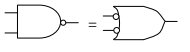
0 Admin
1 Recap
2 Combinational vs. sequential logic
3 Outline: Algorithms in HW
4 State elements, latches, and FF

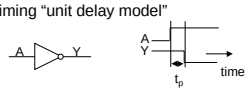
1 Recap

- DeMorgan's Theorem: $\overline{AB} = \overline{A} + \overline{B}$ $\overline{A+B} = \overline{A} \cdot \overline{B}$



A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

- truth table → Boolean Algebra → logic schematic
- timing "unit delay model"



2. COMBINATIONAL LOGIC vs. SEQUENTIAL LOGIC

Combinational

$\underline{y} = \underline{f}$ (inputs)

n inputs, m outputs

Examples:

Adder: y[0:3] = a[0:3] + b[0:3]

memoryless systems

Sequential

$\underline{y} = \underline{f}$ (inputs, time)

(most useful: clocked logic)

Example: (down) counter

Time (e.g. sec)	Y[0:3]
0	10_{10}
1	9
2	8
…	…
10_{10}	0

3. ALGORITHMS IN HW

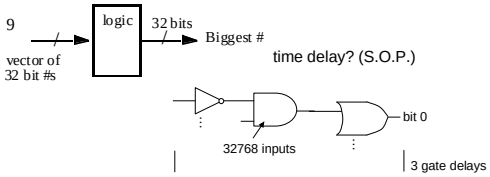
Example: Finding the largest number in a list.

Unsigned int 9[1024]; 32 bit int

3.1 Approach #1: Combinational

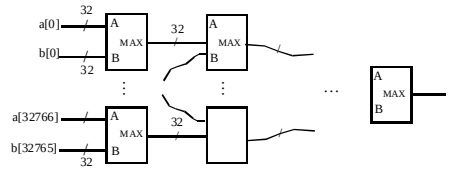
Number of input variables? 32768

Number of lines in truth table? 2^{32768}



3. ALGORITHMS IN HW (cont.)

3.2 Approach #2: Combinational – divide & conquer



$512 + 256 + \ldots + 1 = 1023$ blocks

Each MAX block has: 64 inputs; 32 outputs

2^{64} entries in truth table

Delay = ? 3.9 = 27 units

3. ALGORITHMS IN HW (cont.)

3.3 Approach #3: Sequential

```
max = 0;
for (i=0; i<32768, i#)
    if (a[i]>max) max=a[i]
```

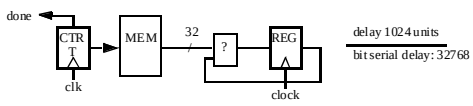
Hardware equivalence:

a[i] → 1024×32 bit memory

for (i=0; i<1024; i#) →15 bit counter

if (a[i]>max) → "comparator" (combinational)

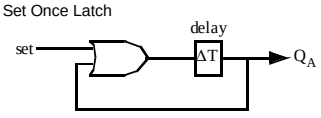
max=a[i] → register



4. STATE ELEMENTS, LATCHES, & FF

Used to store data (1/0)

4.1 Set Once Latch



present state: $Q_A(t) \triangleq QA_n$

next state: $Q_A(t+\Delta t) \triangleq QA_{n+1}$

$\triangleq Q_A^+$

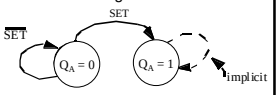
Algebra: $Q_{A_{n+1}} = \text{Set} + Q_{An}$

4. STATE ELEMENTS, LATCHES, & FF (cont.)

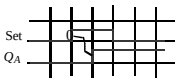
4.1.1 Truth table – "characteristic table" of latch

Inputs Set	Inputs Q_{An}	Output or next state $Q_{An+1} = Q^+$
0	0	0
0	1	1
1	0	1
1	1	1

4.1.2 State diagram



4.1.3 Timing

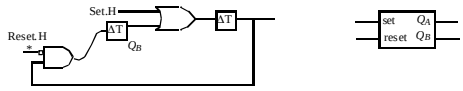


Set	Q^+
0	Q
1	1

Algebra: $Q_{A_{n+1}} = \text{Set} + Q_{An}$

4. STATE ELEMENTS, LATCHES, & FF (cont.)

4.2 Set-reset latch – add reset control



4.2.1 Characteristic table

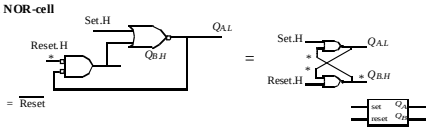
Set	Reset	$Q_B = \overline{R} \cdot Q_A$	Q_{An+1}	Q_{Bn+1}	
0	0	Q_{An}	Q_{An}	Q_{An}	Hold
0	1	0	0	0	Reset
1	0	Q_{An}	1	1	Set
1	1	0	1	0	illegal

Algebra: $Q_{A_{n+1}} = \text{Set} + Q_{An}$

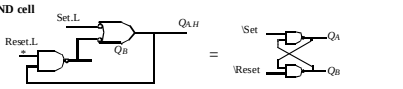
4. STATE ELEMENTS, LATCHES, & FF (cont.)

4.2.2 Alternative forms of set-reset latch

Crossed NOR-cell



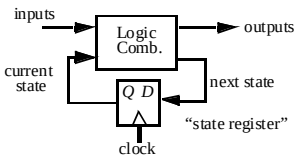
Crossed NAND cell



Algebra: $Q_{A_{n+1}} = \text{Set} + Q_{An}$

4. STATE ELEMENTS, LATCHES, & FF (cont.)

4.3 D or data latch
Goal: add clock to control when data is latched
Fundamental idea: Finite State Machine

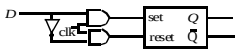


Algebra: $Q_{A_{n+1}} = \text{Set} + Q_{An}$

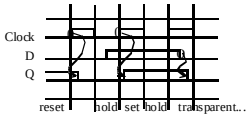
4. STATE ELEMENTS, LATCHES, & FF (cont.)

4.3 D latch: if $D \cdot \text{clock}$, then set
if $\overline{D} \cdot \text{clock}$, then reset
$\Big\}\ Q_{n+1} = \text{clock} \cdot D$

Concept:



Timing diagram (given D and clock), Q(t = 0) = 1



Algebra: $Q_{A_{n+1}} = \text{Set} + Q_{An}$