DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

Assignment 1

Digital Clock

Sahand Sabour

EEE339

Digital System Design with HDL(I)

Student ID

1614650

Introduction

Programmable Logic Devices (PLDs) have obtained an essential role in modern digital systems industry, mainly due to their ability to implement both combinational and sequential logic circuits. Evidently, a Hardware Description Language (HDL) would be used to describe such hardware and accordingly, program PLDs to demonstrate the desired functions. In this project, Verilog was used due to its simplicity and similarity to the C language. In addition, the apparatus for this assignment consisted of a DE1 ALTERA board, its power and USB cable as well as the Quartus II software. In this assignment, each group of two students were provided with a DE1 board and were tasked to create a digital clock with the following main functions:

1. 4-digit minute and second display.
2. Time setting.
3. A stopwatch with 4-digit second and tenth of a second display.

In addition, the students were tasked to implement additional functions to improve the digital clock design. Hence, the following functions were determined to be implemented:

1. Up/Down counting.
2. Timer with flashing LED lights.
3. Simultaneous operation of both clock and stopwatch with changeable modes: allow the user to choose what is displayed.

Hence, the following hardware overview diagram was designed (Figure 1).

Figure 1: Hardware overview diagram for Digital Clock system

It is believed that this assignment was intended to assess each student's understanding of the subject, challenge the ability to design functional hardware, and provide a valuable learning opportunity to explore Verilog as a HDL.

Methodology

In this section, a thorough explanation of each block of the design, including its inputs, outputs, and operating logic, is provided. As the Altera board is equipped with four 7-segment displays, Q_{0-3} would be used to reference each display, with

Q_0 being the right-most display and Q_3 being the left-most display. Hence, it is suggested that these notations be noted for future references. In addition, the connection between the components as well as the pin assignments for the whole circuit is provided at the end of this section.

Second and Decisecond Counter Design

The Altera board is equipped with two oscillators which produce 27MHz and 50MHz clock signals respectively. The prior oscillator was implemented in this assignment. Therefore, each 27000000 clock cycles would account for one second. Accordingly, two counters were designed to output a HIGH signal for each 2700000 and 27000000 clock signals respectively. Consequently, this would allow the other components to be enabled every tenth of a second (Decisecond) or every second depending on their intended functions (check Figure 2, Appendix 1&2).

Figure 2: Second and Decisecond Counters diagram

Clock Design

Inputs

Initially, in order to design a functional clock (MM:SS display), the following inputs were required:

1. **Enc:** allows the clock to operate every second.
2. **Clk:** the 27MHZ clock signal produced by the oscillator.
3. **Ent:** allows the clock to operate when set to HIGH, pauses the clock operation when set to LOW.

The prior two inputs can be obtained from the Altera board itself as clock signal is produced by the 27MHz oscillator and ENC is set to high every 27000000 clock cycles. However, the last input should be determined by flipping a switch, which would allow to the user to resume or pause the operation. Moreover, in order to implement additional functions to the design, the following inputs were added, which were determined by the switches on the Altera board.

1. **Clr:** Restores the clock to the default state (00:00) when set to HIGH.
2. **UpDown:** changes the direction of the count; the clock counts upwards when set to LOW or downwards when set to HIGH.

With the above inputs, a functional clock with functions of clearing the registers, counting both up and down that operates every second was constructed. In order to add the function of time setting, several measures were to be considered.
First, the clock should receive a 4 bit input to determine which digit is to be changed, with each bit representing the digit to be changed when set to HIGH. Second, as declared in the design objectives, concurrent operation between the clock and the stopwatch is to be implemented. Hence, the clock should receive an input corresponding to whether it is being displayed or not. This is necessary as time setting should only be available when the clock is being displayed. Third, an extra counter would be added in order to enhance the user experience. This counter would output a HIGH signal every 0.2 seconds and accordingly, the digits would be increased every 0.2 seconds when a key is hold down. This implementation would allow the user to hold the corresponding keys to set the time rather than pressing them several times. Conclusively, the following inputs were declared:

1. **timeSetter:** a 4 bit input to declare which digit is being set.
2. **mode:** shows that the clock is being displayed when set to LOW.
3. **pause:** represents the pause between each increment in the digit being modified and is set to HIGH every 0.2 seconds.

