ENG632 VHDL & FPGA Systems (M3)

Sam O'Neill – UP879282 – Milestone 3 Design Record

Milestone 3: Digital Modulator, Error Block and Digital Demodulator

Objectives: demonstrate a working system where the FPGA device generates data depending on user inputs and is capable of removing any errors introduced during transmission of the data. Use a digital modulation scheme where specific modulation I and Q waveforms represent each of four possible symbols used in transmission.

Contents

1 Design Requirements

1.1.1 – Design should be implemented on an Artix 7 Basys 3 FPGA board
1.1.2 – Design should be implemented using VHDL and created in Vivado 2020.1
1.1.3 – Top-level design/functionality of M3 is a digital modulation system
1.1.4 – Top-level design will be broken down into hierarchical design of subsystems
1.1.5 – Design subsystems include data generator, modulator, demodulator, and display driver
1.1.6 – Top-level should be driven by a 100MHz system clock
1.1.7 – Subsystems should be driven by divided clock enable signals
1.1.8 – Design should use several 'data select' switches to change data output
1.1.9 – Design should use 2 'error select' switches to change the error introduced
1.1.10 – Data display should be controlled by 'display select' switches
1.1.11 – Additional: implement a cyclic redundancy check and/or bi-directional flow control

1.1.1 Basys 3 Board Diagram and Component Descriptions

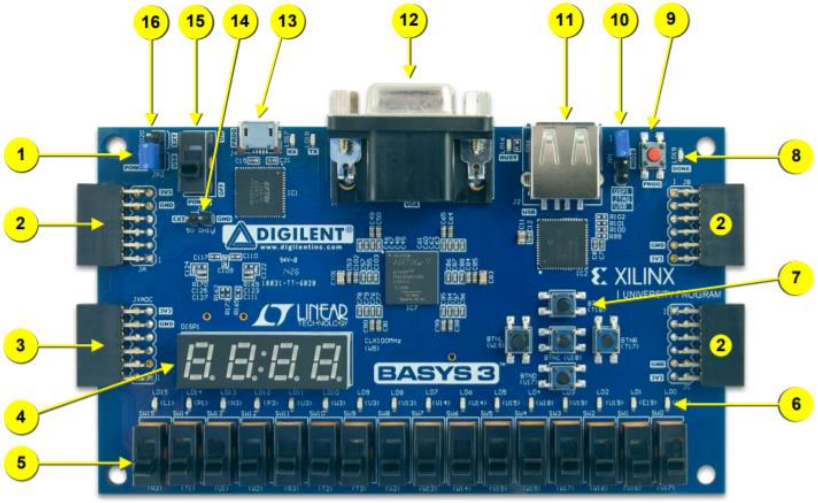


Figure 1: The Basys 3 FPGA board with annotated component descriptions

Callout	Component Description	Callout	Component Description
1	Power good LED	9	FPGA configuration reset button
2	Pmod port(s)	10	Programming mode jumper
3	Analog signal Pmod port (XADC)	11	USB host connector
4	Four digit 7-segment display	12	VGA connector
5	Slide switches (16)	13	Shared UART/ JTAG USB port
6	LEDs (16)	14	External power connector
7	Pushbuttons (5)	15	Power Switch
8	FPGA programming done LED	16	Power Select Jumper

1.1.4/1.1.5 Hierarchical Design

The system will consist of 5 main blocks. The data generator will take an input and respond by outputting a specific 4-bit data sequence in the form of two 2-bit 'symbols.' The modulator will then take these symbols and represent them as modulation I/Q waveforms using modulation scheme B. The channel block is then used to introduce error to these waveforms depending on the current switch setting. The demodulator processes the waveforms and combines symbols to retrieve the original data value. Finally, the display driver block will control the seven segment display, and display the data.

1.1.6 100MHz System Clock

The top-level design should be driven by the built in system clock of the Basys 3. This is configured using the clocking wizard in the Vivado software and set to 100MHz.

1.1.7 Clock Enable Signals

The system clock will be divided into three clock enable signals to drive the necessary system blocks. A 1Hz signal controls the data generator, a 2Hz signal controls the symbol converter, a 16Hz signal and a 250Hz signal controls the display.

1.1.8 Data Select Switches

The design will use 4 switches on the Basys 3 to select the data sequence generated. It is the specific combinations of these switches that will determine which data will be modulated and transmitted.

1.1.9 Error Select Switches

The error to be introduced to the transmission is controlled by two error select switches, providing four different error modes, with varying PRNG ranges: no change, ±16 range, ±32 range, or ±64 range.

1.1.10 Display Select Switches

3 switches will be used to determine the configuration of the seven segment display. This provides 7 different display modes. More detailed information can be found in section 3.8.

1.1.11 Additional Cyclic Redundancy Check and/or Bi-directional flow control

A cyclic redundancy check can be implemented and added to data packets from the data generator. This will be used to check for errors in the packet and request a resend if necessary. Also bi-directional flow control between the data generator and demodulator can be implemented to manage the packet flow.

2 Initial Design

2.1 System Diagram

Defining external and internal signals:
External: Data select switches (i_sw15, i_sw14, i_sw13, i_sw12), start/stop button (i_btnD), reset button (i_btnU), 7-segment display (o_segCathodes, o_segAnodes, o_segDP), error select switches (i_sw11, i_sw10), display select switches (i_sw9, i_sw8, i_sw7), LEDs (i_led15, i_led14, i_led13, i_led12)
Internal: System clock (i_C100MHz), clock enables (i_CE1Hz, i_CE2Hz, i_CE250Hz)
Note: more internal I/O will be used in the design to transmit signals between subsystems, but these will be defined alongside the specific entities where they are required.

