

REPORT JUNE 02

Date:	02 JUNE 2020	Name:	Rakshith B
Course:	Digital Design Using HDL	USN:	4AL16EC409
Topic:	Architecture, Features and Uses of FPGA	Semester & Section:	6th SEM B
Github Repository:	Rakshith-B		

FORENOON SESSION DETAILS

Image of session

The screenshot shows a YouTube video player with the title "WRITING VERILOG TEST BENCHES" and 8,495 views. The video content displays Verilog code for a testbench and a table of simulation results.

```

module testbench;
  reg a, b, c; wire sum, cout;
  integer i;
  full_adder FA (sum, cout, a, b, c);

  initial
  begin
    for (i=0; i<8; i=i+1)
      begin
        {a,b,c} = i; #5;
        $display ("T=%2d, a=%b, b=%b, c=%b, sum=%b, cout=%b",
                  $time, a, b, c, sum, cout);
      end
    #5 $finish;
  end
endmodule
  
```

T= 5,	a=0,	b=0,	c=0,	sum=0,	cout=0
T=10,	a=0,	b=0,	c=1,	sum=1,	cout=0
T=15,	a=0,	b=1,	c=0,	sum=1,	cout=0
T=20,	a=0,	b=1,	c=1,	sum=0,	cout=1
T=25,	a=1,	b=0,	c=0,	sum=1,	cout=0
T=30,	a=1,	b=0,	c=1,	sum=1,	cout=1
T=35,	a=1,	b=1,	c=0,	sum=0,	cout=1
T=40,	a=1,	b=1,	c=1,	sum=1,	cout=1

purpose well how have we done this it is quite simple we have declared an integer of type

Report –

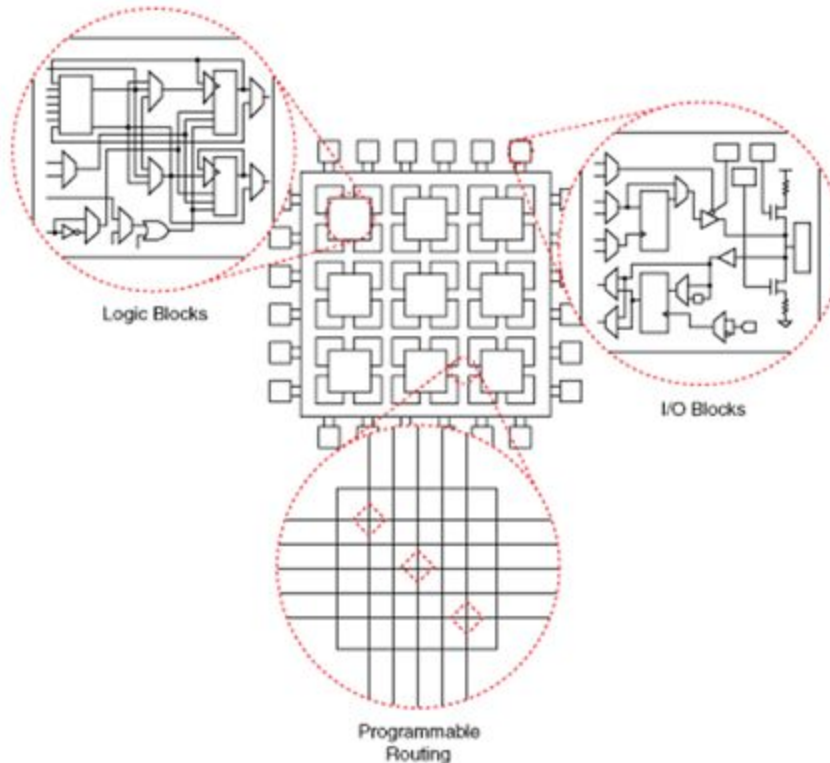
What is FPGA?

The field-programmable gate array (FPGA) is an integrated circuit that consists of internal hardware blocks with user-programmable interconnects to customize operation for a specific application. The interconnects can readily be reprogrammed, allowing an FPGA to accommodate changes to a design or even support a new application during the lifetime of the part.