Outputs

As for the outputs, the clock needed to produce two signals: First, a 16 bit output Q, which represents 4 groups of 4 bits, each group corresponding to a digit that is to be displayed, with the left-most 4 bits correspond to the leftmost digit of the clock etc. Second, an output Rco, which declares that the clock has reached the default state in the down counting mode. With the addition of the latter output, the clock could also function as a timer and trigger a following component when it reaches its lower limit (00:00).

Logic

Initially, the 16 bit output is set to zero. On the positive edge of the clock cycle, the component should analyze the following set of conditional statements and update the outputs accordingly.
If the clear input (Clr) is set to HIGH, the output will be set to zero.
If Clr is set to LOW, the enabled switch is flipped, a second is reached, and the clock is counting upwards, then Q_0 would be incremented by 1. If Q_0 reaches its limit, then Q_1 would be incremented by 1 and Q0 would be reset to 0 etc. As the format of the clock dedicates two digits to display the minutes and two digits to display the seconds, the limits for Q_0 and Q_2 would be 9 while the limits for Q_1 and Q_3 would be 5. Regarding the down counting operation of the clock,

decrements of 1 would be used instead of increments with the limit for all digits being 0. Moreover, each digit would be reset to their respective limits in the up counting mode when they reach 0 in the down counting mode. If all the digits are set to 0 in the down counting mode, Rco would be set to HIGH.
If none of the above conditions are met and one of the keys on the Altera board is pressed by the user, an increment to the corresponding digit is to be applied. Evidently, the increments follow the same limit as the up counting logic and would set the digit to 0 when the limit is reached.

The following figure (Figure 3) delineates the diagram for the designed clock. In addition, the Verilog code for this block is provided in Appendix 3.

Figure 3: Clock diagram

Stopwatch Design

The inputs and outputs for the stopwatch are identical to the ones regarding the clock design. The differentiation to be made between the to components would be regarding both the counter that they were connected to as well as the limit for each digit. Regarding the prior, the stopwatch was connected to the Decisecond counter rather than the second counter, as it should operate on a tenth of a second basis. In addition, regarding the latter difference, the limit for all of the 4 digits is set to 9. The following figure (Figure 4) displays the diagram for the designed stopwatch. The Verilog code for this block is provided in Appendix 4.

Figure 4: Stopwatch diagram

Pause counter

As mentioned, in order to make the time setting feature more convenient for the user, a 0.2 pause is to be made before each increment when the user is pressing the keys on the board. The design of the component for this pause is identical to the second and Decisecond counter, with the slight difference that this component would output a HIGH signal every 5400000 clock cycles: 0.2 seconds (check Figure 5 and Appendix 5).

Figure 5: Pause counter diagram

Priority 4-to-1 Encoder

The Altera is equipped with four keys (buttons). As decided in the design specification, these keys were dedicated to the time setting feature, where each key would

cause an increment in the corresponding digit. For instance, key0 would increase the value of $Q-0$, key1 would increase the value of Q_1, etc. Hence, it was decided that a 4-to-1 priority encoder could be implemented. Accordingly, at any given time, only one digit would be affected, with decreasing priority in the increments from left to right. This is believed to reduce time leading/lagging errors as well as error due to occasional glitches. The block diagram for this component is provided in the below figure (Figure 6). Moreover, the Verilog code for constructing is provided in Appendix 6.

Figure 6: 4-to-1 priority encoder diagram

Display selector

As the Altera board includes merely four 7-segment displays, the simultaneous display of both the clock and the stopwatch is not practical as each of these components would require four displays respectively. Therefore, a component was to be designed to control what is shown by the four 7-segment displays. Accordingly, the display selector component would act as a 2-to-1 selector, which would receive an input from the user indicating the operation to be displayed. More specifically, when the input is set to LOW, the output of the clock would be displayed and when it is set to HIGH, the displays would show the output of the stopwatch. Furthermore, this block divides the 16-bit input to four 4-bit outputs that can be displayed on the 7-segment displays respectively. The block diagram for this component is provided in the figure below (Figure 7) and this block can be constructed by the Verilog code in Appendix 7.

