COL215 Hardware Assignment 2

Mihir Kaskhedikar, 2021CS10551
Dhruv Ahlawat, 2021CS10556

October 2022

1 Introduction

In this assignment, we construct a stopwatch using the 7-segment displays and switches on the Basys 3 board. The format of the clock will be (M:SS:T) where minutes (M) are on one LED display, seconds (SS) are on two LED displays and tenth of second (T) is on one LED display. We would be using the code of Assignment 1 for displaying these 4 numbers on the 4 displays. The stopwatch would have 4 features of **start**, **pause**, **continue** and **reset**. There would be 4 switches assigned on the Basys 3 board for these operations. Their operations are as follows:
Start: When it is switched from 0 to 1, the stopwatch would start (only if it showed 0:00:0 before switching). There is no effect when it is again switched from 1 to 0.
Pause: When it is switched from 0 to 1, it would stop the running stopwatch. There is no effect when it is again switched from 1 to 0.
Continue: It is only switched from 0 to 1 after the stopwatch is paused. It would continue the stopwatch from where it was paused. There is no effect when it is again switched from 1 to 0.
Reset: When it is switched from 0 to 1, the stopwatch is reset to 0:00:0 from whatever it was currently. However it is not started again by this. There is no effect when the reset switch is again switched from 1 to 0.
First, we need to build a clock whose least count is 0.1s for the stopwatch. Next, we have to construct a 4 digit number which would show the time elapsed in M:SS:T format (modulo 10 minutes) since the beginning of its simulation. Finally we add the features mentioned above by introducing some additional variables and signals. We describe each of these three steps below:

2 Setting the least count

For this we simply maintain a variable (initially set to 0) which would get incremented every time there is a rising edge of the in-built clock. Once it attains the value 10000000 (the least count of the in-built clock is 10ns, so for getting 0.1 s, we need 10000000 such iterations) , we change the stopwatch time and again set the variable to 0.

3 Changing the stopwatch time

Initially the time 0:00:0 is shown on the displays and the stopwatch is started. Suppose at some moment the time shown on the displays is A:BC:D. We give a pseudo code of what needs to be done after every 0.1s.

```
D=D+1
if(D==10):
    D=0
    C=C+1
if(C==10):
    C=0
    B=B+1
if(B==6):
    B=0
    A=A+1
if(A==10):
    A=0
```

We feed the new values of A,B,C and D as input signals after converting each of them to 4-bit vectors to the code of Assignment 1. It is clear from the above code that the stopwatch would again reset to 0:00:0 after every 10 minutes.

4 Adding the features

Here we use a trick of storing the previous state of the signal every time the signal is changed. This would enable us to know when there is a rising edge in the value of a signal (we cannot use the command rising edge for any signal other than the in-built clock). Suppose the signals corresponding to start, pause, continue and reset are *start*, *pause*, *continue* and *reset*. We create 4 signals *prevstart*, *prevpause*, *prevcontinue* and *prevreset*, all initialized to '0', to store the previous states of *start*, *pause*, *continue* and *reset*. We also create 3 additional signals $enableWatch$, $isPause$ and $isReset$. The first 2 signals are initialized to '0' while the third is initialized to '1'. If the signal $enableWatch$ is set, it means that the stopwatch needs to keep running while if it isn't set then the stopwatch should not run. If the signal $isPause$ is set, the stopwatch is currently in the paused state while if it isn't set then the stopwatch is not in a paused state. If the signal $isReset$ is set, it means that the clock has been reset while if it isn't set then it means the stopwatch has been started. Having defined the signals we get to how each of them changes after every 0.1s duration. We explain our algorithm part by part.

```
if(start is '1' and prevstart is '0' and isPause is '0' and isReset is '1'):
    Set enableWatch to '1'
    Set isReset to '0'
```

This would ensure that the stopwatch is started only when the *start* is switched from 0 to 1 and the stopwatch was not paused during the switching (to distinguish start from continue).

```
if(pause is '1' and prevpause is '0' and isReset is '0'):
    Set isPause to '1'
    Set enableWatch to '0'
```

