Lecture 18, March 28, 2007

Universal Shift Register

In the previous lecture, we learned about registers that load bits in parallel or serially. The latter called shift registers input bits from one side, say from the left, and the data is read bit-by-bit from the other side, right. A more general type of register is the one that

- Can load data either serially or in parallel
- Can load data either from left or right
- Shift data either to left or right.

Such a register is called Universal Shift Register. To implement such an register, we need n flip-flop to hold the n-bits, we also need to instruct the circuit to:

- Keep the contents of the register as is, i.e., no change.
- Shift bits to the right.
- Shift bits to the left.
- Load bits in parallel.

To give these four (4) commands, we need to bits. We assign them as follows

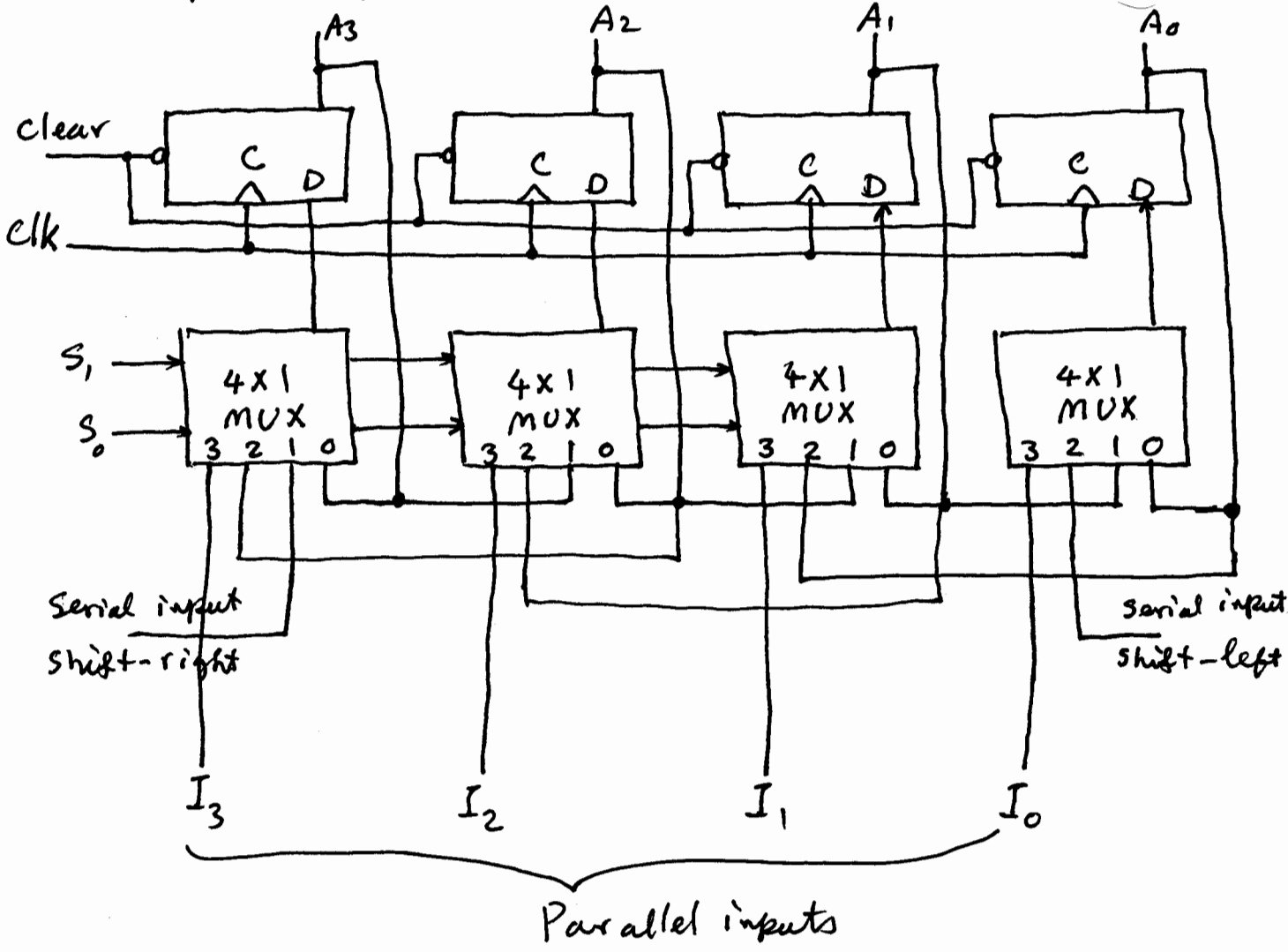
Control bits		Operation
S_1	S_0	
0	0	No change
0	1	shift right
1	0	Shift Left
1	1	Parallel Load

