ECSE 323

Digital System Design

Project: The G06 Rower Machine

McGill

Razi Murshed

260516333

Muhammad Saad Malik

260417438

Table of Contents:

Introduction

The objective of this project is to design a system that computes and displays various pieces of information for a Digital Rowing Machine, such as the exercise Time, Stroke Rate, Pace (time per 500 meters), Power exerted by the rower, the amount of Calories burned, and Heart Rate. The system has been implemented on an Altera Cyclone II DE1 board, and can be attached to any rowing system that is capable of sensing a stroke being made.

Figure 1: A typical Rowing Machine[1]

System Description and Overview

Here we are introduced to our system and find out its major features and

Features

- It can reset all of the accumulated values during exercise at any point of exercise.
- The system can begin exercise after being reset as well. Note that however, before starting the exercise period the system has to be reset at first.
- It can let the user select one of the above values to be displayed at any instance.
- The system is capable of pausing during the exercise period and resuming exercise from where it left off.

[1] Image taken from: http://www.concept2.com/files/images/indoor-rowers/model-e/slides/slide.jpg

- It can compute values depicting the various aspects of the exercise period such as –
    1. Exercise time (in minutes and seconds up to 99 min, 59 sec)
    2. Total stroke count since in an exercise period (up to 9999)
    3. Boat speed (in meters per second)
    4. Total distance covered within the exercise period (in meters)
    5. Current stroke rate (strokes made in the previous 60 second interval)
    6. Pace (Time taken to get to the previous 500 meters interval)
    7. Total number of calories burned (in calories)
    8. Calorie burn rate (in kilocalories per hour)
- It can reset all of the accumulated values during exercise at any point of exercise.
- The system can begin exercise after being reset as well. Note that however, before starting the exercise period the system has to be reset at first.
- It can compute values depicting the various aspects of the exercise period such as –
    9. Exercise time (in minutes and seconds up to 99 min, 59 sec)
    10. Total stroke count since in an exercise period (up to 9999)
    11. Boat speed (in meters per second)
    12. Total distance covered within the exercise period (in meters)
    13. Current stroke rate (strokes made in the previous 60 second interval)
    14. Pace (Time taken to get to the previous 500 meters interval)
    15. Total number of calories burned (in calories)
    16. Calorie burn rate (in kilocalories per hour)

- It can let the user select one of the above values to be displayed at any instance.
- The system is capable of pausing during the exercise period and resuming exercise from where it left off.

Structure of Design and Documentation

The entire design process has been explained in a concise top down format. The .vhdl files explaining the logic used for the system are included with this package. The following pages consist of the detailed explanations about the system and its different components. We'll start off by explaining the entire system as a single circuit, with its relevant inputs and outputs. This can be thought off as the outer skeleton of the circuit. It'll give the reader a clear idea of what the system is supposed to be doing. This will be made easier as we have attached sections of a register transistor level diagram of the system. Next on the list is a description of the various major components of the system. These include all the circuits required to display the information listed above. These have been listed in a manner to correspond with the flow of data. Details about the functionality of these components have also been included, along with the necessary port

mapping to demonstrate each circuit's link to the next. The testing diagrams for these components are also included to show proper working of each part. Finally we will finish off with a short conclusion section. Here we will explain any difficulties we encountered as well as possible improvements to the design.

Overview

Here we take a look at the entire system, its inputs and outputs and how it functions during exercise. Figure 1 shows us a block diagram of our entire system –

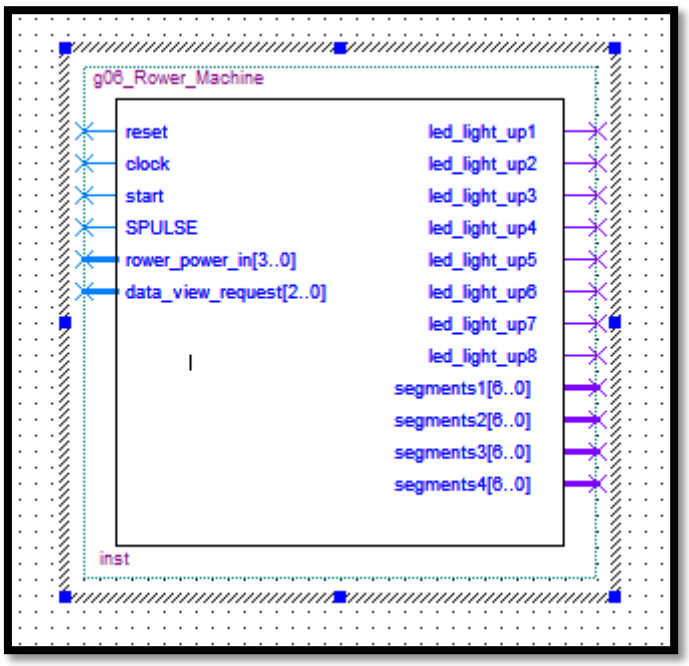

Figure 2: Block Diagram of the Entire System

The device has the following input values in order to compute and produce the information presented above:

- Reset: This button is capable of resetting the entire system to initialize a new exercise period.
- Clock: Being a synchronous system, unless there is a reset, all events occur on a clock edge.
- Start: This input is used in order start an exercise period from a reset, pause exercise and resume exercise again.
- SPULSE: This input takes in the strokes made by the user on the device in order to compute the required output values.
- Rower_power_in: This takes in the rower power exerted by the user on the device in order to compute the necessary output values.
- Data_view_request: This input allows the user to select a value to be displayed.

It also consists of the following output values after all operations on the input have been done –

- Led_light_up (from 1 to 7): These are outputs that light up LED's indicating the values being displayed.
- Segments (from 1 to 4): These outputs are connected to the 7 segment displays on the circuit board in order to display the required values.

The system that results in the block diagram above is tested in our Testing of the System section under Total System Test.

How the System Functions

For the ease of viewing in this report, we have divided the system into 3 major parts, the computation of values, the adjustment for display, and the display unit itself. To illustrate data flow in the system refer to flowchart displayed below.

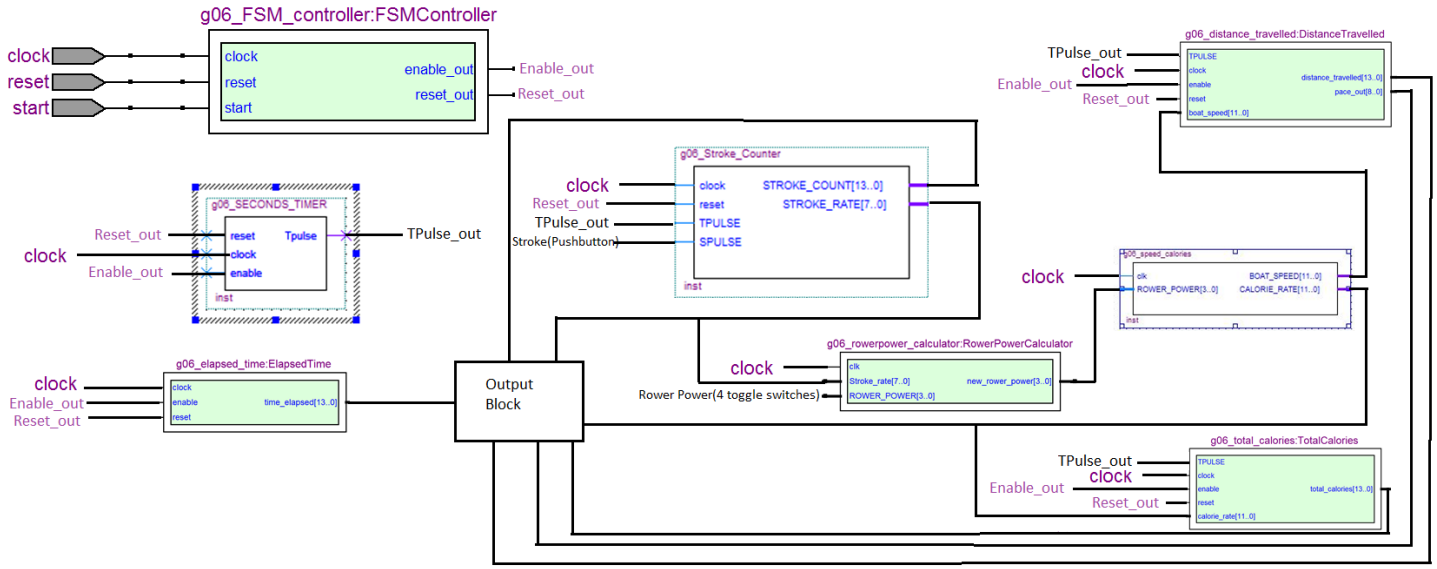

Figure 3: Flow chart describing flow of system

The computation unit shown below is responsible for taking in the inputs clock, reset, start, SPULSE and rower_power_in to the system and calculating the required values that need to be outputted. It consists of the components g06_FSM_controller, g06_seconds_timer, g06_elapsed_time, g06_stroke_counter, g06_rowerpower_calculator,

g06_speed_calories, g06_total_calories and g06_distance_travelled. The detailed functioning of these components are listed in the testing section of this report.

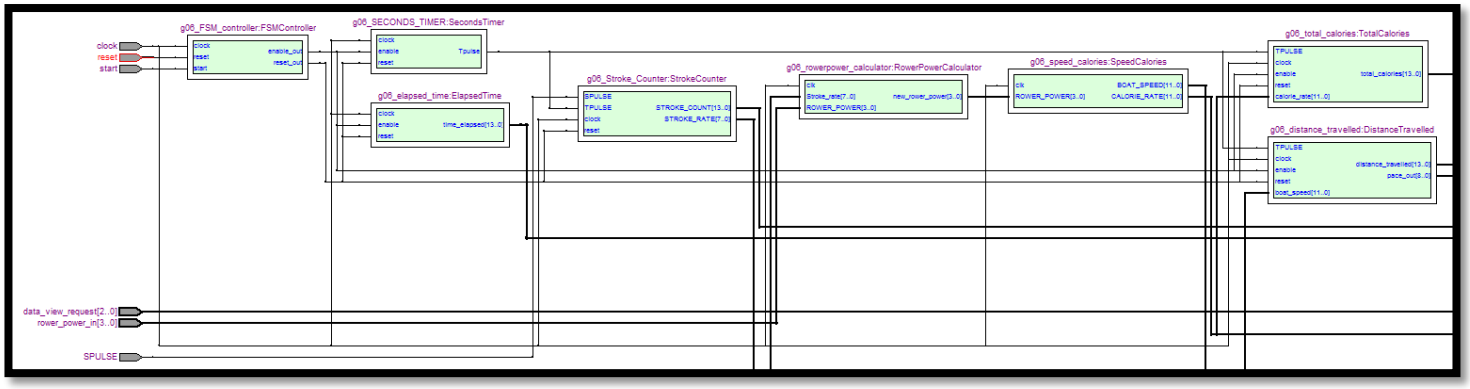

Figure 4: The computation unit

The adjustment for display section shown below takes in the values calculated and formats them for the display unit. It concatenates all values less than 14 bits with 0's for Most Significant Bits (using multiplexers) and sends them to a binary to binary coded decimal (BCD) converter (g06_ 14_bit_binary_to_BCD) which produces the integer numbers to be displayed.

Finally, the display unit itself takes in the data_view_request input in order to determine the value the user wishes to see and displays it through 4 BCD7 units. It also indicates the values being shown using the LED's present on the board.

