2. Examples from Class 1. Car Security System

2.1 System design and Manual Synthesis

Here we had three digital inputs (a) **Key** which was ‘1’ when the alarm is activated, (b) **Door** which is ‘1’ when the door is open and (c) **Ultrasonic detector** which was ‘1’ when there’s movement in the car. We used this description to create a truth table for the system, which looked like this:

<table>
<thead>
<tr>
<th>K</th>
<th>D</th>
<th>U</th>
<th>Horn</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

By inspecting the truth table, we were able to construct a *sum-of-products* circuit which included the three product terms:

\[ K.U.D + K.D.U + K.D.U \]

This reads “(K AND U AND NOT-D) OR (K AND D AND NOT-U) OR (K AND D AND U)”. And (!) here’s the circuit we can produce using the “sum-of-products” methodology:
Now let’s have a look at how this can be translated into the VHDL program, shown below. So let’s do an analysis of this code:

(a) **Lines 10-14** import the required *libraries* of data into our program.

(b) **Lines 16-21** specify the “entity” of our circuit, in particular the inputs and outputs. We see that K,D and U are “standard logic inputs” (in std_logic) while “Horn” is a “standard logic output” (out_std_logic). Here, *standard logic* refers to our understanding of digital logic, with states “1” and “0”. These are wires going in and out of the alarm functional block.

(c) **Lines 23-34** define how our circuit behaviours, its “functionality”.

```
1  ---------------------------------------------------------------
2  --
3  -- CBF Feb 2006
4  --
5  -- Car Security System discussed in class
6  --
7  -- This is coded as a "sum of products"
8  --
9  ---------------------------------------------------------------
10 library ieee;
11 use ieee.std_logic_1164.all;
12
13 library UNISIM;
14 use UNISIM.VComponents.all;
15
16 entity car is port (  17     K : in std_logic;
18     D : in std_logic;
19     U : in std_logic;
20     Horn : out std_logic);
21 end car;
22
23 architecture Dataflow of car is
24   signal prod1, prod2, prod3: std_logic;
25 begin
26
27   prod1 <= K and U and (not D);
28   prod2 <= K and D and (not U);
29   prod3 <= K and D and U;
30
31   Horn <= prod1 or prod2 or prod3;
32
33 end Dataflow;
```

**Line 24,** we declare signal wires which exist within this digital functional module. These are the “product” terms in the sum-of-products circuit. They are defined in **Lines 27-29** and correspond to the three terms in the Boolean expression given above. Finally, in **line 31,** we take the OR of the product terms and assign it to the output wire “Horn”).
Now let’s open up this VHDL file in Xilinx ISE and investigate. First we shall find out how to view the circuit diagram synthesised from the VHDL code, and then run a simulation.

Open the Xilinx ISE project navigator and select File -> Open Project. Navigate the folder CARSECURITY_DFL and open the project CARSECURITY_DFL.ise. You should find the test-bench waveform I have made for you. If all is well, it will look like this. If it’s not there, you’ll have to make one (see previous handout).

2.2 Circuit Synthesized by Xilinx ISE.

Now let’s investigate how the Xilinx tool builds a circuit. First note that we do not expect to get the sum-of-products circuit we made above. The Xilinx Spartan-3 device contains specific functional blocks, and the synthesizer will produce a solution circuit which will run on these blocks.

1. In the “Processes for Source” window, first select the “View RTL Schematic”. This is “Register Transfer Level” and will give a block diagram in terms of high-level components such as registers, ALUs, multiplexers, etc.
More informative is the “View Technology Schematic” which will show how your circuit will be implemented on the Spartan-3 device.

2. Select this tab and you’ll get another block diagram in the window on the right. Right-click on this and choose “Push into the Selected Instance”. This will drop you down to the Spartan-3 hardware level. Here’s what I got:

![Diagram LUT3_A8]

That doesn’t tell us much. What’s a LUT (“Look up Table”)? I’ll explain in class, but basically its RAM used to implement a collection of gates (a bit like that slide I used in Session 2). Can we see the contents of the LUT? Yes by right clicking on the LUT and then choosing “Show LUT Contents” Here’s what I got:

![Diagram LUT]

So what does this all mean? Well you have to read these two diagrams together. U goes into D into I1 and K into I0. This is correct, since K is providing the control signal which is passed to both AND-gates to enable the D and U inputs. The output O goes to the horn. Remember that this circuit does not exist! It is implemented as a lookup table in memory on the Spartan-3.

### 2.3 Simulation of the Synthesized Circuit

Just to complete this section, let’s open up ModelSim and run the simulator on our synthesized circuit (see last handout). Don’t forget to start Modelsim with the test
bench waveform (.tbw) highlighted and not the VHDL (.vhd) file. Here’s the result I go which looks just-about right.