This would ensure that whenever the *pause* is switched from 0 to 1, the stopwatch must stop running and go in the paused state.

```
if(continue is '1' and prevcontinue is '0' and isPause is '1'):
    Set isPause to '0'
    Set enableWatch to '1'
```

This would ensure that whenever *continue* is switched from 0 to 1, the stopwatch must resume and come out of the paused state if it was in a paused state, else nothing would happen (to distinguish continue from start).

```
if(reset is '1' and prevreset is '0'):
    Set isPause to '0'
    Set enableWatch to '0'
    Set isReset to '1'
    Set reading of dislays to 0:00:0
```

This would ensure that the stopwatch is reset to 0:00:0 irrespective of its current state.

```
prevstart<=start
prevpause<=pause
prevcontinue<=continue
prevreset<=reset
```

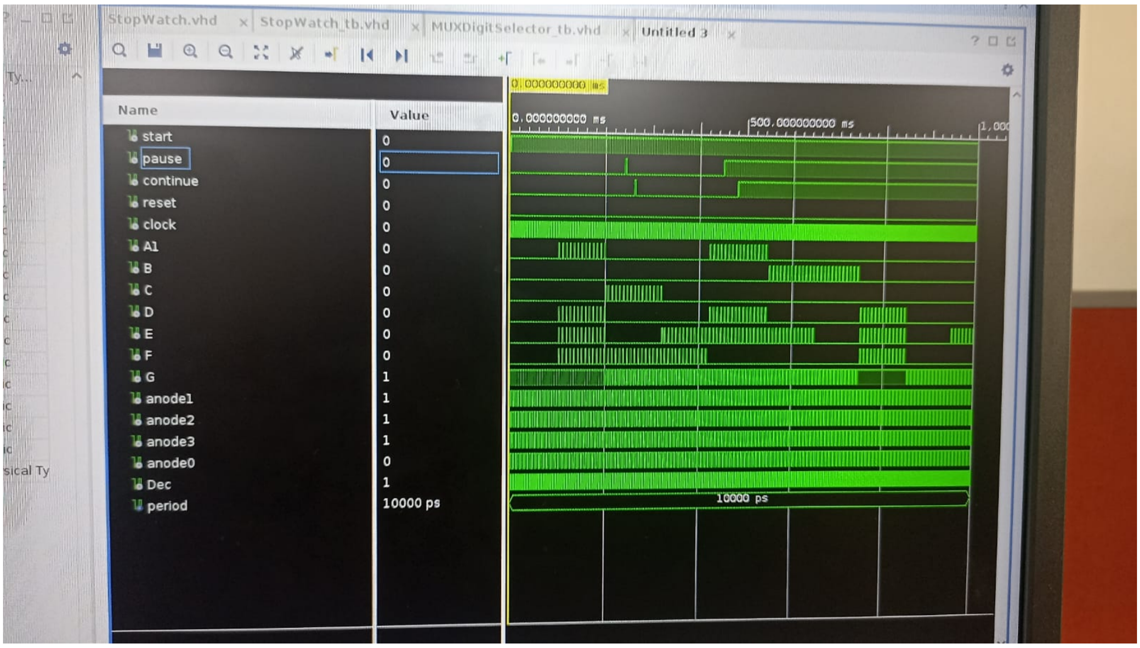
This would update the previous values of *start*, *pause*, *continue* and *reset* to its current values.