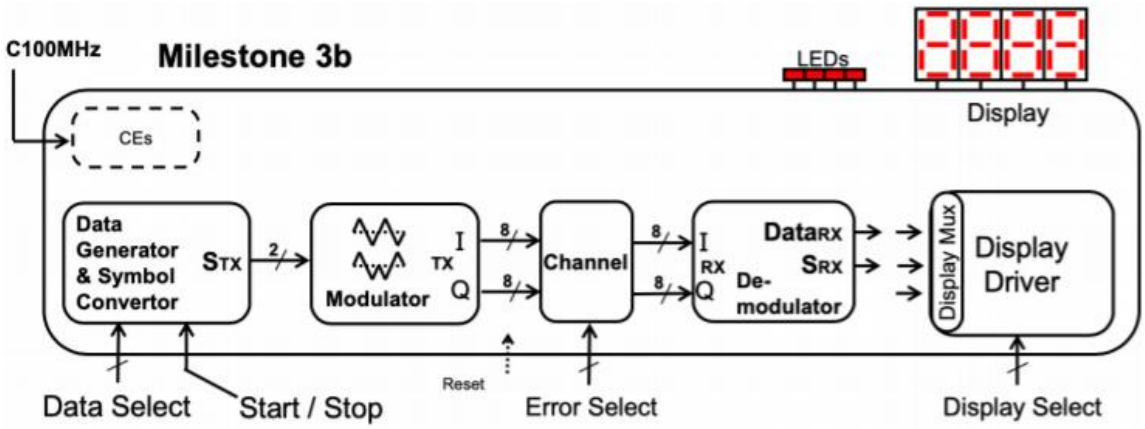


Figure 2.1: Milestone 3 top-level block diagram

Fig 2.1 shows the top-level block diagram of the system, using the main external and internal signals defined above.

2.2 System Explanation

The entire system is driven by the 100MHz clock on the Basys 3. This is used to derive 4 smaller clock enable signals using a clock divider. All clock enable entities have a reset input which allows the internal counters to be reset if necessary.
The data generator/symbol converter block outputs 2-bit symbols to represent the generated data value at 2Hz. A further breakdown of this sub-system is shown in *fig. 3*:

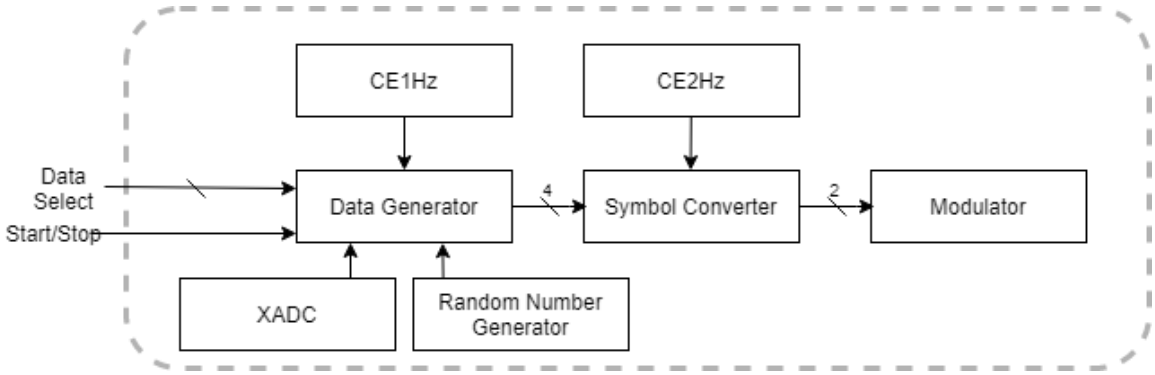
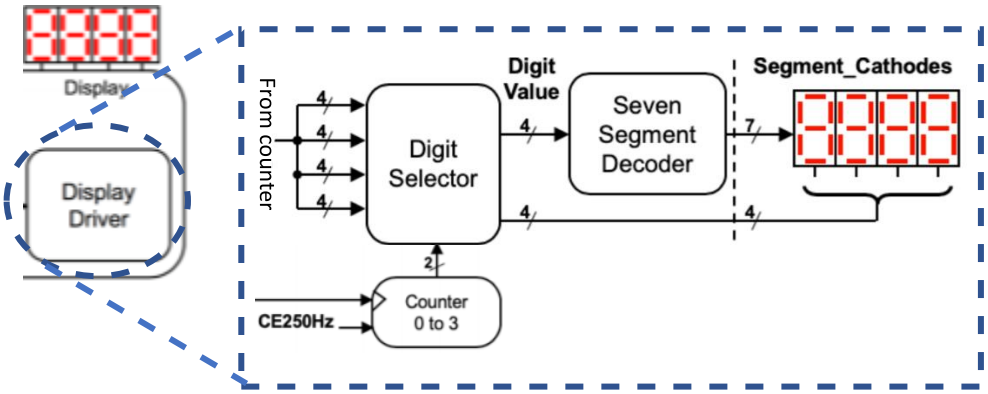


Figure 2.2: data generator block diagram

The modulator block outputs different I/Q waveforms as 8-bit numbers to represent the symbols it receives. The modulation scheme is based on triangular waveforms. The channel block adds noise to the transmission by adding or subtracting pseudo-random values from the transmitted signals. The range of these values is controlled by the error select switches. Then the demodulator will process the received waveform with a multiply accumulate operation to determine the original data and symbol values.
Finally, all data will be displayed on the seven-segment display, controlled by the display driver. A breakdown of the driver is shown in *fig. 2.3*. The display uses a 62.5Hz scanning refresh scheme to ensure all digits are kept illuminated. A timing diagram for this is shown in *fig. 2.4*. The digit selector controls this refresh scheme and also selects the digit to be addressed. The decoder controls the common cathodes to display a number on the current digit.



