UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Rolling Average on 8-bit Data Stream

Group 30411 – Semigroup 1

Man Rareș-Ioan

Matei Octavian-Mihai

Table of contents

I. Task

# Overview

- The Design Assignment will be to develop a simple signal processing system that will calculate the rolling average of a parallel 8-bit data stream as a systems design exercise.

- The design will be implemented on a self contained Xilinx/Digilent Spartan 3 XC3S200 FPGA board to allow demonstration of a working system.

- The system will be developed as a VHDL model using Xilinx ISE WebPack Version 6.3 that includes the use of the Modelsim simulation tools for design verification.

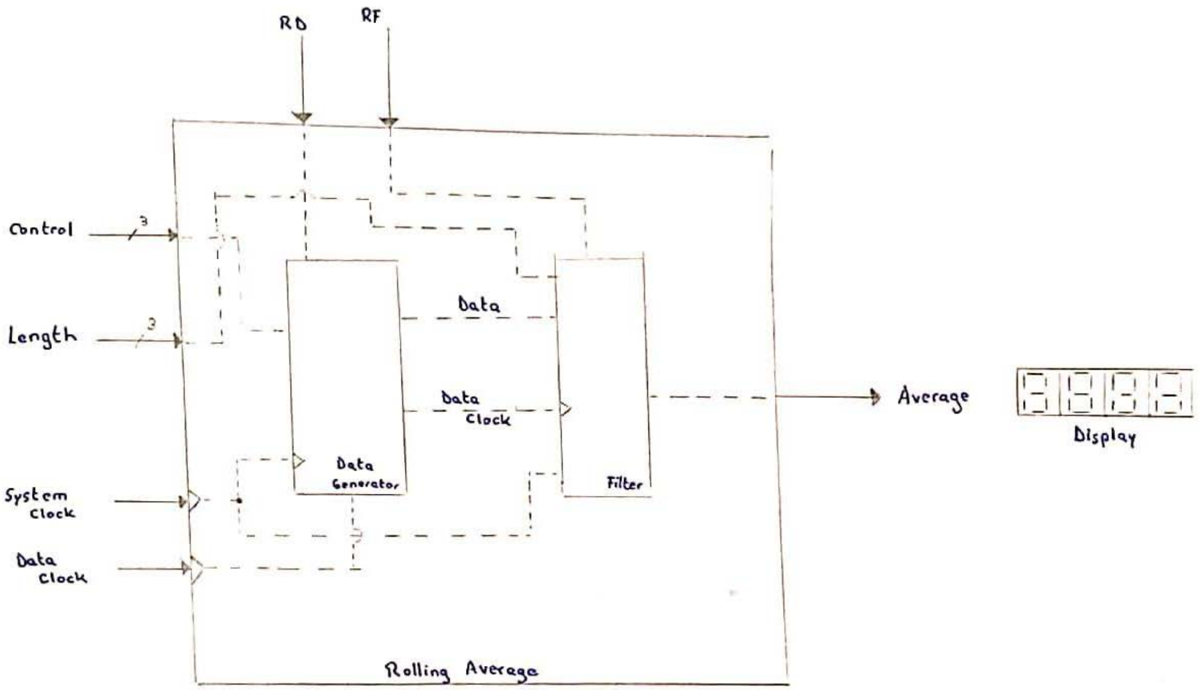
- Assessment will be based primarily on the records kept in individual logbooks supplemented by a short formal report of VHDL listings, system block diagrams and annotated simulation results.

## The Task

Within signal processing systems there is often the need to calculate the numerical average value for an input data stream. This implements a simple low pass filter - smoothing out rapid changes in value in the data stream but maintaining any overall trend. The greater the number of samples used to calculate the average the more smoothing will occur. The filter system will be required to run in "real time" and output the average value at the same rate as the original input data.

The task is to develop a VHDL based model for the "Digital Filter / Rolling Average" system combined with a data stream generator. Switches, Buttons and the Seven Segment Display located on the Diligent S3 board will need to be included to demonstrate correct operation.

