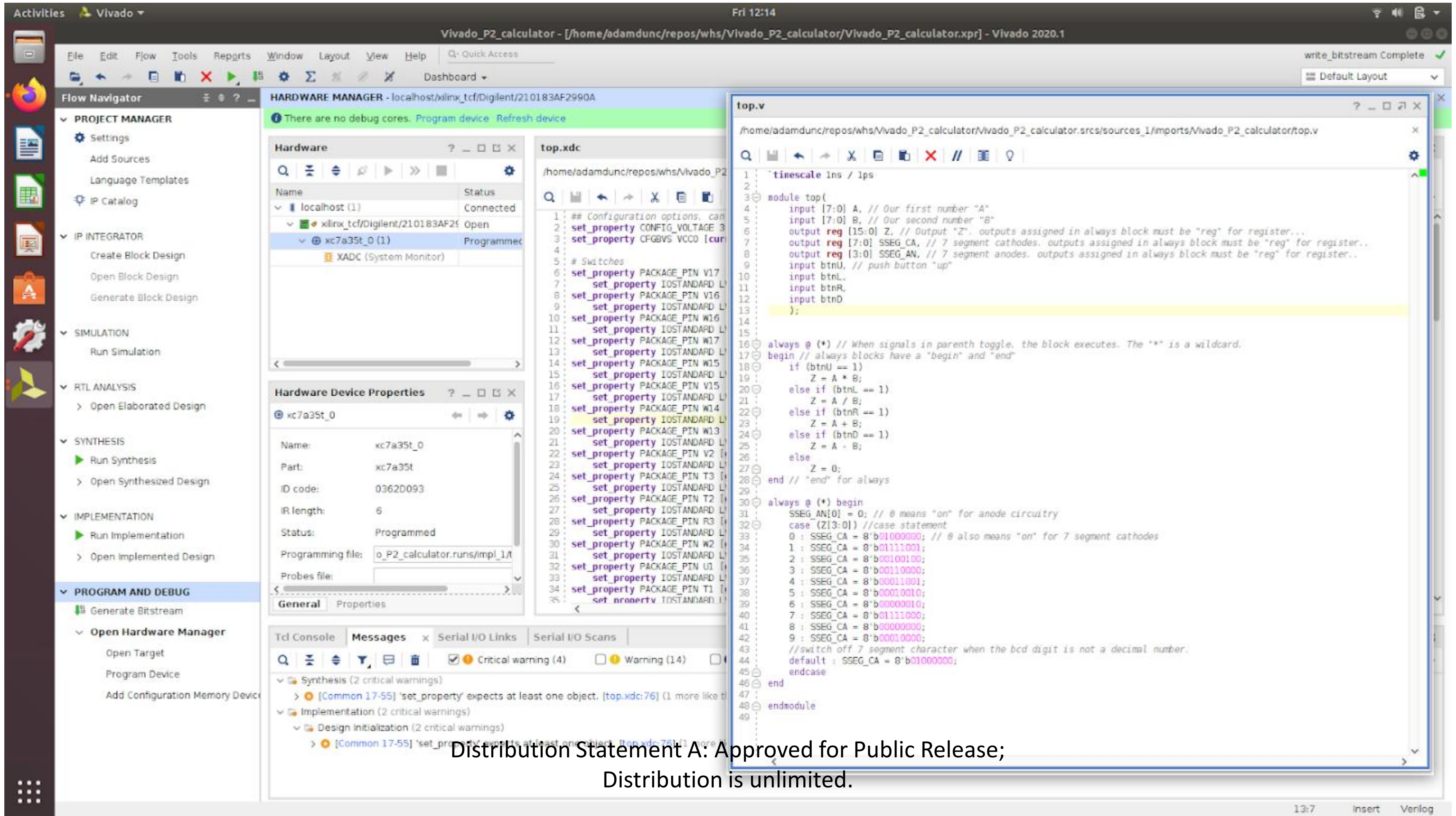
Verilog 101: always, reg

An **always** block executes any time the contents within its parenthesis toggle.

A **reg** is a register

```
module top(
   input [7:0] A, input [7:0] B, output reg [15:0] Z
   );
always @ (*)
       begin
          if (btnU == 1)
                      Z = A + B;
          else if (btnL == 1)
                      Z = A / B;
       end
endmodule
```

Project 4 Step 3: Modify calculator to include always block to select more operands



Vivado_P2_calculator - [/home/adamdunc/repos/whs/Vivado_P2_calculator/Vivado_P2_calculator.xpr] - Vivado 2020.1

HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210183AF2990A

There are no debug cores. Program device Refresh device

Hardware Device Properties: xc7a35t_0

Name:	xc7a35t_0
Part:	xc7a35t
ID code:	03620093
IR length:	6
Status:	Programmed
Programming file:	o_P2_calculator.runs/impl_1/t
Probes file:	

top.v — /home/adamdunc/repos/whs/Vivado_P2_calculator/Vivado_P2_calculator.srcs/sources_1/imports/Vivado_P2_calculator/top.v

```verilog
`timescale 1ns / 1ps

module top(
    input [7:0] A, // Our first number "A"
    input [7:0] B, // Our second number "B"
    output reg [15:0] Z, // Output "Z". outputs assigned in always block must be "reg" for register...
    output reg [7:0] SSEG_CA, // 7 segment cathodes. outputs assigned in always block must be "reg" for register..
    output reg [3:0] SSEG_AN, // 7 segment anodes. outputs assigned in always block must be "reg" for register..
    input btnU, // push button "up"
    input btnL,
    input btnR,
    input btnD
    );


always @ (*) // When signals in parenth toggle, the block executes. The "*" is a wildcard.
begin // always blocks have a "begin" and "end"
    if (btnU == 1)
        Z = A * B;
    else if (btnL == 1)
        Z = A / B;
    else if (btnR == 1)
        Z = A + B;
    else if (btnD == 1)
        Z = A - B;
    else
        Z = 0;
end // "end" for always

always @ (*) begin
    SSEG_AN[0] = 0; // 0 means "on" for anode circuitry
    case (Z[3:0]) //case statement
    0 : SSEG_CA = 8'b01000000; // 0 also means "on" for 7 segment cathodes
    1 : SSEG_CA = 8'b01111001;
    2 : SSEG_CA = 8'b00100100;
    3 : SSEG_CA = 8'b00110000;
    4 : SSEG_CA = 8'b00011001;
    5 : SSEG_CA = 8'b00010010;
    6 : SSEG_CA = 8'b00000010;
    7 : SSEG_CA = 8'b01111000;
    8 : SSEG_CA = 8'b00000000;
    9 : SSEG_CA = 8'b00010000;
    //switch off 7 segment character when the bcd digit is not a decimal number.
    default : SSEG_CA = 8'b01000000;
    endcase
end

endmodule
```

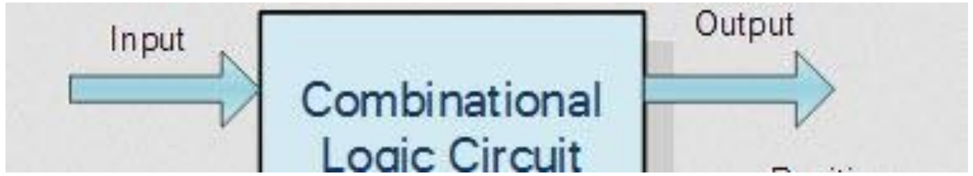
```
Synthesis (2 critical warnings)
  [Common 17-55] 'set_property' expects at least one object. [top.xdc:76] (1 more like this)
Implementation (2 critical warnings)
  Design Initialization (2 critical warnings)
    [Common 17-55] 'set_property' expects at least one object. [top.xdc:76]
```

Distribution Statement A: Approved for Public Release;
Distribution is unlimited.

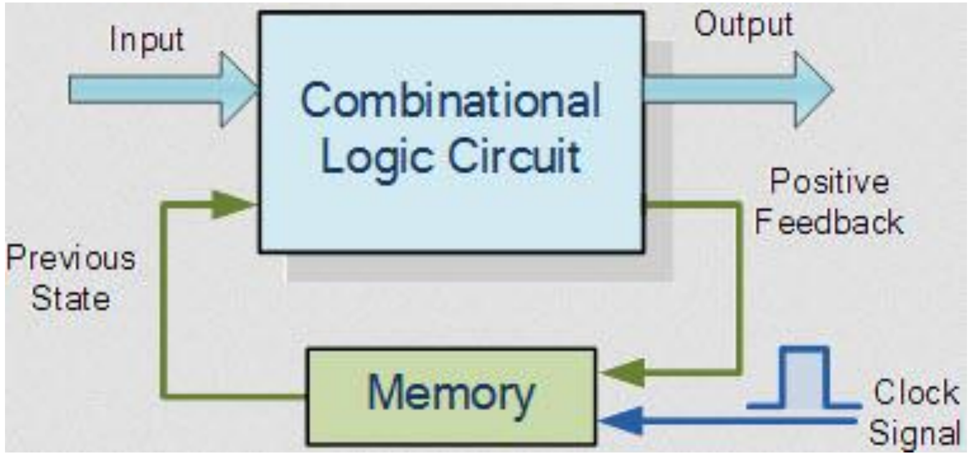
Introduction to FPGAs Lecture 3: Sequential Logic, UART