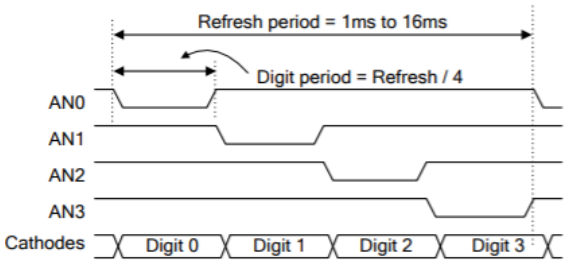


Figure 2.3 (above): display driver breakdown
Figure 2.4 (left): scanning controller timing diagram

3 VHDL Modelling

3.1 Clock Enables

Separate clock enable signals can be generated using a clock divider process and used to control and synchronise any necessary entities. The pseudo code demonstrating this process is below:

```
-- Internal Counter
IF clock rising edge is detected THEN
        IF counter is less than 100,000,000 THEN
                increment counter
        ELSE if counter reaches 100,000,000 THEN
                reset counter

-- Signal Generation
IF clock falling edge is detected THEN
        IF counter is less than 100,000,000 THEN
                change value of signal to '0'
        ELSE if counter reaches 100,000,000 THEN
                change value of signal to '1'
```

The counter increments on every rising edge of the clock pulse, which is 100MHz. The increment of the internal counter determines the frequency of the output signal, for example a 250Hz signal would require an increment of 250.

Clock Enable Entity Record:

T29_M3_ClockEnable

Inputs	Outputs	Generics
• i_clk	• o_CE	• g_maxCount • g_increment

Version	Date	Description
1.0	30-11-20	Created entities. Defined I/O and architecture

3.2 Data Generator

Fig. 3 shows the various modes of the data generator, controlled by the 4 switches which form the 4-bit binary 'Data Select' input.

Start/Stop	Data Select (BIN)	Data Select (DEC)	Data Value
0	XXXX	-	"0000"
1	0000	0	"0001"
1	0001	1	"0111"
1	0010	2	"1110"
1	0011	3	"1000"
1	0100	4	Count 0-F
1	0101	5	Random num
1	0110	6	Student ID 1
1	0111	7	Student ID 2
1	1XXX	8	Temperature

Figure 3 (left): Data generation modes

Data generator pseudo code:

```
Data select = Switch1 and Switch2 and Switch3 and Switch4

IF start/stop = 1 THEN
        IF clock rising edge is detected THEN
                IF Switch1 = 1 THEN
                        output = temperature
                ELSE if Switch1 = 0 THEN
                        CASE data select
                                0: output = “0001”
                                1: output = “0111”
                                2: output = “1110”
                                3: output = “1000”
                                4: output = 0-F counter
                                5: output = random number
                                6: output = “879282”
                                7: output = “879243”
                                8: output = XADC temperature
```

Data Generator Entity Record

T29_M3_DataGenerator

Inputs	Outputs
• i_Sw15, i_Sw14, i_Sw13, i_Sw12 • i_StartStop, i_reset • i_Clk, i_CE1Hz	• Data_Select_bin, o_DataValue

Version	Date	Description
1.0	30-11-20	Created entity. Defined I/O and architecture
1.1	7-12-20	Added 0-F counter, and student number counter
1.2	8-12-20	Added if/else to switches process to correctly reset D4 when start/stop is pressed (see hardware test 3)
1.3	9-12-20	Added PRNG
1.4	20-01-21	Adjusted PRNG to be 4-bits so that it works properly with the display

3.3 Random Number Generator

A random number generator is incorporated into the data generator for data select mode 5 (see *fig. 3*). This uses a linear feedback shift register where the input is a polynomial function of previous outputs. The same number generator will also be used in the error channel to apply errors to the data, so a full explanation of the PRNG design can be found in section 3.6.

3.4 Symbol Converter

The symbol converter uses an internal 0-1 counter to determine which symbol is being outputted. When the counter is 0, the output will be the first symbol. When it is 1, the output will be the second symbol. The counter increments when CE2Hz is detected, so that the symbols are outputted every 0.5 seconds.

Symbol converter pseudo code:

```
-- Internal Counter
IF clock rising edge is detected THEN
        IF counter limit is not reached THEN
                increment counter
        ELSE if counter limit is reached THEN
                reset counter

-- Symbol Generation
sym1 = data first 2 bits
sym2 = data second 2 bits
IF CE2Hz is high THEN
        IF counter is 0 THEN
                output = sym1
        ELSE if counter is 1 THEN
                output = sym2
```

Symbol Converter Entity Record

T29_M3_SymbolConv

Inputs	Outputs
• i_Data • i_CE2Hz • i_Clk	• o_Symbol

Version	Date	Description
1.0	21-11-20	Created entity. Defined I/O and architecture
1.1	21-11-20	- Removed counter and replaced with 'tmp' signal - Removed r_Sym signals which were unnecessary as o_Symbol can just be directly addressed - Added 'reset' signal to solve immediate output problem found during simulations

3.5 Modulator

The modulator will be used to generate triangular modulation waveforms I_{TX} and Q_{TX} based on the symbols received from the symbol converter. This design will modulate the data using **modulation scheme B** (see *fig. 4.1*).

