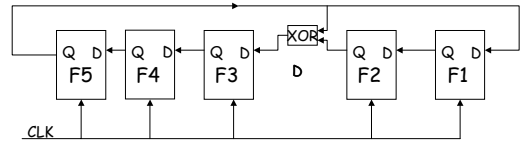


but while we wait for that to happen...

Lab 5 Shift Registers and Counters

Presented By
Neha Kumar

How to build a 5 bit LFSR



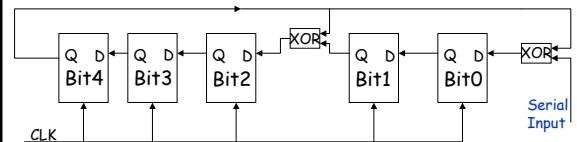
- Look up the "primitive polynomial": $x^5 + x^2 + 1$.
- x^2 implies an XOR between F2 and F3.
- Feed F5's output back to F1.
- Voila!



What is an LFSR?

- Linear Feedback Shift Register
- An n-bit counter that cycles through $2^n - 1$ different non-zero patterns in a pseudo-random fashion.
- In this lab, we use it for error checking.

Parity Bits / Error Checking



- Now let's allow for serial input.
- The last n bits constitute the parity bits.
- These must all be 0 to allow for error checking.
- Our input here, for example, must end with ...00000.

How to build any LFSR

- Place a **linear** array of D Flip Flops and **feed** the output of the last one **back** to the first i.e. $FF(n) \rightarrow FF(1)$.
- For an n-bit LFSR look at the table on Page 6 for a polynomial of degree n.
- For every term x^k in the polynomial compute the XOR of the outputs of $FF(n)$ and $FF(k)$ and feed it to $FF(k+1)$ as input (for $0 < k < n$).
- (By the way, FFs here are numbered 1-n.)

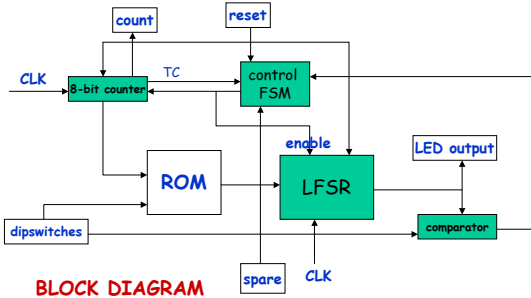
Example: Error Checking

A 4-bit LFSR:

- Serial Input: 1100 1000 111 (Note: 11 = 15-4 bits)
- 1100 1000 111 0000 \rightarrow 1010 (Parity bits = 1010)
- 1100 1000 111 1010 \rightarrow 0000 Correct!
- 1100 0000 111 1010 \rightarrow 0111 Introduce error in 5th bit.



When in lab...



Control FSM

- INPUTS
 - TC = counter output
 - Comparator output
 - Mode button (SPARE)
- STATES
 - **ACTIVE**: Enter on RESET. Enable counter and LFSR.
 - **DONE**: Enter when TC or PAR=LED go high.

Miss Wormwood: What state do you live in?

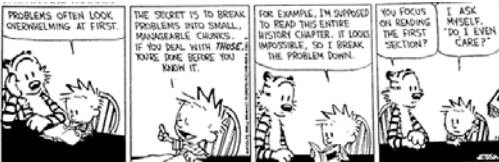
Calvin: Denial.

Miss Wormwood: I don't suppose I can argue with that...



...build the following

- An 8-bit counter (CC8CE) and comparator
 - Standard XILINX parts... easy.
- An 8-bit LFSR
 - Feed it the 255-bit sequence provided by the ROM.
 - Set the DIP switches on the board to provide the last 8 bits.
 - Don't forget the enable signal for the FFs.
- A Control FSM
 - Draw a state diagram and truth table.
 - Design the gate-level implementation.



The Lab!

- Run **MODE 1** with all DIP switches at 0. Get parity bits from LEDs.
- Enter parity bits on the DIP switches and check the LEDs. They should read "00000000".
- Introduce an error in the ROM schematic.
- Use **MODE 1** to detect the error and **MODE 0** to find its position.

Modes of Operation

- Mode 1
 - Shift 255 bits from the ROM.
 - Display the results on the 8 LEDs.
- Mode 0
 - Process the ROM output.
 - Stop when the LFSR pattern matches the DIP switches.
 - The position of the error, if one exists, is displayed.
- Modes are controlled by the SPARE button.

A Side Note

- **GALOIS FIELDS**
 - Fields can be infinite. Like the set of reals or complex numbers.
 - Or they can be finite. These are called Galois fields.
 - Binary numbers form a Galois field where XOR serves as addition and AND serves as multiplication.

AS FAR AS I'M CONCERNED, IF SOMETHING IS SO COMPLICATED THAT YOU CAN'T EXPLAIN IT IN 10 SECONDS, THEN IT'S PROBABLY NOT WORTH KNOWING ANYWAY.





Come prepared...

- Read the handout.
- Read it again.
- Draw the schematics.
- Do the problems on the check-off sheet.