II. Block schemata



Control Settings:

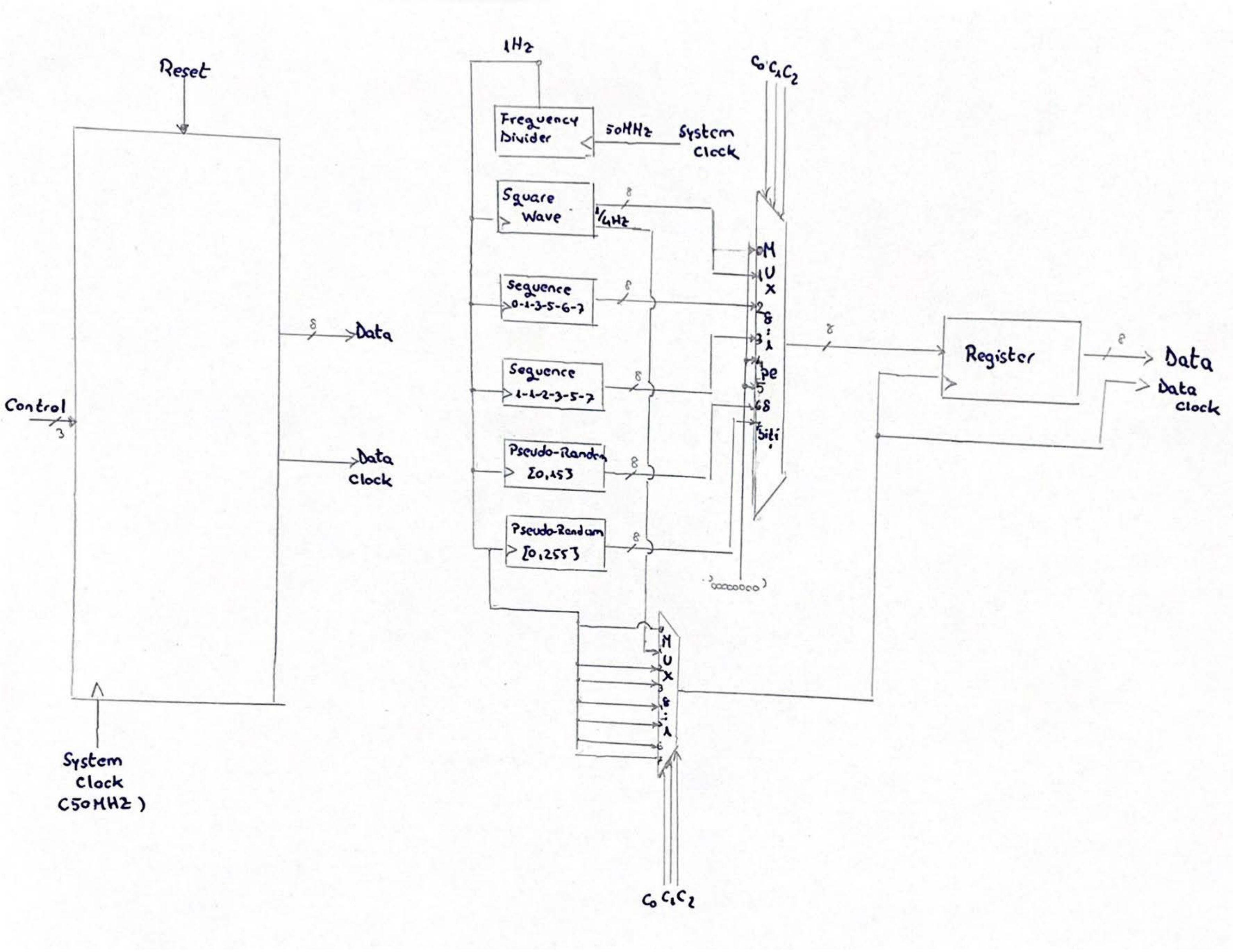
Off - Off - Off	Test Mode o/p 0 (Zero)
Off - Off - On	Square wave (0.25 x data clock)
Off - On - Off	Repeated 6 digit Sequence for Student Number One
Off - On - On	Repeated 6 digit Sequence for Student Number Two
On - On - Off	Pseudo Random Sequence reduced range 0 to 15
On - On - On	Pseudo Random Sequence full range 0 to 255

