I39/2215/2013
AKELLO O. LZRS

OCTOBER 29, 2015

REPORT ON 2 TO 4 DECODER

A decoder is a device that takes in input(s) and outputs different outputs depending on the input value(s). It takes a 'n' number of inputs and outputs '2^n'outputs in which case our 2by4 decoder takes 2 inputs and outputs 4 inputs. It has to be noted though that only one output can be true /(1) for the certain values at the input; hence making a decoder useful for performing selections.

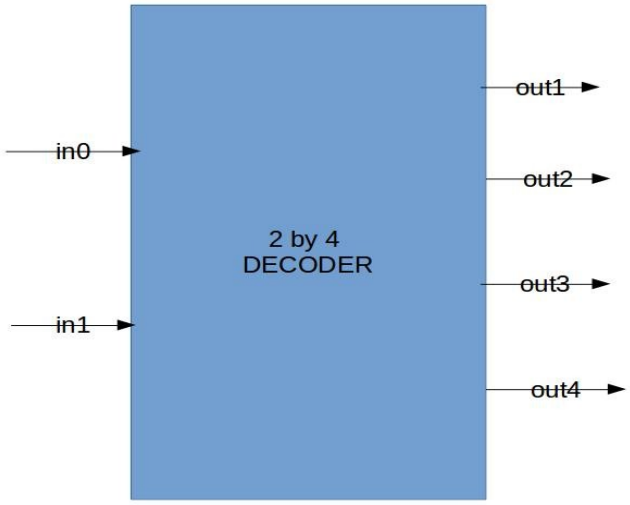
Most code used in this report are excerpts, actual codes can be found in files *decoder2to4.h, decoder2to4.cc, driver2to4.h and monitor2to4.h*

OBJECTIVE

Use systemc to model and simulate how a 2 by 4 decoder works

Input 1	Input 0	Output 3	Output 2	Output 1	Output 0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

A truth table of a 2 by 4 decoder ;output 0 being the first output



PROCEDURE

Three modules were used to model and finally simulate a 2 by 4 decoder. These were namely : driver, monitor and decoder.
First the driver module, as can be found in *driver2to4.h*, is used to generate inputs to the the decoder , after every 5 nanoseconds. It is supposed to do this unlimitedly until the end of simulation.

```
while(1)
        {
                d_a=0; d_b=0;       //0 0 d_b being the most significant
                wait(5, SC_NS);
                d_b=1;              //0 1
                wait(5, SC_NS);
                d_a=1; d_b=0;       //1 0
                wait(5, SC_NS);
                d_b=1;              //1 1
                wait(5, SC_NS);
        }
```

The monitor module, *monitor2to4.h*, is used to check the outputs through the line

```
sensitive<<m_c<<m_d<<m_e<<m_f;
```

and print out the results

```
cout<<"at "<<sc_time_stamp()<<" input is: "<<m_a<<" and "<<m_b<<" outputs are:
"<<m_c<<", "<<m_d<<", "<<m_e<<" and "<<m_f<<endl; //shown as outputted from
the decoder m_c being the first output but not to look as the truth table
```

The last of the three modules is the decoder found in *decoder2to4.h*. It is essentially the most important module of all three since it does the actual decoding computation. It determines output according to the the inputs whereby in this case when both inputs are zero, one is outputted at the first output, c, and the rest zero while when both are one, one is only outputted at the last output, f, and the rest zero.

```
void decode2to4(void)
{
        c = (a == 0 && b == 0)?1:0;
        d = (a == 0 && b == 1)?1:0;
        e = (a == 1 && b == 0)?1:0;
        f = (a == 1 && b == 1)?1:0;
}
```

All these modules were then instantiated in the *decoder2to4.cc*. The driver module instance was only used at the inputs to generate values.

```
decoder2to4 dec("decoder instance");
driver2 dr("driver");
monitor2to4 mn("monitor");
//interconnections between modules
dr.d_a(in1);dr.d_b(in2);
dec.a(in1);dec.b(in2);
mn.m_a(in1);mn.m_b(in2);

dec.c(out1);
mn.m_c(out1);

dec.d(out2);
mn.m_d(out2);

dec.e(out3);
mn.m_e(out3);

dec.f(out4);
mn.m_f(out4);
```

Another function in the *decoder2to4.cc* was to create a trace file that was then used to simulate and store results in *timing_diagram.vcd.* The file *timing_diagram.vcd* can be used by external program “gtkwave” to construct visually simulation of the decoder.

```
c_trace_file *tf;
        tf = sc_create_vcd_trace_file("timing_diagram");
        tf ->set_time_unit(1, SC_NS);
        //trace the signals interconnecting modules
        sc_trace(tf, in1, "first_binary_input");
        sc_trace(tf, in2, "second_binary_input");
        sc_trace(tf, out1, "input_is_zero");
        sc_trace(tf, out2, "input_is_one");
        sc_trace(tf, out3, "input_is_two");
        sc_trace(tf, out4, "input_is_three");

        //run a simulation for 50 systemc nanoseconds
        if (!sc_pending_activity())
                sc_start(55, SC_NS);
        //close the trace file
        sc_close_vcd_trace_file(tf);
        return 0;
```

A *Makefile* file was created to compile and run the code, and output the results to r*esults.tx*t.

RESULTS

These were the results as seen from *results.txt.*

```
Info: (I703) tracing timescale unit set: 1 ns (timing_diagram.vcd)
at 0 s input is: 0 and 0 outputs are: 1, 0, 0 and 0
at 5 ns input is: 0 and 1 outputs are: 0, 1, 0 and 0
at 10 ns input is: 1 and 0 outputs are: 0, 0, 1 and 0
at 15 ns input is: 1 and 1 outputs are: 0, 0, 0 and 1
at 20 ns input is: 0 and 0 outputs are: 1, 0, 0 and 0
at 25 ns input is: 0 and 1 outputs are: 0, 1, 0 and 0
at 30 ns input is: 1 and 0 outputs are: 0, 0, 1 and 0
at 35 ns input is: 1 and 1 outputs are: 0, 0, 0 and 1
at 40 ns input is: 0 and 0 outputs are: 1, 0, 0 and 0
at 45 ns input is: 0 and 1 outputs are: 0, 1, 0 and 0
at 50 ns input is: 1 and 0 outputs are: 0, 0, 1 and 0
```

The results seem to follow on the objective noting that the outputs are in the order of the output1, output2, output3 and output4 respectively.

My gtkwave program did not produce the expected waveform, and I feel this may be a problem with my installation.

CONCLUSION

The objectives of the project were clearly met as can be seen from the results. This is despite the “gtkwave” program not producing a waveform from the vcd file, thereby concluding my project as successful.