Adam Duncan

Sequential logic 101: Clocks and Memory



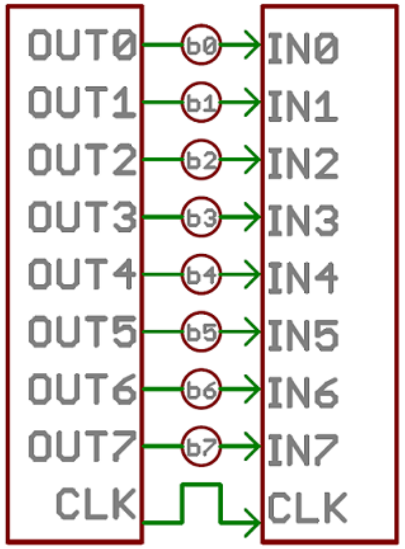
Combinational: output toggles after input (No memory)



Sequential: output toggles with input, previous state, and clock signal edge (Memory)

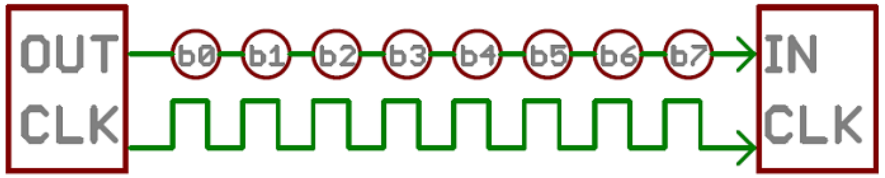
https://www.elprocus.com/tutorial-on-sequential-logic-circuits/

UART 101: Parallel versus Serial



An 8-bit data bus, controlled by a clock, transmitting a byte every clock pulse. 9 wires are used.

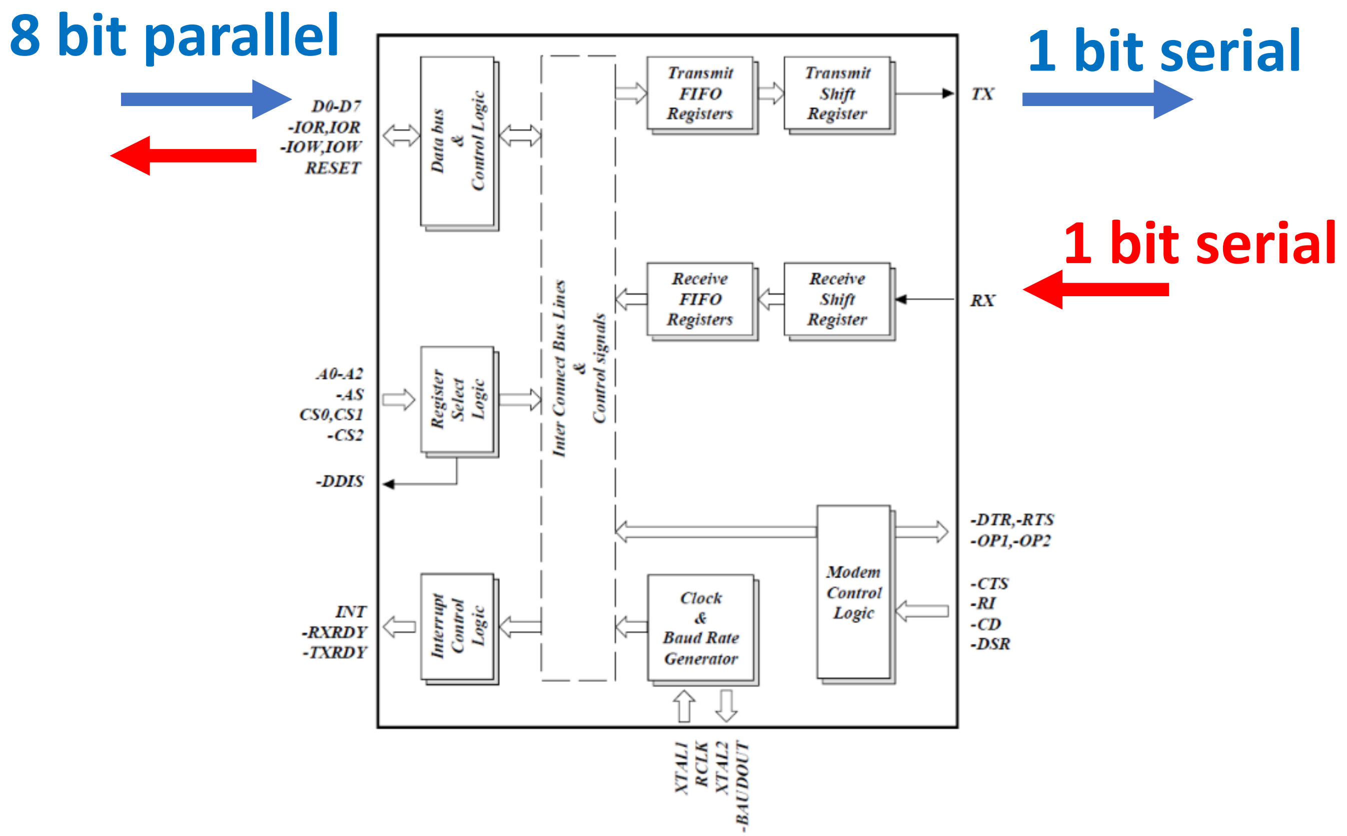
Serial interfaces stream their data, one single bit at a time. These interfaces can operate on as little as one wire, usually never more than four.