Figure 7: Display selector diagram

BCD-to-7-segment Decoder

As implied by the name, the displays on the DE1 board are constructed of seven LED segments. Each of these segments are turned on when subject to a LOW signal. Hence, implementing a BCD-to-7segment is necessary.

Figure 8: Segment index and position for DE1 board

The Above figure illustrates the bit number corresponding to each of these segments. Accordingly, a 7-bit output for each decimal from zero to nine that displays the shape of these digits on the board was to be determined. These outputs are provided in the following table (Table 1).

Input	Output	Input	Output
0	0000001	5	0100100
1	1001111	6	0100000
2	0010010	7	0001111
3	0000110	8	0000000
4	1001100	9	0000100

Table 1: Translation table for decoder

The code for this block is provided in Appendix 8. Moreover, the block diagram for this component is provided below (Figure 9).

Figure 9: BCD-to-7-segment decoder diagram

Alarm

In order to constantly flash a group of LED lights when the clock and/or the stopwatch reach their default state in the down counting mode, it was decided that a component is to be constructed. The input for this component would be provided by its preceding component, whether it is the clock or the stopwatch. Furthermore, this component has an n-bit output, with n corresponding to the number of LEDs on the DE1 board that have the same color. As the Altera board includes eight green LEDs and ten red LEDs, two alarm components with 8-bit and 10-bit outputs respectively were constructed. The logic for this component is significantly similar to the pause counter. That is, it would output a HIGH signal every 5400000 clock signals: 0.2 seconds. With this approach, the LED lights would flash with 0.2-second intervals. Figure 10 displays the block diagram of this component. Moreover, the Verilog code for constructing this component is provided in Appendix 9 and 10.

Figure 10: Alarm components

Pin Assignments

The following table (Table 2) displays the input pin assignments respectively.

Input	Pin
CLK	D12
CLR	L21
ENT	L22
UPDOWN	M22
deciMode	V12
CLR2	U12
ENT2	W12
UPDOWN2	U11
W[0]	R22
W[1]	R21
W[2]	T22
W[3]	T21

Table 2: Table of Pin assignments for inputs

In addtion, the following table (Table 3) includes all the pin assignments regarding the output of the design.

Output	Pin
D[0]	E2
D[1]	F1
D[2]	F2
D[3]	H1
D[4]	H2
D[5]	J1
D[6]	J2
S1[0]	D1
S1[1]	D2
S1[2]	G3
S1[3]	H4
S1[4]	H5

Output	Pin
S1[5]	H6
S1[6]	E1
S2[0]	D3
S2[1]	E4
S2[2]	E3
S2[3]	C1
S2[4]	C2
S2[5]	G6
S2[6]	G5
M[0]	D4
M[1]	F3
M[2]	L8

Output	Pin
M[3]	J4
M[4]	D6
M[5]	D5
M[6]	F4
L[0]	R17
L[1]	R18
L[2]	R18
L[3]	R18
L[4]	R18
L[5]	R18
L[6]	R18
L[7]	R18

Output	Pin
L[8]	D5
L[9]	F4
G[0]	Y21
G[1]	Y22
G[2]	W21
G[3]	W22
G[4]	V21
G[5]	V22
G[6]	U21
G[7]	U22

Table 3: Table of Pin assignments for outputs

Circuitry

The complete circuit, with all the wires connected and the pins assigned, is provided in the figure below (Figure 11).

figures/DigitalClock.jpg

Figure 11: The complete circuit

Results and Discussion

In this section, the simulation results, which demonstrate the functionality of each of the implemented functions, are provided respectively. As simulations allow for micro-analysis of the clock cycles, the counter components were modified to operate at a much less frequency with the same frequency ratio. More specifically, the Decisecond counter, the alarms, the pause counter,and the second counter were modified to output a HIGH signal every 1,2, 2, 10 clock cycles. With this approach, it is believed that the resulting results could be better illustrated and analyzed. In addition, the obtained results would be meticulously observed and discussed. The user interface is provided likewise.

Clock

A functional clock is expected to operate in accordance to the design specification. In this section, the results corresponding to each function with thorough analysis is provided.