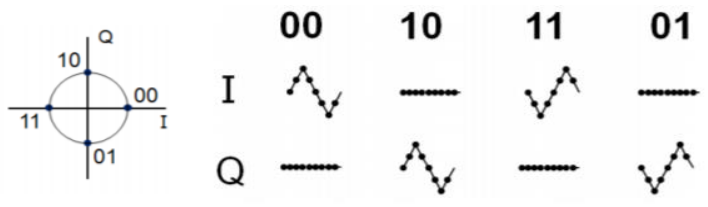
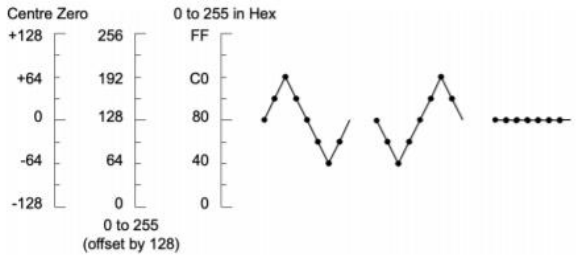


Figure 4.1 (above): the I and Q waveforms generated by modulation scheme B
Figure 4.2(below): the scale of I and Q waveforms



As seen in *fig. 4.2*, the generated waveforms will have a range of $\pm 64_{DEC}$, though the total available range will be $\pm 128_{DEC}$ to allow for noise to be added. Values are offset by $+128_{DEC}$ to circumvent the need for signed bits, so that the range is actually $0\text{-}255_{DEC}$. This can be represented in hex as 0-FF.

The triangular waveforms will output at the same rate as the symbol converter, so that the frequency of I_{TX} and Q_{TX} is 2Hz. Each cycle of the waveforms must be represented by eight 8-bit values, so this output will be controlled by a 16Hz clock enable.
To generate the waveforms, the following calculations were used to determine the required value between ±64 at a specific time:

1 second = 100,000,000 clock pulses
2 waveforms per second, so 8 steps of 64_{DEC}
$8*64 = 512_{DEC}$ per second
100,000,000 / 512 = 195,312.5 pulses per value

Pseudo code for the modulator is below:

```
Initialise data to 128
Function to determine initial direction of data (<128 or >128)
IF clock rising edge detected THEN
        IF ce2hz detected THEN
                call direction function
        IF counter is less than 195312 THEN
                increment counter
        ELSE if counter limit reached THEN
                WHEN symbol = 00:
                        IF max/min values reached THEN change data direction
                        ELSE increment value
                WHEN symbol = 10:
                        IF max/min values reached THEN change data direction
                        ELSE increment value
                WHEN symbol = 10:
                        IF max/min values reached THEN change data direction
                        ELSE increment value
                WHEN symbol = 01:
                        IF max/min values reached THEN change data direction
                        ELSE increment value
        IF ce16hz detected THEN
                output the current value
```

Scheme B Modulator Entity Record

T29_M3_ModulatorB

Inputs	Outputs
• i_clk • i_CE2Hz, i_CE16Hz • i_Symbol	• o_Idata, o_Qdata (used to construct the waveforms) • o_I_tx, o_Q_tx (data values taken at 8 points per wave cycle)

Version	Date	Description
1.0	07-01-20	Created entity. Defined I/O and architecture
1.1	08-01-20	Adjusted design following simulations by adding 'direction' function to fix incorrect outputs
1.2	08-01-20	Added CE16Hz clock enable to control data output

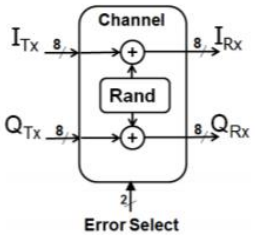
3.6 Error Channel

To test the robustness of the modulation scheme, the error channel block will be used to apply 'noise' to the transmission using randomly generated numbers. The range of these numbers is decided using the error select switches (*fig. 5.1*).

Error Select	PRNG Range
00	No change
01	±16
10	±32
11	±64

Figure 5.1 (left): the range of random numbers used in each error mode

Figure 5.2 (right): the design of the channel block



There are several methods for implementing a PRNG (pseudo-random number generator) in VHDL. One of the most common is a linear feedback shift register (LFSR). This uses a shift register comprised of D flip-flops where the input is a linear function of the previous states, producing a sequence of pseudo-random numbers. It is pseudo-random because the sequence will eventually repeat and will also follow a mathematically predictable sequence. The number of bits used in the LFSR determines its repetition time and the range of numbers it can produce. An n-bit counter has a maximum sequence of 2^n-1. Since this design requires a maximum range of ±64, a 7-bit counter is needed. This provides a maximum sequence of (2^7-1) 127 random numbers. In modes where the numbers must be smaller, the result of the PRNG will be divided by 2 or 4.

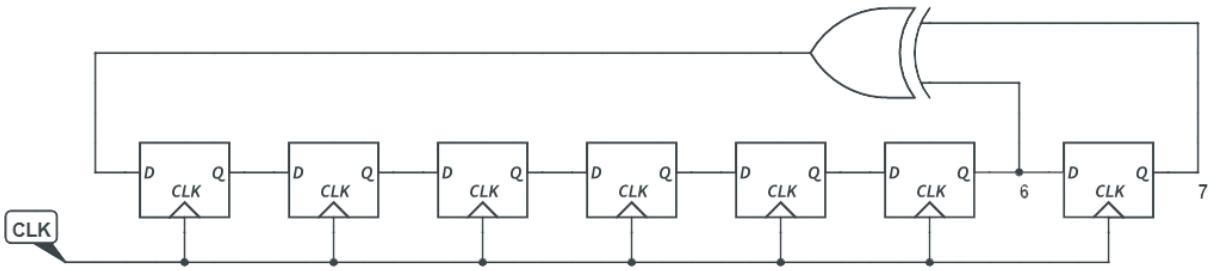


Figure 5.3: a 7-bit LFSR PRNG

Pseudo code for the channel is below:

```
Initialise output to 128

-- PRNG
IF clock rising edge detected THEN
        IF reset is not pressed THEN
                LFSR tap 6 XOR LFSR tap 5
                Shift register by 1
        ELSE if reset is pressed THEN
                reset random num to seed value

-- Output
IF clock rising edge detected THEN
        CASE error select
                WHEN “00”, output has no change
                WHEN “01”, divide random num by 4 and add to output
                WHEN “10”, divide random num by 2 and add to output
                WHEN “11”, add random num to output
```

