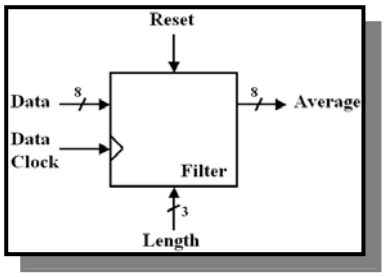
Overview

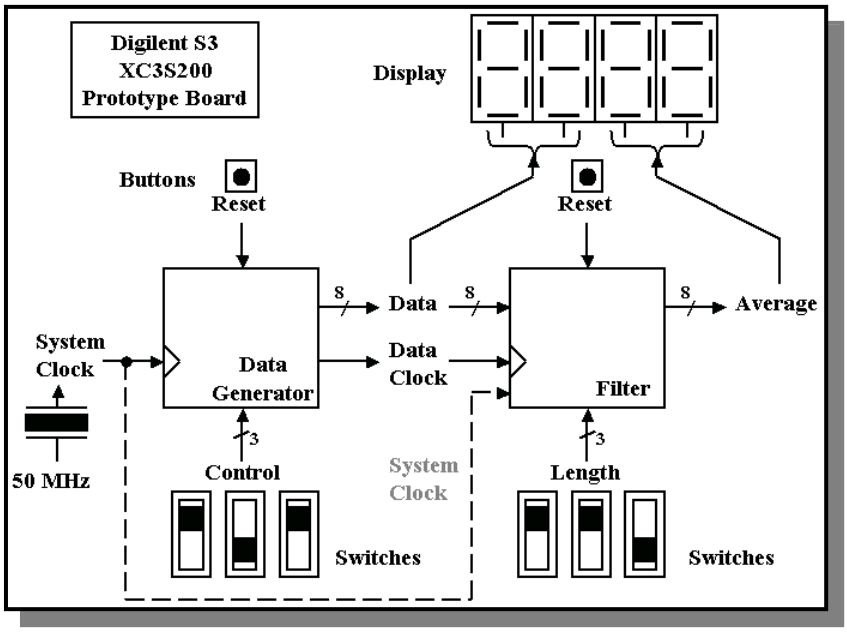
- The Design Assignment will be to develop a **simple signal processing system** that will calculate the **rolling average** of a **parallel 8-bit data stream** as a systems design exercise.
- The design will be implemented on a self contained **Xilinx/Digilent Spartan 3 XC3S200 FPGA board** to allow 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.



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

"Data Clock" rate: 1Hz.

Summary of Spartan-3 FPGA Attributes

XILINX®

Device	System Gates	Equivalent Logic Cells	CLB Array (One CLB = Four Slices)			Distributed RAM Bits (K=1024)	Block RAM Bits (K=1024)	Dedicated Multipliers	Maximum User I/O
			Rows	Columns	Total CLBs				
XC3S200	200K	4,320	24	20	480	30K	216K	12	173

Notes: Logic Cell = 4-input Look-Up Table (LUT) plus a 'D' flip-flop. 8 Logic Cells/CLB

/Continued….

Solving the "problem"

There are many possible approaches that may be taken in implementing the required system functionality - but generally the input data has to be buffered (or stored) in some way such that the calculation of an average over a specific number of values can be performed.

One technique, for example, is to store and add the latest input data value to a running total, and to subtract the first value at the other end of the buffer. The "running total" may then be shifted by a number of bits as a simple method of performing division. (Shifting by one bit divides by 2, shifting by two bits divides by 4, etc.) The stored values may then moved along the buffer, or queue, the oldest value being discarded from the front of the queue such that the next input value may then be stored at the end. The cycle then repeats for the next input data value. (Alternatively - one might keep the data static in the queue and read and write to the queue via two pointers to form a circular buffer.)

Other approaches exist to define a working system - the choice is yours. The goal is to achieve a working system that fulfils the specification with the minimum of complexity and resources.

As a starting point it is an idea to keep things simple and start with a minimum system - for example - Generate only the square wave and average over 4 samples. Once this phase is working then other features may then be progressively developed and integrated.