Buffer "Length" Settings:

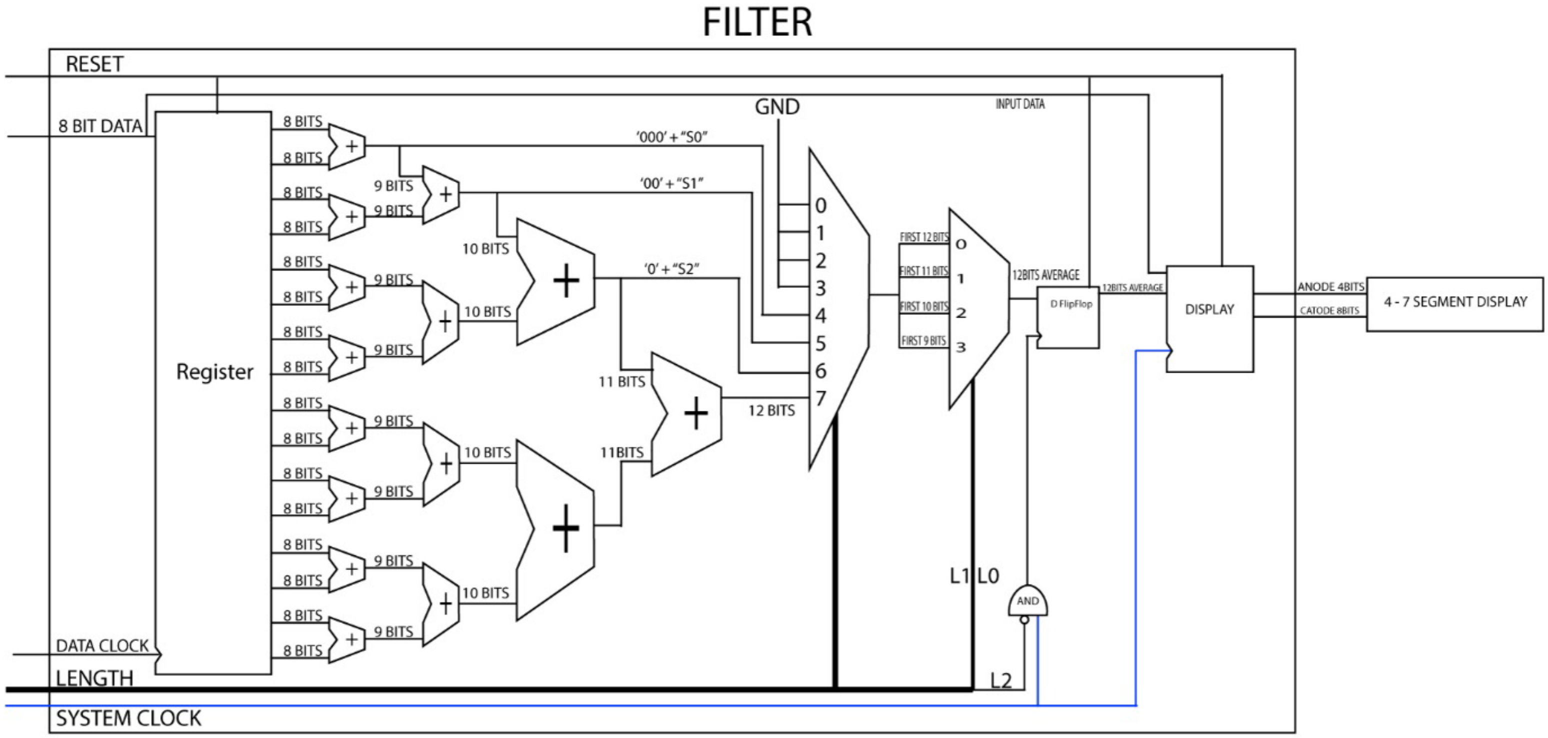
Off - Off - Off	Stop - Hold Value
On - Off - Off	2 Sample Average
On - Off - On	4 Sample Average
On - On - Off	8 Sample Average
On - On - On	16 Sample Average