Error Channel Entity Record

T29_M3_Channel

Inputs	Outputs
• i_clk, i_CE16Hz • i_sw11, i_sw10, i_reset • i_I_tx, i_Q_tx	• o_I_rx, o_Q_rx • o_errSel

Version	Date	Description
1.0	12-01-20	Created entity. Defined I/O and architecture
1.1	13-01-20	Modified LFSR design to fix problems with random number generation
1.2	13-01-20	Added error select switches/output processes

3.7 Digital Demodulator

The digital demodulator will convert the received error-containing waveform back into its original data values using a multiply and accumulate operation against a reference waveform.

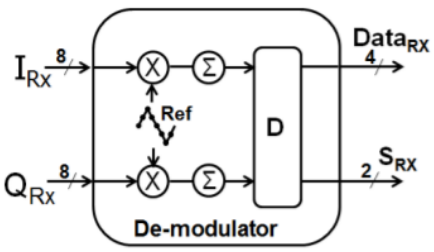


Figure 6.1 (left): the design of the demodulator block

Figure 6.2 (right): the reference waveform to be used during demodulation

The multiply-accumulate operation takes the product of two numbers (X on *fig. 6.1*) and then adds it to an accumulator (Σ on *fig. 6.1*). This operation will be performed on both the I and Q waveforms. The results of the operation will be used for the decision process (D on *fig. 6.1*) which will be able to decide which symbol has been received.

Demodulator Entity Record

T29_M3_Demodulator

Inputs	Outputs
• i_clk, i_CE16Hz, i_CE2Hz, i_reset • i_IData, i_QData • i_errSel	• o_DataRX • o_SymbolRX

Version	Date	Description
1.0	20-01-20	Created entity. Defined I/O and architecture
1.1	04-02-21	Changed values to integers for easier calculations
1.2	04-02-21	Instantiated decision entities

3.8 Display Driver

The display driver consists of:

- digit selector case statement, controlling the anodes and which digit is currently being addressed
- display decoder with/select statement, controlling the cathodes

The binary to 7-segment cathode truth table to be used in this entity is shown below. It maps a binary input to its corresponding cathode output.

Binary	Hex	CA	CB	CC	CD	CE	CF	CG	Cathodes
0000	0	L	L	L	L	L	L	H	0000001
0001	1	H	L	L	H	H	H	H	1001111
0010	2	L	L	H	L	L	H	L	0010010
0011	3	L	L	L	L	H	H	L	0000110
0100	4	H	L	L	H	H	L	L	1001100
0101	5	L	H	L	L	H	L	L	0100100
0110	6	L	H	L	L	L	L	L	0100000
0111	7	L	L	L	H	H	H	H	0001111
1000	8	L	L	L	L	L	L	L	0000000
1001	9	L	L	L	L	H	L	L	0000100
1010	a	L	L	L	L	L	H	L	0000010
1011	b	H	H	L	L	L	L	L	1100000
1100	c	L	H	H	L	L	L	H	0110001
1101	d	H	L	L	L	L	H	L	1000010
1110	e	L	H	H	L	L	L	L	0110000
1111	f	L	H	H	H	L	L	L	0111000

Figure 7.1: binary to 7-segment decoder truth table

The display is controlled by 3 display select switches, so that there is a total of 7 display modes. Each mode is described in *fig. 7.2*. The first mode uses the data generator values only, while the other 6 modes focus on modulation and display the data values before and after modulation/demodulation. 3 modes display data for modulation scheme A, and 3 display data for scheme B.

Display Select				D4	D3	D2	D1
X00	As for M2		See M2	DS	DV	Symbol	
001	(M3a)	A	I & Q Tx	I Tx		Q Tx	
010	(M3b)		Data Values Tx, Rx	Data Value Tx		Data Value Rx	
011	(M3b)		I & Q Rx	I Rx		Q Rx	
101	(M3a)	B	I & Q Tx	I Tx		Q Tx	
110	(M3b)		Data Values Tx, Rx	Data Value Tx		Data Value Rx	
111	(M3b)		I & Q Rx	I Rx		Q Rx	

Figure 7.2: the 7 display modes

Display driver record

T29_M3_DisplayDriver

Inputs	Outputs
• i_dataMode, i_dataValue • i_digitSelect • i_Symbol • i_sw9, i_sw8, i_sw7	• o_segAnodes • o_segCathodes

Version	Date	Description
1.0	20-11-20	Created entity. Defined I/O and architecture
1.1	03-12-20	Made several changes to digit selector to incorporate all possible outputs of the data generator
1.2	17-01-21	Added display select switches
1.3	17-01-21	Changed digitSelect process to incorporate new display options

3.9 Top Level Entity

This is the top level file which instantiates all other entities used in the system, and provides connections to/from all external I/O.