NOTE: All components that are mentioned above will be discussed in further detail in the testing section.

Flow Status	Successful - Thu Dec 04 14:08:02 2014
Quartus II 64-Bit Version	9.1 Build 350 03/24/2010 SP 2 SJ Full Version
Revision Name	g06_Lab5
Top-level Entity Name	g06_Rower_Machine
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Met timing requirements	Yes
Total logic elements	728 / 18,752 (4 %)
Total combinational functions	728 / 18,752 (4 %)
Dedicated logic registers	250 / 18,752 (1 %)
Total registers	250
Total pins	47 / 315 (15 %)
Total virtual pins	0
Total memory bits	192 / 239,616 (< 1 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

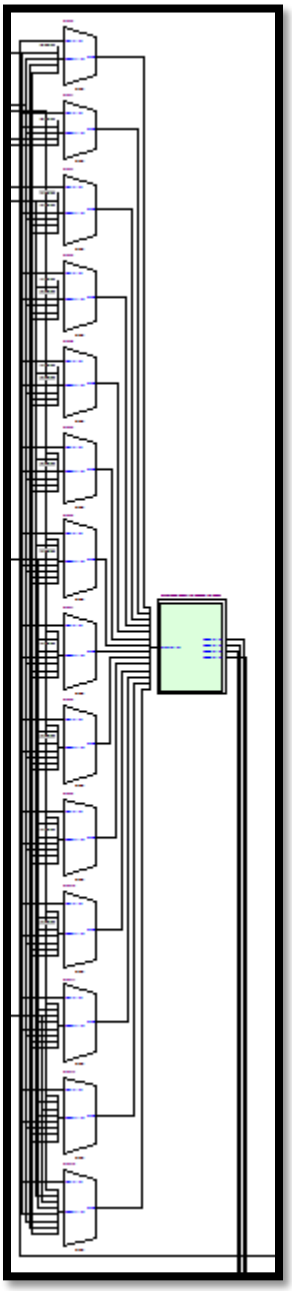

Figure 5: Adjustment for display unit

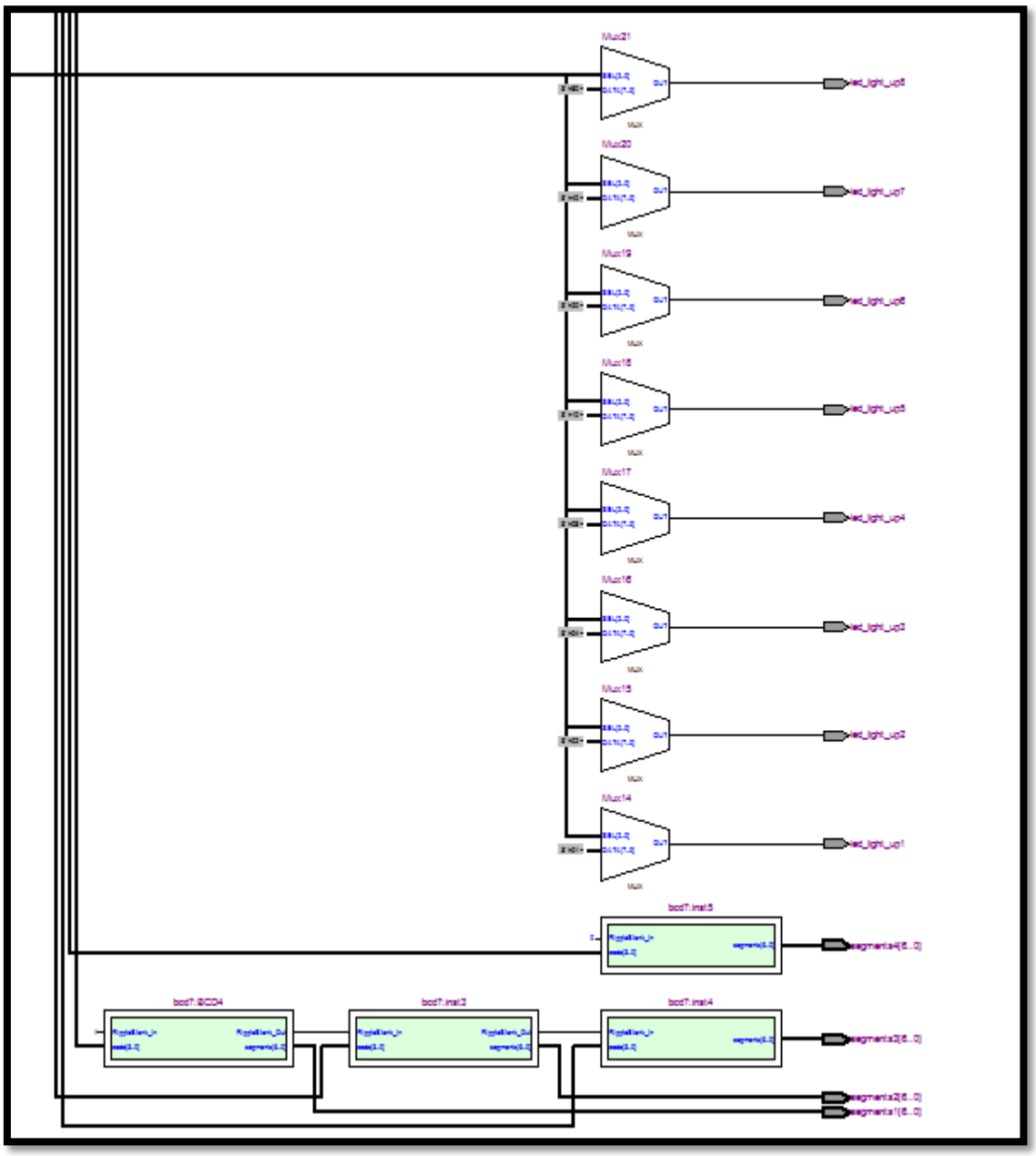

Figure 6: Display unit

User Interface

The board: The different switches to be used by the user are indicated in the figure below.

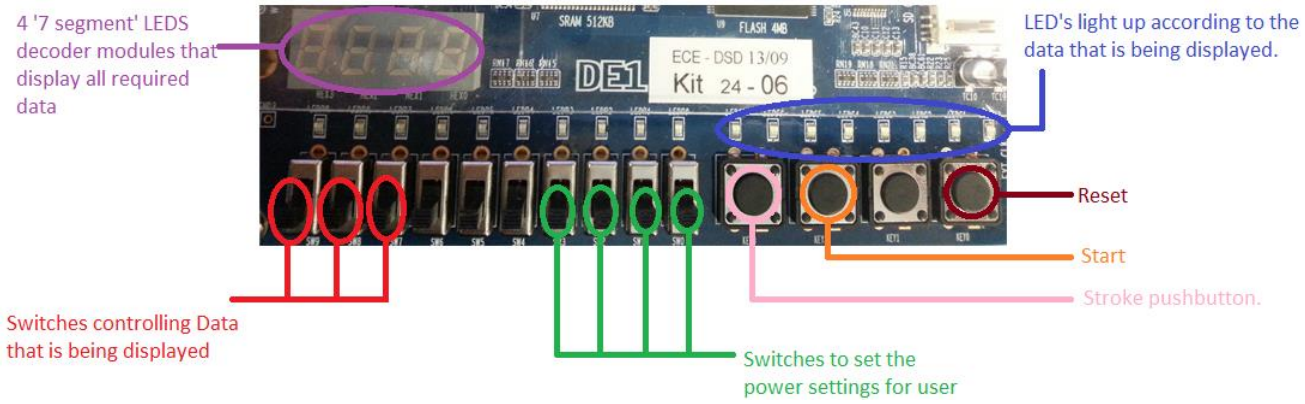

Figure I-1: View of the Cyclone 2 board with regards to the user interface.

Functionality of Reset:

Reset: Sets all values of circuit to 0.

Figure I-2: Functionality of reset displayed for Time elapsed, Boat speed and rower pace.

Functionality of remaining items:

Switches controlling display:

000 = Total time Elapsed & Green LED 1

Before start = 1

As displayed below value stays at 0 until start goes high.

Figure I-3: Time elapsed before start.

After start =1:

As displayed in Figures below the total time elapsed is being displayed.

Figure I-4: Functionality of seconds in time elapsed.

In the following figures functionality of the minutes display is being shown.

Figure I-5: Functionality of minutes and seconds display in time elapsed.

001 = Total Stroke count & Green LED 2

The figures below display different values of the total stroke count depending upon how many times the Stroke pushbutton is pressed.

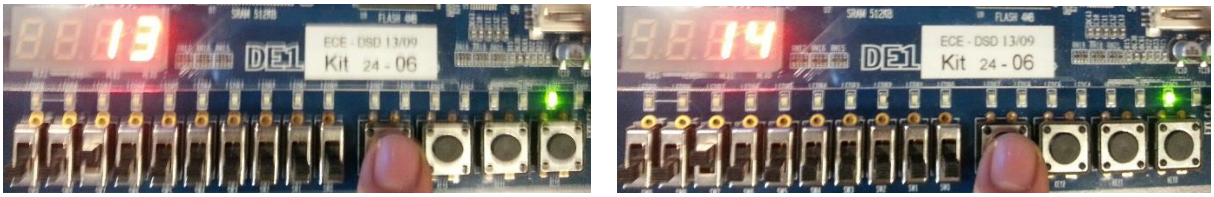

Figure I-6: Functionality of Stroke counter and dependence on SPULSE.

010 = Distance Travelled & Green LED 3

The figures below display different values of the total distance travelled.

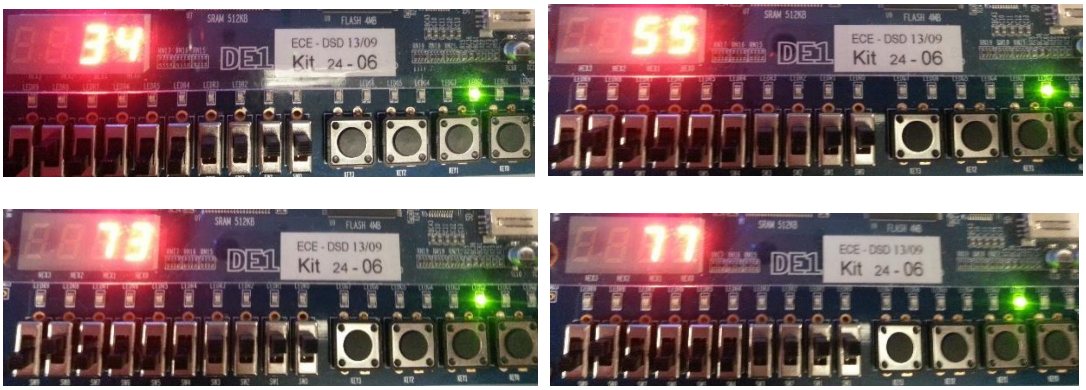

Figure I-7: Functionality of distance travelled along with rower power variation.

011 = Total Calories Burned & Green LED 4

The figures below display different values of the total calories burned dependant on the number of strokes.