Example of a serial interface, transmitting one bit every clock pulse. Just 2 wires required!

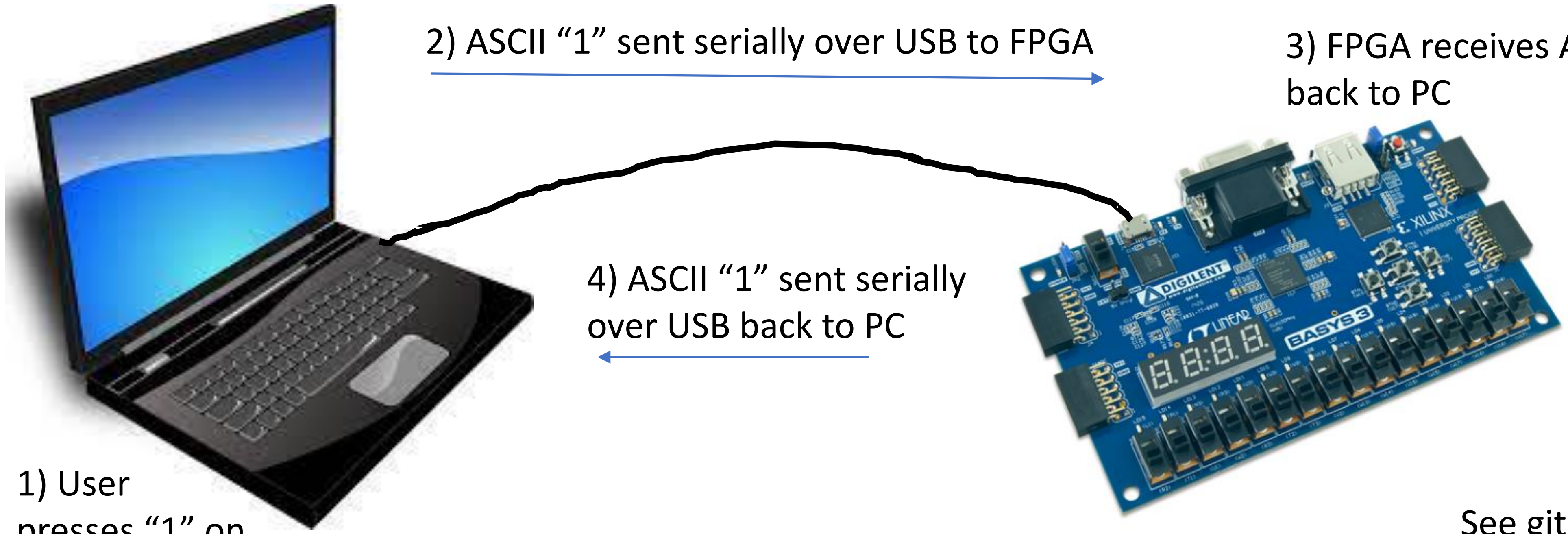
https://learn.sparkfun.com/tutorials/serial-communication/all

UART 101: Hardware blocks to convert between serial and parallel data



https://learn.sparkfun.com/tutorials/serial-communication/all

Project 5: Design/(simulate)/Program FPGA to function as UART to communicate with PC