Up Counting

The most fundamental function of a clock is to count upwards and update the 7-segment displays accordingly. In order to simulate the results for this function, the waveforms were simulated during $10\mu s$, $20\mu s$, $100\mu s$, and $420\mu s$ duration to clearly illustrate the increments for each digit respectively. For the BCD-to-binary decoding translation, refer to Table 1.

Figure 12: Up counting for the clock (Q_0)

According to the above figure (Figure 12), the clock is able to accurately increment the right-most digit (Q_0) from 0 to 9, reset the digit to 0, and accordingly, increment Q_1 by one unit.

Figure 13: Up counting for the clock (Q_1)

The above figure (Figure 13) shows that the up-counting operation is conducted properly: Q_1 is accurately incremented from 0 to 5, reset to 0, and Q_2 is incremented by one unit in correspondence. Furthermore, as shown in the figure below (Figure 14), the incremental operation is successfully achieved for Q_2 and Q_3 has been updated in accordance.

Figure 14: Up counting for the clock (Q_2)

Figure 15: Up counting for the clock (Q_3)

As illustrated in Figure 15, similar to the previous results, the operation successfully incremented Q_3 from 0 to 5 and reset to 0 after the limit has been reached. Therefore, it can be concluded that the up-counting operation for the clock is functional.

Time setting

The second significant function of a functional clock is providing the ability to set the time. The waveforms for this function were simulated for duration of $400ns$. The time setting feature was tested for all four digits and the results are provided below respectively.

Figure 16: Time setting for the clock (Q_0)

The above figure (Figure 16) indicates that by holding down KEY0 (setting W[0] to LOW), Q_0 is incremented from 0 to 9 by units of one and reset to 0 after reaching 9, which corresponds to the design specification. In addition, the below figure (Figure 17) illustrates that the incremental operation is accurately conducted for Q_1 likewise; that is, the digits are incremented from 0 to 5 and reset to 0 when the limit is reached.

Figure 17: Time setting for the clock (Q_1)

According to the below figure (Figure 18), Q_2 is incremented by units of one from 0 to 9 and reset to 0 when the digit has reached 9.

Figure 18: Time setting for the clock (Q_2)

Figure 19: Time setting for the clock (Q_3)

The above figure (Figure 19) illustrates that the incremental operation is performed accurately and hence, as the operation has been functional for all the displays (Q_{0-3}), it can be concluded that the time setting feature is successfully functioning.

Down Counting

Designed as an additional feature, the down-counting operation is performed similarly to the up-counting operating, with decrements in the digits rather then increments. In order to simulate results for the down-counting process and provide a clear illustration of the performance, the time-setting feature must initially be operated. Therefore, for each digit's analysis, an increment is simulated in prior and accordingly, the down-counting process for each digit is observed respectively.

Figure 20: Down counting for the clock (Q_0)

According to the above figure (Figure 20), Q_0 is initially incremented to 2 and by after the UPDOWN input is set to HIGH, the value of Q_0 is decremented by units of one from 2 to 0.

Figure 21: Down counting for the clock (Q_1)

Evidently, Figure 21 indicates the same results for Q_1, as the decrements are performed accurately. Additionally, it can be observed that after Q_1 is decremented, the value of Q_0 is set to 9 and the down-counting process effectively proceeds. The same observation can be made from the below figure (Figure 22), where each decrement in Q_2 causes Q_1 to be set to 5.

Figure 22: Down counting for the clock (Q_2)

As illustrated in the below figure (Figure 23), the down-counting operation for Q_3 is performed correctly. From this Figure, it can be observed that Q_3 is properly decremented until it reaches 0 and correspondingly, sets the value of Q_2 to 9. Hence, based on the obtained results, it is believed that the down-counting process is fully functional.

Figure 23: Down counting for the clock (Q_3)

Reset

The results for the reset feature are provided in the below figure (Figure 24). As illustrated in the figure, all the digits are set to 0 when the CLR input is set to HIGH. Hence, this feature is properly functioning.

Figure 24: Reset function for clock

Alarm

As displayed in Figure 25, when the clock reaches 00:00 in the down-counting mode, output G is continuously set to HIGH (green LEDs are turned on) and reset to LOW (green LEDs are turned off). This is an accurate demonstration of the flashing effect, as requested by the design specifications.

