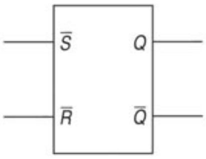
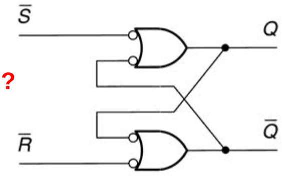
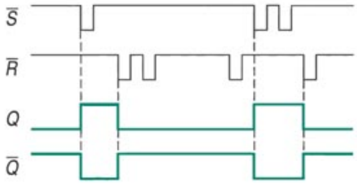
Class 05
Sequential Logic: Latch

SR NAND Latch



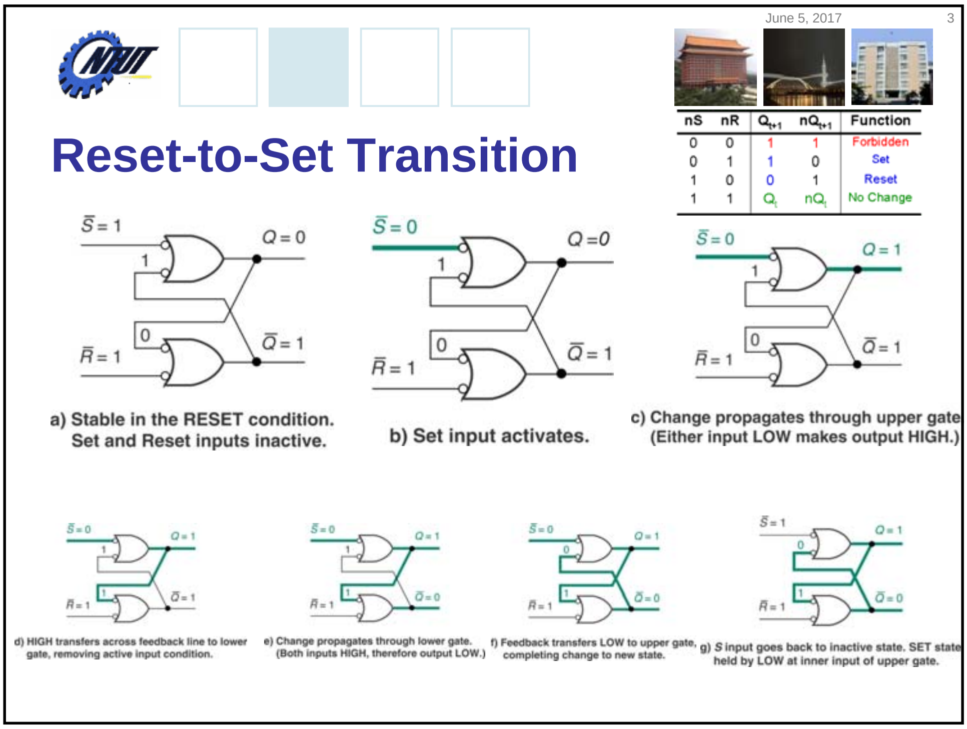
Why called NAND Latch?





nS	nR	Q_{t+1}	nQ_{t+1}	Function
/S	/R		$/Q_{t+1}$	
0	0	1	1	Forbidden
0	1	1	0	Set
1	0	0	1	Reset
1	1	Q_t	nQ_t	No Change

SR NAND Latch Function Table



Reset-to-Set Transition

nS	nR	Q_{t+1}	nQ_{t+1}	Function
0	0	1	1	Forbidden
0	1	1	0	Set
1	0	0	1	Reset
1	1	Q_t	nQ_t	No Change

a) Stable in the RESET condition. Set and Reset inputs inactive.

b) Set input activates.

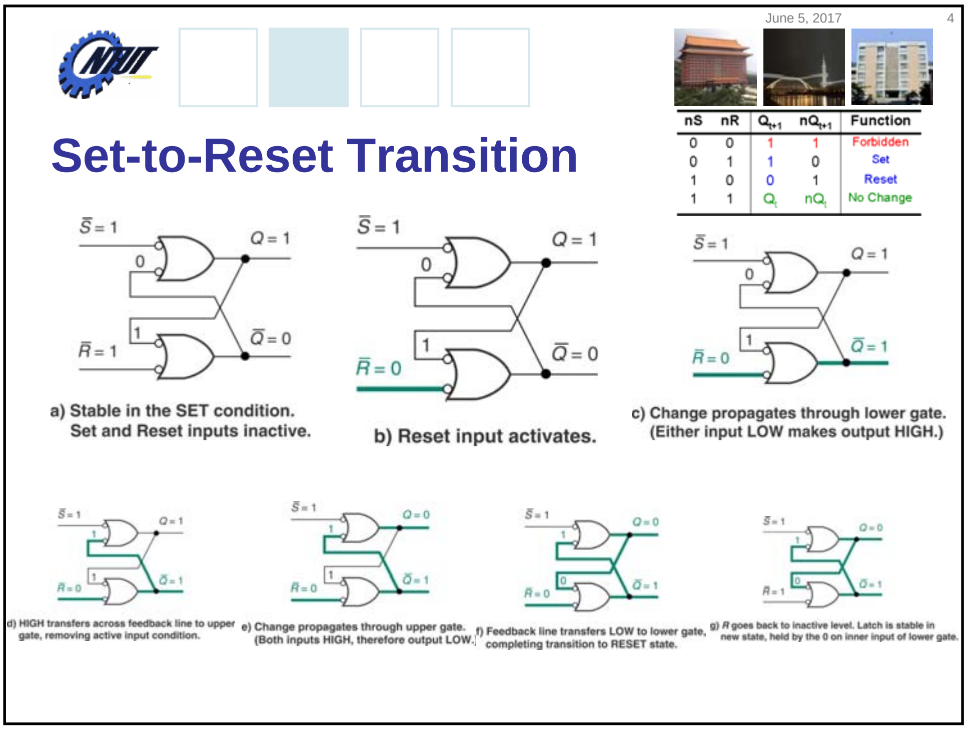
c) Change propagates through upper gate (Either input LOW makes output HIGH.)