1) User presses “1” on PC keyboard

2) ASCII “1” sent serially over USB to FPGA

3) FPGA receives ASCII “1”, and sends it back to PC

4) ASCII “1” sent serially over USB back to PC

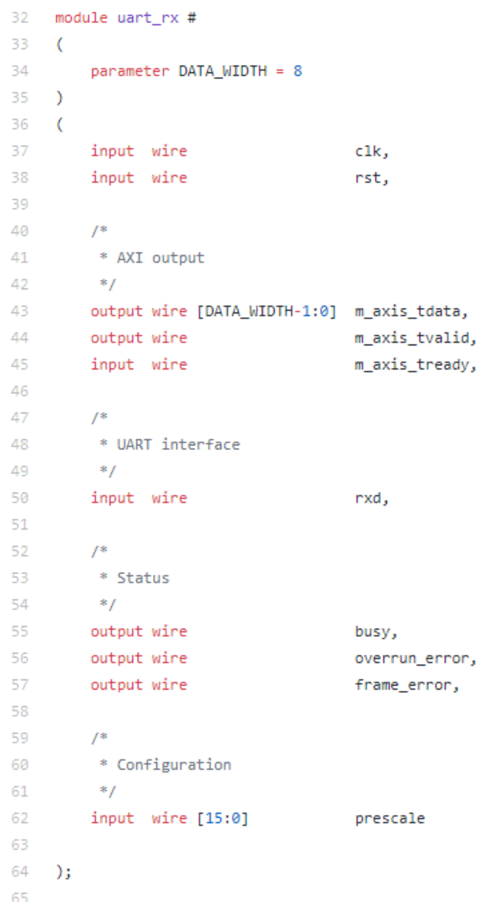
See github/adamdunc/whs for README

Project 5 Goals:

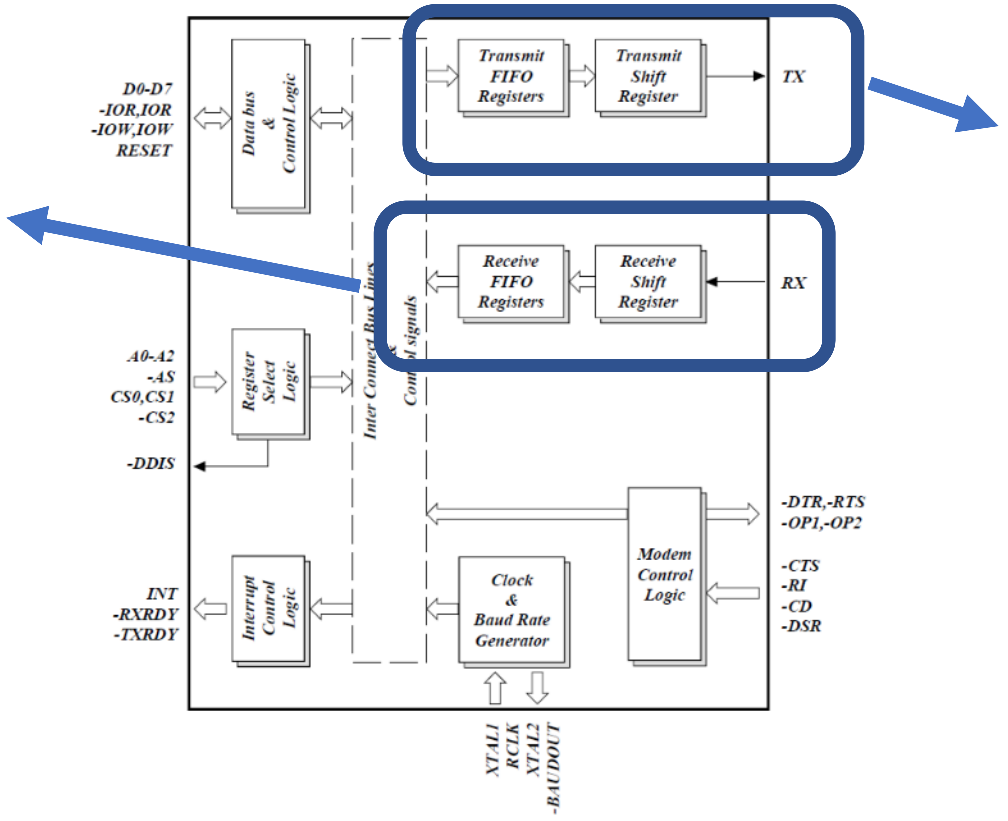
1) Communicate with FPGA from PC
2) Experience sequential logic and UARTs

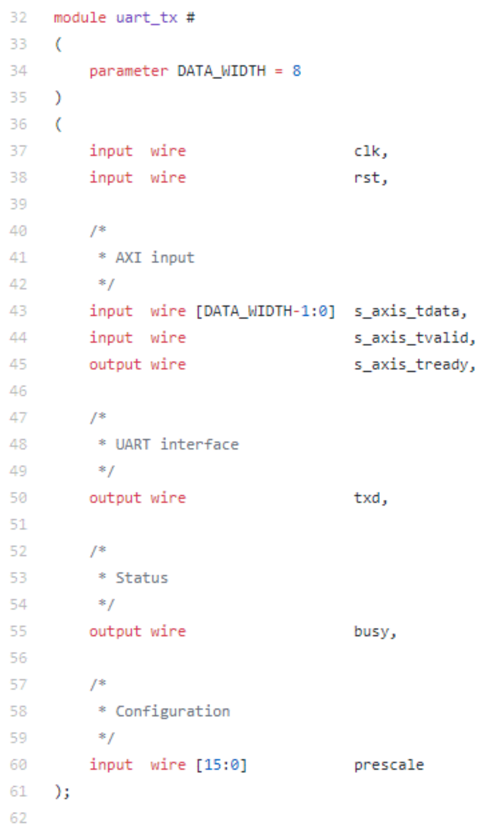
https://github.com/alexforencich/verilog-uart/