Figure 25: Alarm function for clock

Stopwatch

A functional stopwatch is expected to be able to perform the same functions as the clock. In this section, the results corresponding to each function with thorough analysis is provided. For all of the following simulations, deciMode is set to HIGH as functions of the stopwatch are to be analyzed.

Up Counting

Similar to the clock, the most fundamental function of a stopwatch is also to count upwards and update the 7-segment displays accordingly. In order to simulate the results for this function, the waveforms were simulated during $1\mu s$, $2\mu s$, $20\mu s$, and $200\mu s$ duration to illustrate the increments for each digit respectively. For the BCD-to-binary decoding translation, refer to Table 1.

Figure 26: Up counting for stopwatch (Q_0)

As displayed in the above figure (Figure 26), the stopwatch is able to accurately increment the right-most digit (Q_0) from 0 to 9, reset the digit to 0, and accordingly, increment Q_1 by one unit.

Figure 27: Up counting for stopwatch (Q_1)

The above figure (Figure 27) shows that the up-counting operation is properly functioning: Q_1 is accurately incremented from 0 to 9, reset to 0, and Q_2 is incremented by one unit in correspondence. In addition, as displayed in the figure below (Figure 28), the incremental operation is successfully achieved for Q_2 and Q_3 has been updated in correspondence .

Figure 28: Up counting for stopwatch (Q_2)

Figure 29: Up counting for stopwatch (Q_3)

By observing Figure 29 similar results to the previous simulations is derived: Q_3 is successfully incremented from 0 to 9 and reset to 0 after the limit has been reached. Therefore, it can be concluded that the up-counting operation for the stopwatch is functional.

Time setting

The stopwatch should also be able to perform the time setting function. Similarly, the waveforms for this function were simulated for $400ns$, the time setting feature was tested for all four digits and the results are provided below respectively.

Figure 30: Time setting for stopwatch (Q_0)

Figure 30 clearly displays that by holding down KEY0 (setting W[0] to LOW), Q_0 is incremented from 0 to 9 by units of one and reset to 0 after reaching 9. Moreover, the below figure (Figure 31) indicates that the incremental operation is accurately performed for Q_1 as well: the digits are incremented from 0 to 9 and reset to 0 when the limit is reached.

Figure 31: Time setting for stopwatch (Q_1)

According to the below figure (Figure 32), Q_2 is incremented by units of one from 0 to 9 and reset to 0 when the digit has reached 9.

Figure 32: Time setting for stopwatch (Q_2)

Figure 33: Time setting for stopwatch (Q_3)

The above figure (Figure 33) displays that the incremental operation is performed accurately and hence, as the operation has been functional for all the displays (Q_{0-3}). Therefore, the time setting feature for stopwatch is functional.

Down Counting

The down-counting operation in the stopwatch is performed similarly to the up-counting operation, with decrements in the digits rather then increments. In order

to simulate the results for the down-counting process and provide a clear illustration of the performance, the time-setting feature must initially be operated. Therefore, for each digit's analysis, an increment is simulated in prior and correspondingly, the down-counting process for each digit is observed respectively.

Figure 34: Down counting for stopwatch (Q_0)

According to the above figure (Figure 34), Q_0 is initially incremented to 5 and by after the UPDOWN input is set to HIGH, the value of Q_0 is decremented by units of one from 5 to 0.

Figure 35: Down counting for stopwatch (Q_1)

Figure 35 indicates the same results for Q_1, as the decrements are performed accurately. Moreover, it can be derived that after Q_1 is decremented, the value of Q_0 is set to 9 and the down-counting process effectively proceeds. The same observation can be made from the below figure (Figure 36), where each decrement in Q_2 causes Q_1 to be set to 9.

Figure 36: Down counting for stopwatch (Q_2)

As shown in the below figure (Figure 37), the down-counting operation for Q_3 is performed correctly. From this Figure, it can be observed that Q_3 is properly decremented until it reaches 0 and correspondingly, sets the value of Q_2 to 9. Hence, based on the obtained results, it is believed that the down-counting process is functional.

Figure 37: Down counting for stopwatch (Q_3)

Reset

The results for the reset feature are provided in the below figure (Figure 38). As displayed in the figure, all the digits are set to 0 when the CLR input is set to HIGH. Hence, this feature is properly functioning.