The FPGA has its roots in earlier devices such as programmable read-only memories (PROMs) and programmable logic devices (PLDs). These devices could be programmed either at the factory or in the field, but they used fuse technology (hence, the expression “burning a PROM”) and could not be changed once programmed. In contrast, FPGA stores its configuration information in a

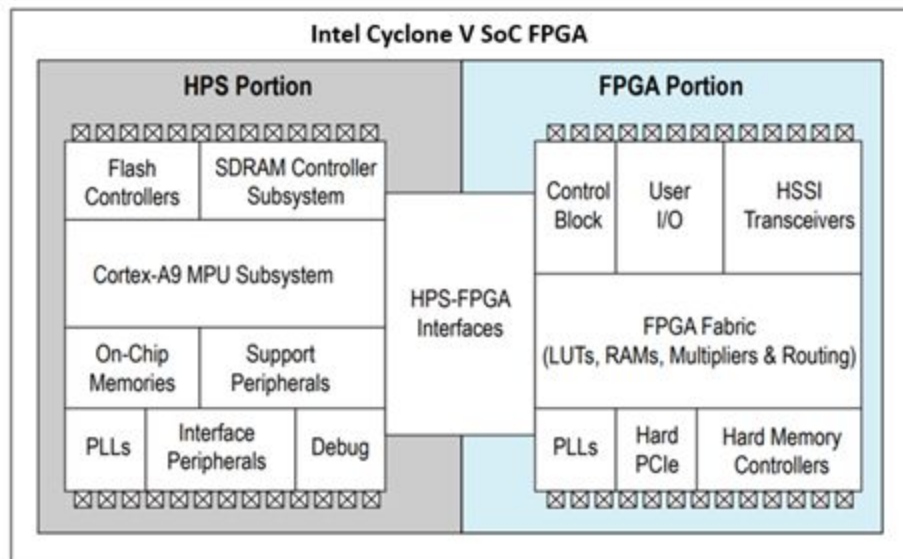
re-programmable medium such as static RAM (SRAM) or flash memory. FPGA manufacturers include Intel, Xilinx, Lattice Semiconductor, Microchip Technology and Microsemi.

FPGA Architecture



SoC FPGAs

SoC FPGAs include a wide range of processing capabilities to suit different applications. A low-cost, low-power SoC FPGA such as Intel's Cyclone V, for example, targets high-volume applications such as industrial motor control drives, protocol bridging, video processing cards and handheld devices. The device (Figure 3) has two distinct parts: the FPGA portion and a hard processor system (HPS) based around a single- or dual-core 32-bit Arm Cortex-A9 MPCORE running at 925 MHz. Each part contains its own set of peripherals, which includes hard IP from third-party vendors.



At the other end of the scale, the Stratix 10 SX targets high-performance applications in communications, data center acceleration, high-performance computing (HPC), radar processing and ASIC prototyping; that FPGA includes a quad-core 64-bit Arm Cortex-A53 running at up to 1.5 GHz.

FPGA Design

How do we transform this collection of thousands of hardware blocks into the correct configuration to execute the application? An FPGA-based design begins by defining the required computing tasks in the development tool, then compiling them into a configuration file that contains information on how to hook up the CLBs and other modules. The process is similar to a software development cycle except that the goal is to architect the hardware itself rather than a set of instructions to run on a predefined hardware platform. Designers have traditionally used a hardware description language (HDL) such as VHDL (Figure 4) or Verilog to design the FPGA configuration.

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.numeric_std.all;
4
5 entity signed_adder is
6   port
7   (
8     aclr : in    std_logic;
9     clk  : in    std_logic;
10    a     : in    std_logic_vector;
11    b     : in    std_logic_vector;
12    q     : out   std_logic_vector
13  );
14 end signed_adder;
15
16 architecture signed_adder_arch of signed_adder is
17   signal q_s : signed(a'high+1 downto 0); -- extra bit wide
18
19 begin -- architecture
20   assert(a'length >= b'length)
21     report "Port A must be the longer vector if different sizes!"
22     severity FAILURE;
23   q <= std_logic_vector(q_s);
24
25   adding_proc:
26   process (aclr, clk)
27   begin
28     if (aclr = '1') then
29       q_s <= (others => '0');
30     elsif rising_edge(clk) then
31       q_s <= ('0'&signed(a)) + ('0'&signed(b));
32     end if; -- clk'd
33   end process;
34
35 end signed_adder_arch;

```

Once the FPGA design has been created and verified using HDL, the compiler takes the text-based file and generates a configuration file that contains information on how the components should be wired together. Even if the HDL code has no errors, choosing the wrong FPGA may still cause the compilation to fail—for example, the FPGA runs out of a specific resource type or the compiler cannot create the required routes between components. One challenge with an HDL approach is that configuring an FPGA requires both coding skills and a detailed knowledge of the underlying hardware, and the required expertise is not widely available. As a result, vendors are offering software development kits (SDKs) that allow designers to develop FPGA solutions in popular high-level languages such as C/C++, Python and OpenCL. High-level synthesis (HLS) design tools are also available; these run on a framework such as National Instrument's LabVIEW and feature graphical block diagrams instead of lines of code.

FPGA Uses:

The ability to configure the hardware of the FPGA, reconfigure it when needed and optimize it for a particular set of functions makes the FPGA an attractive option in many applications.

FPGAs are often used to provide a custom solution in situations in which developing an ASIC would be too expensive or time-consuming. An FPGA application can be configured in hours or days instead of months. Of course, the flexibility of the FPGA comes at a price: An FPGA is likely to be slower, require more PCB area and consume more power than an equivalent ASIC.

Even when an ASIC will be designed for high-volume production, FPGAs are widely used for system validation, including pre-silicon validation, post-silicon validation and firmware development. This allows manufacturers to validate their design before the chip is produced in the factory.

FPGA Applications

Many applications rely on the parallel execution of identical operations; the ability to configure the FPGA's CLBs into hundreds or thousands of identical processing blocks has applications in image processing, artificial intelligence (AI), data center hardware accelerators, enterprise networking and automotive advanced driver assistance systems (ADAS).

Many of these application areas are changing very quickly as requirements evolve and new protocols and standards are adopted. FPGAs enable manufacturers to implement systems that can be updated when necessary.

A good example of FPGA use is high-speed search: Microsoft is using FPGAs in its data centers to run Bing search algorithms. The FPGA can change to support new algorithms as they are created. If needs change, the design can be repurposed to run simulation or modeling routines in an HPC application. This flexibility is difficult or impossible to achieve with an ASIC.

Other FPGA uses include aerospace and defense, medical electronics, digital television, consumer electronics, industrial motor control, scientific instruments, cybersecurity systems and wireless communications.

Implement a 4:1 MUX and write the test bench code to verify the module

Using Behavioral Modeling:

```
module m41(out, a, b, c, d, s0, s1);  
output out;  
input a, b, c, d, s0, s1;  
wire sobar, s1bar, T1, T2, T3, T4;  
not (s0bar, s0), (s1bar, s1);  
and (T1, a, s0bar, s1bar), (T2, b, s0bar, s1), (T3, c, s0, s1bar), (T4, d, s0, s1);  
or(out, T1, T2, T3, T4);  
endmodule
```

Test Bench Code

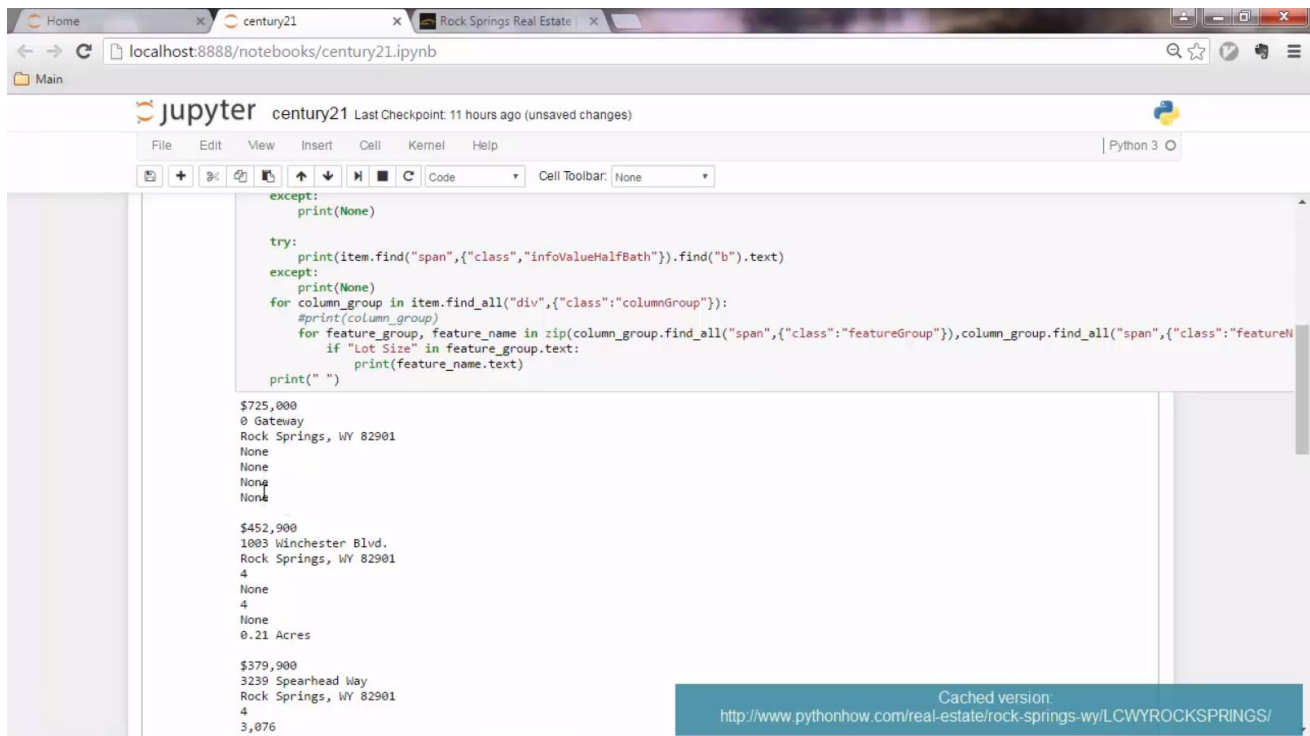
```
module muxt_b;  
reg [3:0] a;  
reg [1:0] s;  
wire o;  
mux4bit uut (.a(a), .s(s),.o(o));  
initial begin  
#10 a=4'b1010;  
#10 s=2'b00;  
#10 s=2'b01;  
#10 s=2'b10;  
#10 s=2'b11;  
#10 $stop;  
end  
endmodule
```

Date: 02 JUNE 2020
Course: Python On Udemy
Topic: Scrape Real Estate
Property Data from the Web

Name: RAKSHITH B
USN: 4AL16EC409
Semester & Section: 6 B

AFTERNOON SESSION DETAILS

Image of session: Output