III. Detailed Block schemata

Data Generator

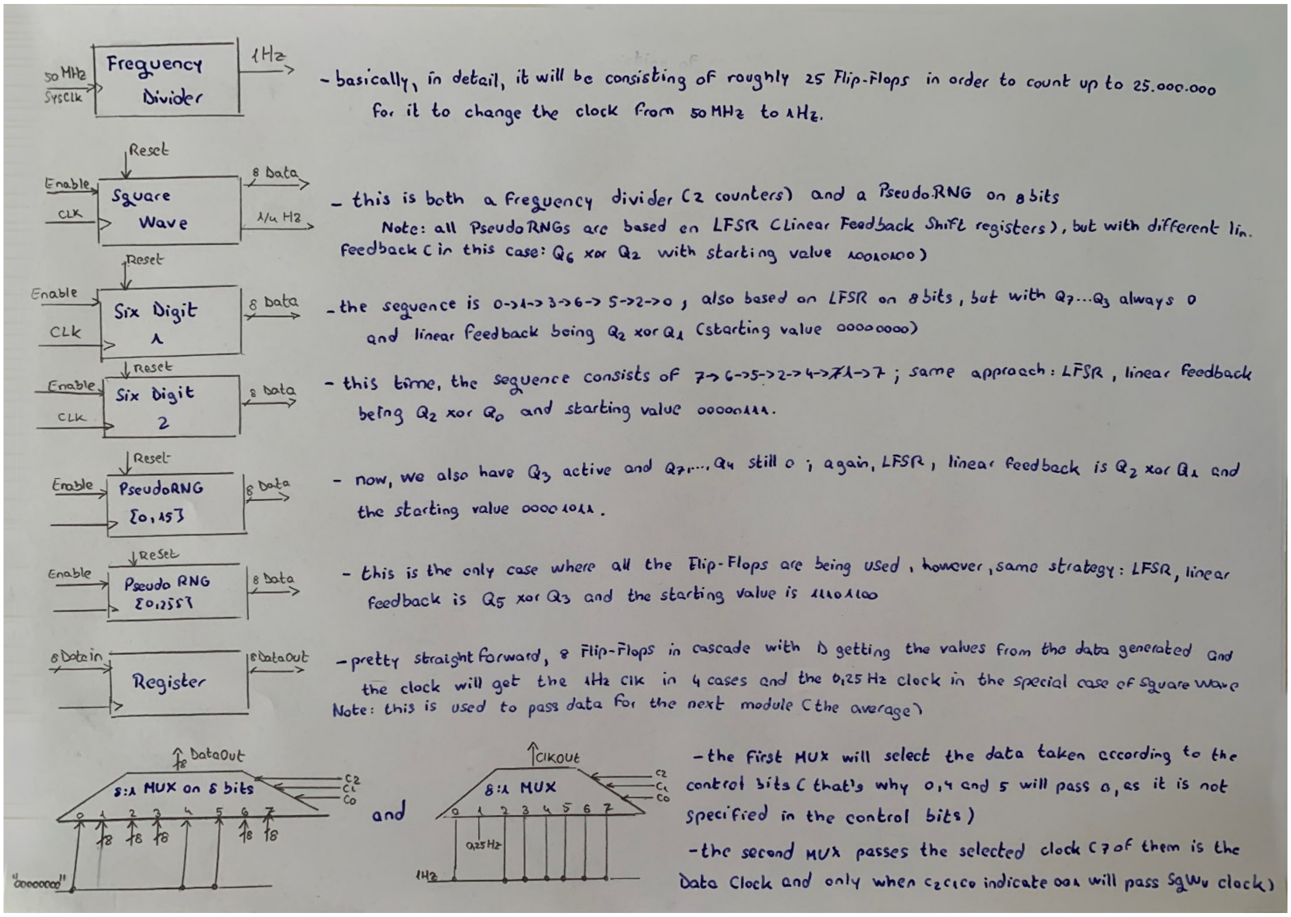


Filter


FILTER
RESET
8 BIT DATA
Register
DATA CLOCK
LENGTH
SYSTEM CLOCK
8 BITS
9 BITS
10 BITS
11 BITS
11BITS
12 BITS
'000' + "S0"
'00' + "S1"
'0' + "S2"
GND
0
1
2
3
4
5
6
7
INPUT DATA
FIRST 12 BITS
FIRST 11 BITS
FIRST 10 BITS
FIRST 9 BITS
12BITS AVERAGE
D FlipFlop
12BITS AVERAGE
DISPLAY
ANODE 4BITS
CATODE 8BITS
4 - 7 SEGMENT DISPLAY
L1
L0
L2
AND

IV. Components, signals and implementation

Data Generator



- basically, in detail, it will be consisting of roughly 25 Flip-Flops in order to count up to 25.000.000 for it to change the clock from 50 MHz to 1Hz.