d) HIGH transfers across feedback line to lower gate, removing active input condition.

e) Change propagates through lower gate. (Both inputs HIGH, therefore output LOW.)

f) Feedback transfers LOW to upper gate, completing change to new state.

g) *S* input goes back to inactive state. SET state held by LOW at inner input of upper gate.



Set-to-Reset Transition

nS	nR	Q_{t+1}	nQ_{t+1}	Function
0	0	1	1	Forbidden
0	1	1	0	Set
1	0	0	1	Reset
1	1	Q_t	nQ_t	No Change

a) Stable in the SET condition. Set and Reset inputs inactive.

b) Reset input activates.

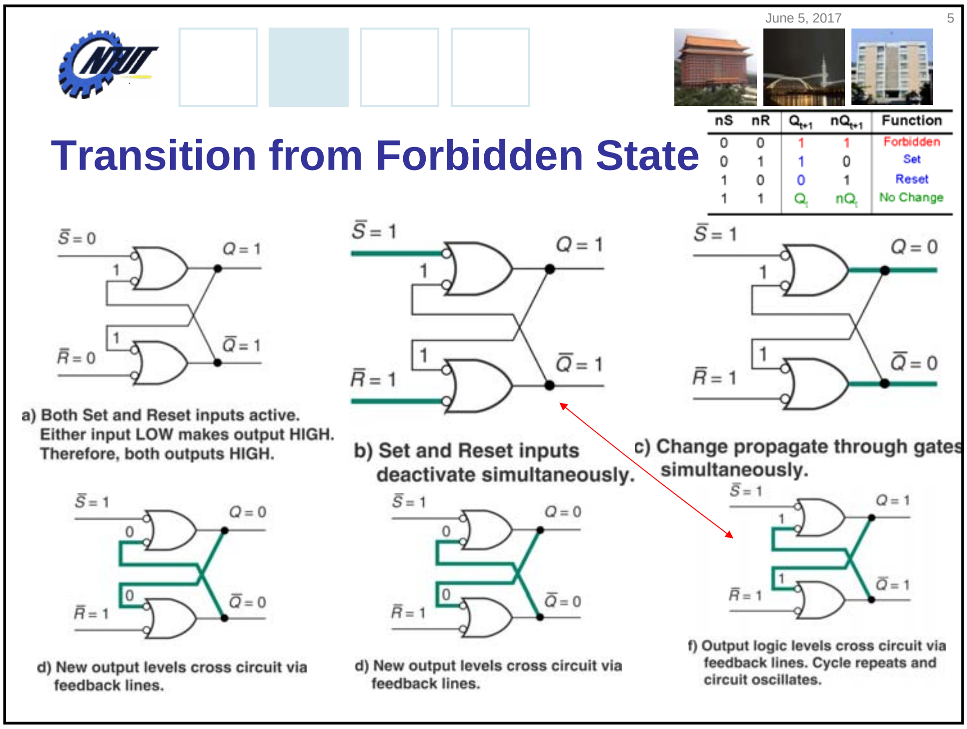
c) Change propagates through lower gate. (Either input LOW makes output HIGH.)

d) HIGH transfers across feedback line to upper gate, removing active input condition.

e) Change propagates through upper gate. (Both inputs HIGH, therefore output LOW.)

f) Feedback line transfers LOW to lower gate, completing transition to RESET state.

g) *R* goes back to inactive level. Latch is stable in new state, held by the 0 on inner input of lower gate.



Transition from Forbidden State

nS	nR	Q_{t+1}	nQ_{t+1}	Function
0	0	1	1	Forbidden
0	1	1	0	Set
1	0	0	1	Reset
1	1	Q_t	nQ_t	No Change

a) Both Set and Reset inputs active. Either input LOW makes output HIGH. Therefore, both outputs HIGH.