Figure 38: Reset function for stopwatch

Alarm

As shown in Figure 39, when the stopwatch reaches 000:0 in the down-counting mode, output L is continuously set to HIGH (red LEDs are turned on) and reset to LOW (red LEDs are turned off), which corresponds to the desired flashing effect.

Figure 39: Alarm function for stopwatch

Simultaneous operation

As specified in the design objectives, simultaneous operation of both the clock and the stopwatch was required. The following figure (Figure 40) illustrates the results for this operation for simulation with duration of $20\mu s$. As displayed in the figure, the displayed outputs correspond to the clock when deciMode is set to LOW, and they correspond to the stopwatch when deciMode is set to HIGH. Hence, the implementation of this feature is successfully achieved.

Figure 40: Simultaneous operation of clock and stopwatch

User Interface

The complete user manual to access all of the functions of the design is provided in the table below (Tables 4 and 5).

Switch	**Function**
SW0	Start the clock (switch down to pause)
SW1	Reset the clock
SW2	Change the counting direction for clock (Up for low counting)
SW3	Change the display (Down for MM:SS and up for SSS:Ds)
SW4	Start the stopwatch (switch down to pause)
SW5	Reset the stopwatch
SW6	Change the counting direction for stopwatch (Up for low counting)

Table 4: User interface manual for switches

Key	**Function**
KEY0	increment the value of Q0 by 1
KEY1	increment the value of Q1 by 1
KEY2	increment the value of Q2 by 1
KEY3	increment the value of Q3 by 1

Table 5: User interface manual for keys

Note that in order for increments to be functioning, the counter must be paused in prior. The keys will not have any affect on an operating clock/stopwatch.

Conclusion

In this assignment, a digital clock circuit was designed and implemented. The clock had two main components: normal clock (MM:SS) and stopwatch (SSS:D). As regards to the functionality, all of the required functions as well as the additional functions were designed and implemented with success. It is believed that after successful completion of this assignment, valuable experience regarding the Quartus II software, the DE1 Altera board, and Verilog was obtained. In addition, it is believed that deeper understanding of the lecture materials, which were taught in the EEE339 module, was achieved. Conclusively, this assignment was considered a valuable learning opportunity as well as a successful implementation of a functional digital clock.

References

[1] *EEE205 Lab Altera Experiment*, 4 ed., Xi'an Jiaotong Liverpool University, Department of Electrical and Electronic Engineering, Suzhou, 2008

Appendices

```
module deciCount (clk, Q);
input clk;
output reg Q;
reg [25:0] count = 0; // 26 bit output to store 2700000

always @(posedge clk)
begin
Q <= 0;
if (count == 26'd2700000) // 0.1 second is reached
begin
count <= 0;
Q <= 1;
end
else
count <= count+1;
end
endmodule
```

Appendix 1: Verilog code for Decisecond counter

```
module secondCount (clk, Q);
input clk;
output reg Q;
reg [25:0] count = 0; // 26 bit output to store 27000000

always @(posedge clk)
begin
Q <= 0;
if (count == 26'd27000000) // 1 second is reached
begin
count <= 0;
Q <= 1;
end
else
count <= count+1;
end
endmodule
```

