Objective: The objective of this lab is to learn the fundamental components of the Xilinx FPGA tools required to enter and assemble HDL code using a schematic entry tool and program an FPGA (Artix-7).

Intro: In this lab you will program an FPGA to display a rectangle on a monitor using a VGA cable. The color of the rectangle will depend on the configuration of the twelve switches located on the FPGA board. You will then modify the code provided for this tutorial to provide slightly different functionality, such as displacing the rectangle.

NOTE: Before starting this homework, you must have finished Tutorial 2 (and Tutorial 1 if you are using your own ISE 14.7 software) to make sure you can create a project and program FPGA.
Equipment Needed:

- Working computer with Xilinx ISE 14.7 installed
- NEXYS4 DDR (Artix-7) FPGA board + USB Programming Cable
- Second monitor
- VGA cable

Tutorial:

1. Start Xilinx Tool
2. In the ISE Project Navigator Window Create a New Project: 
   File → New Project

3. Name your project, setup directories, and set top-level source type.
3.1. If any “_pn.exe” error appears, perform the following steps:

i) Open the following directory: C:\Xilinx\14.7\ISE_DS\ISE\lib\nt64

ii) Find and rename libPortability.dll to libPortability.dll.orig

iii) Make a copy of libPortabilityNOSH.dll (copy and paste it to the same directory) and rename it libPortability.dll

iv) Copy libPortabilityNOSH.dll again, but this time navigate to C:\Xilinx\14.7\ISE_DS\common\lib\nt64 and paste it there

v) In C:\Xilinx\14.7\ISE_DS\common\lib\nt64 Find and rename libPortability.dll to libPortability.dll.orig

vi) Rename libPortabilityNOSH.dll to libPortability.dll

3.2. If any Licensing error appears, go to Help→License Manager and make sure the following settings are configured:
4. Hit next and configure the Project Settings:

5. Click Next/Finish until the “New Project Wizard” window closes.

6. Click Project→New Source
7. Select “Verilog Module” then continue to hit Next/Finish until the “New Source Wizard” window closes.

8. Add an “Implementation Constraints File”, continue to select Next/Finish until the “New Source Wizard” window closes, and copy the code provided in the Appendix D into its contents. (Mention which file)
9. Create two additional “Verilog Module” files, shown in the figures below, and copy the code provided in the Appendix B and C into their respective files. (mention which file to add)
10. Copy provided code for the top Verilog module (vgaproject.v) from the appendix A (mention the file name). Now edit the code to instantiate three modules (vgaPulse Horizontal, vgaPulse Vertical and clockDiv pixel). Please see the comments in the code to understand where you have to instantiate these modules.

Example of instantiation:

Usually module instantiation is done as follows,

`Module_name Instance_name (Port_Association_List).
For example, you have declared a module as follows,

`module addbit (a,b,ci,sum,co);

where a,b, ci are the inputs and sum,co are the outputs. Now you can instantiate this module to any other module as follows,

```
addbit addbit1 ( 
  a(0) , 
  b(1) , 
  ci(0) , 
  sum(sum) , 
  co(co) 
);
```

Remember in the parentheses we provide the values or variable name from where values are coming for the input or output variables. Now save the project. There should be three “.v” files and one “.ucf” file listed below the top module in the Hierarchy listing. Also at the end of the vgaProject module you will see a always block. In this always block you will see some values of x and y. According to the values of x and y the rectangle will be placed in the monitor. Think yourself how you can move the square in any place of monitor by changing the values of x and y. You can do this thing after completing the whole project also. The project hierarchy should be look like as below. Your project
11. Synthesize the program by first selecting the top Verilog Module file, expand “Configure Target Device” inside the Processes section (lower-left corner), right click “Manage Configuration Project”, then click “Run”.

12. Make sure that there are minimal warnings and no errors shown after Synthesis is complete.
13. Ensure that your device is connected to your computer.

14. An ISE iMPACT window should appear. Double click on Boundary Scan and right click in the center of the white space, then select Initialize Chain.
15. Select Yes in response to assigning configuration files, then select the “.bit” file.

16. Then select No for the next pop-up request, and Cancel for the final pop-up.
17. Right click on the square that is now shown in the white space and select “Program”. You should momentarily see a “Program Succeeded” message appear at the bottom of the white space.

18. Make sure the VGA cable is properly connected to the FPGA and a different monitor than the one you are currently using.