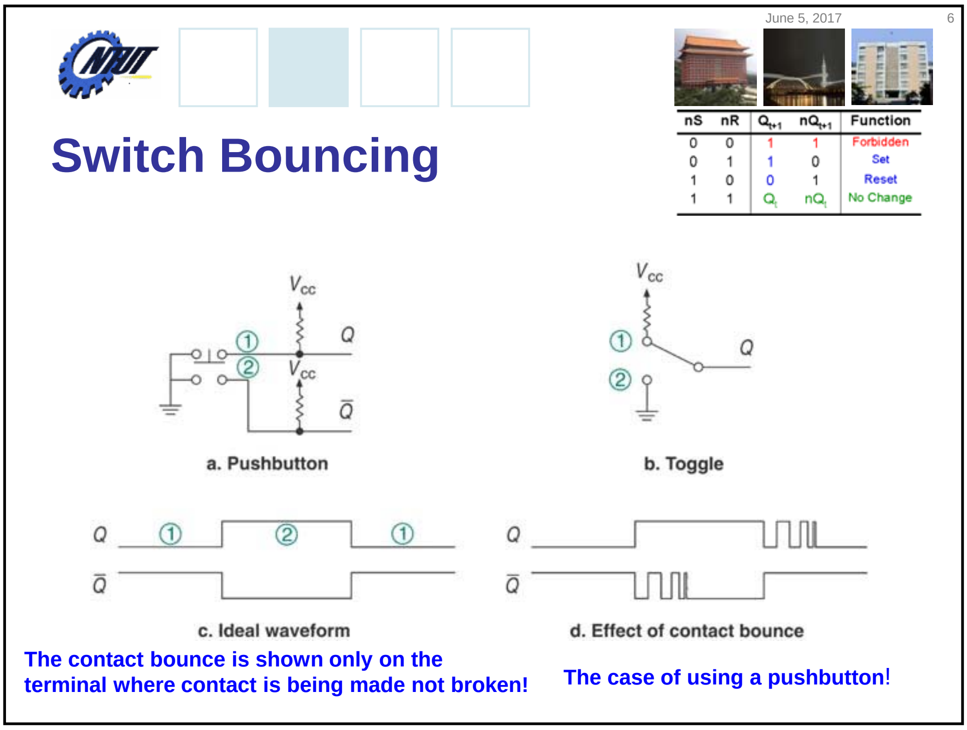
b) Set and Reset inputs deactivate simultaneously.

c) Change propagate through gates simultaneously.

d) New output levels cross circuit via feedback lines.

d) New output levels cross circuit via feedback lines.

f) Output logic levels cross circuit via feedback lines. Cycle repeats and circuit oscillates.



Switch Bouncing

nS	nR	Q_{t+1}	nQ_{t+1}	Function
0	0	1	1	Forbidden
0	1	1	0	Set
1	0	0	1	Reset
1	1	Q_t	nQ_t	No Change

a. Pushbutton

b. Toggle

c. Ideal waveform

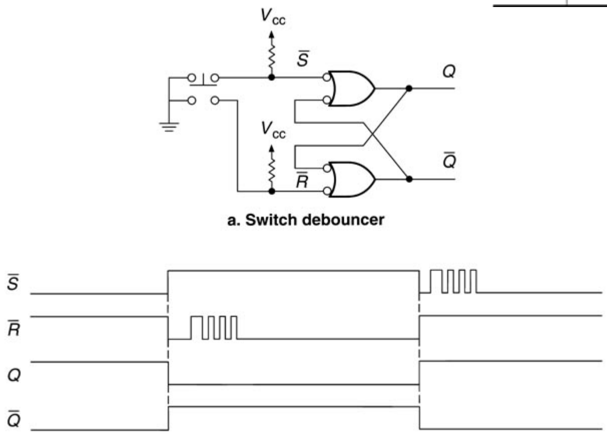
d. Effect of contact bounce

The contact bounce is shown only on the terminal where contact is being made not broken!

The case of using a pushbutton!

nS	nR	Q_{t+1}	nQ_{t+1}	Function
0	0	1	1	Forbidden
0	1	1	0	Set
1	0	0	1	Reset
1	1	Q_t	nQ_t	No Change

Switch Debouncing



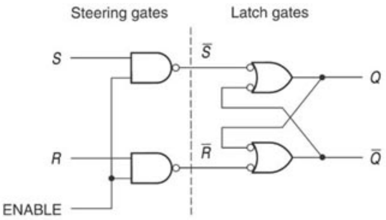
a. Switch debouncer

b. Timing diagram

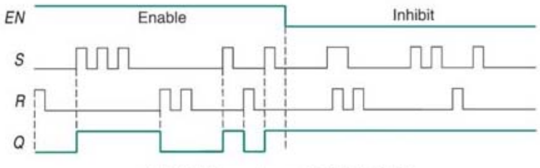
nS	nR	Q_{t+1}	nQ_{t+1}	Function
0	0	1	1	Forbidden
0	1	1	0	Set
1	0	0	1	Reset
1	1	Q_t	nQ_t	No Change

SR NAND Latch Truth Table

Gated SR NAND Latch



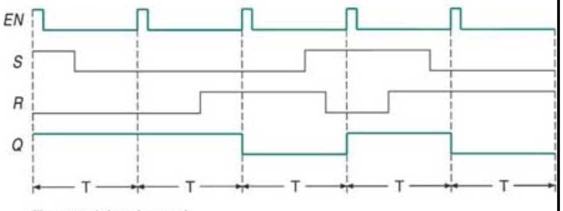
a. Circuit



b. ENABLE used as an ON/OFF signal

EN	S	R	Q_{t+1}	nQ_{t+1}	Function
1	0	0	Q_t	nQ_t	No Change
1	0	1	0	1	Reset
1	1	0	1	0	Set
1	1	1	1	1	Forbidden
0	X	X	Q_t	nQ_t	Inhibited