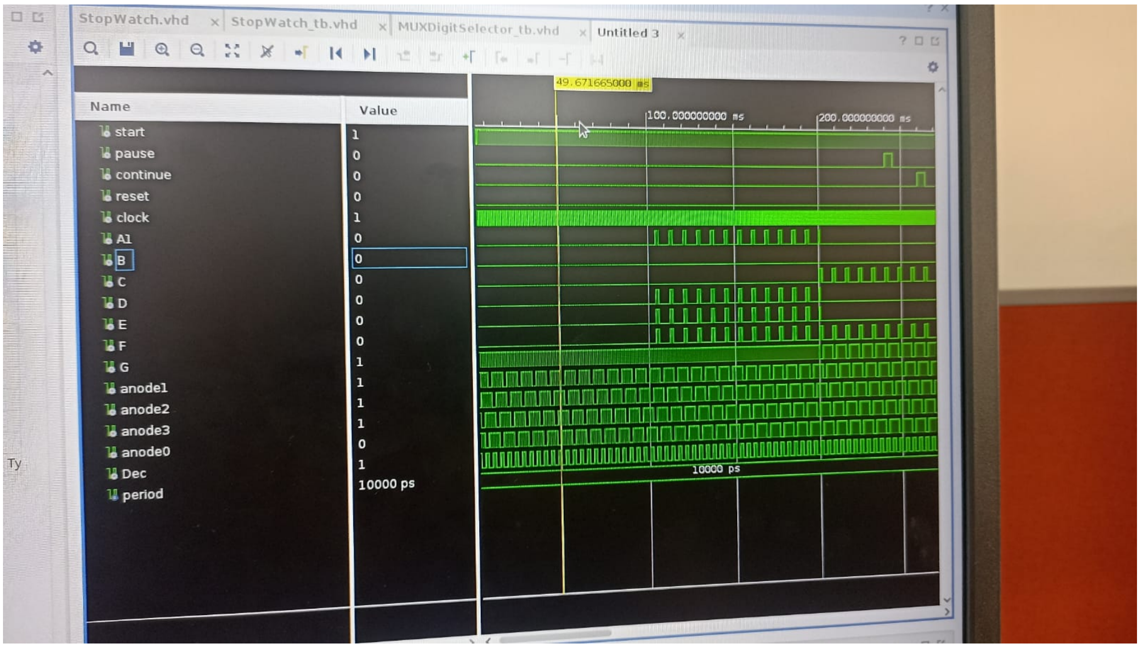
```
if(enableWatch is '1'):
    Proceed with changing the numbers on the displays
```

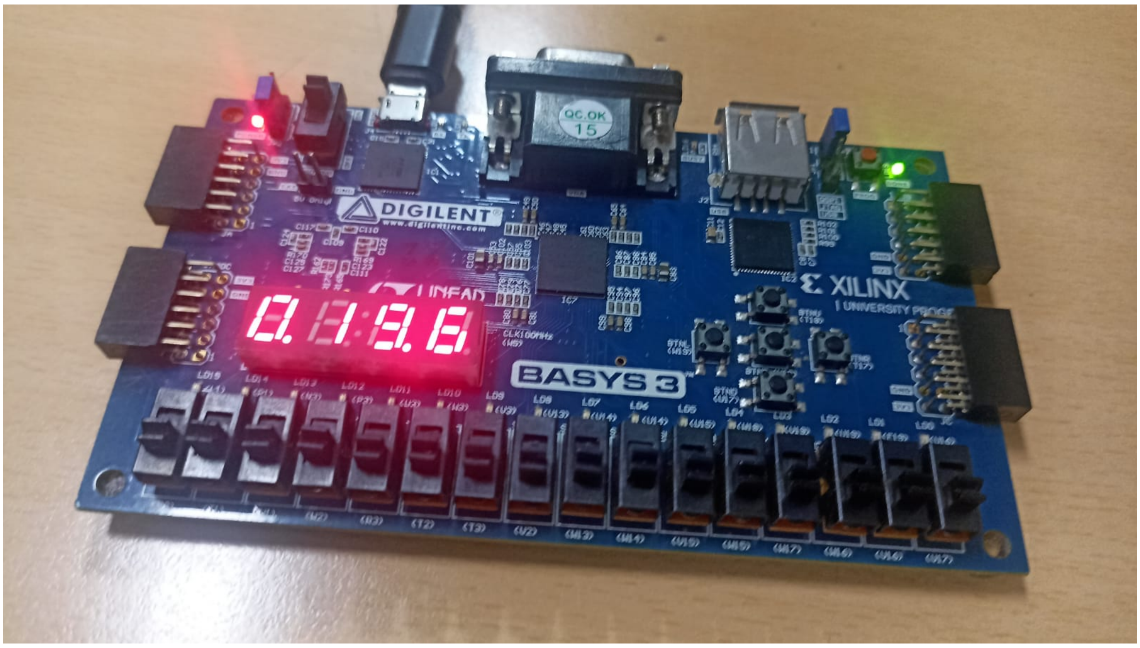
The stopwatch must run only when $enableWatch$ is set.