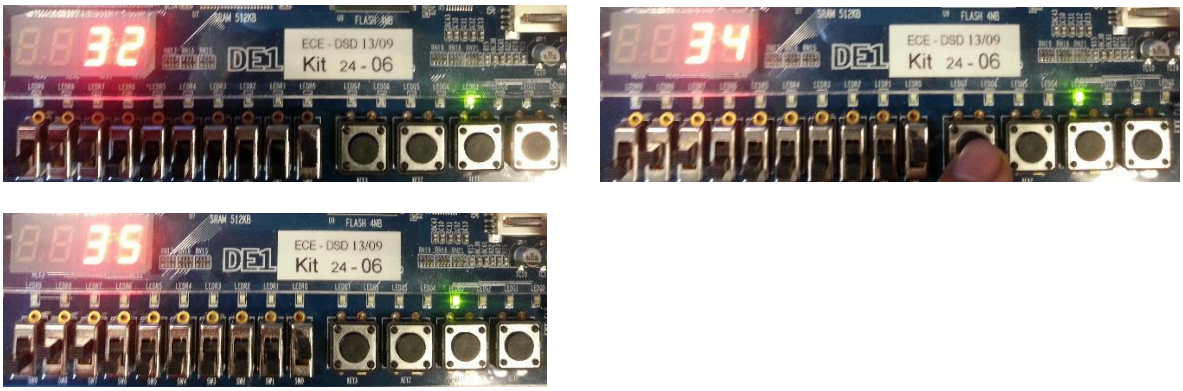

Figure I-8: Functionality of Total calories burned, along with dependence on power and SPULSE.

100 = Stroke rate & Green LED 5

The figures below display the stroke rates in two different minutes.

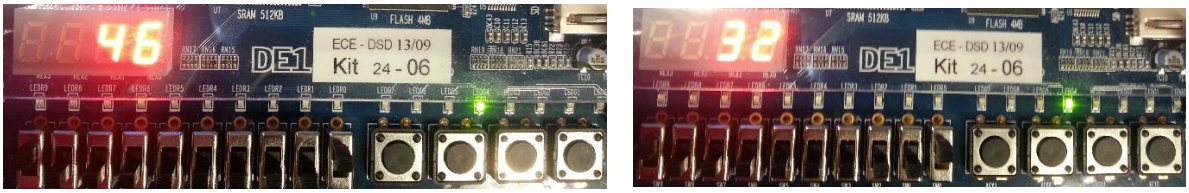

Figure I-9: Functionality of stroke rate in two different minutes.

101 = Boat Speed & Green LED 6

The figures below display the boat speed at different intervals of time

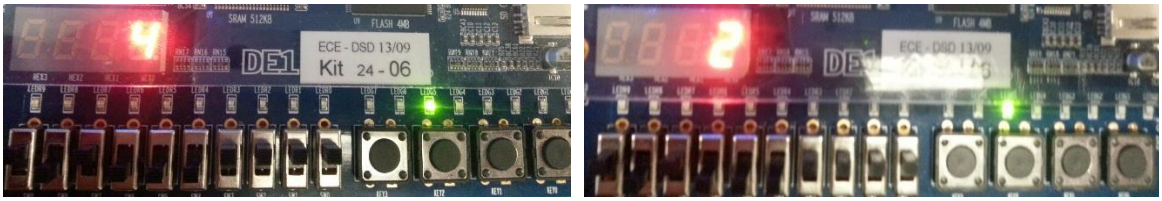

Figure I-10: Functionality of Boat speed along with varying power value.

110 = Calorie Rate & Green LED 7

The figures below display the rate at which the user is burning calories.

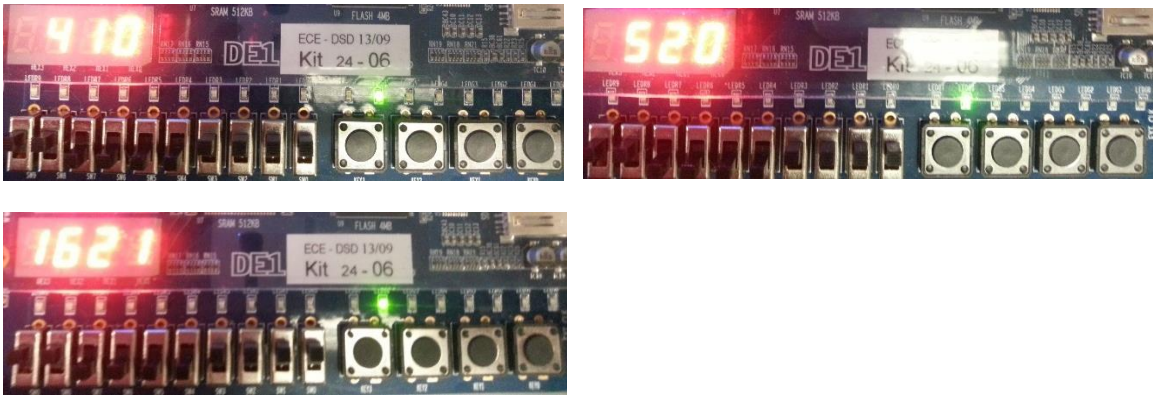

Figure I-11: Functionality of calorie burn rate along with different power settings.

111 = Rower Pace & Green LED 8

The figure below display the rower pace at two different intervals in time.

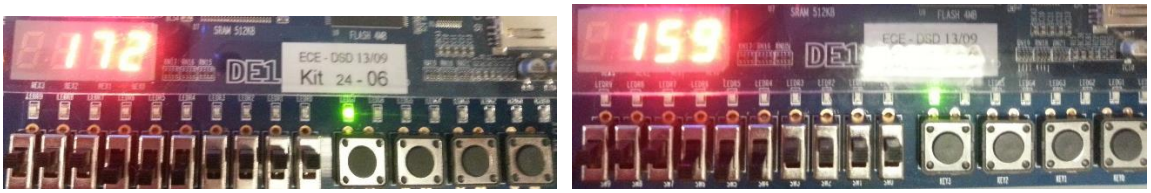

Figure I-12: Functionality of rower pace in two consecutive minutes, with varied power settings.

Testing of the System

In this section of the report we detail the functioning of each and every component we have created and our test of the entire system. Our approach to testing was to create every component and carry out a functional and timing simulations of it by assuming that whatever inputs a component might be getting from another is a correct output from the source. We then assembled all the components together into a test bed and uploaded the program into the circuit board in order to carry out hardware testing. A few results of these hardware tests are available through the user interface section of this report and the simulations are attached at the end of each component.

Components and their Tests

The following components and their tests are listed below -

- g06_FSM_controller
- g06_seconds_timer
- g06_elapsed_time
- g06_stroke_counter
- g06_rowerpower_calculator
- g06_speed_calories, g06_total_calories
- g06_distance_travelled
- g06_14_bit_binary_to_BCD
- BCD7

G06_FSM_controller

Component Description: The overall control of the rowing machine is implemented with a Finite State Machine (FSM). This simple FSM monitors the state of the reset and start buttons and controls the operations of the components within the system by issuing different conditions for different states of the machine. The initial idea for the controller is shown in the state diagram below –

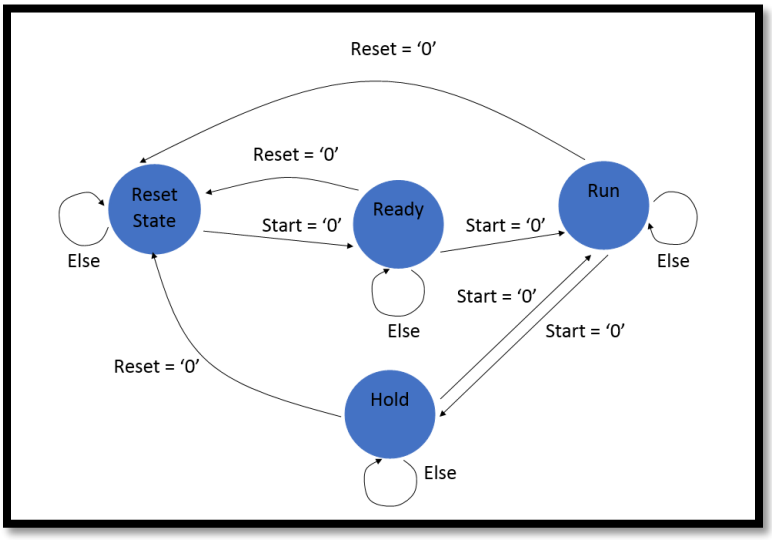

Figure 7: Theoretical Design

The actual implementation after the synthesis of the VHDL code is shown below.

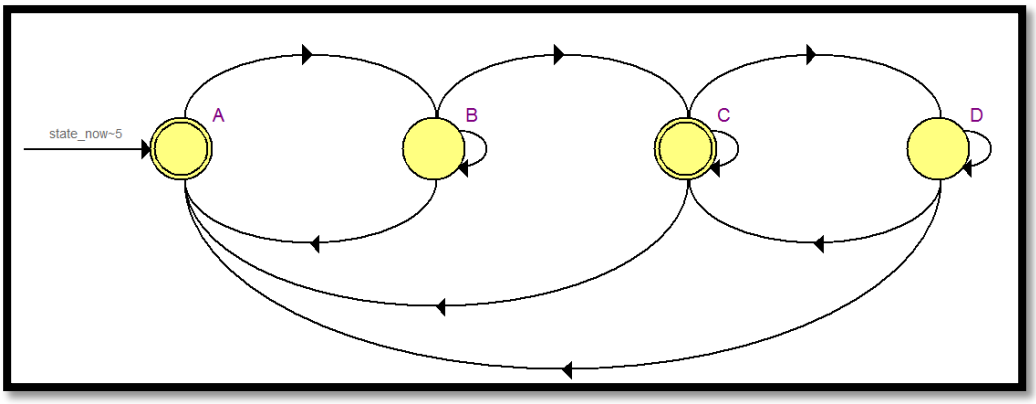

Figure 8: Implemented FSM

In the above figure A is the reset state where the circuit goes after a reset has been asserted, B is the Ready state that indicates that the circuit is ready to run, C is the run state when the exercise period starts and takes place and finally D is the Hold state that holds the values accumulated during a pause in exercise. The state table below describes the

transition from one state to another. **NOTE:** For the design it has been taken into account that the push buttons are active low and to prevent recording of multiple readings from the same input has been prevented by the use of a flag.

	Future State			
Current State	Reset = '0'	Reset = '1'	Start = '0'	Start = '1'
A	A	A	B	A
B	A	B	C	B
C	A	C	D	C
D	A	D	C	D

Figure 9: State table for controller module

Function:

A block diagram for the circuit is included below to illustrate the inputs and outputs to the system. A schematic diagram below, displays the transistor level logic in the circuit.

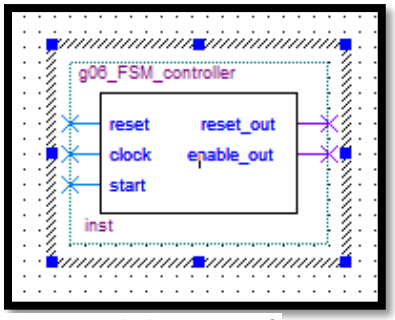

Figure 10: Block Diagram of FSM controller

Inputs:

- *Clock*: Universal clock input to all components in circuit set at 50 MHz
- *Reset*: A single bit input when low takes machine to reset state.
- *Start*: A single bit input when low stars the circuit from reset or pause.

Outputs:

- *Reset_out*: A single bit output that resets all the components in the circuit.
- *Enable_out*: Enables the components with an enable input to compute required outputs to the circuit.