Appendix 2: Verilog code for second counter

```
module normalClock (Clk, Pause, Enc, Ent, Clr, UpDown,
```

```
deciMode, timeSetter, Q, Rco);
input Clk, Pause, Enc, Ent, Clr, UpDown, deciMode;
input [3:0] timeSetter;
reg[3:0] Q1; // output for Q0 display
reg[3:0] Q2; // output for Q1 display
reg[3:0] Q3; // output for Q2 display
reg[3:0] Q4; // output for Q3 display
output reg [15:0] Q; // The cumulative output
output reg Rco =0;

always @(posedge Clk, posedge Clr)
begin
if (Clr)
begin
Q1 <= 4'b0000;
Q2 <= 4'b0000;
Q3 <= 4'b0000;
Q4 <= 4'b0000;
Rco <= 0;
end
else if (Enc && Ent)
begin
Rco <= 0;
if (!UpDown) //Incremental
begin
if (Q1==4'b1001) //If the first digit is 9 (LSB)
begin
Q1 <= 4'b0000;
if (Q2 == 4'b0101) //If the second digit is 5
begin
Q2 <= 4'b0000;
begin
if (Q3==4'b1001) //If the third digit is 9
begin
Q3 <= 4'b0000;
if (Q4 == 4'b0101) //If the fourth digit is 5
Q4 <= 4'b0000;
else
Q4 <= Q4+1;
end
else
Q3 <= Q3+1;
end
```

```
end
else
Q2 <= Q2+1;
end
else
Q1 <= Q1+1;
end
else //Decremental
begin
if (Q1==4'b0000)
begin
if (Q2==4'b0000 && Q3==4'b0000 && Q4==4'b0000)
Rco <= 1;
else
begin
Q1 <= 4'b1001;
if (Q2 == 4'b0000)
begin
Q2 <= 4'b0101;
begin
if (Q3==4'b0000)
begin
Q3 <= 4'b1001;
if (Q4 == 4'b0000)
Q4 <= 4'b0101;
else
Q4 <= Q4-1;
end
else
Q3 <= Q3-1;
end
end
else
Q2 <= Q2-1;
end
end

else
Q1 <= Q1-1;
end
end
else if (!Ent &&Pause && !deciMode) // for time setting
begin
```

```
case (timeSetter)
4'b0001: begin
if (Q1 == 4'b1001)
Q1 <= 4'b0000;
else
Q1 <= Q1+1;
end
4'b0010: begin
if (Q2 == 4'b0101)
Q2 <= 4'b0000;
else
Q2 <= Q2+1;
end
4'b0100: begin
if (Q3 == 4'b1001)
Q3 <= 4'b0000;
else
Q3 <= Q3+1;
end
4'b1000: begin
if (Q4 == 4'b0101)
Q4 <= 4'b0000;
else
Q4 <= Q4+1;
end
endcase
end
Q <= {Q4, Q3,Q2,Q1};
end
endmodule
```

Appendix 3: Verilog code for clock

```
module stopwatch (Clk, Pause, Enc, Ent, Clr, UpDown, deciMode
,timeSetter, Q, Rco);
input Clk, Pause, Enc, Ent, Clr, UpDown, deciMode;
input [3:0] timeSetter;
reg[3:0] Q1; // output for Q0 display
reg[3:0] Q2; // output for Q11 display
reg[3:0] Q3; // output for Q2 display
reg[3:0] Q4; // output for Q3 display
output reg [15:0] Q; // The cumulative output
output reg Rco =0;
```

```
always @(posedge Clk, posedge Clr)
begin
if (Clr)
begin
Q1 <= 4'b0000;
Q2 <= 4'b0000;
Q3 <= 4'b0000;
Q4 <= 4'b0000;
Rco <= 0;
end
else if (Enc && Ent)
begin
Rco <= 0;
if (!UpDown) //Incremental
begin
if (Q1==4'b1001) //If the first digit is 9 (LSB)
begin
Q1 <= 4'b0000; //Set it to zero
if (Q2 == 4'b1001) //If the second digit is 9
begin
Q2 <= 4'b0000;
begin
if (Q3==4'b1001) //If the third digit is 9
begin
Q3 <= 4'b0000;
if (Q4 == 4'b1001) //If the fourth digit is 9
Q4 <= 4'b0000;
else
Q4 <= Q4+1;
end
else
Q3 <= Q3+1;
end
end
else
Q2 <= Q2+1;
end
else
Q1 <= Q1+1;
end
else //Decremental
begin
if (Q1==4'b0000)
```

```
begin
if (Q2==4'b0000 && Q3==4'b0000 && Q4==4'b0000)
Rco <= 1;
else
begin
Q1 <= 4'b1001;
if (Q2 == 4'b1001)
begin
Q2 <= 4'b1001;
begin
if (Q3==4'b0000)
begin
Q3 <= 4'b1001;
if (Q4 == 4'b1001)
Q4 <= 4'b1001;
else
Q4 <= Q4-1;
end
else
Q3 <= Q3-1;
end
end
else
Q2 <= Q2-1;
end
end

else
Q1 <= Q1-1;
end
end
else if (!Ent &&Pause &&deciMode) // Time setting
begin
case (timeSetter)
4'b0001: begin
if (Q1 == 4'b1001)
Q1 <= 4'b0000;
else
Q1 <= Q1+1;
end
4'b0010: begin
if (Q2 == 4'b1001)
Q2 <= 4'b0000;
```

```
else
Q2 <= Q2+1;
end
4'b0100: begin
if (Q3 == 4'b1001)
Q3 <= 4'b0000;
else
Q3 <= Q3+1;
end
4'b1000: begin
if (Q4 == 4'b1001)
Q4 <= 4'b0000;
else
Q4 <= Q4+1;
end
endcase
end
Q <= {Q4, Q3,Q2,Q1};
end
endmodule
```