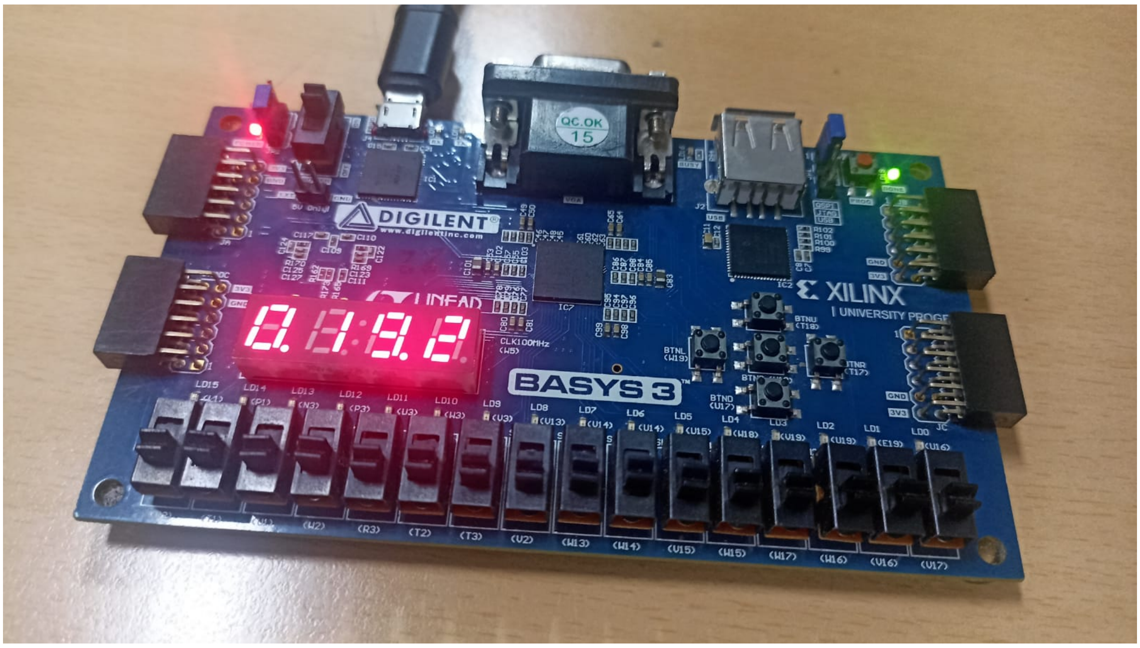
5 Simulation and Basys3 board snapshots

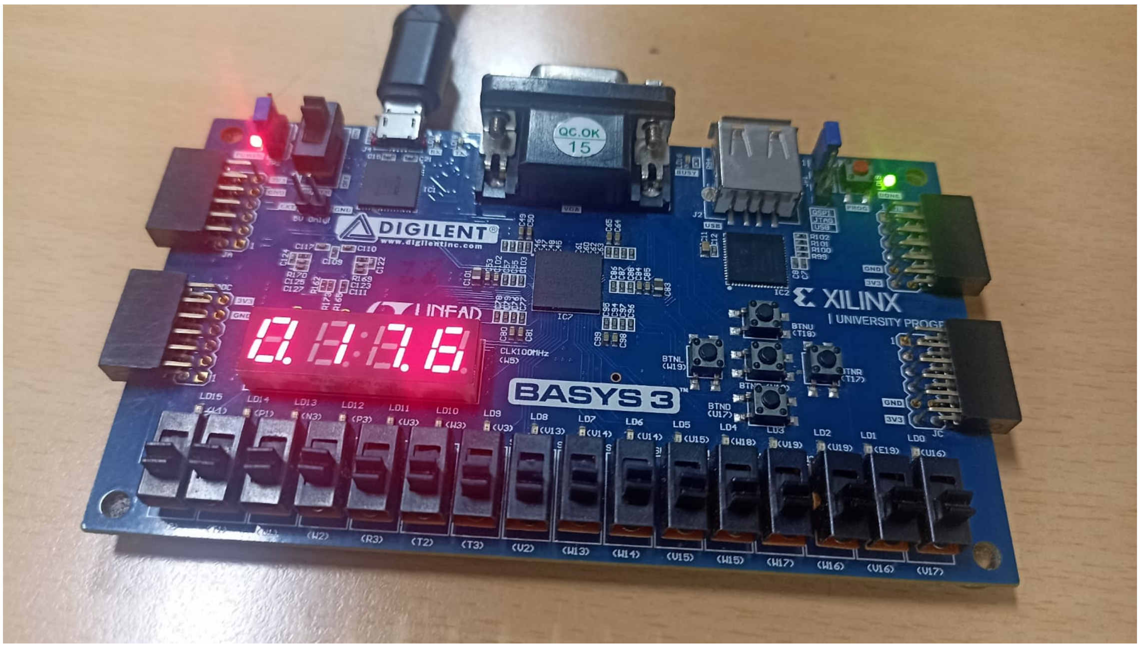
link to hardware test video-
Video


StopWatch.vhd
StopWatch_tb.vhd
MUXDigitSelector_tb.vhd
Untitled 3
Name
Value
start
pause
continue
reset
clock
A1
B
C
D
E
F
G
anode1
anode2
anode3
anode0
Dec
period
10000 ps

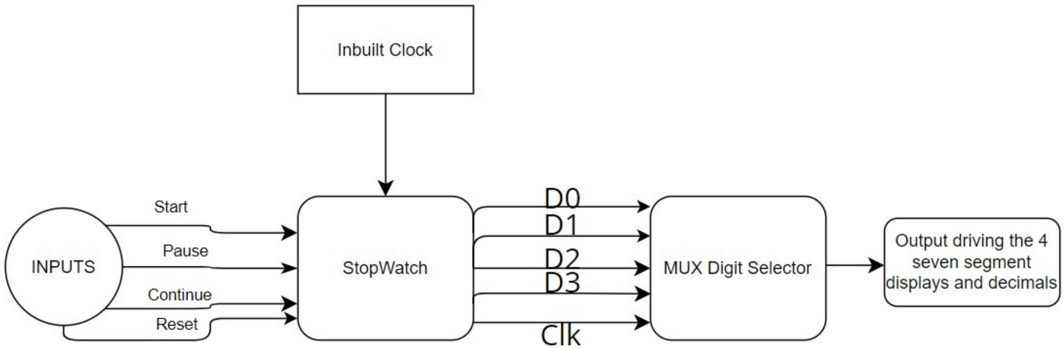

StopWatch.vhd
StopWatch_tb.vhd
MUXDigitSelector_tb.vhd
Untitled 3
Name
Value
start
pause
continue
reset
clock
A1
B
C
D
E
F
G
anode1
anode2
anode3
anode0
Dec
period
10000 ps


QC OK
15
DIGILENT
XILINX
BASYS 3


QC OK
15
DIGILENT
XILINX
BASYS 3


DIGILENT
XILINX
BASYS 3

6 Basic Block Diagram



7 Synthesis Report

```
    Copyright 1986-2022 Xilinx, Inc. All Rights Reserved.
------------------------------------------------------------------------------------------------
| Tool Version : Vivado v.2022.1 (lin64) Build 3526262 Mon Apr 18 15:47:01 MDT 2022
| Date         : Wed Oct 26 17:07:27 2022
| Host         : dhd running 64-bit Ubuntu 20.04.3 LTS
| Command      : report_utilization -file StopWatch_utilization_synth.rpt -pb StopWatch_utilizatio
| Design       : StopWatch
| Device       : xc7a35tcpg236-1
| Speed File   : -1
| Design State : Synthesized
------------------------------------------------------------------------------------------------

Utilization Design Information

Table of Contents
-----------------
1. Slice Logic
1.1 Summary of Registers by Type
2. Memory
3. DSP
4. IO and GT Specific
5. Clocking
6. Specific Feature
7. Primitives
8. Black Boxes
9. Instantiated Netlists

1. Slice Logic
--------------
```

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	187	0	0	20800	0.90
LUT as Logic	187	0	0	20800	0.90
LUT as Memory	0	0	0	9600	0.00
Slice Registers	200	0	0	41600	0.48
Register as Flip Flop	200	0	0	41600	0.48
Register as Latch	0	0	0	41600	0.00
F7 Muxes	0	0	0	16300	0.00
F8 Muxes	0	0	0	8150	0.00

* Warning! The Final LUT count, after physical optimizations and full implementation, is typically lower. Run opt_design after synthesis, if not already completed, for a more realistic count.

