DAILY ASSESSMENT FORMAT

Date:	**1st June 2020**	**Name:**	**Sushmitha R Naik**
Course:	**Digital design using HDL**	**USN:**	**4AL16EC090**
Topic:	✓ **Industry Applications of FPGA.** ✓ **FPGA Business Fundamentals.** ✓ **FPGA vs ASIC Design Flow.**	**Semester & Section:**	**6th sem 'B' sec**
GitHub Repository:	**Sushmitha_naik**		

FORENOON SESSION DETAIL
Image of session

FPGA?

An FPGA is a (mostly) digital, (re-)configurable ASIC. I say mostly because there are analog and mixed-signal aspects to modern FPGAs. For example, some have A/D converters and PLLs. I put *re-* in parenthesis because there are actually one-time-programmable FPGAs, where once you configure them, that's it, never again. However, most FPGAs you'll come across are going to be re-configurable.

The advantage is that you're not tying up a centralized processor. Each function can operate on its own. Large quantities of deterministic I/O – the amount of determinism that you can achieve with an FPGA will usually far surpass that of a typical sequential processor. If there are too many operations within your required loop rate on a sequential processor, you may not even have enough time to close the loop to update all of the I/O within the allotted time. Signal processing – includes algorithms such as digital filtering, demodulation, detection algorithms, frequency domain processing, image processing, or control algorithms.

Core components:

LUT (Look-Up Table)

The name LUT in the context of FPGAs is actually misleading, as it doesn't convey the full power of this logical resource. There are however two other common uses for a LUT:

LUT as a shift register – shift registers are very useful for things like delaying the timing of an operation to align the outputs of one algorithm with another. Size varies based on FPGA.
LUT as a small memory – you can configure the LUT logic as a VERY small volatile random-access memory block. Size varies based on FPGA

FF (Flip-flop)

Flip-flops store the output of a combinational logic calculation. This is a critical element in FPGA design because you can only allow so much asynchronous logic and routing to occur before it is registered by a synchronous resource (the flip-flop), otherwise the FPGA won't make timing. It's the core of how an FPGA works. Flip-flops can be used to register data every clock cycle, latch data, gate off data, or enable signals.

Block Memory

It's important to note that there are generally several types of memory in an FPGA. We mentioned the configuration of a LUT resource. Another is essentially program memory, which is intended to store the compiled version of the FPGA program itself (this may be part of the FPGA chip or as a separate non-volatile memory chip). What we're referring to here though, is

neither of those types of memory. Here we're referring to dedicated blocks of volatile user memory within the FPGA. This memory block is generally on the order of thousands of bits of memory, is configurable in width and depth, and multiple blocks of memory can be chained together to create larger memory elements. They can generally be configured as either single-port or dual-port random access, or as a FIFO. There will generally be many block memory elements within an FPGA.

Multipliers or DSP blocks

Have you ever seen the number of digital logic resources that it takes to create a 16-bit by 16-bit multiplier? It's pretty crazy, and would chew through your logical and routing resources pretty quickly. Check out the 2-bit by 2-bit example here: https://en.wikipedia.org/wiki/Binary_multiplier. FPGA vendors solve this problem with dedicated silicon to lay down something on the order of 18-bit multiplier blocks.

I/O (Input/Output)

If you're going to do something useful with an FPGA, you generally have to get data from and/or provide data outside the FPGA. To facilitate this, FPGAs will include I/O blocks that allow for various voltage standards (e.g. LVCMOS, LVDS) as well as timing delay elements to help align multiple signals with one another (e.g. for a parallel bus to an external RAM chip).

Clocking and routing

This is really a more advanced topic, but critical enough to at least introduce. We use an external oscillator and feed it into clocking resources that can multiply, divide, and provide phase-shifted versions of your clock to various parts of the FPGA.Routing resources not only route your clock to various parts of the FPGA, but also your data. Routing resources within an FPGA are one of the most underappreciated elements, but so critical.

TASK1:

Write a Verilog code to implement NAND gate in all different styles.

Gate Level modeling:

```
module NAND_2_gate_level (output Y, input A, B);
    wire Yd;
    and (Yd, A, B);
    not (Y, Yd);
endmodule
```

Data flow modelling:

```
module NAND_2_data_flow (output Y,input A,B);
   assign Y=~(A&B);
endmodule
```

Behavioural Modeling:

