A: 1010 (10)
B: 0101 (5)
=110010

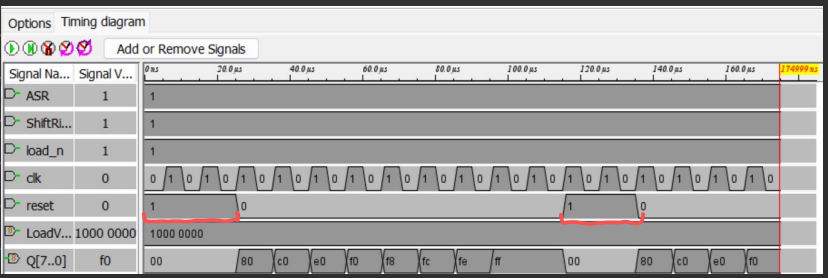
```
  1010
  0101 x
 ------
  1010
 0000 x
1010 xx
0000 xxx +
--------
0110010
```

$D_{3:0}$	Character
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	b
1100	c
1101	d
1110	E
1111	F

5. Simulate your modules with *Poke*. Choose test cases that make you feel confident about your shifter's correctness, in preparation for your in-lab demo. Make sure to include a few selected screenshots of these cases when you hand in your prelab.

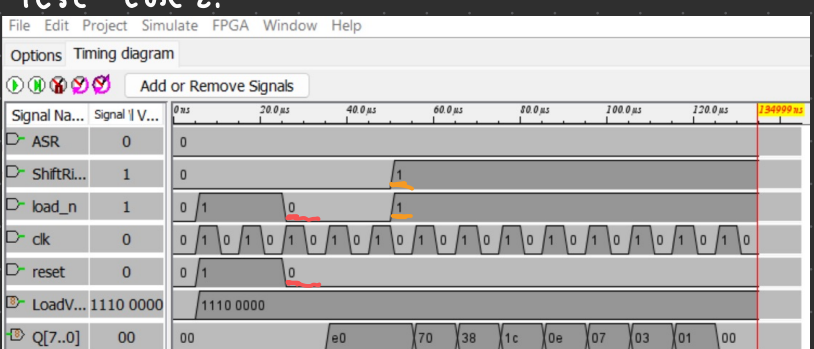
   In your simulation, you should perform the reset operation on the first clock cycle, then do a parallel load of your register on the next cycle. Finally, clock the register for several cycles to demonstrate both types of shifts. *(NOTE: If you do not perform a reset first, your simulation will not work! Try simulating without doing reset first and see what happens. Can you explain the results?)*

Test - Case 1:



- Asr and ShiftRight = 1
  → Preforms an Arethmetic-Right-Shift
- load_n = 1:
  → We are loading 10000000, but since n is never 0, we never load
- Reset
  → Reset a few times
- load_val = 10000000:
  → Set a load_val, to 10000000
- Out:
  → [-128, -64, -32, -16, -8, -4, -2, -1] (signed)
  → Shifting Right => dividing by 2
  We keep - sign since we do ARS

Test - Case 2:



- Testing logical right shift,
- Turned reset = 0, and loan_n = 0, to load 1110 0000 *
- Turned shifter = 1, and load_n = 1, to stop loading, and start shifter *
  Output is [1110 0000, 0111 0000, 0011 1000, ...]
  Each clock cycle preforms a logical right shift.