1.1 Summary of Registers by Type

Total	Clock Enable	Synchronous	Asynchronous
0	_	-	-
0	_	-	Set
0	_	-	Reset
0	_	Set	-
0	_	Reset	-
0	Yes	-	-
0	Yes	-	Set
0	Yes	-	Reset
0	Yes	Set	-
200	Yes	Reset	-

2. Memory

Site Type	Used	Fixed	Prohibited	Available	Util%
Block RAM Tile	0	0	0	50	0.00
RAMB36/FIFO*	0	0	0	50	0.00
RAMB18	0	0	0	100	0.00

* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only \newline one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a \newline Block RAM Tile, that tile can still accommodate a RAMB18E1

3. DSP

Site Type	Used	Fixed	Prohibited	Available	Util%
DSPs	0	0	0	90	0.00

4. IO and GT Specific

Site Type	Used	Fixed	Prohibited	Available	Util%
Bonded IOB	17	0	0	106	16.04
Bonded IPADs	0	0	0	10	0.00
Bonded OPADs	0	0	0	4	0.00

PHY_CONTROL	0	0	0	5	0.00
PHASER_REF	0	0	0	5	0.00
OUT_FIFO	0	0	0	20	0.00
IN_FIFO	0	0	0	20	0.00
IDELAYCTRL	0	0	0	5	0.00
IBUFDS	0	0	0	104	0.00
GTPE2_CHANNEL	0	0	0	2	0.00
PHASER_OUT/PHASER_OUT_PHY	0	0	0	20	0.00
PHASER_IN/PHASER_IN_PHY	0	0	0	20	0.00
IDELAYE2/IDELAYE2_FINEDELAY	0	0	0	250	0.00
IBUFDS_GTE2	0	0	0	2	0.00
ILOGIC	0	0	0	106	0.00
OLOGIC	0	0	0	106	0.00

5. Clocking

Site Type	Used	Fixed	Prohibited	Available	Util%
BUFGCTRL	1	0	0	32	3.13
BUFIO	0	0	0	20	0.00
MMCME2_ADV	0	0	0	5	0.00
PLLE2_ADV	0	0	0	5	0.00
BUFMRCE	0	0	0	10	0.00
BUFHCE	0	0	0	72	0.00
BUFR	0	0	0	20	0.00

6. Specific Feature

Site Type	Used	Fixed	Prohibited	Available	Util%
BSCANE2	0	0	0	4	0.00
CAPTUREE2	0	0	0	1	0.00
DNA_PORT	0	0	0	1	0.00
EFUSE_USR	0	0	0	1	0.00
FRAME_ECCE2	0	0	0	1	0.00
ICAPE2	0	0	0	2	0.00
PCIE_2_1	0	0	0	1	0.00
STARTUPE2	0	0	0	1	0.00
XADC	0	0	0	1	0.00

7. Primitives

Ref Name	Used	Functional Category
FDRE	200	Flop & Latch
LUT6	74	LUT
LUT5	56	LUT
CARRY4	56	CarryLogic
LUT4	46	LUT

```
| LUT3     |   18 |                 LUT |
| LUT2     |   17 |                 LUT |
| OBUF     |   12 |                  IO |
| IBUF     |    5 |                  IO |
| LUT1     |    4 |                 LUT |
| BUFG     |    1 |               Clock |
+----------+------+---------------------+


8. Black Boxes
--------------

+----------+------+
| Ref Name | Used |
+----------+------+


9. Instantiated Netlists
------------------------

+----------+------+
| Ref Name | Used |
+----------+------+
```