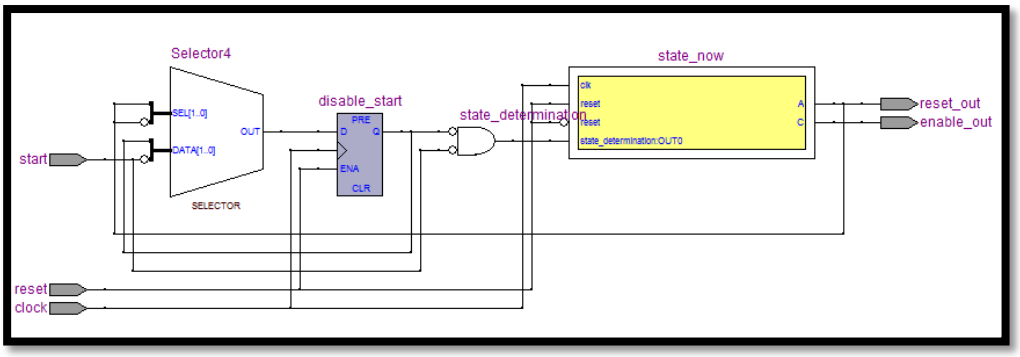

Figure 11: Schematic of controller

Testing:

The results of the tests for the FSM controller can be seen in the vector waveform file below. We can see the effect of a reset on the system at around 320ns in to the simulation. When a rest is signalled all states go to A. After the reset the system goes to state A from which it moves to sate B in the next clock cycle. The effect of a start signal is also a few times throughout. We can see that a start from state B takes the system to state C. A start in state C takes it to state D.

Another start in state D takes it back to C. We can also observe that the circuit does not detect a start signal several times even if it is held high for several clock cycles.

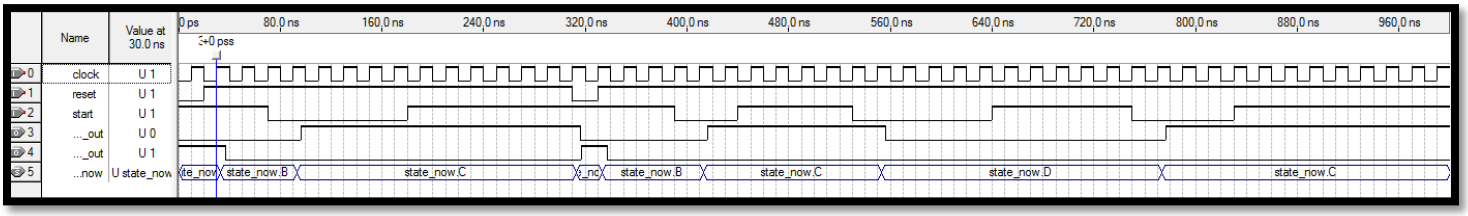

Figure 12: Timing diagram for FSM controller

G06_seconds_timer

Component Description:

Function: The g06_seconds_counter is a frequency divider circuit. It divides the input Master clock frequency, *CLOCK,* by a certain number, 49999999 in our case, and outputs a pulse signal, *TPULSE*, with a frequency of 1 Hz. The output is used as a reference signal in all circuits that require any sampling to be done on a 'per second' or 'per minute' basis. A block diagram for the circuit is included below to illustrate the inputs and outputs to the system. A schematic diagram below displays the transistor level logic in the circuit.

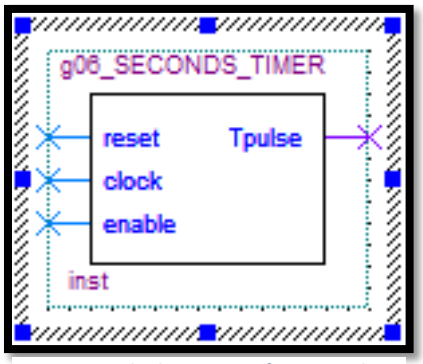

Figure 13: Block Diagram for Seconds Timer

Inputs:

- *Clock*: Universal clock input to all components in circuit set at 50 MHz.
- *Reset*: A single bit input when high resets all values to '0'. This is connected to the reset_out from the g06_FSM_controller.
- *Enable*: A single bit input when high initializes the circuit to be active. This is connected to the enable_out from the g06_FSM_controller.

Outputs:

- *TPULSE*: The required single bit pulse signal with frequency 1 Hz. Is used as input to any circuit that requires data to be sampled each second.

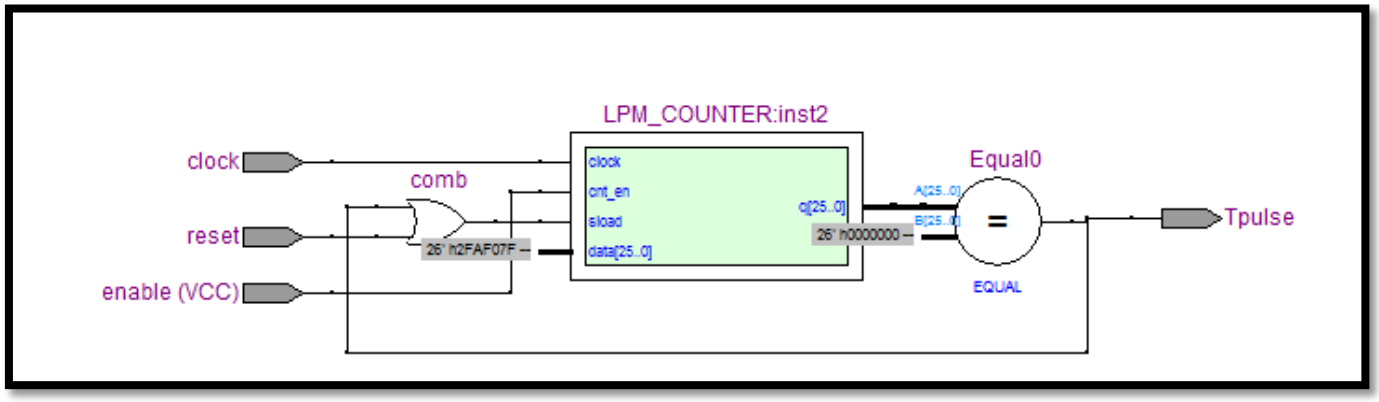

Figure 14: Schematic of seconds timer

Testing:

A timing simulation was run with a reduced constant frequency divider value to give us an accurate simulation in a lesser amount of time. The value used for testing was 1000. The clock frequency was 20 ns and by division by a 1000 we obtain a pulse every 20 us.

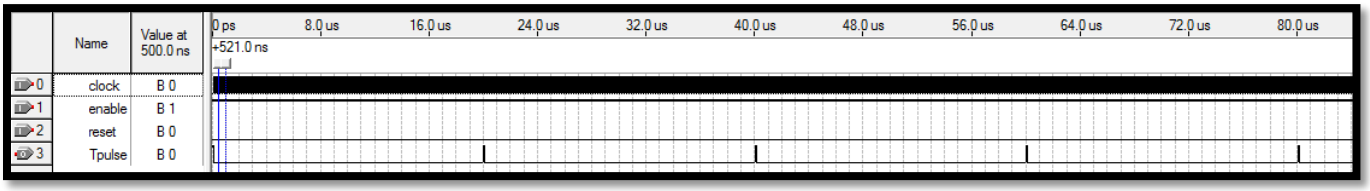

Figure 15: Seconds timer test

G06_elapsed_time

Component Description:

Functionality: Displays total time elapsed since the enable signal goes high. Counter will stop if enable goes low and will reset to '0' if reset goes high. The output is displayed on the 4 LED's on the board; the first 2 LED's display the minutes elapsed and the last 2 led's displaying the seconds elapsed. The minutes and seconds display is split by allowing the seconds to upto 59, and then adding 100 to the value of the total number upon reaching 60. This ensures the value of the minutes increase by 1 every sixty seconds.

A block diagram, for the circuit is included below to illustrate the inputs and outputs to the system. A schematic diagram below displays the transistor level logic in the circuit.

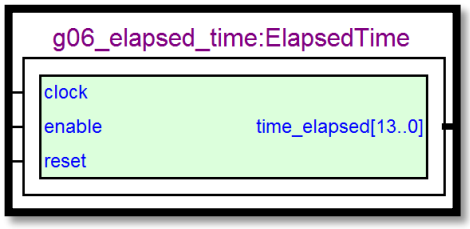

Figure 16: Block diagram of elapsed time

Inputs:

- *Clock*: Universal clock input to all components in circuit set at 50 MHz.
- *Reset*: A single bit input when high resets all values to '0'. This is connected to the the reset_out from the g06_FSM_controller
- *Enable*: A single bit input when high initializes the circuit to be active. This is connected to the enable_out from the g06_FSM_controller.

Outputs:

- *Time_Elapsed*: A 14 bit output which displays the total time elapsed since enable has been active. Displays values on the LED's from 0 seconds till 99 minutes 59 seconds.

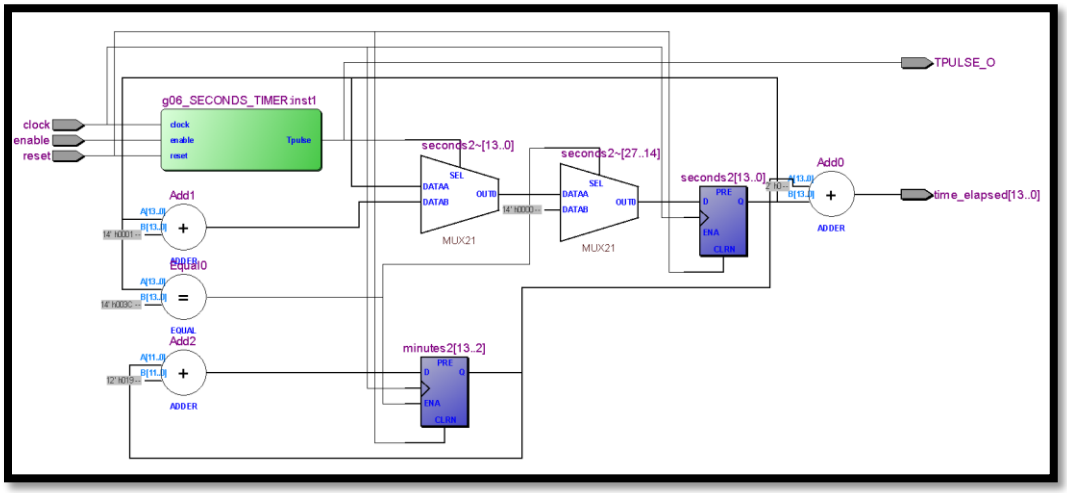

Figure 17: Time elapsed schematic

Testing:

The following two waveform files show that every TPULSE increments seconds by one and after every 60 seconds a minute pulse can be seen.