```
except:
    print(None)

try:
    print(item.find("span", {"class": "infoValueHalfBath"}).find("b").text)
except:
    print(None)
for column_group in item.find_all("div", {"class": "columnGroup"}):
    print(column_group)
    for feature_group, feature_name in zip(column_group.find_all("span", {"class": "featureGroup"}), column_group.find_all("span", {"class": "featureName"})):
        if "Lot Size" in feature_group.text:
            print(feature_name.text)
    print(" ")

$725,000
0 Gateway
Rock Springs, WY 82901
None
None
None
None

$452,900
1003 Winchester Blvd.
Rock Springs, WY 82901
4
None
4
None
0.21 Acres

$379,900
3239 Spearhead Way
Rock Springs, WY 82901
4
3,076
```

Cached version:
<http://www.pythonhow.com/real-estate/rock-springs-wy/LCWYROCKSPRINGS/>

```

#Get the first page to extract page numbers
import requests, re
from bs4 import BeautifulSoup

r=requests.get("http://www.pythonhow.com/real-estate/rock-springs-wy/LCWYROCKSPRINGS/")
c=r.content

soup=BeautifulSoup(c,"html.parser")

all=soup.find_all("div",{"class":"propertyRow"})

all[0].find("h4",{"class":"propPrice"}).text.replace("\n","").replace(" ","")

page_nr=soup.find_all("a",{"class":"Page"})[-1].text
print(page_nr,"number of pages were found")

l=[]
base_url="http://www.pythonhow.com/real-estate/rock-springs-wy/LCWYROCKSPRINGS/t=0&s="
for page in range(0,int(page_nr)*10,10):
    print( )
    r=requests.get(base_url+str(page)+".html")
    c=r.content
    #c=r.json()["list"]
    soup=BeautifulSoup(c,"html.parser")
    all=soup.find_all("div",{"class":"propertyRow"})
    for item in all:
        d={}

d["Address"]=item.find_all("span",{"class","propAddressCollapse"})[0].text
        try:

d["Locality"]=item.find_all("span",{"class","propAddressCollapse"})[1].text
        except:
            d["Locality"]=None

d["Price"]=item.find("h4",{"class","propPrice"}).text.replace("\n","").replace(" ","")
        try:

d["Beds"]=item.find("span",{"class","infoBed"}).find("b").text
        except:
            d["Beds"]=None

```



```

        try:
d["Area"]=item.find("span",{ "class","infoSqFt"}).find("b").text
        except:
            d["Area"]=None

        try:
            d["Full
Baths"]=item.find("span",{ "class","infoValueFullBath"}).find("b").t
ext
        except:
            d["Full Baths"]=None

        try:
            d["Half
Baths"]=item.find("span",{ "class","infoValueHalfBath"}).find("b").t
ext
        except:
            d["Half Baths"]=None
        for column_group in
item.find_all("div",{ "class":"columnGroup"}):
            for feature_group, feature_name in
zip(column_group.find_all("span",{ "class":"featureGroup"}),column_g
roup.find_all("span",{ "class":"featureName"})):
                if "Lot Size" in feature_group.text:
                    d["Lot Size"]=feature_name.text

        l.append(d)
l
import pandas
df=pandas.DataFrame(l)
df
df.to_csv("Output.csv")

```