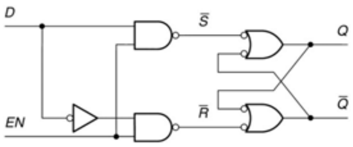
Gated NAND Latch Function Table



T = equal time interval

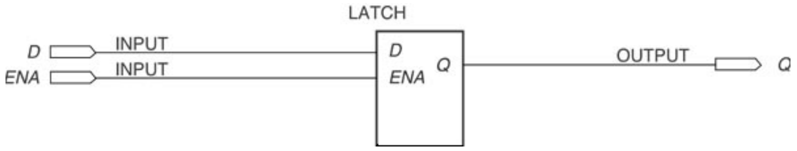
c. ENABLE used as a synchronizing signal

Gated D Latch (Transparent Latch)



EN	D	Q_{t+1}	nQ_{t+1}	Function	Comment
1	1	1	0	Set	
1	0	0	1	Reset	Transparent
0	X	Q_t	nQ_t	No Change	Store

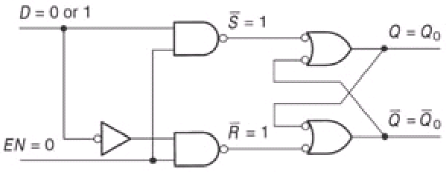
Gated D Latch Function Table



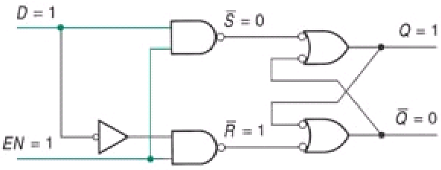
Gated D Latch (Cont.)

nS	nR	Q_{t+1}	nQ_{t+1}	Function
0	0	1	1	Forbidden
0	1	1	0	Set
1	0	0	1	Reset
1	1	Q_t	nQ_t	No Change

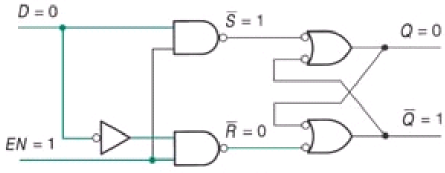
SR NAND Latch Truth Table



a. No change state



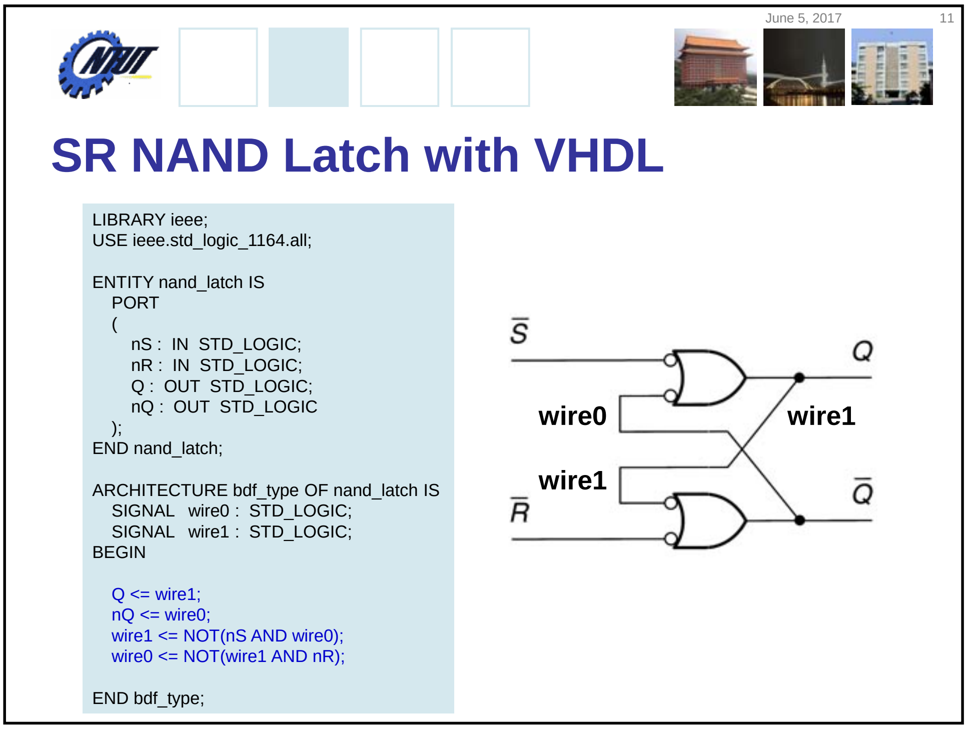
b. Set condition



c. Reset condition

EN	D	Q_{t+1}	nQ_{t+1}	Function	Comment
1	1	1	0	Set	
1	0	0	1	Reset	Transparent
0	X	Q_t	nQ_t	No Change	Store