19. With all the switches configured in the “up-ward” or “on” position, a white rectangle should appear at the bottom left of your second monitor:
Homework deliveries and report

1. Follow all the instructions and make sure the project is fully working on the FPGA board.

2. Show at least 6 different colors with different configuration of switches

3. Modify the top level vgaProject.v code, so the location of the square on monitor changes (ex. top right or top left).

4. Write a complete report to explain your design and how you found the numbers to move the square. Also in your report always mention on the first page if your design is Fully working and if it’s not working upto which stage you could get it working.
module vgaProject(input clk, // System clk
    input [3:0]r, // input from switches
    input [3:0]g,
    input [3:0]b,
    output reg [3:0]red, // display on VGA
    output reg [3:0]green,
    output reg [3:0]blue,
    output VS, // Vertical sync obtained from
    output HS, // Horizontal sync obtained from
    output vFree,
    output hFree);

reg [3:0] free;
wire [10:0] x,y;
// Instantiate vgaPulse Horizontal with clk = pixelClk, stage1 = 21'd96, stage2 = 21'd144, stage3 = 21'd784, endStage = 21'd800, syncPulse = HS, free = hFree and position = x
// Instantiate vgaPulse Vertical with clk = HS, stage1 = 21'd2, stage2 = 21'd35, stage3 = 21'd515, endStage = 21'd525, syncPulse = VS, free = vFree, position = y
// Instantiate clockDiv pixel with div = 32'd4, out =pixelClk
always begin

free[0]=hFree&&vFree&&$(x<500)\&\&(x>450)\&\&(y<450)\&\&(y>400); // value of 'x' and 'hfree' obtained from vgaPulse
Horizontal
free[1]=hFree&&vFree&&$(x<500)\&\&(x>450)\&\&(y<450)\&\&(y>400); // value of 'y' and 'vfree' obtained from vgaPulse
Vertical
free[2]=hFree&&vFree&&$(x<500)\&\&(x>450)\&\&(y<450)\&\&(y>400);
free[3]=hFree&&vFree&&$(x<500)\&\&(x>450)\&\&(y<450)\&\&(y>400);

red=r&free; // if free show red on VGA
green=g&free; // if free show green on VGA
blue=b&free; // if free show blue on VGA

end

endmodule

APPENDIX B | vgaPulse.v

module vgaPulse(input clk,
input [21:0]stage1, // 96 for Horizontal & 2 for Vertical (Passed from top module)
input [21:0]stage2, // 144 for Horizontal & 35 for Vertical (Passed from top module)
input [21:0]stage3, // 784 for Horizontal & 515 for Vertical (Passed from top module)
input [21:0]endStage, // 800 for Horizontal & 525 for Vertical (Passed from top module)
output syncPulse, // Sync pulse HS or VS
output free, // singla indicating pixel can be passed to VGA port
output [10:0]position); // position of the pixel
// Temporary registers
reg [12:0] count;
reg inc;
reg[10:0] posCount;
reg S0,S1,S2,S3;
reg free_reg,sp;

assign position=posCount;
assign free=free_reg;
assign syncPulse=sp;

always begin
  inc=(count>endStage);
  S0<=(((count>21'd0)||(count==21'd0))&&(count<stage1)||(count==endStage));
  S1<=(((count>stage1)||(count==stage1))&&(count<stage2));
  S2<=(((count>stage2)||(count==stage2))&&(count<stage3));
  S3<=(((count>stage3)||(count==stage3))&&(count<endStage));
end

always@(negedge clk) begin
  free_reg<=S2;                          // free at S2 for displaying
  sp<=S1||S2||S3;                        // Syncpulse if either of S1,S2 or S3 are 1
end

always@(posedge clk) begin              // All always blocks run in parallel so this block will
  case(free)                            // check for updated status of free and inc at every posedge
    S0: break;
    S1: break;
    S2: break;
    S3: break;
    default: break;
  endcase
end
APPENDIX C | clockDiv.v

module clockDiv(input clk, input [31:0]div, output reg out);

reg [31:0] count;
reg inc;
reg max;

// All Always blocks run in parallel don't get confused
always begin  // This block will execute continuously and will update 'inc'
max=div>>1;
inc=(count>max);
end

always@(posedge clk) begin
```verilog
case(inc)
  1:begin
    count=0;
    out=~out;
  end
  0:begin
    count=count+1;
  end
endcase
endmodule
```