Appendix 4: Verilog code for stopwatch

```
module pauseCount (clk, Q);
input clk;
output reg Q;
reg [25:0] count = 0; //26 output to store 5400000

always @(posedge clk)
begin
Q <= 0;
if (count == 26'd5400000) // 0.2 seconds reached
begin
count <= 0;
Q <= 1;
end
else
count <= count+1;
end
endmodule
```

Appendix 5: Verilog code for pause counter

```
module enc4to1 (W, Y);
input [3:0] W;
output reg [3:0] Y;

always @(W)
begin
casex(W)
4'b0xxx: Y=4'b1000; // if key3 is pressed
4'b10xx: Y=4'b0100; // if key2 is pressed
4'b110x: Y=4'b0010; // if key1 is pressed
4'b1110: Y=4'b0001; // if key0 is pressed
default: Y=4'b0000; // no key is pressed
endcase
end
endmodule
```

Appendix 6: Verilog code for 4-to-1 priority encoder

```
module displaySelector (deciMode,R1, R2, Q1, Q2, Q3, Q4);
input deciMode;
input [15:0] R1; // 16 bit input from the clock
input [15:0] R2; // 16 bit input from the stopwatch
output reg [3:0] Q1; // output for Q0 display
output reg [3:0] Q2; // output for Q1 display
output reg [3:0] Q3; // output for Q2 display
output reg [3:0] Q4; // output for Q3 display

always @(deciMode)
if (!deciMode) // clock is displayed
{Q4, Q3, Q2, Q1}= R1;
else // stopwatch is displayed
{Q4, Q3, Q2, Q1}= R2;
endmodule
```

Appendix 7: Verilog code for display selector

```
module seg7Decoder(bcd, leds);
input[3:0] bcd;
output reg[7:1] leds;

always @(bcd)
```

```
case(bcd)
4'b0000: leds= 7'b0000001; //Displaying 0
4'b0001: leds= 7'b1001111; //Displaying 1
4'b0010: leds= 7'b0010010; //Displaying 2
4'b0011: leds= 7'b0000110; //Displaying 3
4'b0100: leds= 7'b1001100; //Displaying 4
4'b0101: leds= 7'b0100100; //Displaying 5
4'b0110: leds= 7'b0100000; //Displaying 6
4'b0111: leds= 7'b0001111; //isplaying 7
4'b1000: leds= 7'b0000000; //Displaying 8
4'b1001: leds= 7'b0000100; //Displaying 9
default: leds= 7'bx; //Error! input is outside [0,9] range
endcase
endmodule
```

Appendix 8: Verilog code for 7-segment decoder

```
module alarmWatch (Clk, Ent, leds);
input Clk, Ent;
output reg [7:0] leds =0; // for the 8 green LEDs
reg [22:0] count = 0;

always @(posedge Clk)
begin
if (Ent)
begin
if (count == 0)
begin
leds <= ~leds;
end
count <= count +1;
end
else
begin
leds <= 0;
count <= 1;
end
end
endmodule
```

Appendix 9: Verilog code for display selector

```
module alarmClock (Clk, Ent, leds);
input Clk, Ent;
output reg [9:0] leds =0; // for the 10 red LEDs
reg [22:0] count = 0;

always @(posedge Clk)
begin
if (Ent)
begin
if (count == 0)
begin
leds <= ~leds;
end
count <= count +1;
end
else
begin
leds <= 0;
count <= 1;
end
end
endmodule
```

Appendix 10: Verilog code for alarm