Project 5 Step 1: Instantiate “uart_rx” and “uart_tx” modules



```
module uart_rx #
(
    parameter DATA_WIDTH = 8
)
(
    input  wire                   clk,
    input  wire                   rst,

    /*
     * AXI output
     */
    output wire [DATA_WIDTH-1:0]  m_axis_tdata,
    output wire                   m_axis_tvalid,
    input  wire                   m_axis_tready,

    /*
     * UART interface
     */
    input  wire                   rxd,

    /*
     * Status
     */
    output wire                   busy,
    output wire                   overrun_error,
    output wire                   frame_error,

    /*
     * Configuration
     */
    input  wire [15:0]            prescale

);
```



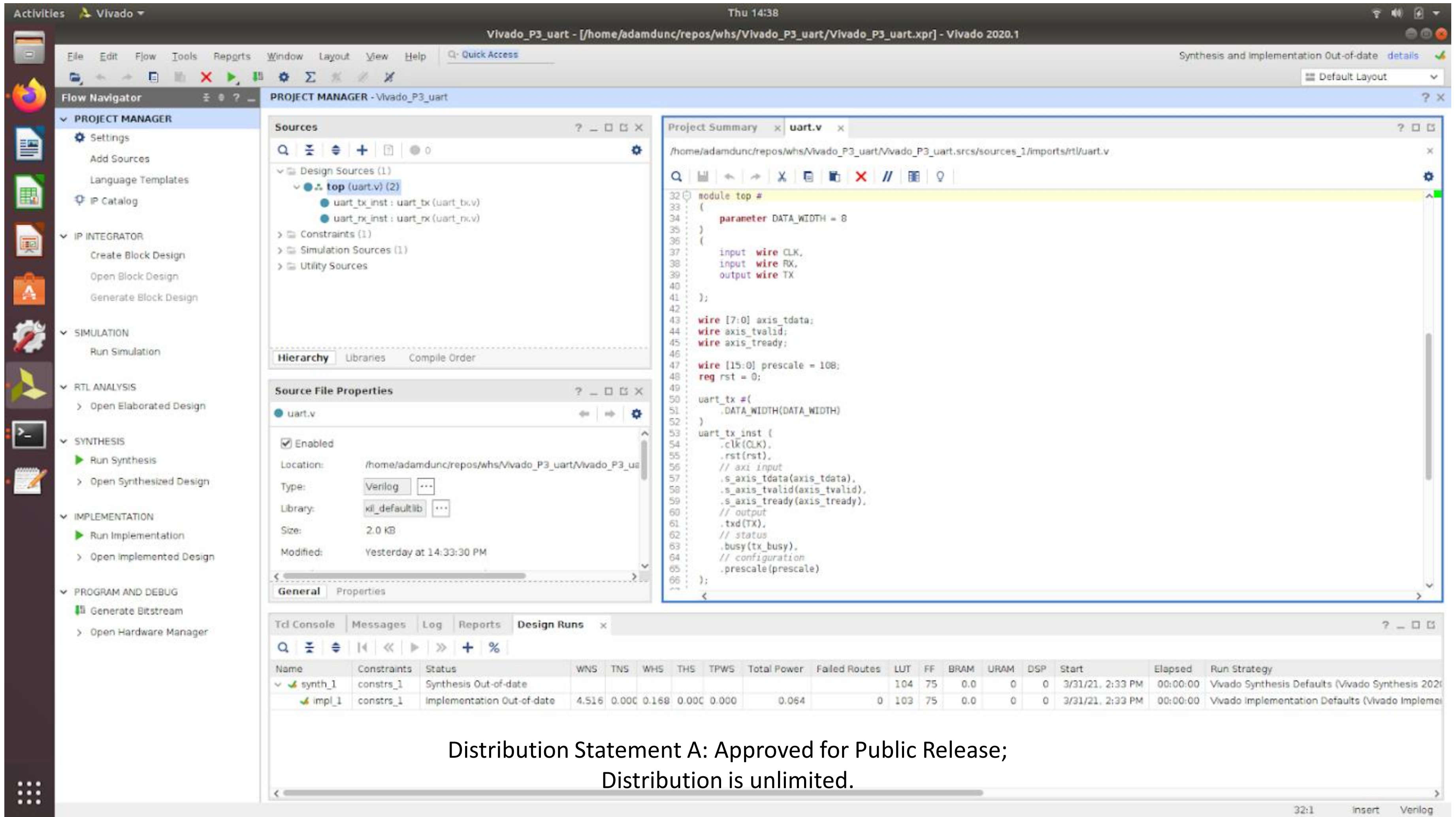


```
module uart_tx #
(
    parameter DATA_WIDTH = 8
)
(
    input  wire                   clk,
    input  wire                   rst,

    /*
     * AXI input
     */
    input  wire [DATA_WIDTH-1:0]  s_axis_tdata,
    input  wire                   s_axis_tvalid,
    output wire                   s_axis_tready,

    /*
     * UART interface
     */
    output wire                   txd,

    /*
     * Status
     */
    output wire                   busy,

    /*
     * Configuration
     */
    input  wire [15:0]            prescale
);
```