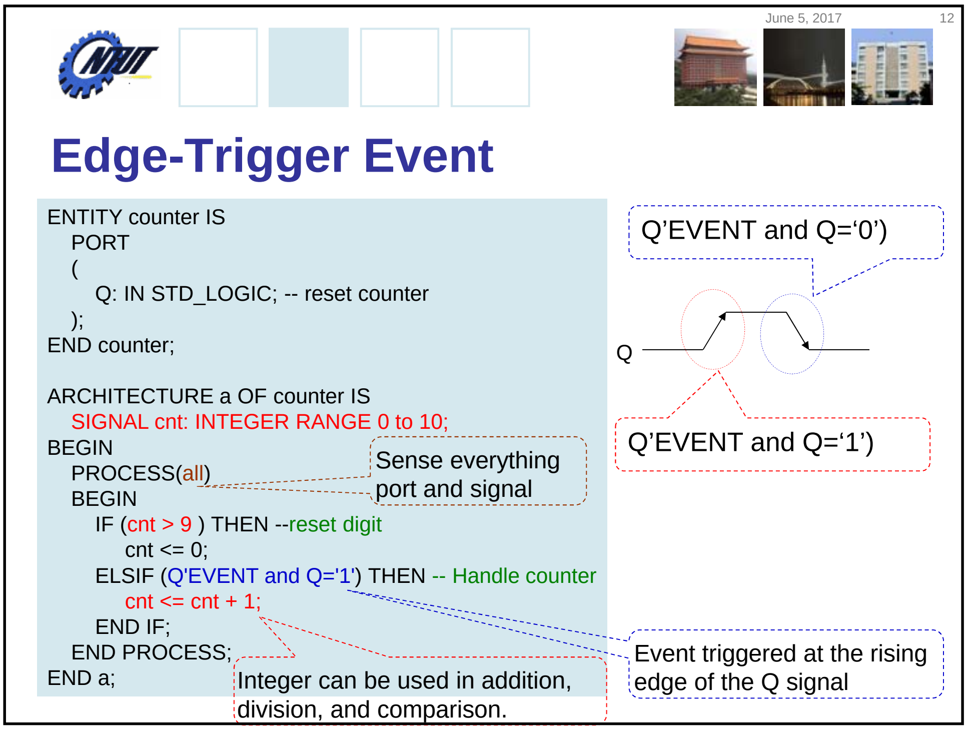
Gated D Latch Truth Table



June 5, 2017 11

SR NAND Latch with VHDL

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY nand_latch IS
    PORT
    (
        nS :  IN  STD_LOGIC;
        nR :  IN  STD_LOGIC;
        Q :  OUT  STD_LOGIC;
        nQ :  OUT  STD_LOGIC
    );
END nand_latch;

ARCHITECTURE bdf_type OF nand_latch IS
    SIGNAL   wire0 :  STD_LOGIC;
    SIGNAL   wire1 :  STD_LOGIC;
BEGIN

    Q <= wire1;
    nQ <= wire0;
    wire1 <= NOT(nS AND wire0);
    wire0 <= NOT(wire1 AND nR);

END bdf_type;
```



June 5, 2017 12

Edge-Trigger Event

```
ENTITY counter IS
    PORT
    (
        Q: IN STD_LOGIC; -- reset counter
    );
END counter;