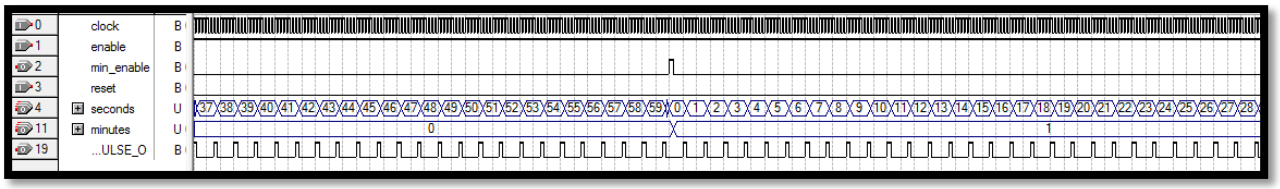

Figure 18: Time Elapsed Test

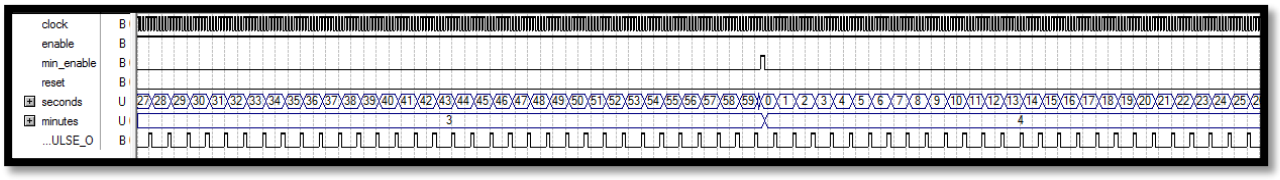

Figure 19: Time elapsed test

g06_stroke_counter

Component Description:

Function: The g06_stroke_counter is a circuit that counts the total strokes made by the user, *STROKE_COUNT*, as well as calculating the strokes per minute, *STROKE_RATE*. The g06_stroke_counter takes in a general clock, a TPULSE and an SPULSE. The TPULSE is the output of the g06_seconds_timer that outputs a signal every second in order to measure the number of strokes carried out. At the rising edge of every clock cycle, if any SPULSE is set, the STROKE COUNT value is incremented by one. The stroke rate output samples the number of SPULSES over 60 seconds or TPULSES in our case and outputs the Stroke rate over the last minute.

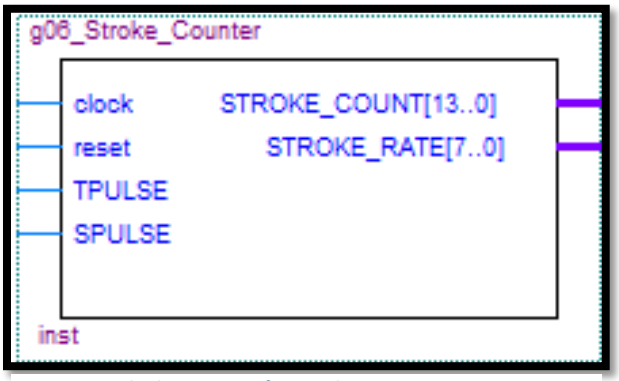

Figure 20: Block Diagram for Stroke Counter

Inputs:

- *Clock*: Universal clock input to all components in circuit set at 50 MHz.
- *Reset*: A single bit input when high resets all values to '0'. This is connected to the the reset_out from the g06_FSM_controller.
- *TPULSE*: A single bit input that gives us a pulse with a time period of 1 second used as reference for all per minute

calculations. This is connected to the TPULSE output from the g06_seconds_timer circuit.

- *SPULSE*: A single bit input that records a stroke being made. This is connected to a switchbutton, the pressing of which signifies a stroke being made.

Outputs:

- *STROKE_COUNT*: A 13 bit output that gives the total strokes made by user. The output is displayed in integer format using the 4 LED's on the board.
- *STROKE_RATE*: A 7 bit unsigned binary output that gives a 'per minute' rate for the *STROKE_COUNT.*

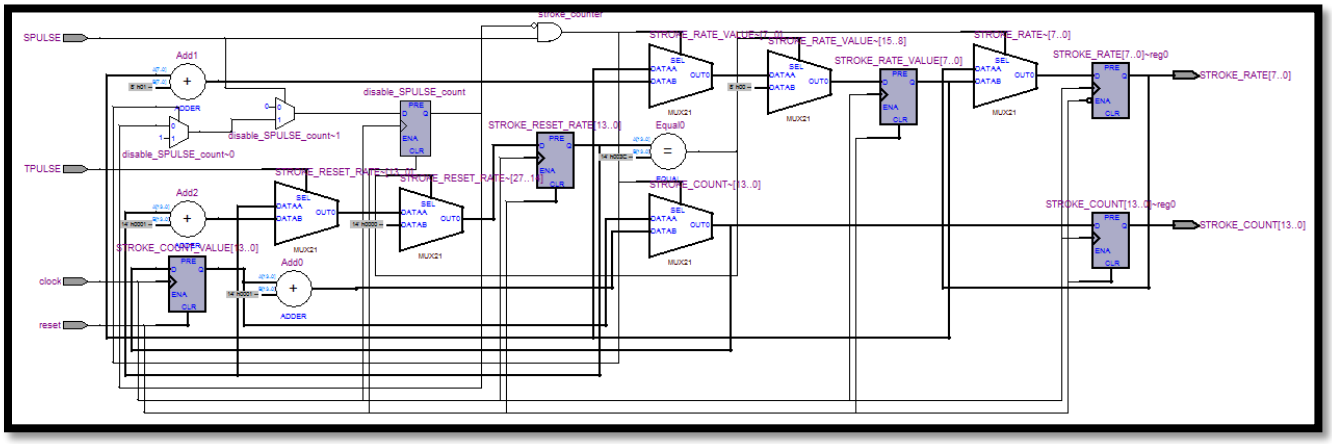

Figure 3.2: Schematic Diagram for g06_stroke_counter

Testing:

Functional Simulation: A functional simulation of the circuit was done with fast pulse signals for *TPULSE* and *SPULSE* signals to save on time. Below are snapshots of the simulation for different possible cases.

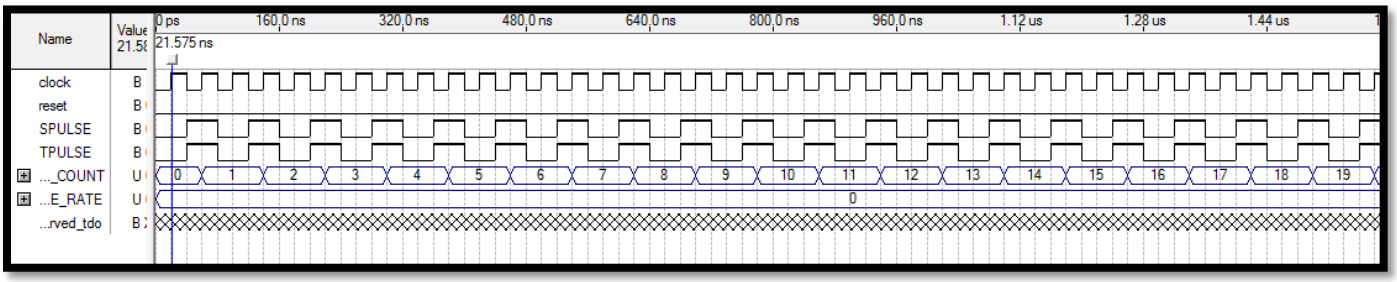

Figure 21: Functional simulation for g06_stroke_counter with a SPULSE having equal frequency as TPULSE.

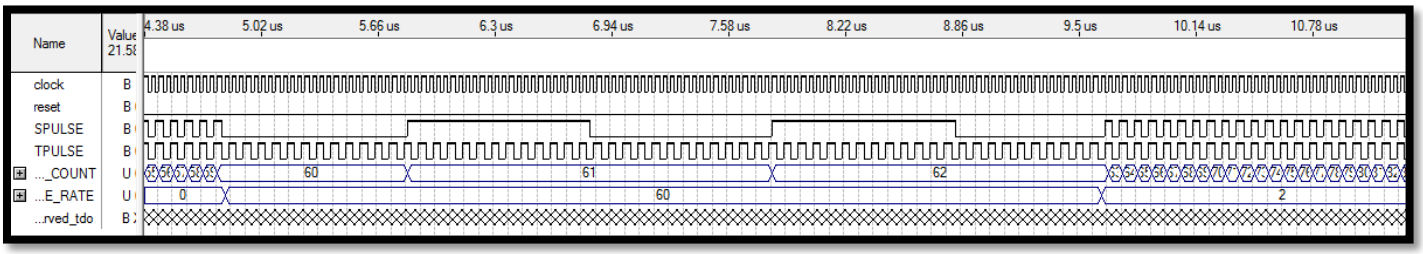

Figure 22: Functional simulation for g06_stroke_counter with a SPULSE having lower frequency than TPULSE

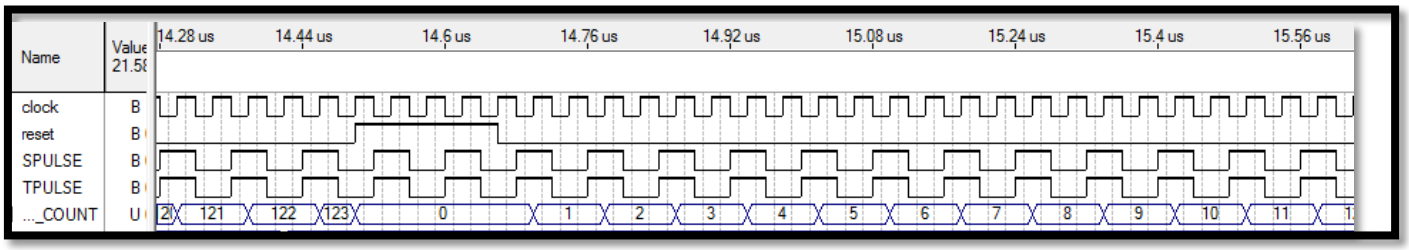

Figure 23: Functional simulation for g06_stroke_counter showing functionality of reset data line.

G06_rower_power_calculator

Component Description:

Functionality: Computes the average power exerted by the user in 1 second. This is done by scaling the rower power selected in the system with the user's stroke rate. This gives a better approximation of the user's power exertion.

The output is used in the computation of the user's calorie burn rate and the boat speed in the g06_speed_calories circuit.

A block diagram for the circuit is included below to illustrate the inputs and outputs to the system. A schematic diagram below displays the transistor level logic in the circuit.

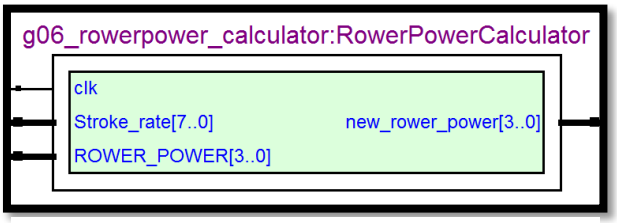

Figure 24: Rower Power Calculator

Inputs:

- *Clock*: Universal clock input to all components in circuit set at 50 MHz
- *Stroke_rate*: An 8 bit unsigned input that represents the user's stroke rate. It is used in the computation of the average power

exerted by the user. This is connected to the STROKE_RATE output of the g06_Stroke_counter.

- *ROWER_POWER*: A 4 bit unsigned input which is the value power setting at which the user has set the machine. It is used in the computation of the average power exerted by the user. This is connected to 4 switches on the board, with each switch representing a Bit. Can be set between 0 and 480 Watts, with increments of 32 watts.

Outputs:

- *New_rower_power*: A 4 bit output which is the value of the average power exerted by the user in a second. This is used in computations in the g06_speed_calories circuit.