Top level pseudo code:

```
INSTANTIATE system clock
INSTANTIATE CE1Hz
INSTANTIATE CE2Hz
INSTANTIATE CE16Hz
INSTANTIATE CE250Hz
INSTANTIATE data generator
INSTANTIATE symbol converter
INSTANTIATE modulator A
INSTANTIATE modulator B
INSTANTIATE error channel A
INSTANTIATE error channel B
INSTANTIATE demodulator A
INSTANTIATE demodulator B
INSTANTIATE 0-3 counter
INSTANTIATE display driver
```

Top level record
T29_M3_topLevel

Inputs	Outputs
• i_C100MHz • i_SW15, i_SW14, i_SW13, i_SW12, i_SW11, i_SW10, i_SW9, i_SW8, i_SW7 • i_StartStop, i_reset	• o_segCathodes • o_segAnodes

Version	Date	Description
1.0	7-01-20	Created entity. Instantiated completed entities.
1.1	20-01-20	Added debounce to start/stop button
1.2	04-02-21	Added demodulator B and modulator A entities

4 Simulations

CE1Hz:

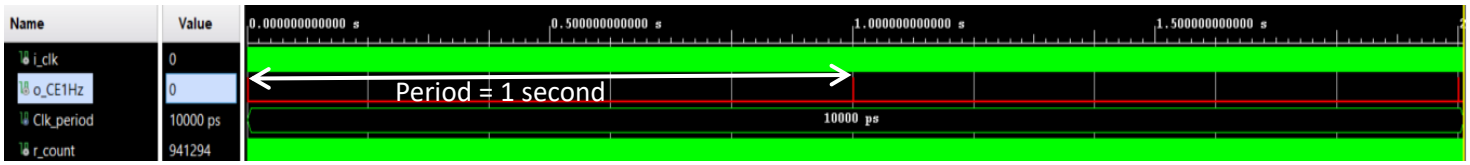


Figure 8: CE1Hz simulation output

The entity behaves correctly. The output o_CE1Hz pulses once every second (indicated by the red signal), once r_count reaches the limit of 100,000,000 (which takes 1 second due to the counter using an increment of 1). This means the output has a period of 1 second, and therefore a frequency of 1Hz.

CE2Hz:

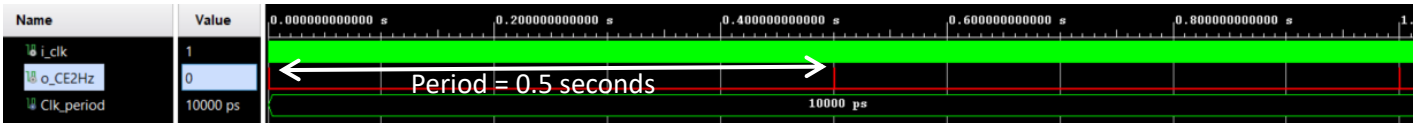


Figure 9: CE2Hz simulation output

The entity behaves correctly. The output o_CE2Hz pulses twice every second (indicated by the red signal), once r_count reaches the limit of 100,000,000 (which takes half a second due to the counter using an increment of 2). This means the output has a period of 0.5 seconds, and therefore a frequency of 2Hz.

CE250Hz:

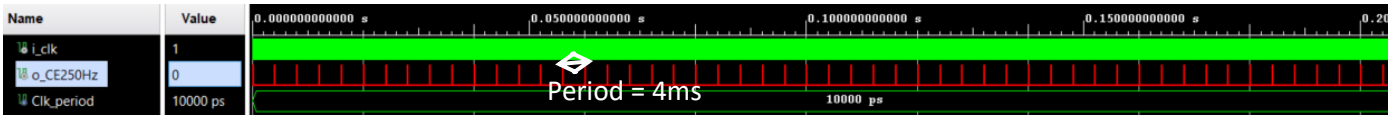


Figure 10: CE250Hz simulation output

The entity behaves correctly. The output o_CE250Hz pulses 250 times every second (indicated by the red signal), once r_count reaches the limit of 100,000,000 (which takes 0.004 seconds due to the counter using an increment of 250). This means the output has a period of 0.004 seconds, and therefore a frequency of 250Hz.

0-3 Counter:

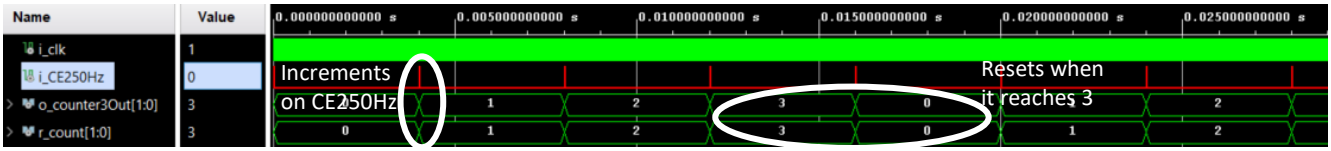


Figure 11: 0-3 counter simulation output

The entity behaves correctly. The output o_counter3Out increments on each pulse of the 250Hz clock enable entity (indicated by the red signal). The output counts from 0 to 3, and then resets back to 0. This is a 62.5Hz refresh scheme for driving the seven-segment display and ensuring all digits are properly illuminated.

Symbol Converter:

Testbench uses a test data value of '1101' ('d'), which should be split into two symbols of '11' and '01'. Initial simulation below shows that the data is correctly split (shown by the values of r_Sym1 and r_Sym2 in the objects list), however only the first of these symbols is outputted. The output o_Symbol does not change with each pulse of CE2Hz as it should, and instead remains at the first value ('11').

