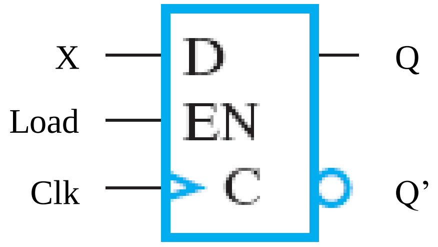
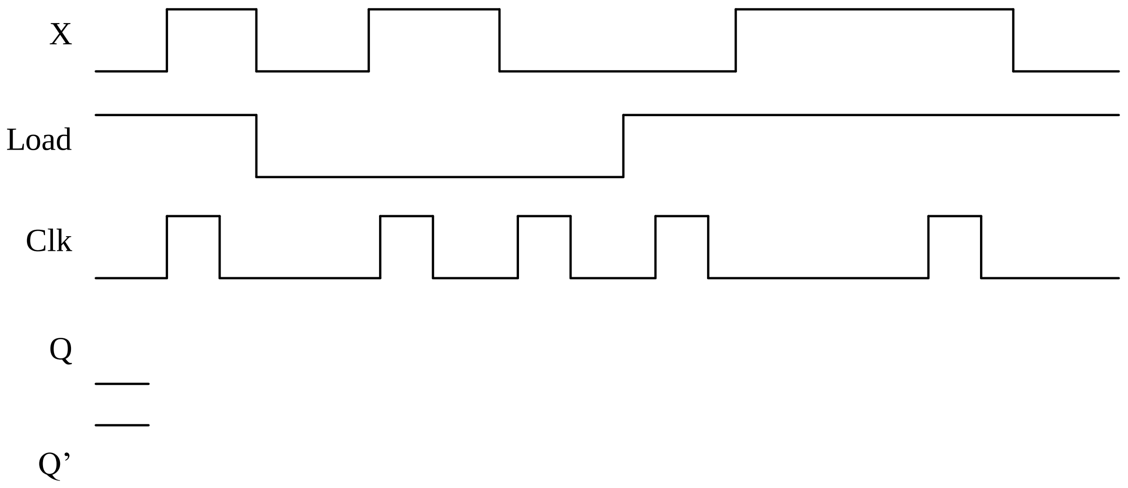
Tutorial 7
Registers and RTL

CO 2206 Computer Organization

Task 1. Register

- Knowing the function of a D flip-flop with enable, complete the following timing diagram (for the outputs). Assume initially Q=0 and Q'=1.





Task 2. Shift Register

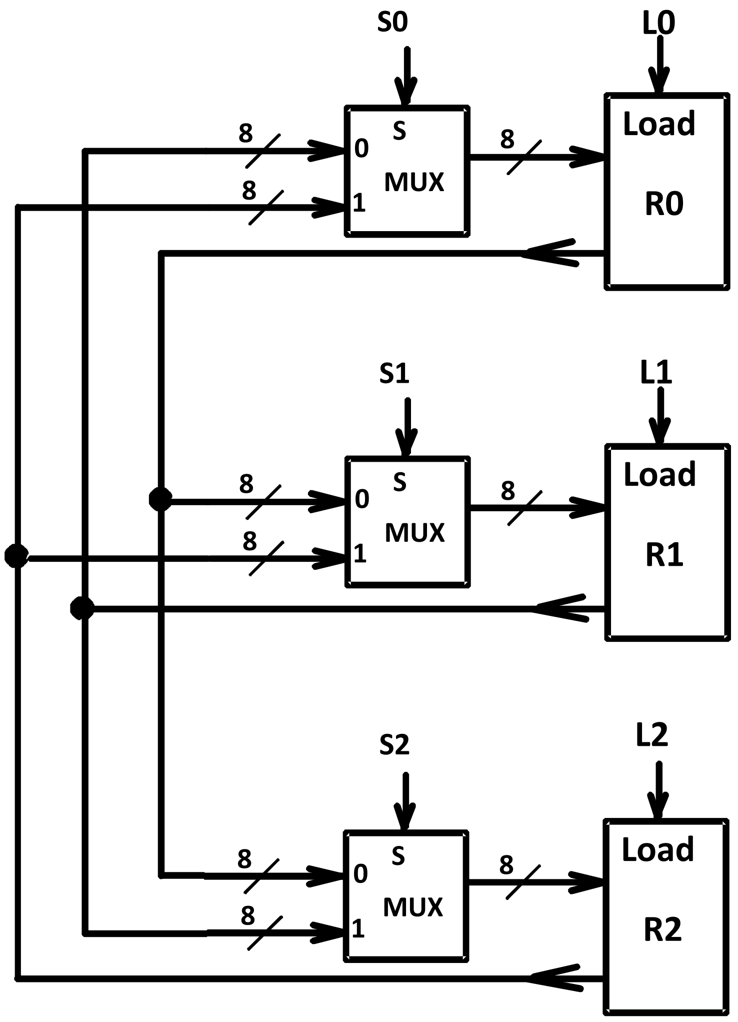
- Modify the 4-bit universal shift register on Slide 19, Chapter 7, such that the functions are:
  - when s1s0 = 00,
    - no change of states
  - when s1s0 = 01,
    - parallel load
  - when s1s0 = 10,
    - shift right
  - when s1s0 = 11,
    - shift left