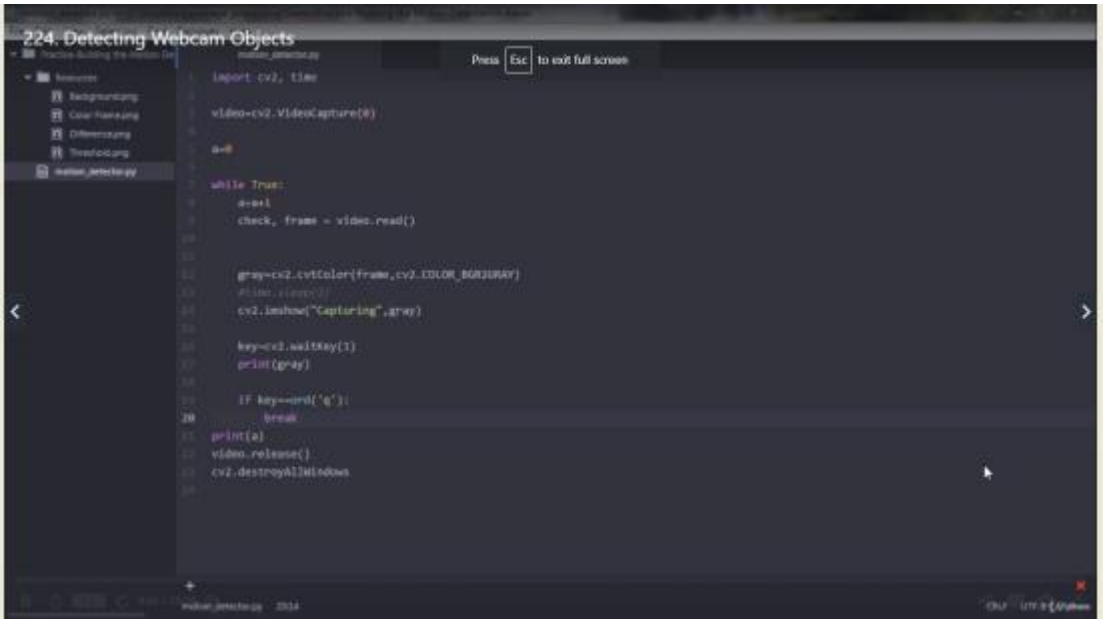
```
module NAND_2_behavioral (output reg Y,input A,B);
always@ (A or B) begin
     if (A==1'b1 & B==1'b1) begin
         Y=1'b0;
      end
      else
          Y=1'b1;
end
endmodule
```

DAILY ASSESSMENT FORMAT

Date:	**1st June 2020**	**Name:**	**Sushmitha R Naik**
Course:	**Python**	**USN:**	**4AL16EC090**
Topic:	**Build a webcam motion detector**	**Semester & Section:**	**6th sem 'B' sec**
GitHub Repository:	**Sushmitha_naik**		

AFTERNOON SESSION DETAILS

Image of session



Motion detection is the detection of the change in the position of an object with respect to its surroundings and vice-versa. Buckle up your seat belts to drive through this motion detector application along with me and your lovable Python.we may be able to perform the following tasks using this application, though the list is non-exhaustive:

1) Find in front of screen time during working from home.

2)Monitor your child's in front of screen time.

3) Find trespassing in your backyard.

4)Locate unwanted public/animal movements around yoroom/house/alley and what not…….Photo by Williapm Thomas on Unsplash Hardware Requirements: A computer with a webcam or any type of camera installed.

- **Software Requirements: Python 3 or above.Additional Requirements: 30 mins of your time,.**
- **Firstly, you will capture the first frame via webcam. This frame will be treated as the baseline frame. Motion will be detected by calculating the phase difference between this baseline frame and the new frame with some object. The new frames will be called Delta frame.**

- **Then you will refine your delta frame using pixel intensity. The refined frame will be called the Threshold frame.**
- **Then you will apply some intricate image processing techniques like Shadow Removal, Dilation, Contouring, etc. on the Threshold frame to capture substantial objects.**
- **Detected Object You will be able to capture the time stamp when an object entered the frame and exited the frame.**
- **Thus, you will be able to find the screen-on time.I won't embed my code here as I would like you to improve the blood circulation on your fingertips.To start with basic installations, please install python 3 or above, pandas, and opencv via pip. Once done, you are ready to begin:**

STEP 1: Import required libraries:

STEP 2: Initialize variables, lists, data frames:You will get to know when each one of the above will be required in the below code.

STEP 3: Capture the video frames using webcam:OpenCV has in-built functions to open the camera and capture video frames. "0" denotes the camera at the hardware port number 0 in your computer. If you have multiple cameras or external cameras or a CCTV setup installed, you may provide the port number accordingly.

STEP 4: Converting the captured frame to gray-scale and applying Gaussian Blur to remove noise:We convert the color frame to gray frame as an extra layer of color is not required. GaussianBlur is used for image smoothing and it will, in turn, enhance the detection accuracy. In the GaussianBlur function, for the 2nd parameter, we define the width and height of the Gaussian Kernel and for the 3rd parameter, we provide standard deviation value. These are set of higher order differential calculus theorems, so you may use standard values of the kernel size as (21,21) and standard-deviation as 0.

GUVI
tech deserves you

Sushmitha R naik

is here by awarded the certificate of achievement for
the successful completion of

Step into Robotic Process Automation

during GUVI's RPA SKILL-A-THON 2020

S.P.Balamurugan
Co-founder, CEO

Valid certificate ID 19gV109fj10uk25124

Verified certificate issue on June 1 2020

Verify certificate at www.guvi.in/certificate?id=19gV109fj10uk25124

In association with
UiPath Academic Alliance