To implement the commands, we can use n multiplexers, each with 4 inputs and two control (select) lines.

- When $S_1 S_0$ is 00 the input 0 is selected in order to make no change, the output of the corresponding FF has to be connected to the input 0.
- When $S_1 S_0$ is 01 (shift right), we need to connect the output of the flip-flop that is on the left side to the input of the FF. So, we need to connect the output of the adjacent left FF

to input 1 of the multiplexer.

- Similarly we connect the output of the right-adjacent FF to input 2 of the MUX.
- The parallel load input I_i will be connected to input 3, i.e., the one selected when $S_1 S_0 = 11$.



Counters

A counter is a register whose content (equivalently, its state) changes according to a prescribed sequence. For a example a binary counter starts from 00...00 moves to 00...01, 00...10, ... and 11...11 and then to 00...00.

A BCD counter counts from 0000 to 1001.

There are two types of Counters:

- Ripple Counters.
- Synchronous Counters.

In a Synchronous Counter, a common clock connected to C inputs of all flip-flops dictates the transition of the state of flip-flops.

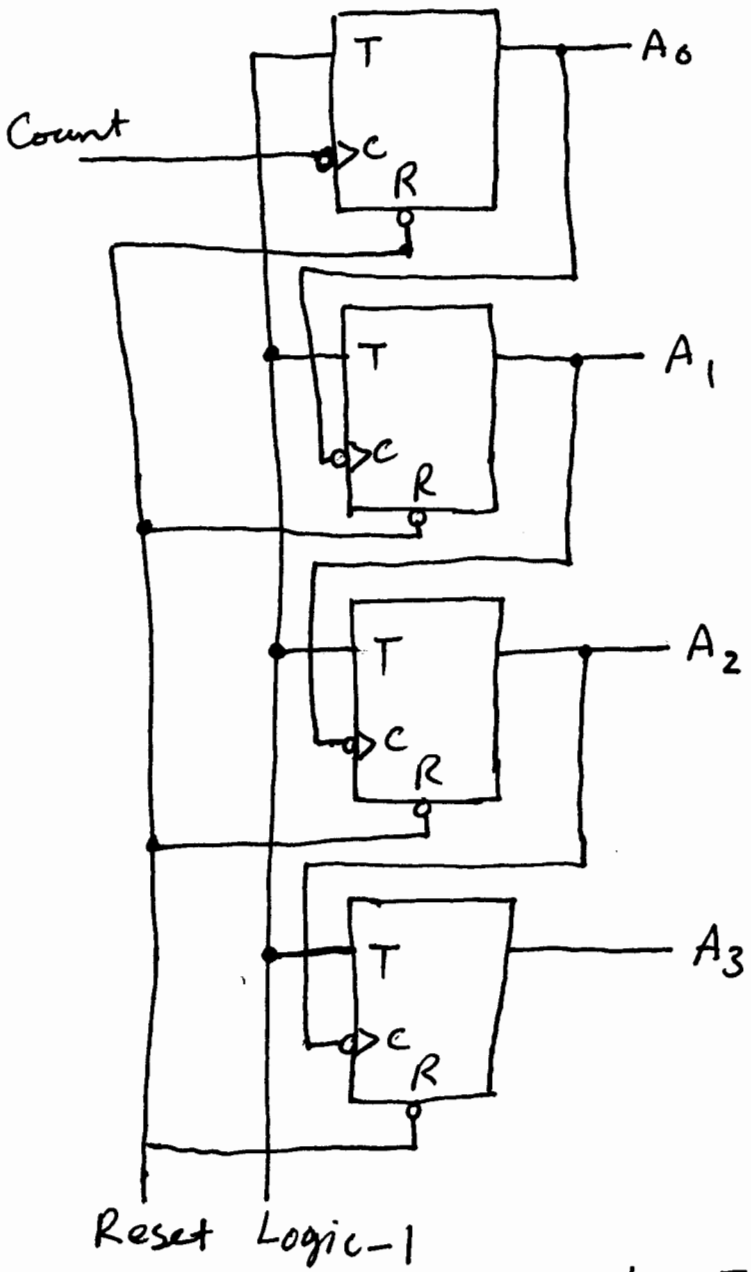
In a ripple counter, the output of one flip-flop has effect on the next state of the following flip-flop.