ARCHITECTURE a OF counter IS
    SIGNAL cnt: INTEGER RANGE 0 to 10;
BEGIN
    PROCESS(all)
    BEGIN
        IF (cnt > 9 ) THEN --reset digit
            cnt <= 0;
        ELSIF (Q'EVENT and Q='1') THEN -- Handle counter
            cnt <= cnt + 1;
        END IF;
    END PROCESS;
END a;
```

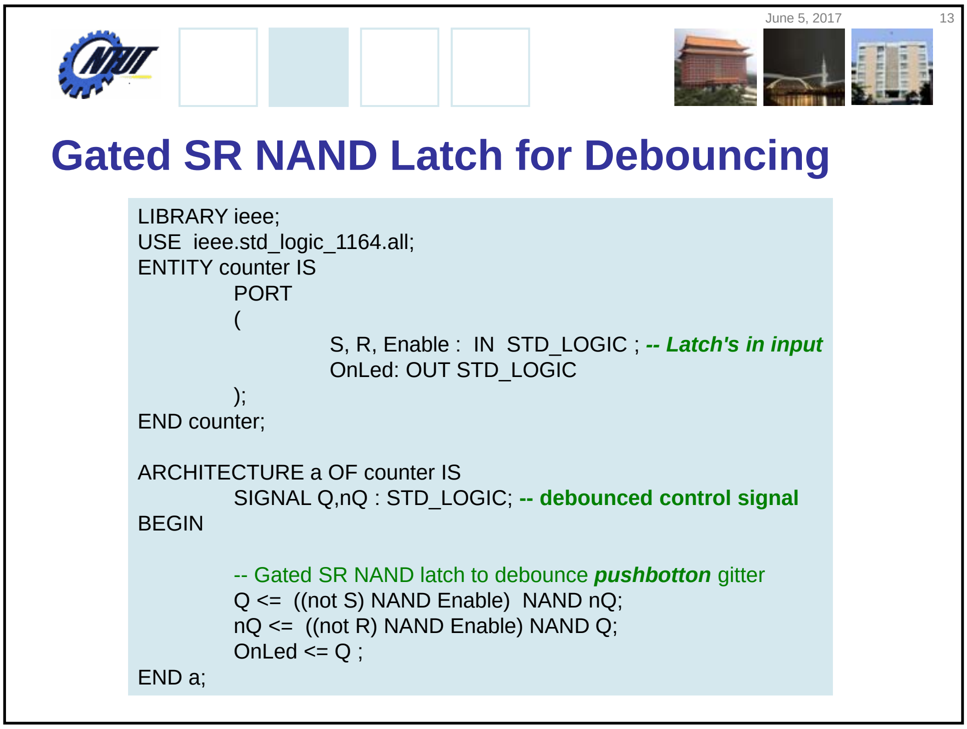
Sense everything port and signal

Integer can be used in addition, division, and comparison.

Event triggered at the rising edge of the Q signal

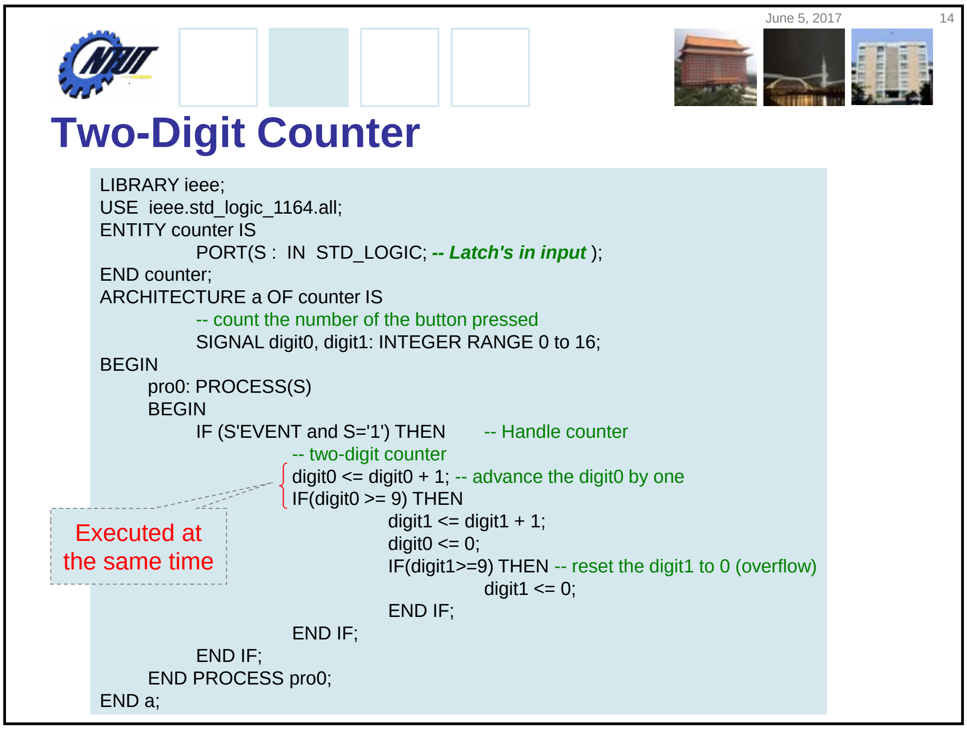
Q'EVENT and Q='0')

Q'EVENT and Q='1')



Gated SR NAND Latch for Debouncing

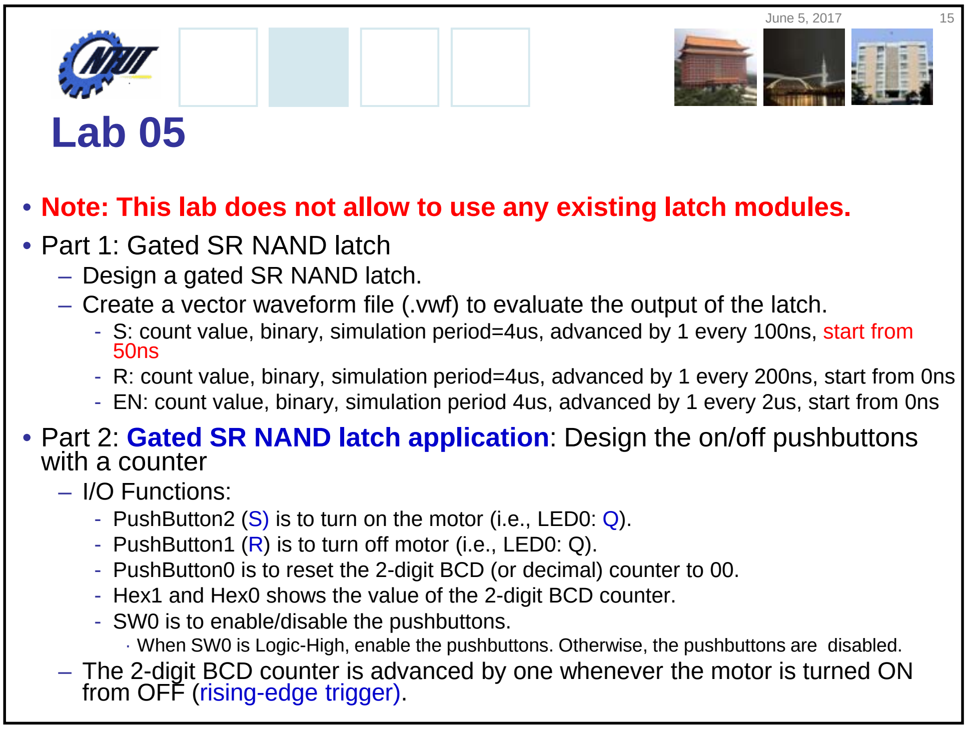
```
LIBRARY ieee;
USE  ieee.std_logic_1164.all;
ENTITY counter IS
        PORT
        (
                S, R, Enable :  IN  STD_LOGIC ; -- Latch's in input
                OnLed: OUT STD_LOGIC
        );
END counter;