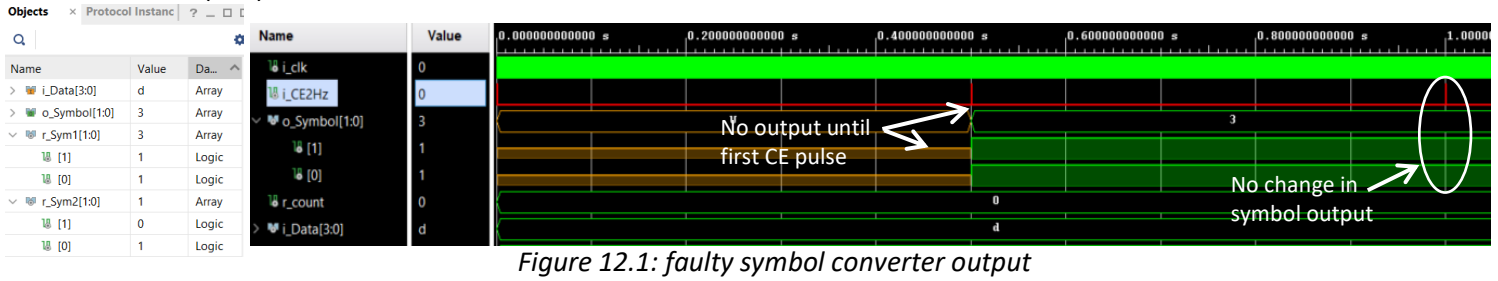


Figure 12.1: faulty symbol converter output

This is due to the counter not working correctly. R_count should count from 0-1 on each pulse, but it stays at 0 so only the first symbol is outputted. Also, the symbols should be outputted straight away, but the output is 'U' until the first clock enable signal.

To fix both of these issues, the counter was removed and replaced with a signal that simply inverts on each clock enable signal. The state of this signal then determines which symbol is outputted. Another signal was also added to provide the immediate output. This signal is only used once for the initial output, then remains unchanged afterwards to stop it from affecting the rest of the code. These changes produced the correct simulation below:

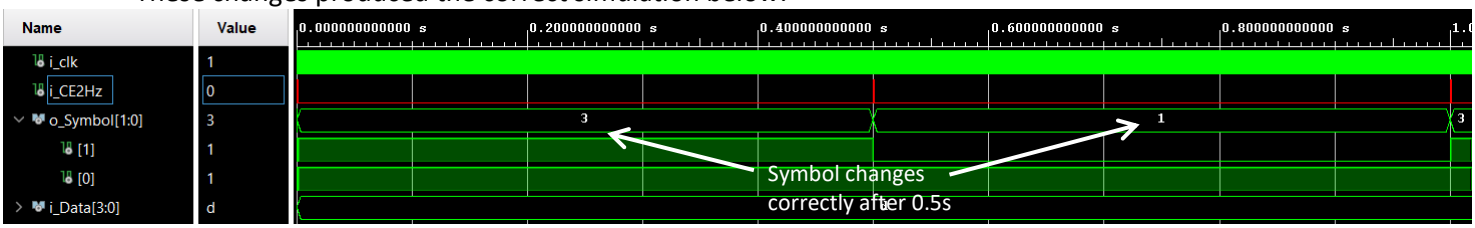


Figure 12.2: correct symbol converter output

Modulator Scheme B:

Figure 13 shows a simulation demonstrating the I and Q waveforms in response to each possible symbol input. The simulation was run for two seconds to demonstrate all 4 I/Q combinations. The reference image below shows all intended waveforms depending on symbol input and confirms that the entity is working as intended.

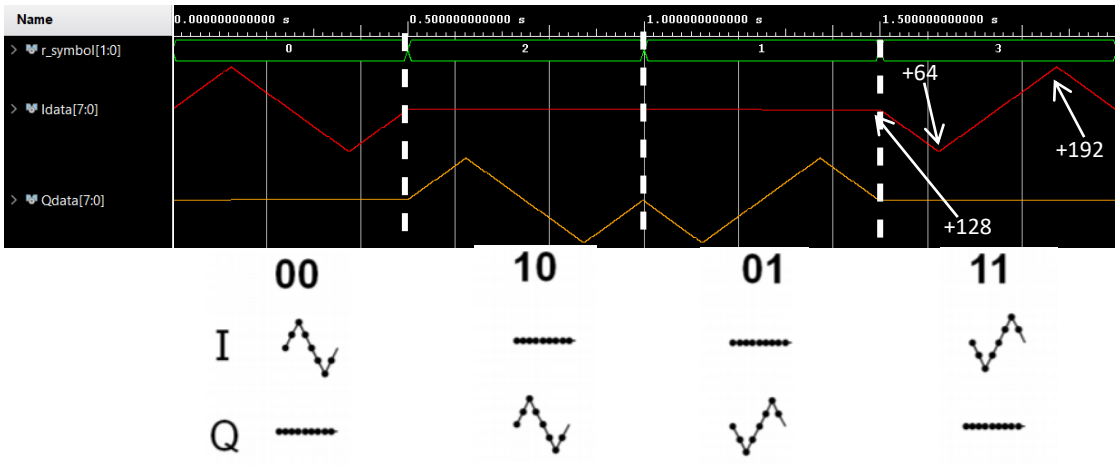


Figure 13: I/Q waves from each of the 4 symbol inputs

Error Channel/PRNG:

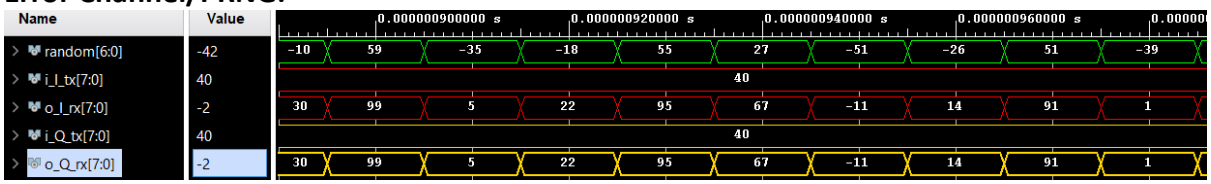


Figure 14.1: random number generator simulation

The 'random' signal produces a random signed decimal number between -64 and +64 on each clock pulse. Using a test value of 40 for the inputs from the modulator (ITX and QTX), the output values are correctly modified by adding the current random number to this. The waveforms in *fig. 14.2* show the 'clean' input I/Q waves as well as those with errors introduced at the output.