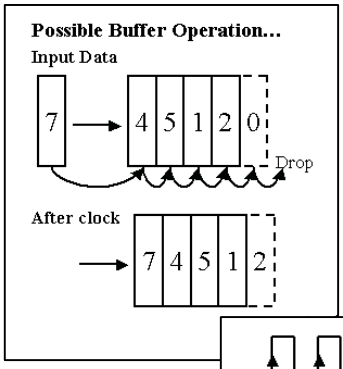
Some Initial Ideas…

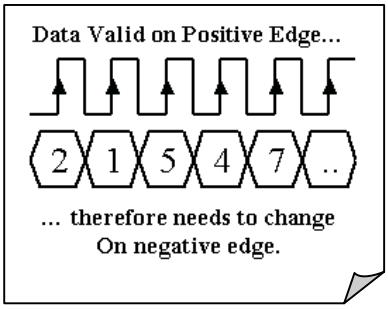
Example Rolling Average Over 4 Samples

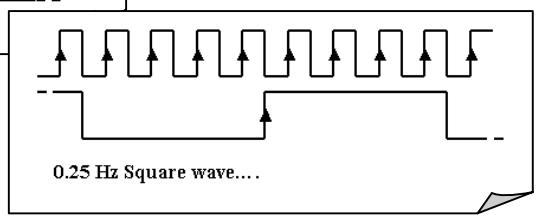
Sequence: 2 1 5 4 7 8 3 9 9 …

Values	Total	Average	Output
2 1 5 4	12	3.00	3
1 5 4 7	17	4.25	4
5 4 7 8	24	6.00	6
4 7 8 3	22	5.50	5 (or 6?)
7 8 3 9	27	6.75	6 (or 7?)
8 3 9 9	29	7.25	7

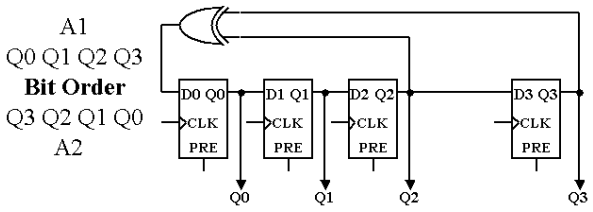
What happens at start up until buffer is full?







Generating Random Sequences



Sequence

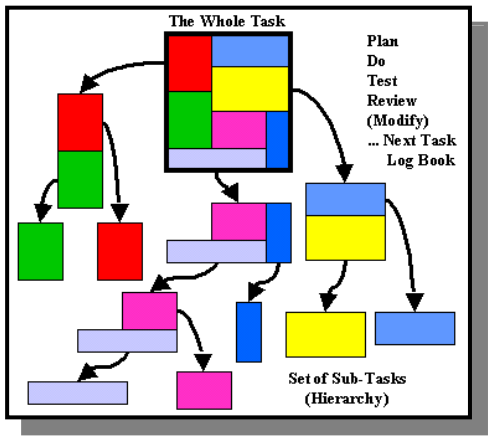
A1 - 15,14,12,8,1,2,4,9,3,6,13,10,5,11,7...

4-bit LFSR - Tap Points 0,1,4
8-bit LFSR - Tap Points 0,1,5,6,8
9-bit LFSR - Tap Points 0,4,9

The diagram on the left shows a **linear feedback shift register (LFSR)**. The exclusive-OR gates and shift register act to produce a **pseudo-random binary sequence (PRBS)** at each of the flip-flop outputs. By correctly choosing the **"Tap points"** at which we take the feedback from an n -bit shift register, we can produce a PRBS of length $2^n - 1$, a **maximal-length sequence** that includes all possible patterns (or vectors) of n bits, **excluding the all-zeros pattern**.
The diagram shows a **4-bit** linear feedback shift register (LFSR) that produces a repeating string of 15 pseudo-random binary numbers. Shown is one of four XOR Flip-flop configurations that exist for PBRS generators.
For a greater range of values (eg 255) then the number of bits in the LFSRs may be increased with specific tap points to generate the required maximum length pseudo-random binary sequences.

/Continued….

Method of Working



As with any systems design task reaching a solution may seem at first complex, difficult and overwhelming. The answer is to apply a methodical approach and break the task down into a set of simpler building blocks that are far easier to understand and model using VHDL i.e. **"Divide and Conquer"** - The systems integration phase then combines these building blocks together, again in a methodical way, to form the completed system. There is no specific order that building blocks need to be implemented but each block MUST be independently verified and tested before it is integrated into a system. It is also expected that the process of integration will also be a staged process (incremental) and that there will be verification and testing at each stage. **Trying to implement and test a complete system in "one step" normally ends in disaster and is not an acceptable approach.**

The design process therefore is a repeated cycle of: **"*Plan Do Test Review (modify... Next Task* "**