- this is both a frequency divider (2 counters) and a PseudoRNG on 8 bits
  Note: all PseudoRNGs are based en LFSR (Linear Feedback Shift registers), but with different lin. feedback (in this case: Q_6 xor Q_2 with starting value 10010100)

- the sequence is 0->1->3->6->5->2->0 ; also based on LFSR on 8 bits, but with $Q_7 ... Q_3$ always 0 and linear feedback being Q_2 xor Q_1 (starting value 00000000)

- this time, the sequence consists of 7->6->5->2->4->~~7~~1->7 ; same approach: LFSR, linear feedback being Q_2 xor Q_0 and starting value 00000111.

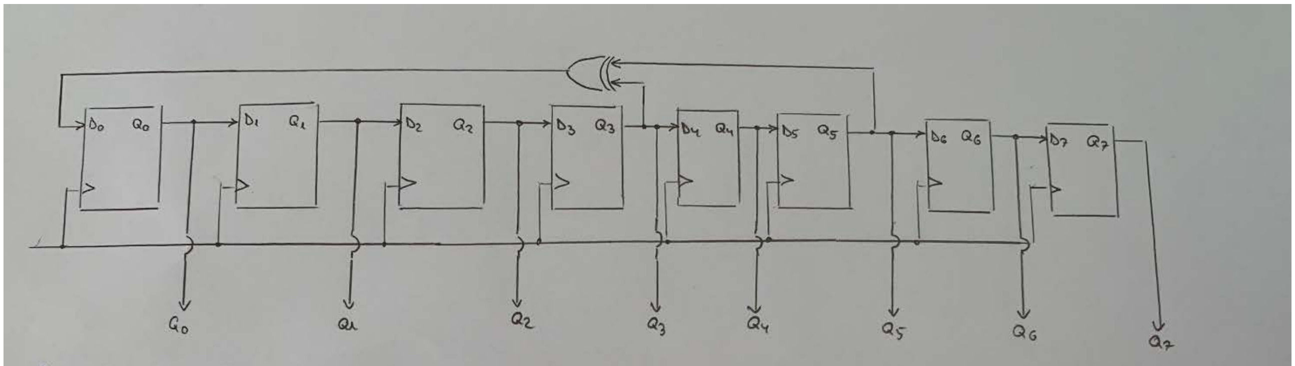
- now, we also have Q_3 active and $Q_7, ..., Q_4$ still 0 ; again, LFSR, linear feedback is Q_2 xor Q_1 and the starting value 00001011.

- this is the only case where all the Flip-Flops are being used, however, same strategy: LFSR, linear feedback is Q_5 xor Q_3 and the starting value is 11101100

- pretty straight forward, 8 Flip-Flops in cascade with D getting the values from the data generated and the clock will get the 1Hz clk in 4 cases and the 0,25 Hz clock in the special case of Square Wave
Note: this is used to pass data for the next module (the average)