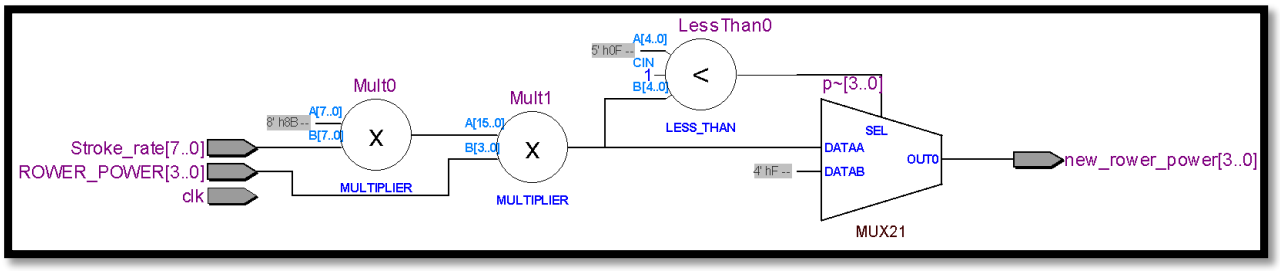

Figure 25: Schematic of Rower Power Calculator

G06_speed_calories

Component Description:

Functionality: The g06_speed_calories circuit calculates the Boat_speed, and Calorie_rate, given the Rower_power. This is done through the use of two formulas used to calculate the requisite outputs given a certain input.

The outputs are displayed using the 4 LED's on the Altera board. As well as in computations in further circuits.

A block diagram for the circuit is included below to illustrate the inputs and outputs to the system. A schematic diagram below displays the transistor level logic in the circuit.

Figure 26: Block diagram for g06_speed_calories.

Inputs:

- *Clock*: Universal clock input to all components in circuit set at 50 MHz.
- *Rower_power*: A 4 bit unsigned binary input used to specify the average power being exerted by the user. This is used in the computation of both the outputs. This is connected to the new_rower_power output of g06_rower_power_calculator.

Outputs:

- *Boat_Speed*: A 12 Bit unsigned binary output used to display the speed of the boad given a certain power level. This is calculated using the formula: Boat_speed= 0.7095 * cuberoot(Rower_power). This is displayed on the board using the 4 LED's as well as in computations in other circuits.

- *Calorie_rate*: A 12 Bit unsigned binary output used to display rate at which the rower is burning calories given a certain power level. This is calculated using the formula Calorie_rate = 3.4415*Rower_Power +300. This is

displayed on the board using the 4 LED's as well as in computations in other circuits.

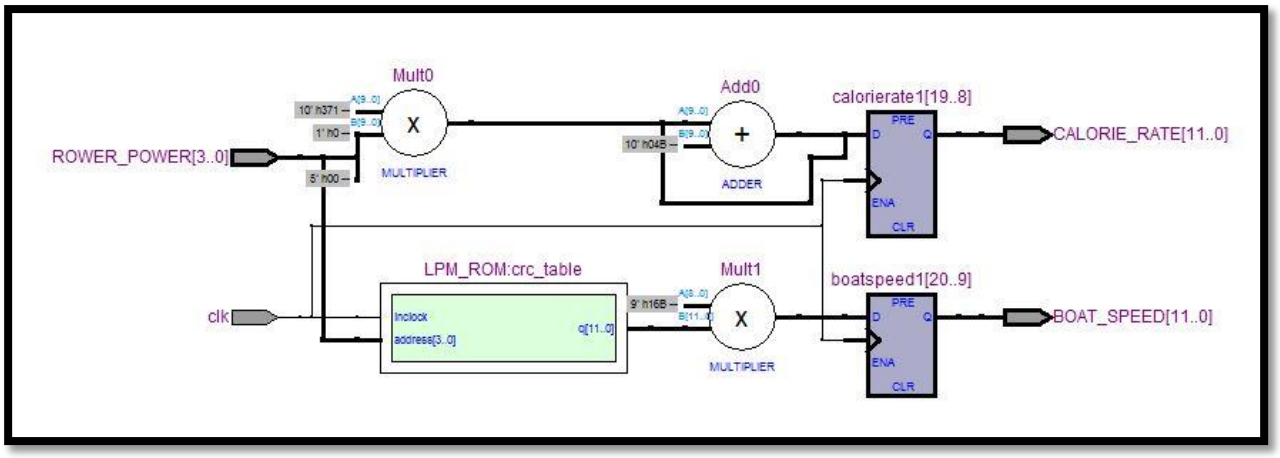

Figure 27: Schematic diagram for g06_speed_calories.

Testing:

We have done a simulation of the circuit above and have found the results corresponding to the rower power and have found the results to match that of the calculations provided by the formulas.

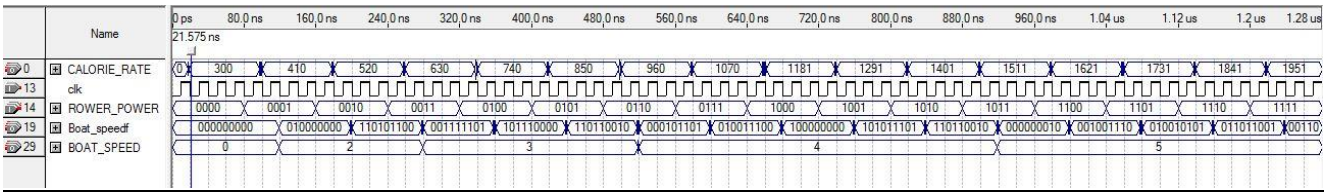

Figure 28: Timing simulation for 16 different input values to g06_speed_calories.

g06_distance_travelled

Component Description:

Functionality: Displays total distance travelled by the user depending on the speed at which the user is rowing. This is done by updating the distance travelled by the user each second and displaying the total distance at any point in time.

The circuit also displays the pace at which the user is rowing. The pace is the time it takes the user to row 500 metres.

The output is displayed on the 4 LED's on the board. The results are displayed in integer format.

A block diagram for the circuit is included below to illustrate the inputs and outputs to the system. A schematic diagram below, displays the transistor level logic in the circuit.

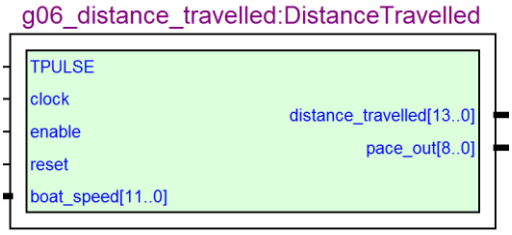

Figure 29: Block Diagram of distance travelled

Inputs:

- *Clock*: Universal clock input to all components in circuit set at 50 MHz
- *Reset*: A single bit input when high resets all values to '0'. This is connected to the reset_out from the g06_FSM_controller.
- *Enable*: A single bit input when high initializes the circuit to be active. This is connected to the enable_out from the g06_FSM_controller.
- *TPULSE*: A single bit input which goes high each second. It is used to sample the boat speed at each second and then calculate the distance travelled as well as the pace. This represents the time parameter of the distance

equation. When TPULSE goes to high, the value of the distance travelled is updated with the boat speed. This ensures that value of distance is updated each second. After value of distance has reached 500 metres the total

number of active edges of Tpulses determines the pace. This is connected to the TPULSE output from the g06_seconds_timer circuit.

- *Boat_speed*: A 14 bit unsigned input which is the value of the speed at which the user is rowing. It is used to calculate the distance travelled as well as the pace. This represents the speed parameter of the distance equation. This is connected to the Boat_speed output from the g06_speed_calories circuit.

Outputs:

- *Distance_Travelled*: A 14 bit output which displays the total distance travelled by the user since the circuit has been enabled. Displays values on the LED's in meters from 0 till 9999 meters.
- *Pace_out:* An 8 bit output which displays the pace at which user is rowing. It is a display of the amount of time it takes for the user to row 500 meters. Displays values on the LED's from 0 to 255.

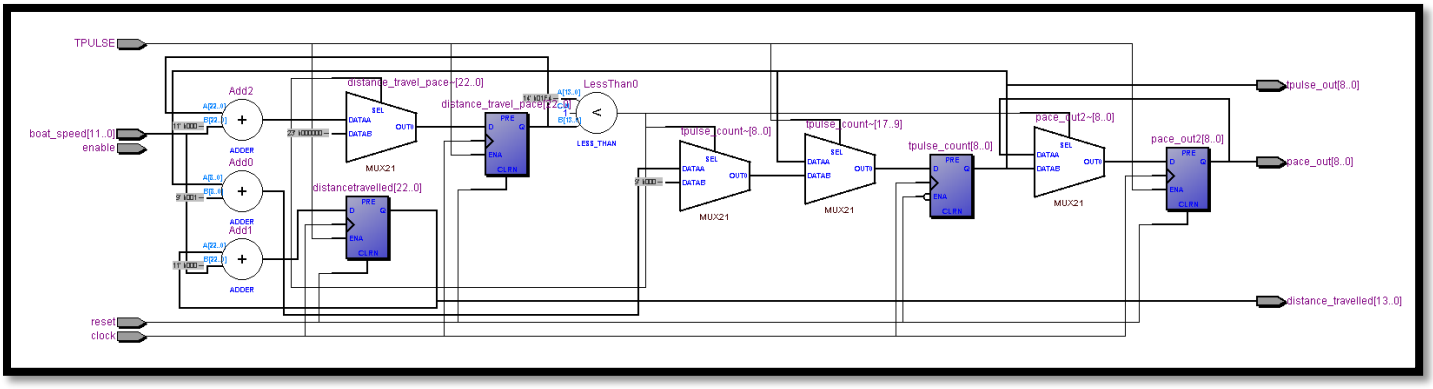

Figure 30: Schematic of Distance Travelled

Testing: It can be seen in the diagrams below that the distance is incremented by the speed at each second and every 500 metres the pace is updated.

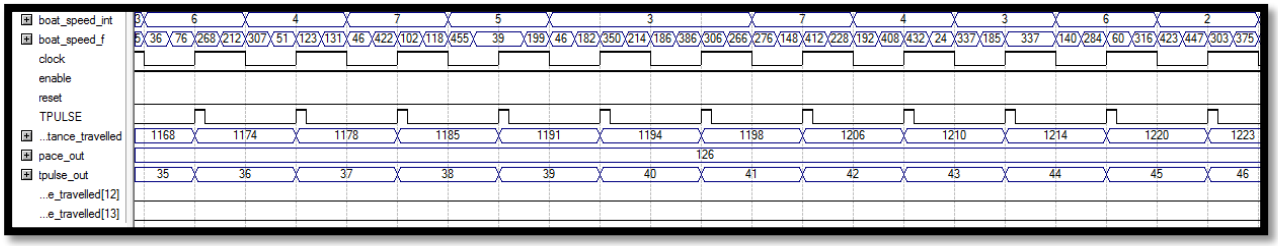

Figure 31: Distance Travelled test

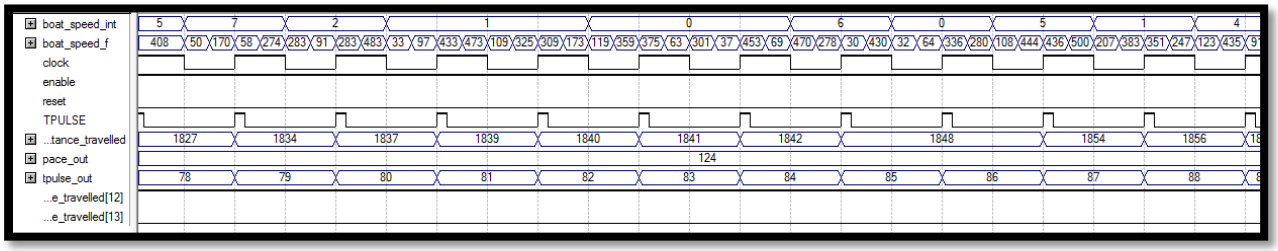

Figure 32: Waveform showing distance travelled and pace

G06_total_calories

Component Description:

Function: Displays total number of calories burned by the user after the enable signal has gone high. This is done by using the calorie burn rate as an input to the circuit. This value is then multiplied by a constant, c= 0.28, to give the final required total number of calories burned.