**APPENDIX D | introVGA.ucf**

```ucf
## This file is a general .ucf for the Nexys4 DDR Rev C board
## To use it in a project:
## - uncomment the lines corresponding to used pins
## - rename the used signals according to the project

## Clock signal
NET "clk" LOC = "E3" | IOSTANDARD = "LVCMOS33";
  #Bank = 35, Pin name = #IO_L12P_T1_MRCC_35,
    Sch name = clk100mhz
#NET "clk100mhz" TNM_NET = sys_clk_pin;
#TIMESPEC TS_sys_clk_pin = PERIOD sys_clk_pin 100 MHz HIGH 50%;

## Switches
NET "r<0>" LOC=J15 | IOSTANDARD=LVCMOS33; #IO_L24N_T3_RS0_15
NET "r<1>" LOC=L16 | IOSTANDARD=LVCMOS33; #IO_L3N_T0_DQS_EMCCLK_14
NET "r<2>" LOC=M13 | IOSTANDARD=LVCMOS33; #IO_L6N_T0_D08_VREF_14
NET "r<3>" LOC=R15 | IOSTANDARD=LVCMOS33; #IO_L13N_T2_MRCC_14
NET "g<0>" LOC=R17 | IOSTANDARD=LVCMOS33; #IO_L12N_T1_MRCC_14
```
<table>
<thead>
<tr>
<th>NET</th>
<th>LOC</th>
<th>IOSTANDARD</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;hfree&quot;</td>
<td>H17</td>
<td>LVCMOS33;</td>
<td>I/O_L18P_T2_A24_15</td>
</tr>
<tr>
<td>&quot;vfree&quot;</td>
<td>K15</td>
<td>LVCMOS33;</td>
<td>I/O_L24P_T3_RS1_15</td>
</tr>
<tr>
<td>#NET &quot;led&lt;2&gt;&quot;</td>
<td>J13</td>
<td>LVCMOS33;</td>
<td>I/O_L17N_T2_A25_15</td>
</tr>
<tr>
<td>#NET &quot;led&lt;3&gt;&quot;</td>
<td>N14</td>
<td>LVCMOS33;</td>
<td>I/O_L8P_T1_D11_14</td>
</tr>
<tr>
<td>#NET &quot;led&lt;4&gt;&quot;</td>
<td>R18</td>
<td>LVCMOS33;</td>
<td>I/O_L7P_T1_D09_14</td>
</tr>
<tr>
<td>#NET &quot;led&lt;5&gt;&quot;</td>
<td>V17</td>
<td>LVCMOS33;</td>
<td>I/O_L18N_T2_A11_D27_14</td>
</tr>
<tr>
<td>#NET &quot;led&lt;6&gt;&quot;</td>
<td>U17</td>
<td>LVCMOS33;</td>
<td>I/O_L17P_T2_A14_D30_14</td>
</tr>
<tr>
<td>#NET &quot;led&lt;7&gt;&quot;</td>
<td>U16</td>
<td>LVCMOS33;</td>
<td>I/O_L18P_T2_A12_D28_14</td>
</tr>
<tr>
<td>#NET &quot;led&lt;8&gt;&quot;</td>
<td>V16</td>
<td>LVCMOS33;</td>
<td>I/O_L16N_T2_A15_D31_14</td>
</tr>
<tr>
<td>#NET &quot;led&lt;9&gt;&quot;</td>
<td>T15</td>
<td>LVCMOS33;</td>
<td>I/O_L14N_T2_SRCC_14</td>
</tr>
<tr>
<td>#NET &quot;led&lt;10&gt;&quot;</td>
<td>U14</td>
<td>LVCMOS33;</td>
<td>I/O_L22P_T3_A05_D21_14</td>
</tr>
<tr>
<td>#NET &quot;led&lt;11&gt;&quot;</td>
<td>T16</td>
<td>LVCMOS33;</td>
<td>I/O_L15N_T2_DQS_DOUT_CSO_B_14</td>
</tr>
<tr>
<td>#NET &quot;led&lt;12&gt;&quot;</td>
<td>V15</td>
<td>LVCMOS33;</td>
<td>I/O_L16P_T2_CSI_B_14</td>
</tr>
<tr>
<td>#NET &quot;led&lt;13&gt;&quot;</td>
<td>V14</td>
<td>LVCMOS33;</td>
<td>I/O_L22N_T3_A04_D20_14</td>
</tr>
<tr>
<td>#NET &quot;led&lt;14&gt;&quot;</td>
<td>V12</td>
<td>LVCMOS33;</td>
<td>I/O_L20N_T3_A07_D23_14</td>
</tr>
<tr>
<td>#NET &quot;led&lt;15&gt;&quot;</td>
<td>V11</td>
<td>LVCMOS33;</td>
<td>I/O_L21N_T3_DQS_A06_D22_14</td>
</tr>
</tbody>
</table>