https://github.com/alexforencich/verilog-uart/

See github/adamdunc/whs for README

Project 5 Step2: Connect UART RX and UART TX modules together with axis_tdata, axis_tvalid, axis_tready



Vivado_P3_uart - [/home/adamdunc/repos/whs/Vivado_P3_uart/Vivado_P3_uart.xpr] - Vivado 2020.1

Sources

- Design Sources (1)
  - top (uart.v) (2)
    - uart_tx_inst : uart_tx (uart_tx.v)
    - uart_rx_inst : uart_rx (uart_rx.v)
- Constraints (1)
- Simulation Sources (1)
- Utility Sources

Source File Properties: uart.v

Enabled	✓
Location:	/home/adamdunc/repos/whs/Vivado_P3_uart/Vivado_P3_ua
Type:	Verilog
Library:	xil_defaultlib
Size:	2.0 KB
Modified:	Yesterday at 14:33:30 PM

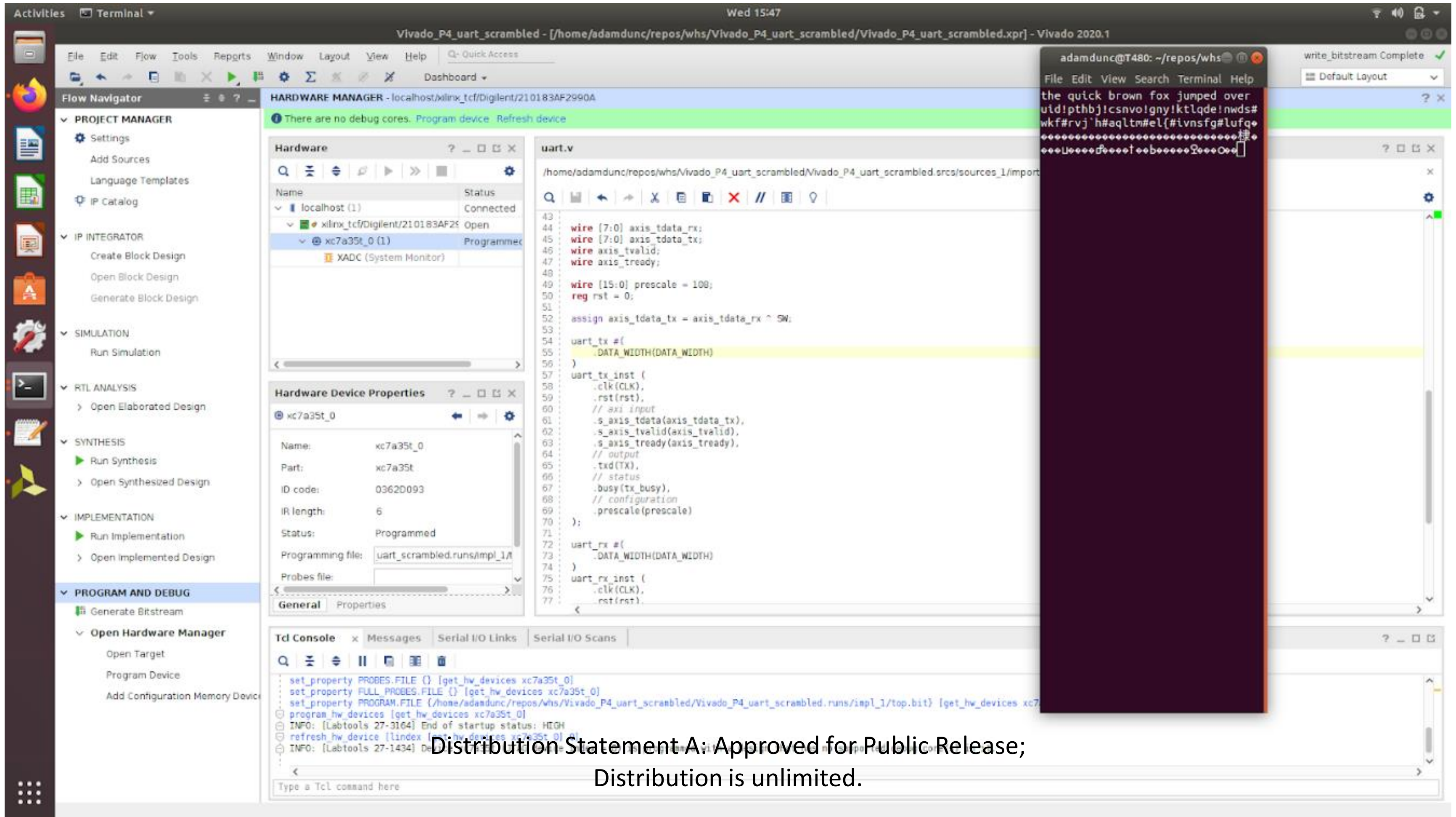
/home/adamdunc/repos/whs/Vivado_P3_uart/Vivado_P3_uart.srcs/sources_1/imports/rtl/uart.v

```verilog
module top #
(
    parameter DATA_WIDTH = 8
)
(
    input  wire CLK,
    input  wire RX,
    output wire TX

);

wire [7:0] axis_tdata;
wire axis_tvalid;
wire axis_tready;

wire [15:0] prescale = 108;
reg rst = 0;

uart_tx #(
    .DATA_WIDTH(DATA_WIDTH)
)
uart_tx_inst (
    .clk(CLK),
    .rst(rst),
    // axi input
    .s_axis_tdata(axis_tdata),
    .s_axis_tvalid(axis_tvalid),
    .s_axis_tready(axis_tready),
    // output
    .txd(TX),
    // status
    .busy(tx_busy),
    // configuration
    .prescale(prescale)
);
```

Design Runs

Name	Constraints	Status	WNS	TNS	WHS	THS	TPWS	Total Power	Failed Routes	LUT	FF	BRAM	URAM	DSP	Start	Elapsed	Run Strategy
synth_1	constrs_1	Synthesis Out-of-date								104	75	0.0	0	0	3/31/21, 2:33 PM	00:00:00	Vivado Synthesis Defaults (Vivado Synthesis 202
impl_1	constrs_1	Implementation Out-of-date	4.516	0.000	0.168	0.000	0.000	0.064	0	103	75	0.0	0	0	3/31/21, 2:33 PM	00:00:00	Vivado Implementation Defaults (Vivado Impleme

Project 5 step 3: Add logic inside FPGA to “modify” the ASCII values inside the FPGA



```
wire [7:0] axis_tdata_rx;
wire [7:0] axis_tdata_tx;
wire axis_tvalid;
wire axis_tready;

wire [15:0] prescale = 108;
reg rst = 0;

assign axis_tdata_tx = axis_tdata_rx ^ SW;

uart_tx #(
    .DATA_WIDTH(DATA_WIDTH)
)
uart_tx_inst (
    .clk(CLK),
    .rst(rst),
    // axi input
    .s_axis_tdata(axis_tdata_tx),
    .s_axis_tvalid(axis_tvalid),
    .s_axis_tready(axis_tready),
    // output
    .txd(TX),
    // status
    .busy(tx_busy),
    // configuration
    .prescale(prescale)
);

uart_rx #(
    .DATA_WIDTH(DATA_WIDTH)
)
uart_rx_inst (
    .clk(CLK),
    .rst(rst),
```

```
the quick brown fox jumped over
uid!pthbj!csnvo!gny!ktlqde!nwds#
wkf#rvj`h#aqltm#el{#ivnsfg#lufq◆
◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
```

```
set_property PROBES.FILE {} [get_hw_devices xc7a35t_0]
set_property FULL_PROBES.FILE {} [get_hw_devices xc7a35t_0]
set_property PROGRAM.FILE {/home/adamdunc/repos/whs/Vivado_P4_uart_scrambled/Vivado_P4_uart_scrambled.runs/impl_1/top.bit} [get_hw_devices xc7
program_hw_devices [get_hw_devices xc7a35t_0]
INFO: [Labtools 27-3164] End of startup status: HIGH
refresh_hw_device [lindex [get_hw_devices xc7a35t_0] 0]
INFO: [Labtools 27-1434]
```

Distribution Statement A: Approved for Public Release;
Distribution is unlimited.

Project 6: Generate Basic “Pong” Game

- Open Project 6 Vivado Pong project
  - Add Verilog to drive VGA output from FPGA
  - (See README for specific files and lines to add)
  - Generate bitstream
  - Program bitstream on FPGA
- Connect VGA output from FPGA board to monitor
- Use “left” and “right” keys to control paddle
- We will use this as the base for our final project!

See github/adamdunc/whs for README

Project 6 Goals:

1) Build simple Pong game on FPGA
2) Customize Pong with your group!

https://www.fpga4fun.com/PongGame.html