ARCHITECTURE a OF counter IS
        SIGNAL Q,nQ : STD_LOGIC; -- debounced control signal
BEGIN

        -- Gated SR NAND latch to debounce pushbotton gitter
        Q <=  ((not S) NAND Enable)  NAND nQ;
        nQ <=  ((not R) NAND Enable) NAND Q;
        OnLed <= Q ;
END a;
```



Two-Digit Counter

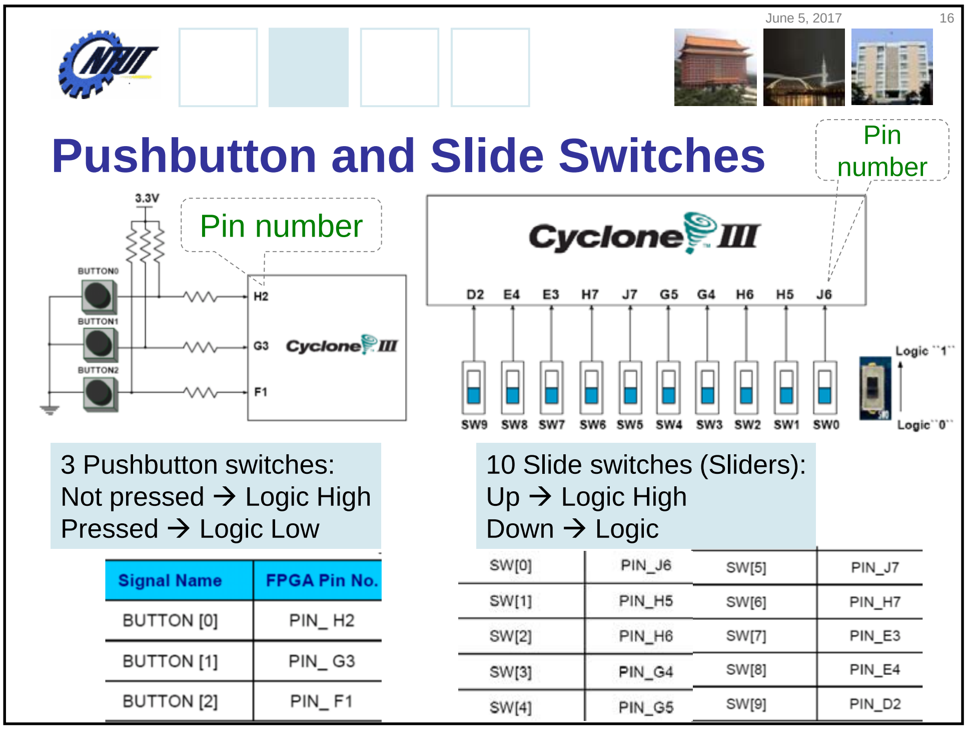
```
LIBRARY ieee;
USE  ieee.std_logic_1164.all;
ENTITY counter IS
        PORT(S :  IN  STD_LOGIC; -- Latch's in input );
END counter;
ARCHITECTURE a OF counter IS
        -- count the number of the button pressed
        SIGNAL digit0, digit1: INTEGER RANGE 0 to 16;
BEGIN
    pro0: PROCESS(S)
    BEGIN
        IF (S'EVENT and S='1') THEN       -- Handle counter
            -- two-digit counter
            digit0 <= digit0 + 1; -- advance the digit0 by one
            IF(digit0 >= 9) THEN
                digit1 <= digit1 + 1;
                digit0 <= 0;
                IF(digit1>=9) THEN -- reset the digit1 to 0 (overflow)
                    digit1 <= 0;
                END IF;
            END IF;
        END IF;
    END PROCESS pro0;
END a;
```

Executed at the same time



Lab 05

- **Note: This lab does not allow to use any existing latch modules.**
- Part 1: Gated SR NAND latch
  - Design a gated SR NAND latch.
  - Create a vector waveform file (.vwf) to evaluate the output of the latch.
    - S: count value, binary, simulation period=4us, advanced by 1 every 100ns, start from 50ns
    - R: count value, binary, simulation period=4us, advanced by 1 every 200ns, start from 0ns
    - EN: count value, binary, simulation period 4us, advanced by 1 every 2us, start from 0ns
- Part 2: **Gated SR NAND latch application**: Design the on/off pushbuttons with a counter
  - I/O Functions:
    - PushButton2 (S) is to turn on the motor (i.e., LED0: Q).
    - PushButton1 (R) is to turn off motor (i.e., LED0: Q).
    - PushButton0 is to reset the 2-digit BCD (or decimal) counter to 00.
    - Hex1 and Hex0 shows the value of the 2-digit BCD counter.
    - SW0 is to enable/disable the pushbuttons.
      - When SW0 is Logic-High, enable the pushbuttons. Otherwise, the pushbuttons are disabled.
  - The 2-digit BCD counter is advanced by one whenever the motor is turned ON from OFF (rising-edge trigger).



Pushbutton and Slide Switches

3 Pushbutton switches:
Not pressed → Logic High
Pressed → Logic Low

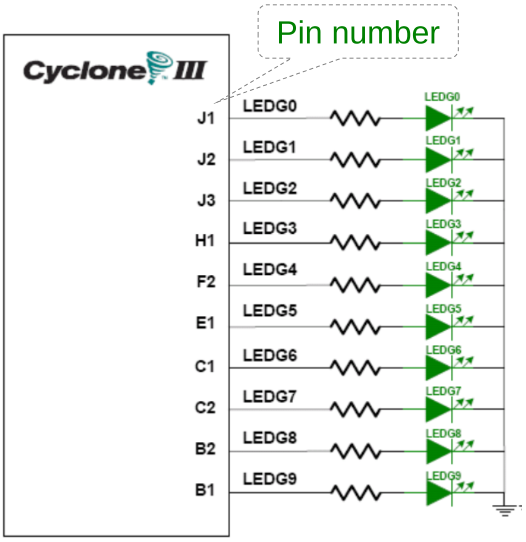
10 Slide switches (Sliders):
Up → Logic High
Down → Logic

Signal Name	FPGA Pin No.
BUTTON [0]	PIN_ H2
BUTTON [1]	PIN_ G3
BUTTON [2]	PIN_ F1

SW[0]	PIN_J6	SW[5]	PIN_J7
SW[1]	PIN_H5	SW[6]	PIN_H7
SW[2]	PIN_H6	SW[7]	PIN_E3
SW[3]	PIN_G4	SW[8]	PIN_E4
SW[4]	PIN_G5	SW[9]	PIN_D2

LEDs

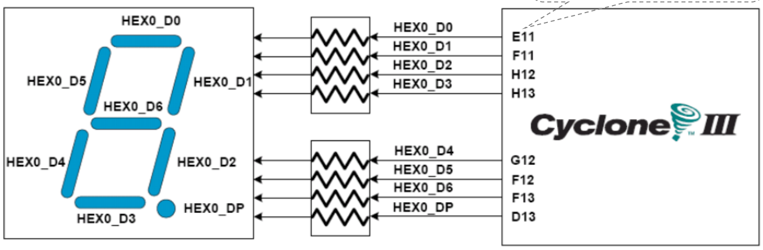
10 LEDs
Opuput high → LED on
Output low → LED off



Signal Name	FPGA Pin No.
LEDG[0]	PIN_J1
LEDG[1]	PIN_J2
LEDG[2]	PIN_J3
LEDG[3]	PIN_H1
LEDG[4]	PIN_F2
LEDG[5]	PIN_E1
LEDG[6]	PIN_C1
LEDG[7]	PIN_C2
LEDG[8]	PIN_B2
LEDG[9]	PIN_B1

7-Segment Displays

Pin number
(active-low)



Signal Name	FPGA Pin No.						
HEX0_D[0]	PIN_E11	HEX1_D[0]	PIN_A13	HEX2_D[0]	PIN_D15	HEX3_D[0]	PIN_B18
HEX0_D[1]	PIN_F11	HEX1_D[1]	PIN_B13	HEX2_D[1]	PIN_A16	HEX3_D[1]	PIN_F15
HEX0_D[2]	PIN_H12	HEX1_D[2]	PIN_C13	HEX2_D[2]	PIN_B16	HEX3_D[2]	PIN_A19
HEX0_D[3]	PIN_H13	HEX1_D[3]	PIN_A14	HEX2_D[3]	PIN_E15	HEX3_D[3]	PIN_B19
HEX0_D[4]	PIN_G12	HEX1_D[4]	PIN_B14	HEX2_D[4]	PIN_A17	HEX3_D[4]	PIN_C19
HEX0_D[5]	PIN_F12	HEX1_D[5]	PIN_E14	HEX2_D[5]	PIN_B17	HEX3_D[5]	PIN_D19
HEX0_D[6]	PIN_F13	HEX1_D[6]	PIN_A15	HEX2_D[6]	PIN_F14	HEX3_D[6]	PIN_G15
HEX0_DP	PIN_D13	HEX1_DP	PIN_B15	HEX2_DP	PIN_A18	HEX3_DP	PIN_G16