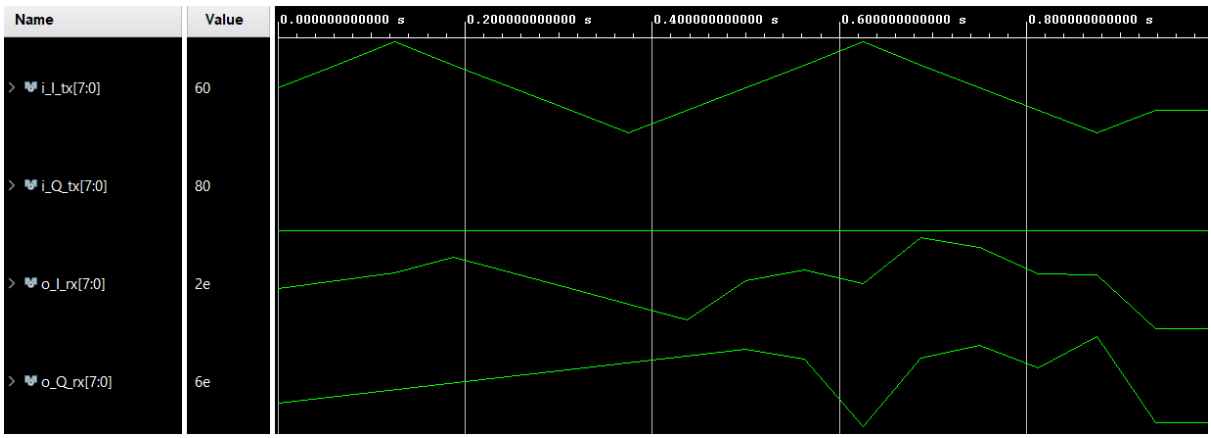


Figure 14.2: waveforms with error introduced

5 Synthesis

Synthesis on the final design was successfully completed, verifying that the model will correctly translate to hardware.

5.1 Synthesis Errors

No critical errors or fatal warnings were detected during synthesis.

6 Implementation

6.1 Pin Mapping

In order to map the required physical pins of the FPGA to the inputs and outputs of the VHDL, a constraints file is added to the project.
The inputs and outputs and their mapped pins are listed below.

I/O	Mapped to pin…
System Clock (i_C100MHz)	W5
Switches: Data Select — 1. i_SW15 2. i_SW14 3. i_SW13 4. i_SW12 Error Select — 5. i_SW11 6. i_SW10 Display Select — 7. i_SW9 8. i_SW8 9. i_SW7	As specified in coursework brief: 1. R2 2. T1 3. U1 4. W2 5. R3 6. T2 7. T3 8. V2 9. W13
Buttons: 1. i_StartStop 2. i_Reset	As specified in coursework brief: 1. U17 2. T18
7-Segment Cathodes (o_segCathodes)	(In order from cathodes 6 to 0) W7, W6, U8, V8, U5, V5, U7
7-Segment Anodes (o_segAnodes[0-3]	(In order from anodes 0 to 3) U2, U4, V4, W4

6.2 Implementation Errors

Implementation of the design produced 1 critical error:
[Common 17-55] 'set_property' expects at least one object. [T29_M2_constraints.xdc:25]

This error was the result of adding a decimal point pin map in the constraints file when it was not used at any point in the design. While this should not cause any issues with the rest of the design, it was removed from the constraints for safety.

7 Hardware Test Record

7.1 Test Record

<table>
<tr><th>Date</th><th>Test Number</th><th colspan="2">Results</th></tr>
<tr><td rowspan="3">29/01/21</td><td rowspan="3">1</td><td colspan="2">Initial hardware test, before demodulator has been implemented. Purpose of test is to ensure M2 display modes still work correctly</td></tr>
<tr><td>Working</td><td>Non-working</td></tr>
<tr><td>- All M2 modes working correctly except for 5</td><td>- M2 mode 5 (random number) not working due to using a 7-bit LFSR when the data should be 4-bit
- All M3 modes not working as they haven’t been implemented yet</td></tr>
<tr><td rowspan="3">29/01/21</td><td rowspan="3">2</td><td colspan="2">Follow-up test after fixing M2 mode 5</td></tr>
<tr><td>Working</td><td>Non-working</td></tr>
<tr><td>- All M2 modes working correctly</td><td>- All M3 modes not working as they haven’t been implemented yet</td></tr>
<tr><td rowspan="3">01/02/21</td><td rowspan="3">3</td><td colspan="2">Testing the function of M3 display modes (display select 1-7).</td></tr>
<tr><td>Working</td><td>Non-working</td></tr>
<tr><td>- Display select 5 (101) working, displaying correct waveform data</td><td>- All other M3 modes show zeros on the display as they are not correctly implemented</td></tr>
<tr><td rowspan="3">04/02/21</td><td rowspan="3">4</td><td colspan="2">Second test of M3 display modes</td></tr>
<tr><td>Working</td><td>Non-working</td></tr>
<tr><td>- Modulation scheme B modes 5 & 7 work correctly, mode 6 has small problem</td><td>- Scheme A modes not yet implemented
- Mode 6 only displays DataTX, not DataRX, because demodulator is not working</td></tr>
</table>