Task 3. Register Cell Design

- Draw the circuit diagram to implement the ad-hoc register cell design on Slide 41, Chapter 7
- By referring to the ad-hoc design above, design (and draw the circuit of) a register cell to implement the following register transfer:
  - Op1: $A \leftarrow A'$
  - Op2: $A \leftarrow A + B$
  - Op3: $A \leftarrow A - B$
  - Assume that
    - Only one of Op1, Op2, Op3 is equal to 1
    - For Op1, Op2, Op3 equal to 0, A remains unchanged
    - You can use NOT gate(s) and a Full Adder in your design.

Task 4. Mux-Based Transfer

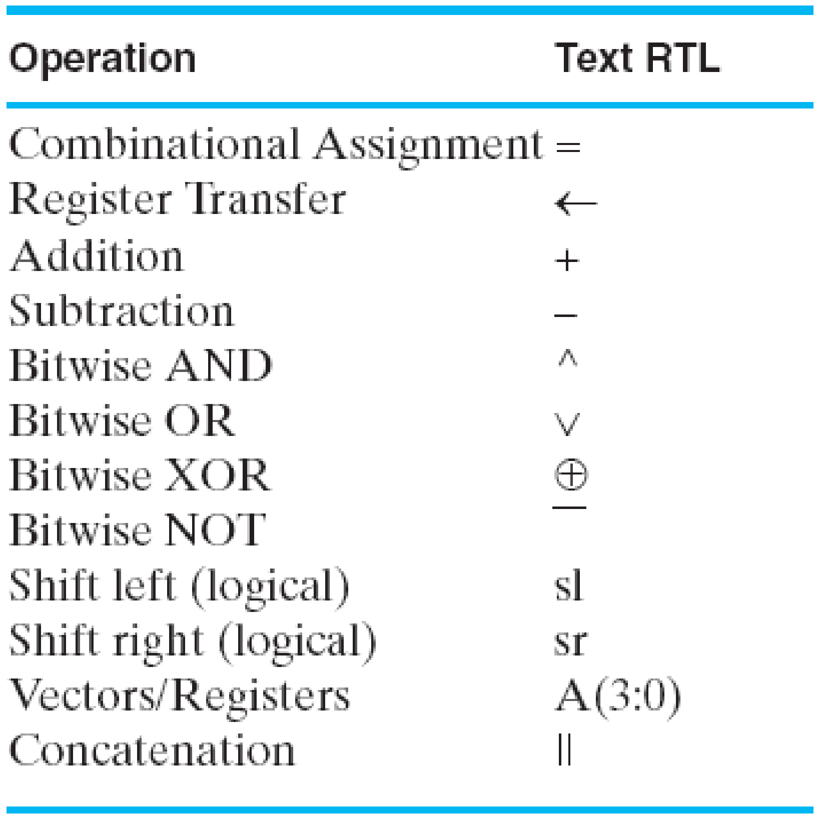
- Determine the value for the select (S2, S1, S0) signals and load (L2, L1, L0) signals to perform the following register transfer operations:
  - `R0 ← R1`
  - `R1 ← R2`
  - `R1 ← R0, R2 ← R0`



Task 5. RTL

- Express the following statement in *RTL*

```
if (ABC = 001) then
(R0 = R1) else if
(ABC = 010) then (R0
= R2) else if (ABC =
011) then (R0 = R3)
else if (ABC = 101)
then (R0 = R0 OR R1)
```



Operation	Text RTL
Combinational Assignment	=
Register Transfer	$\leftarrow$
Addition	+
Subtraction	−
Bitwise AND	^
Bitwise OR	$\vee$
Bitwise XOR	$\oplus$
Bitwise NOT	$\overline{}$
Shift left (logical)	sl
Shift right (logical)	sr
Vectors/Registers	A(3:0)
Concatenation	||

Task 6. Serial Register Operation

- Modify the circuit on Slide 59 of Chapter 7 to perform a serial operation of the following logic function:
  - **`B = A'B where A and B are both 16-bit`**

Task 7

- **Task 7:** Use 4-bit binary counter with synchronous parallel load (in block diagram) and logic gates to design the following counters:
  a. Modulo-3 that counts 0,1,2 repeatedly
  b. Modulo-6 that counts 2,3,4,5,6,7 repeatedly