The output is displayed on the 4 LED's on the board. The results are displayed in integer format.

A block diagram for the circuit is included below to illustrate the inputs and outputs to the system. A schematic diagram below, displays the transistor level logic in the circuit.

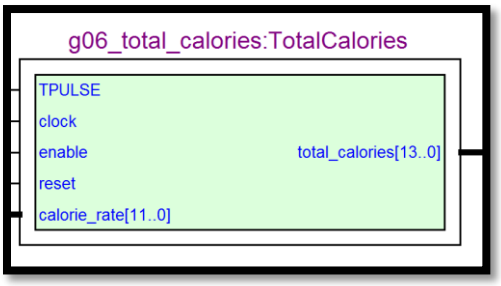

Figure 33: Schematic of Total Calories

Inputs:

- *Clock*: Universal clock input to all components in circuit set at 50 MHz.
- *Reset*: A single bit input when high resets all values to '0'. This is connected to the the reset_out from the g06_FSM_controller.
- *Enable*: A single bit input when high initializes the circuit to be active. This is connected to the the enable_out from the g06_FSM_controller.
- *TPULSE*: A single bit input which goes high each second. It is used to sample the calorie rate each second and then compute the resultant total calories. This is connected to the TPULSE output from the g06_seconds_timer circuit.
- *Calorie_rate*: A 12 bit unsigned input which is the value of rate at which the user is burning calories. Used in the calculation of total calories burned by user. This is connected to the Calorie_rate output from the g06_speed_calories circuit.

Outputs:

- *Total Calories*: A 14 bit output which displays the total calories burned by the user since the circuit has been enabled. Displays values on the LED's in 0 till 9999 Calories.

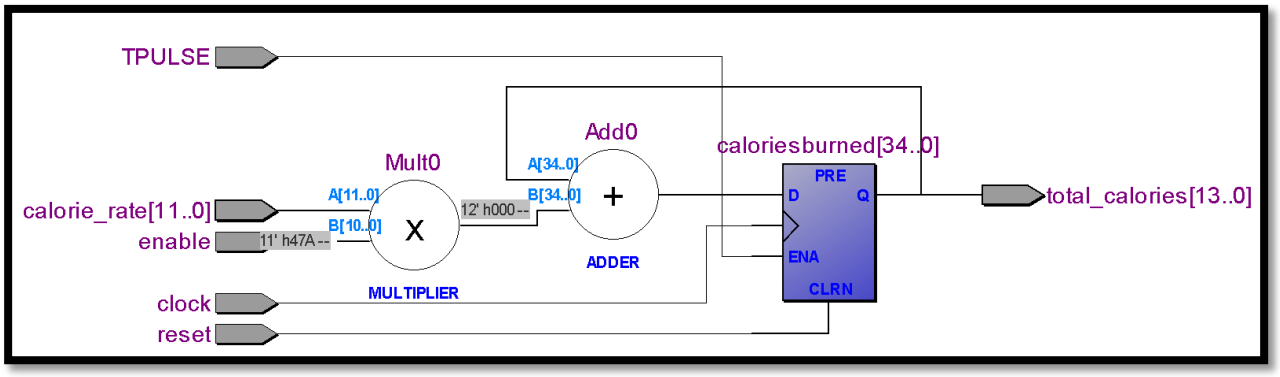

Figure 34: Schematic of Total Calories

Testing:

The waveform below depicts how the total calories increase with a constant calorie rate (maximum in this case) as the time progresses.

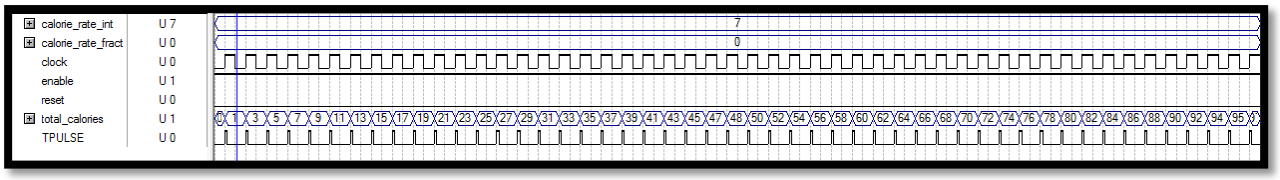

Figure 35: Schematic showing total calories in relation to the calorie rate

G06_14_bit_binary_to_BCD

Component Description:

Function: The g06_binary_to_BCD circuit is used to determine the binary coded decimal of a 14 bit binary number input to it. The least 4 significant bits have simple binary to BCD representations. We then add 3 to the number before bringing in the next bit if and only if the number is greater than 4. We continue doing this, shifting in bits until we have run out of bits. The resultant output is the BCD equivalents of the input number

The output is displayed on the 4 LED's on the board. The results are displayed in integer format.

A block diagram for the circuit is included below to illustrate the inputs and outputs to the system. A schematic diagram below displays the transistor level logic in the circuit.

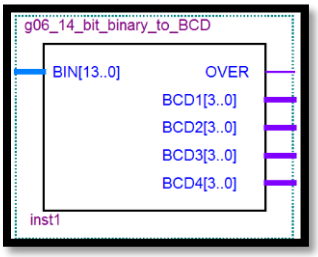

Figure 36: Block diagram of a 14 bit binary to BCD module

Inputs:

- Bin: 14 bit unsigned numbers whose binary coded decimal is to be calculated. All values that need to be displayed on the LED's are first concatenated into 14 bits and then connected to the input line of the circuit.

Outputs:

- BCD1, BCD2, BCD3, and BCD4: The 4 bit binary coded decimal equivalents of the input to the system. These are then connected to the inputs of the g06_7_segment_display component, to be displayed on the LED's.

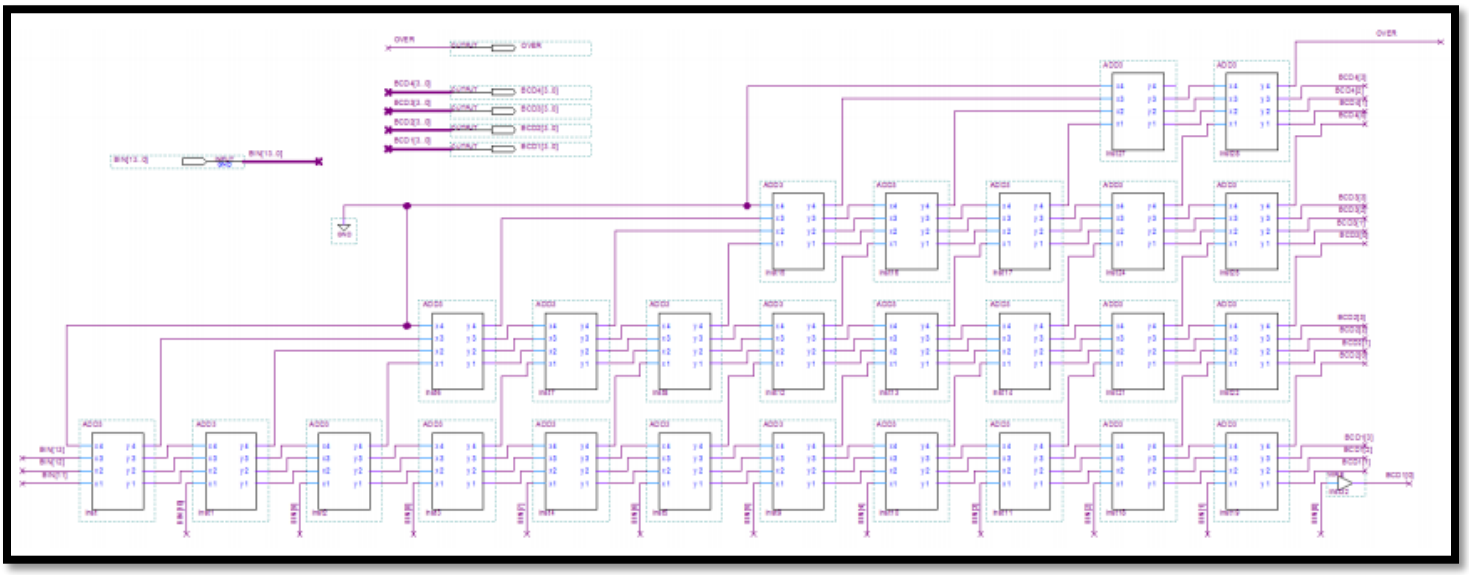

Figure 37: Schematic of Binary to BCD converter[2]

Testing:

[2] Taken from lab 2 description slides

Several values were tested. The total testing time was set to 65536 ns. The BIN values were set to change ever 40ns by a values of 100. These were done to ensure that a wide range of values were tested in order to cover all possible cases as displayed by the waveform files below.

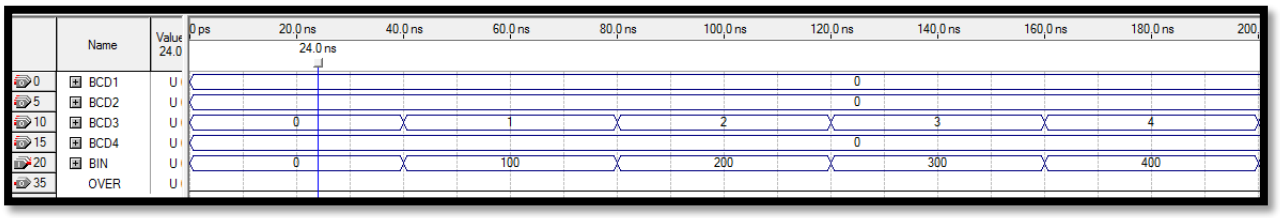

Figure 38: For low values

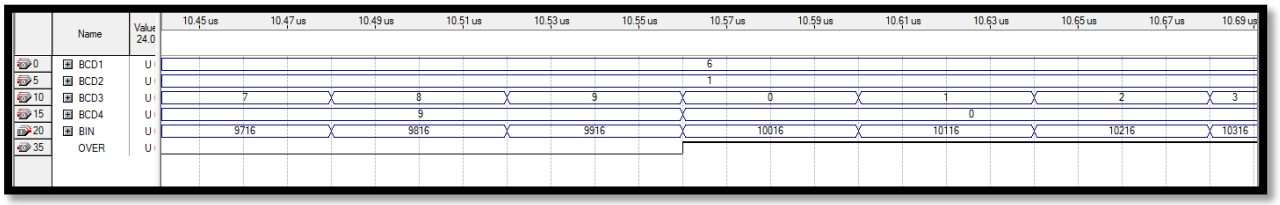

Figure 39:For overflowing

BCD7

Component Description:

Function: The g06_7_segment decoder is used to drive a seven segment display depending on the values input to it. Depending on the hexadecimal value input to the circuit it will drive the corresponding LED segments to display the requisite number. The component also has the ability to determine if the screen is to be kept blank in the case of an irregular or zero input. In the case of cascading seven segment displays it will only display a zero in the least significant bit.

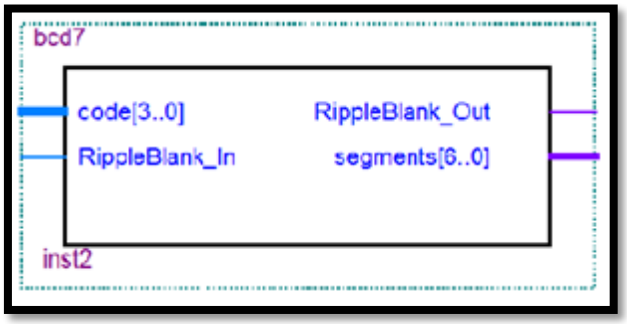

Figure 40: Block diagram of bcd7

The output is displayed on the LED on the board. The results are displayed in integer format.

A block diagram for the circuit is included below to illustrate the inputs and outputs to the system. A schematic diagram below displays the transistor level logic in the circuit.

Inputs:

- *Code:* the 4 bit binary equivalent of a hexadecimal number(0 to F). This represents the value to be displayed by the LED. This is connected to the relevant BCDn output from the g06_14_bit_binary_to_BCD.

- *RippleBlank_in:* a single bit variable which will determine if the screen is to be kept blank or not. Connected to the Rippleblank_out of next g06_7_segment decoder.

Outputs:

- *Segments:* 7 bit number which determines which of the seven segments on an LED is to be lit up. Connected to the LED's on the board.
- *RippleBlank_Out:* single bit output which determines in the case of cascading LED's if the next LED is to be blanked or not. Connected to the Rippleblank_in of previous g06_7_segment decoder.

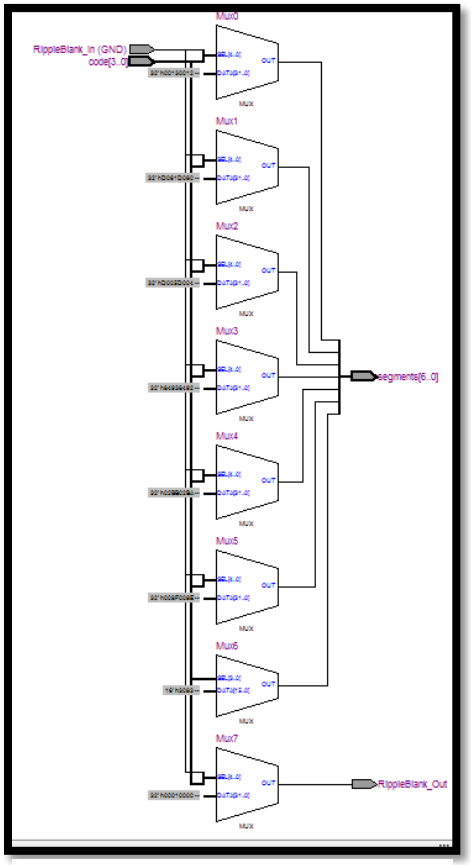

Figure 41:BCD 7 schematic

Functional Simulation:

Initially the value of *RippleBlank_In* was set to low while the values of code went through the 16 different hexadecimal cases.

Then the sixteen different hexadecimal cases were tested with *RippleBlank_In* set to high. This changed the output only in the case of 0 input and drove the *RippleBlank_Out* to high.

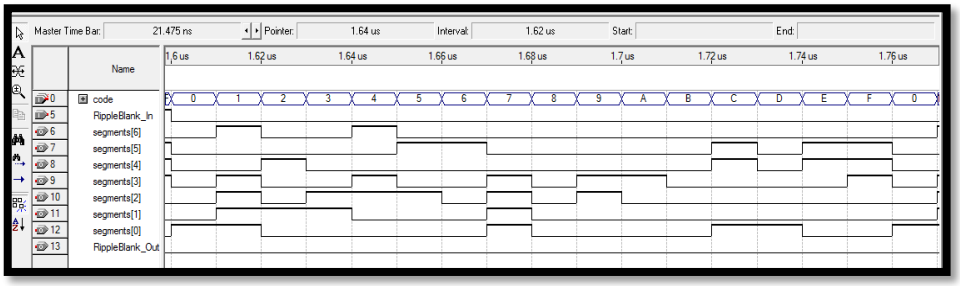

Figure 42: BCD7 Testing

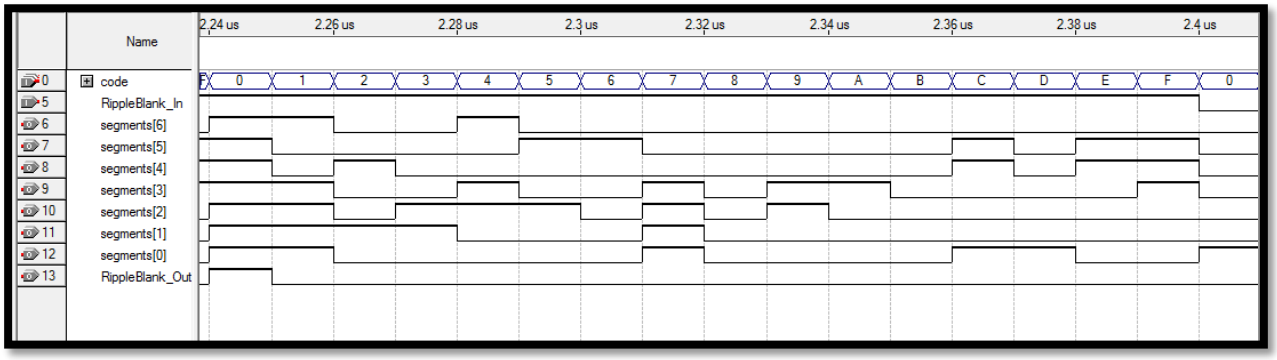

Figure 43:BCD7 testing waveform

Total System Test

After creating each component we then went on to assemble our rowing machine circuit in a test bed that was uploaded onto the board for hardware testing. Our test bed itself serves as the final circuit of the rowing machine. However to ensure we had correct functioning of the circuit a functional simulation of the outputs had been done without the adjustments and display unit mentioned earlier.

The simulation results are given below and the hardware tests can be found in the images of the user interface.

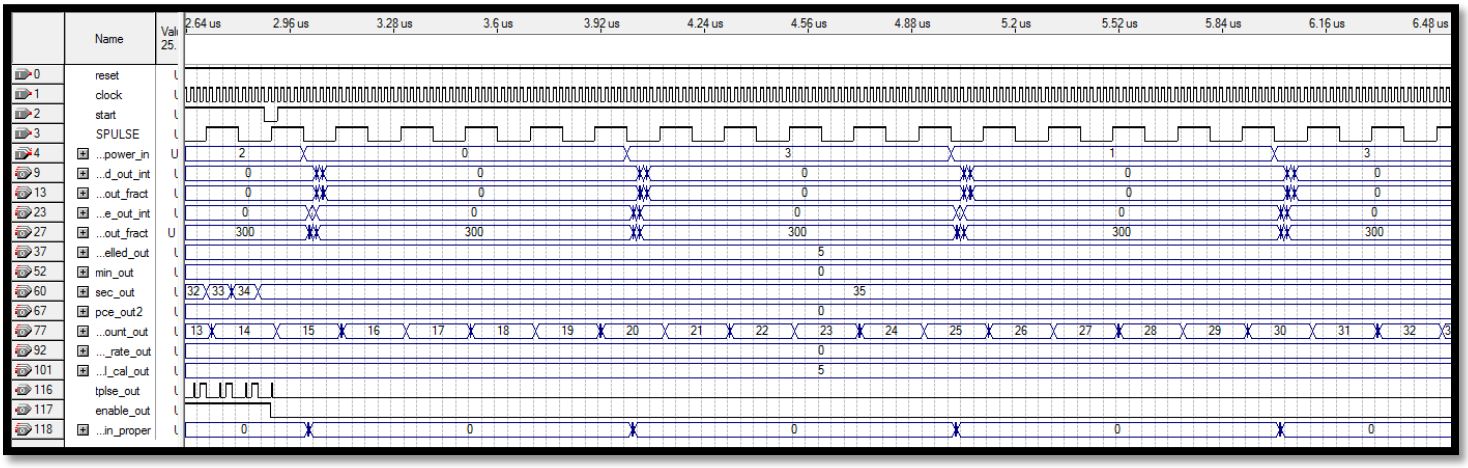

Figure 44: Our System in a HOLD state

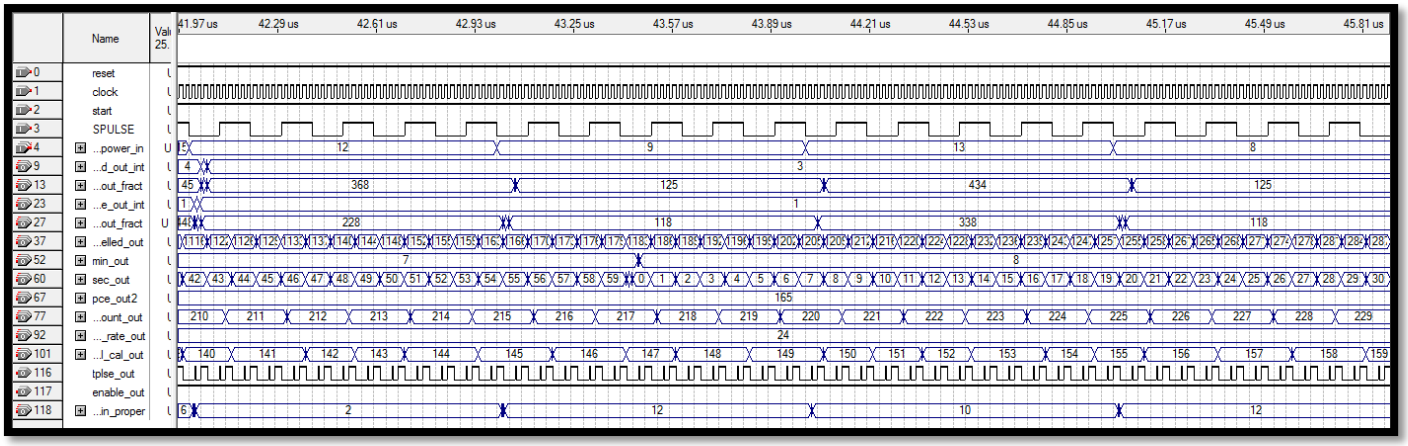

Figure 45: In a Running state

Conclusion

During the development of the g06_Rower_Machine a few issues were encountered by the team. Although none of the issues in themselves were too significant, the cumulative time spent on error fixing added up. A few issues that are worth pointing out are:

- The total time available for the report as well as final project was too little.
- A recurring issue that arose was keeping track of the integer and fractional bits each time floating point arithmetic was used.
- The logic for giving a minutes output for the total time elapsed was causing difficulty. Instead of creating a new circuit we solved the problem by adding 100 every sixty seconds.
- Sampling any pushbutton only once when pressed. This was solved by putting up flag conditions every time an input from the pushbuttons was used.

Apart from the issues there are a few design changes that could have improved the performance of our circuit:

- Ability to display decimals and fractional parts.
- Using a few extra sensors more information such as heart rate display could be added.
- If more LED's were available multiple items could be displayed at one time.
- By introducing a few more circuits different modes could be introduced to the system such as High intensity mode, interval training, cardio training etc.