Binary Ripple Counter

A binary ripple-counter consists of a series connection of flip-flops where the output of

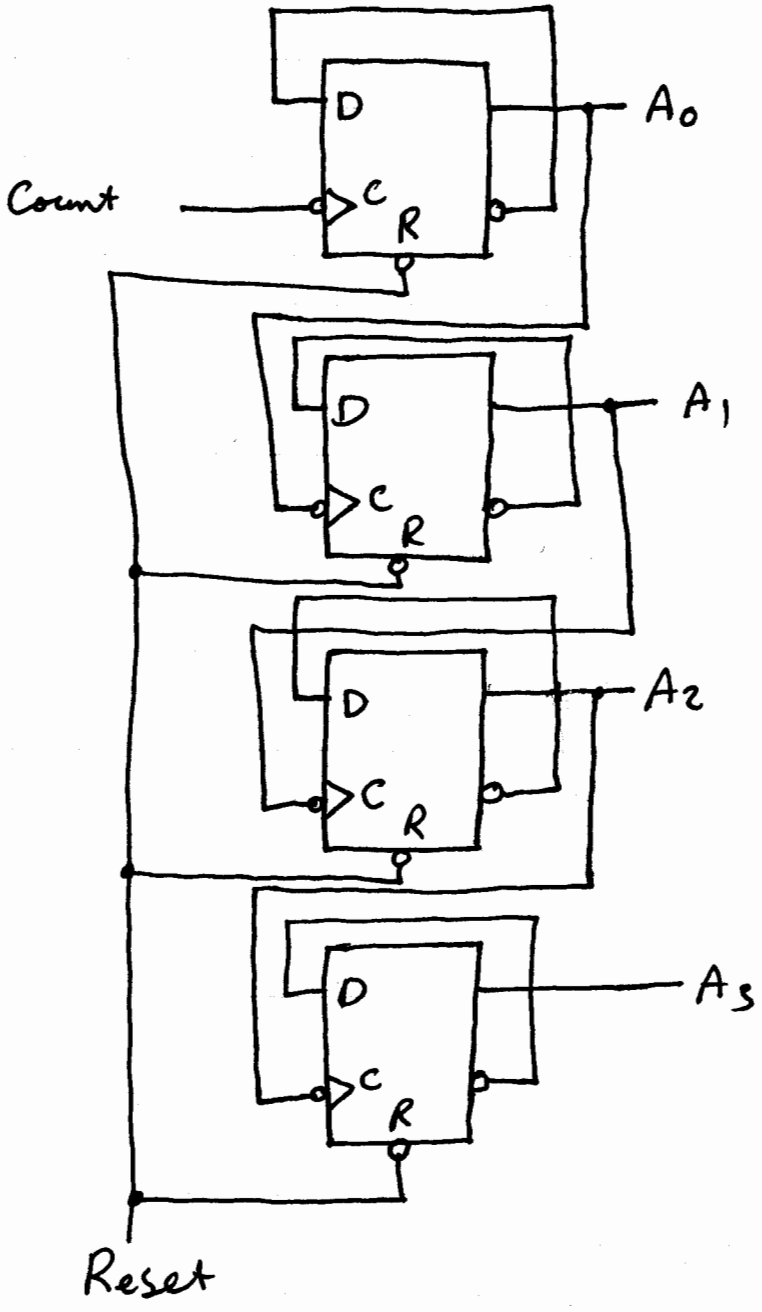
each flip-flop is connected to the clock (C) input of the next flip-flop. So, change of state of each flip-flop affects the next state of the following flip-flop.