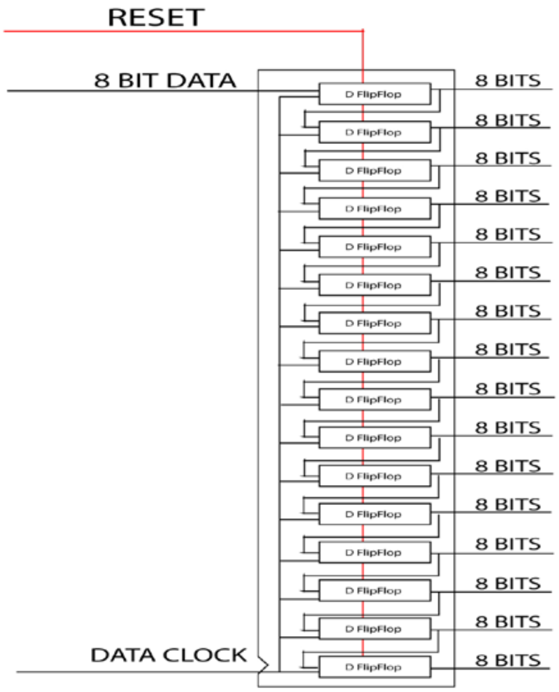
- the first MUX will select the data taken according to the control bits (that's why 0,4 and 5 will pass 0, as it is not specified in the control bits)

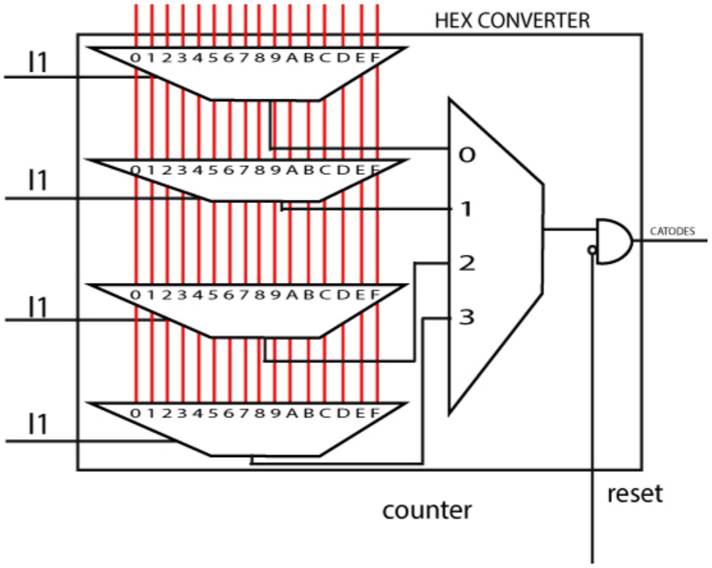
- the second MUX passes the selected clock (7 of them is the Data Clock and only when $c_2c_1c_0$ indicate 001 will pass SqWv clock)



Example of a LFSR. Note: each PseudoRNG is based on this principle, resulting in similar approaches (each of them having different tap points for the linear feedback as well as the starting values (tried to make them as dissimilar as possible))

Filter




HEX CONVERTER
I1
0 1 2 3 4 5 6 7 8 9 A B C D E F
I1
0 1 2 3 4 5 6 7 8 9 A B C D E F
I1
0 1 2 3 4 5 6 7 8 9 A B C D E F
I1
0 1 2 3 4 5 6 7 8 9 A B C D E F
0
1
2
3
CATODES
counter
reset

RED WIRES HAVE THE FOLLOWING VALUES:

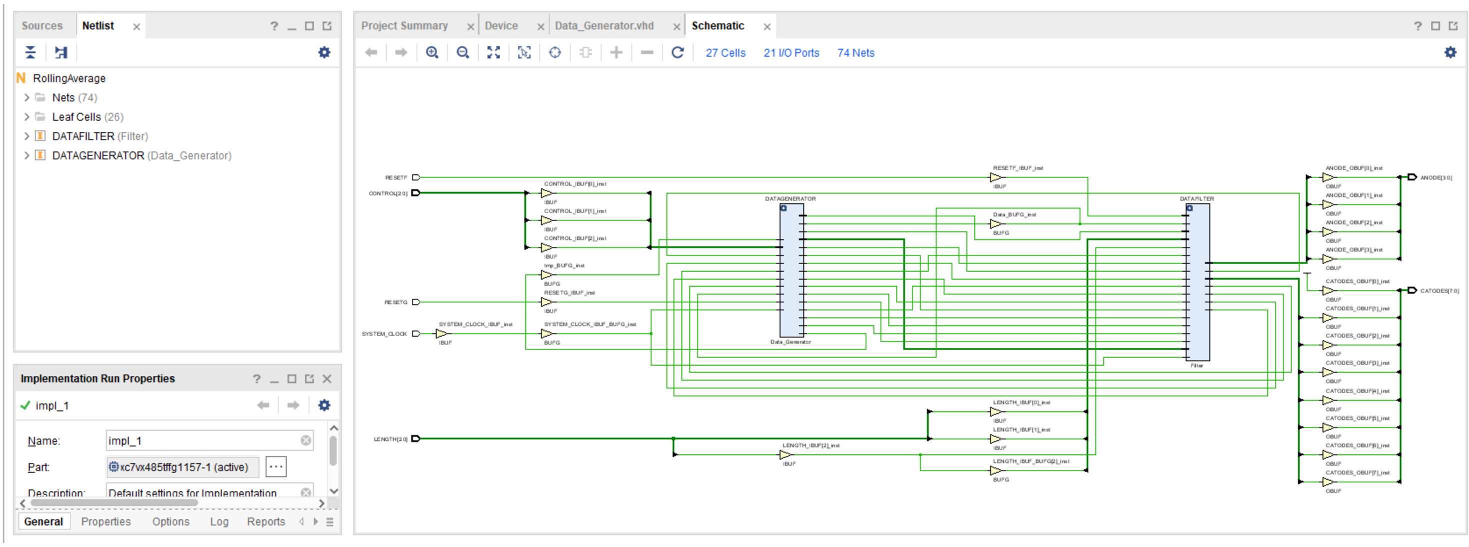
0 – 00000011; 1 – 10011111; 2 – 00100101; 3 – 00001101; 4 – 10011001; 5 – 01001001; 6 – 01000001; 7 – 00011111; 8 – 00000001; 9 – 00001001; A – 00010001; B – 11000001; C – 11100101; D – 10000101; E – 01100001; F – 01110001;

V. Justification of the Chosen Solution

Given the chosen solution, the interconnection of these components was done relatively easily which also makes it a lot easier for a less experimented user to understand the system's functionality by taking a look at the block scheme or from the source code. In this solution that we chose, the processes and the sensitivity lists play a crucial role in defining the overall system.

The interconnection of all these compnents was done in the process of the Filter component. We chose this solution by pipelining in order to ensure a faster and better functionality of the system after the "dead" clock cycles (2-16 cycles). Nevertheless, the cost of doing such implementation is far greater than a more rudimentary approach, thus we believe that the end-user will acknowledge the performance increase.

The FPGA compatibility test was run in ISE VIVADO with the following result:



RollingAverage
- Nets (74)
- Leaf Cells (26)
- DATAFILTER (Filter)
- DATAGENERATOR (Data_Generator)

Implementation Run Properties

✓ impl_1

Name: impl_1

Part: xc7vx485tffg1157-1 (active)

Description: Default settings for Implementation

27 Cells 21 I/O Ports 74 Nets

VI. Maintenance and user’s manual

User’s Manual:

Step 1: Power on the system;

Step 2: Choose your desired configuration by moving the control and length switches;

Step 3: Wait for the pipelining to occur;

Step 4: Check the set of displays that show both the data that is generated and the average computed;

Step 5: If the user wants to reset the whole system and compute another average, make sure that the reset is set on high voltage (logic ‘1’);

Step 6: Repeat step 2.

Maintenance:

- Make sure that the FPGA is functional and runs properly;
- Ensure that the FPGA’s number of ports is greater or equal to the required minimum specification (7 switches and 4 anodes).

VII. Possibilities for subsequent improvements

A first improvement which could be brought to this system could be a more exact approximation of the average value of the numbers by using floating point. However, this involves a rethinking and remapping of the system, as well as a much higher number of displays. In this way, the accuracy of the computed average will be higher.

Another improvement would be performance-wise: in spite of using a single pipeline, there is a possibility of acheiving a multi-level pipelining, thus speeding up the whole system after the mandatory initial clock cycles. Moreover, a performance gain would be given by reducing the computational power required to achieve the goal.

VIII. Conclusion

All things considered, even though the task is rudimentary, it is of great impression by giving a look at how the work is done in VHDL using the ISE software family (Vivado) and a physical environment for testing solutions and maybe even publishing real hardware prototypes.