

# Notes: Standard Combinatorial Logic Circuits; Storage Elements and Sequential Logic Circuits

## A. Why?

Certain combinatorial logic circuits that select inputs appear commonly. Storage elements are the basic circuits that store data, which is used in logic circuits that use memory.

## B. Outcomes

By the end of the class you should

- Know what decoders and multiplexers (muxes) do (and that they're combinatorial logic circuits).
- Know what half adders and full adders do.
- Know how an R-S Latch works.
- Know why we extend an R-S Latch to make a D-Latch.

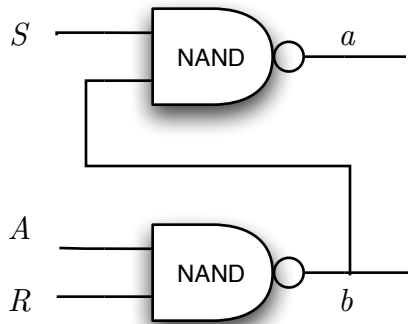
## C. Some Standard Combinatorial Circuits

- **Decoder:** n-bit input (binary number),  $2^n$  outputs (named 00...0 to 11...1); the one output specified by the input will be 1, the others will be 0. [Textbook Figure 3.11]
  - Example: Instruction op code is a binary number; decoding it tells you which instruction is being specified.
- **Multiplexer** ("mux") works the other way:  $2^n$  data inputs (named 00...0 to 11...1) and an n-bit "select signal" input. The one output = the named input. [Figures 3.12 and 3.13] Example: Select a memory bit given its address.
- **Half adder:** Consider adding 1 bit + 1 bit to get a 2-bit output ( $0+0 = 00$ ,  $0+1 = 1+0 = 01$ ,  $1+1 = 10$ ); the left bit is the carry out. [Figure 3.15 with  $C_i = 0$ ]
  - Could be used to add rightmost bit of two unsigned binary numbers
- **Full adder:** For addition of two unsigned binary numbers, there's also a carry in to worry about ( $0+0+0 = 00$ ,  $0+0+1 = 0+1+0 = 1+0+0 = 01$ ,  $0+1+1 = 1+0+1 = 1+1+0 = 10$ ,  $1+1+1 = 11$ ). [Figure 3.15]
  - (A half adder is a full adder with carry in = 0.)
  - To add two n-bit numbers, you need n copies of a full adder.

## D. Storage Element - the R-S Latch

- When input signals change in circuit without loops, the change ripples forward through the circuit to the outputs.
- If a circuit has a loop, then some gate's output is some gate's input and the changes that occur can depend on the past value of the input and output gates.
  - The circuit can have a state that depends on past history.

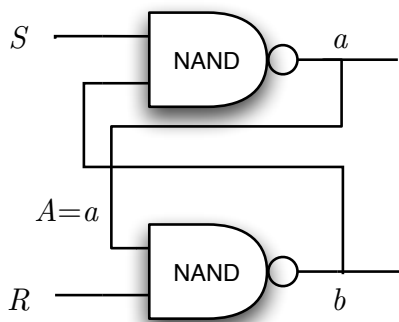
- R-S latch is a simple example; we'll approach it in steps. Device below has 3 inputs and 2 outputs. When  $a = \neg b$ , we'll say the device is in state  $a$ .



Not Quite an R-S Latch

$R$	$S$	$A$	$a = \neg(Sb)$	$b = \neg(RA)$
0	0	0	1	1
0	0	1	1	1
0	1	0	0	1
0	1	1	0	1
1	0	0	1	1
1	0	1	1	0
1	1	0	0	1
1	1	1	1	0

- Connecting  $a$  to  $A$  gives us an R-S latch. Normally  $R = S = 1$ ; this maintains the state of  $a$  and  $b$ . (The parenthesized numbers indicate no change in a value.) Changing  $R$  (Reset) to 0 changes  $b$  then  $a$  to end with  $a=0, b=1$ . Changing  $S$  (Set) to 0 changes  $a$  then  $b$  to end with  $a=1, b=0$ . Either way, we go back to  $R = S = 1$  to maintain  $a$  and  $b$ .



An R-S Latch

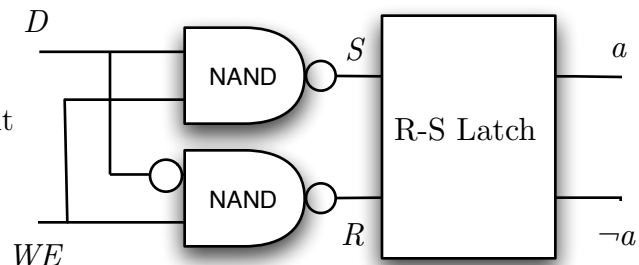
$R$	$S$	$a$	$b$	$New\ a = \neg(Sb)$	$New\ b = \neg(Ra)$
1	1	0	1	(0)	(1)
1	1	1	0	(1)	(0)
0	1	0	1	(0)	(1)
0	1	1	0	(1)	Chg to 1
0	1	1	1	Chg to 0	(1)
1	0	0	1	Chg to 1	(1)
1	0	1	0	(1)	(0)
1	0	1	1	(1)	Chg to 0

- We don't set  $R = S = 0$  because that doesn't leave  $a$  and  $b$  in a good state.

$R$	$S$	$a$	$b$	$New\ a = \neg(Sb)$	$New\ b = \neg(Ra)$
0	0	0	1	Chg to 1	(1)
0	0	1	0	(1)	Chg to 1
0	0	1	1	(1)	(1)

- Gated D-Latch

- To ensure that  $S$  and  $R$  aren't both 0, we can add more gates.
- Gated D-latch sets R-S latch to input  $D$ , but only if "write enable"  $WE$  input = 1.
  - Reset  $a = 0$  if  $\neg D = WE = 1$
  - Set  $a = 1$  if  $D = WE = 1$



Gated D-Latch

- Register = List of bits

- Gated D-latch stores 1 bit; combine multiple  $D$  latches to get multiple bits; use common  $WE$  line.
- 2-bit register has two input data bits  $D_1, D_0$ , one input control bit  $WE$ , two output bits  $Q_1, Q_0$ .
- Setting  $WE = 1$  causes latches to store  $D_1, D_0$  (and output  $Q_1, Q_0$ ).
- Setting  $WE$  back = 0 causes  $D_1, D_0$  to be ignored (but output  $Q_1, Q_0$  stays as is).