Below is the circuit diagram of a 4-bit ripple counter implemented with T flip-flops.



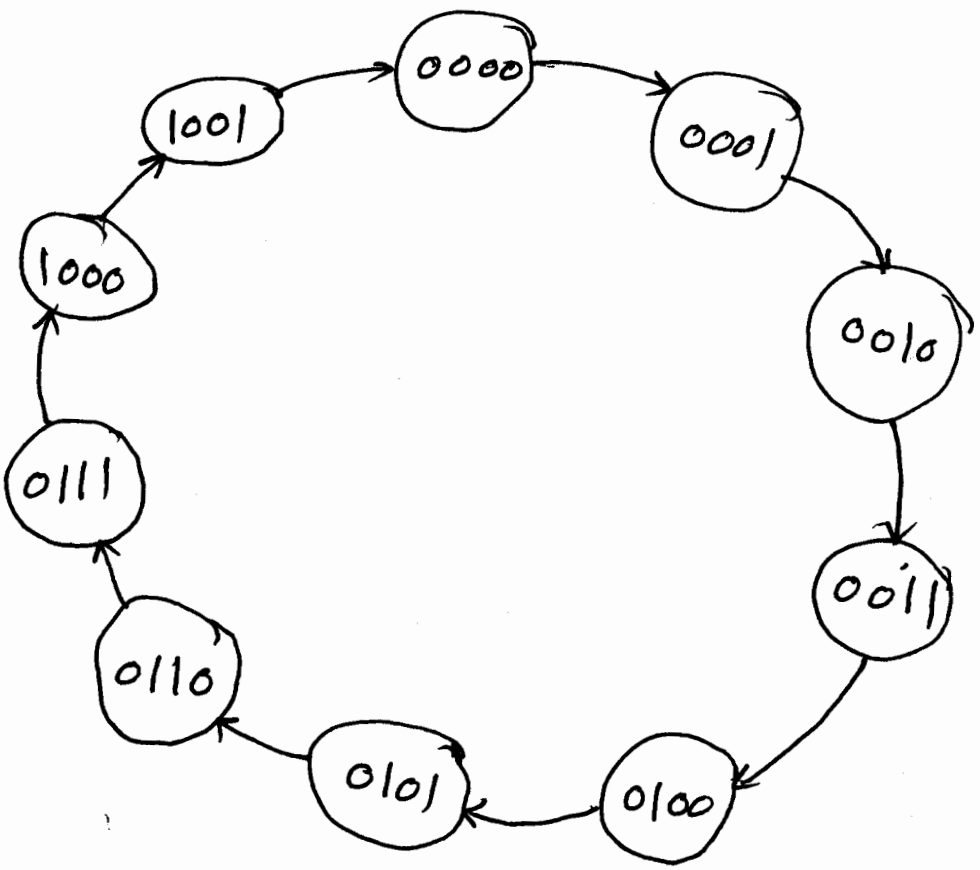
Note that all T inputs are connected to Logic-1.
So, anytime the C input goes from 1 to 0 the output of the flip-flop changes.
Assume that the counter is in state 0000, i.e., $A_3 = A_2 = A_1 = A_0 = 0$.
If a count pulse is applied, on the falling edge the count pulse (connected to C of the least significant flip-flop), A_0 goes from 0 to 1.
So, the counter will show 0001.
The next Count will toggle A_0 to 0. This will cause A_1 to go to 1. So, the count will be 0010.
The third count will change A_0 to 1 $\Rightarrow$ 0011.
The fourth countpulse will reset A_0. Resetting of A_0 resets A_1. This in turn will set $A_2 \rightarrow$ 0100.
This will continue untill the count is 1111.
A count applied when Count is 1111 results in reseting of $A_0 \rightarrow A_1 \rightarrow A_2 \rightarrow A_3$, i.e., State = 0000.

The implementation of a 4-bit ripple counter using D-flip-flops is:



BCD ripple Counter

A BCD counter counts from 0000 (zero) to 1001 (nine) and goes back to 0000. So after 1001, we need a logic to prevent the state to go to 1010 (ten).
We can implement a BCD ripple counter using 4 JK flip-flops. The state diagram is as follows



The circuit diagram is: