Digital Logic Design

Part 4

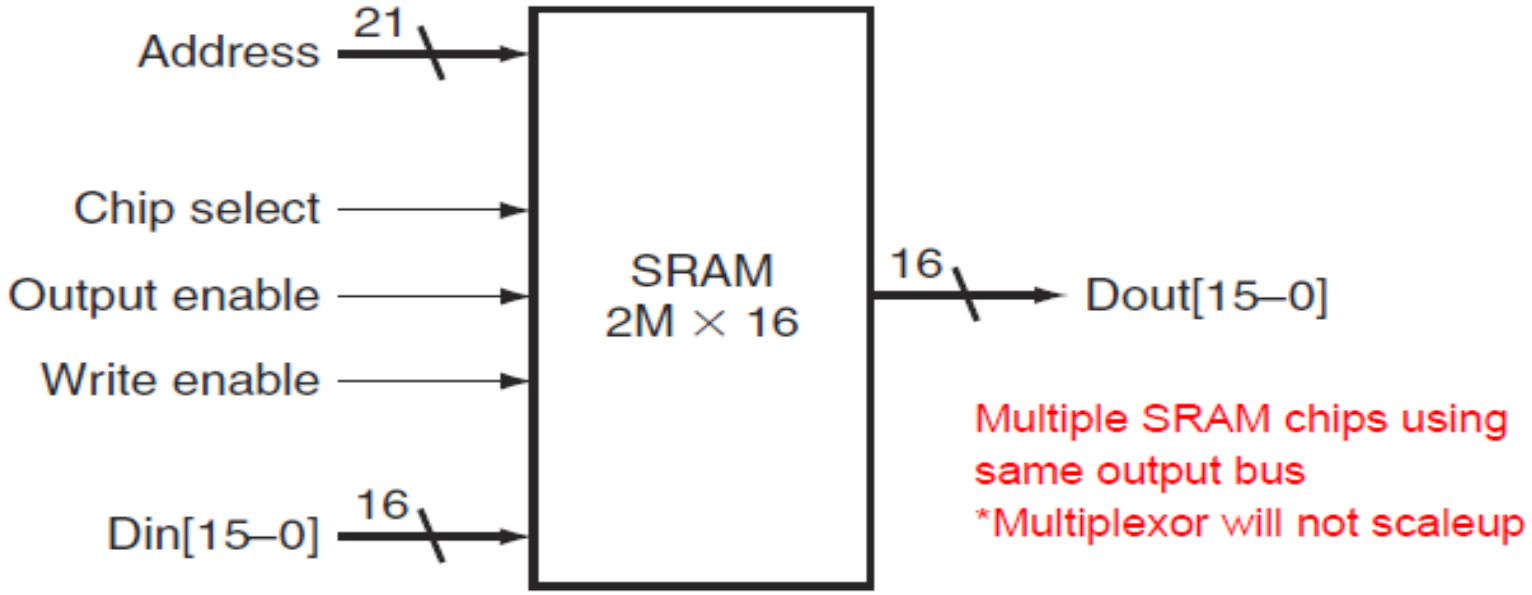
A2 :
NEW SUBMISSION GUIDELINES USING MARKUS

Do NOT Submit into Drop Boxes

15-1-5

Random Access Memories

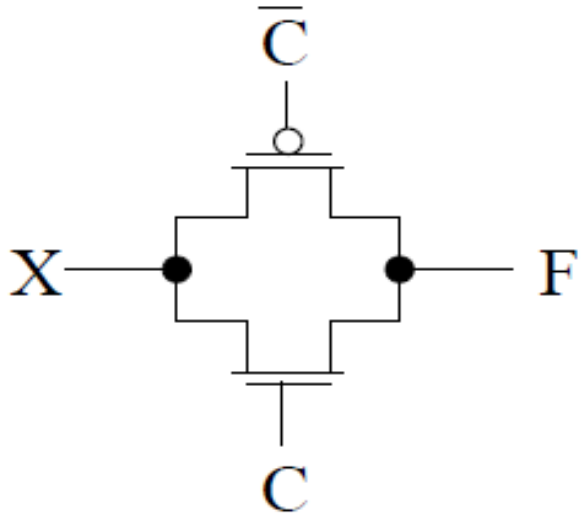
- Static random access memories (SRAM) use D latches



- Register file idea won't scale up; decoder and multiplexors too big
- Fix multiplexor problem by using three-state buffers
- Fix decoder problem by using two-level decoding
- This type of memory is **not** clocked

Three-state buffer or transmission gate

- Has three outputs 0, 1, and *floating* (connected to neither power or ground)

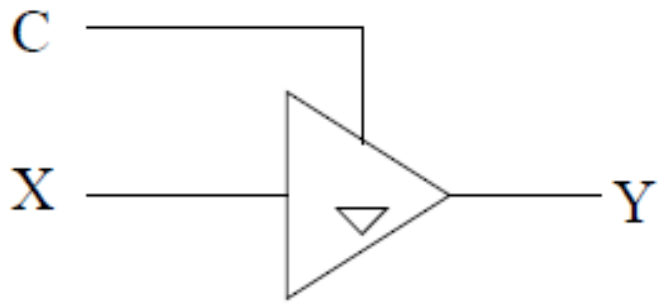


If C=0 Both Transistors are off
Signal X does not pass at all
If C=1 Both Transistors are on.
X passes through both of them to Output F

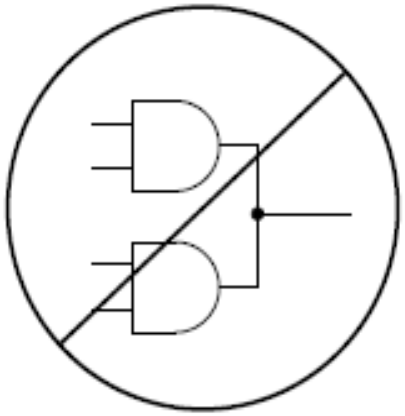
- $C = 1$, then
  - NMOS gate passes 0 well
  - $\bar{C} = 0$ and PMOS gate passes 1 well
- $C = 0$, then $\bar{C} = 1$ and both transistors are off (output is floating).

High Impedance : No data passage

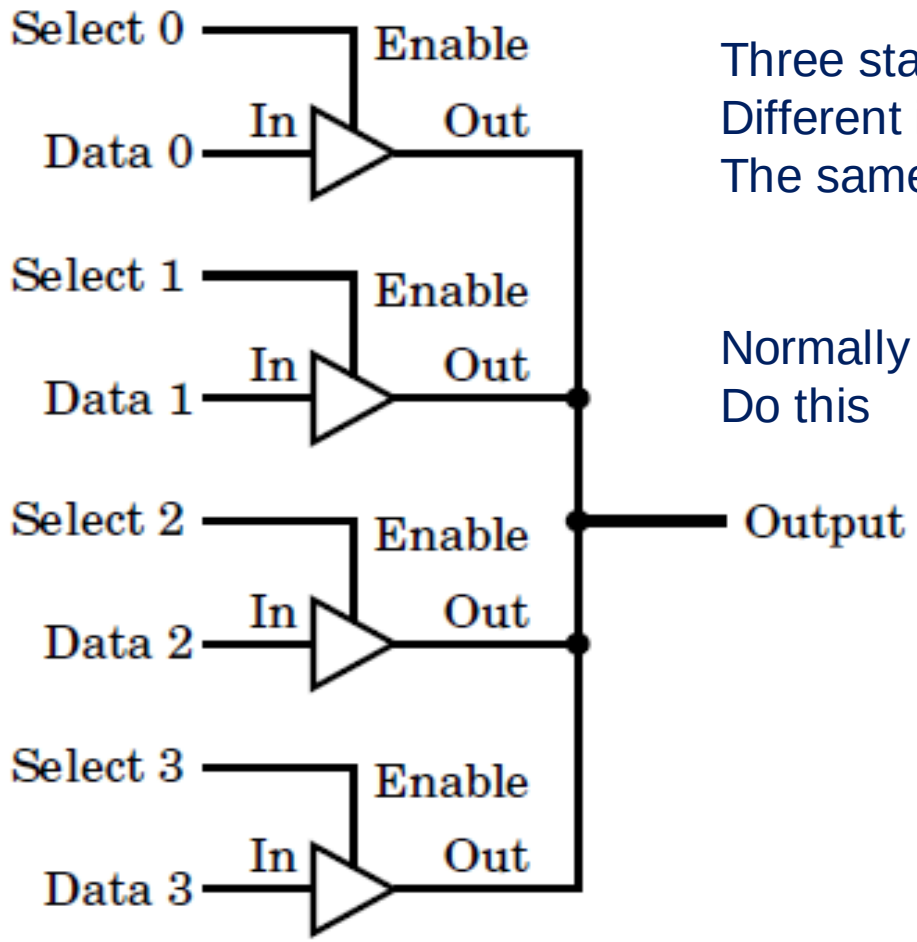
Using Three-State Buffers



- High-impedance outputs can be "tied together" without problems
- Normally, do not tie output lines together



Making Multiplexors from Three-State Buffers

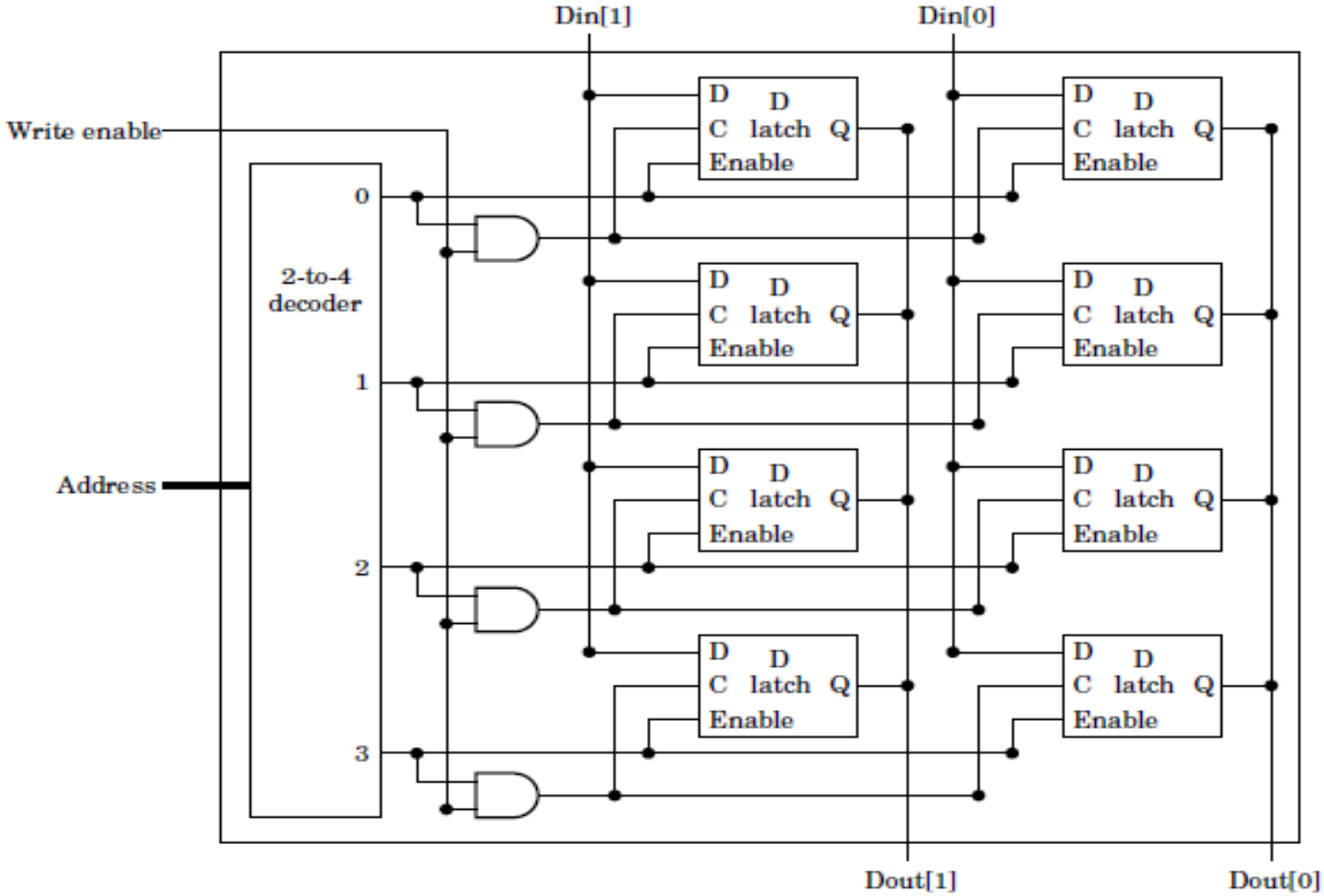


Three state buffers allow us to tie Different input sources together to use The same Output Line.

Normally -with simple gates we cannot Do this

IMPORTANT: Must ensure that at most one select input is 1, or short-circuit may result (physical meltdown)

Example of SRAM Structure

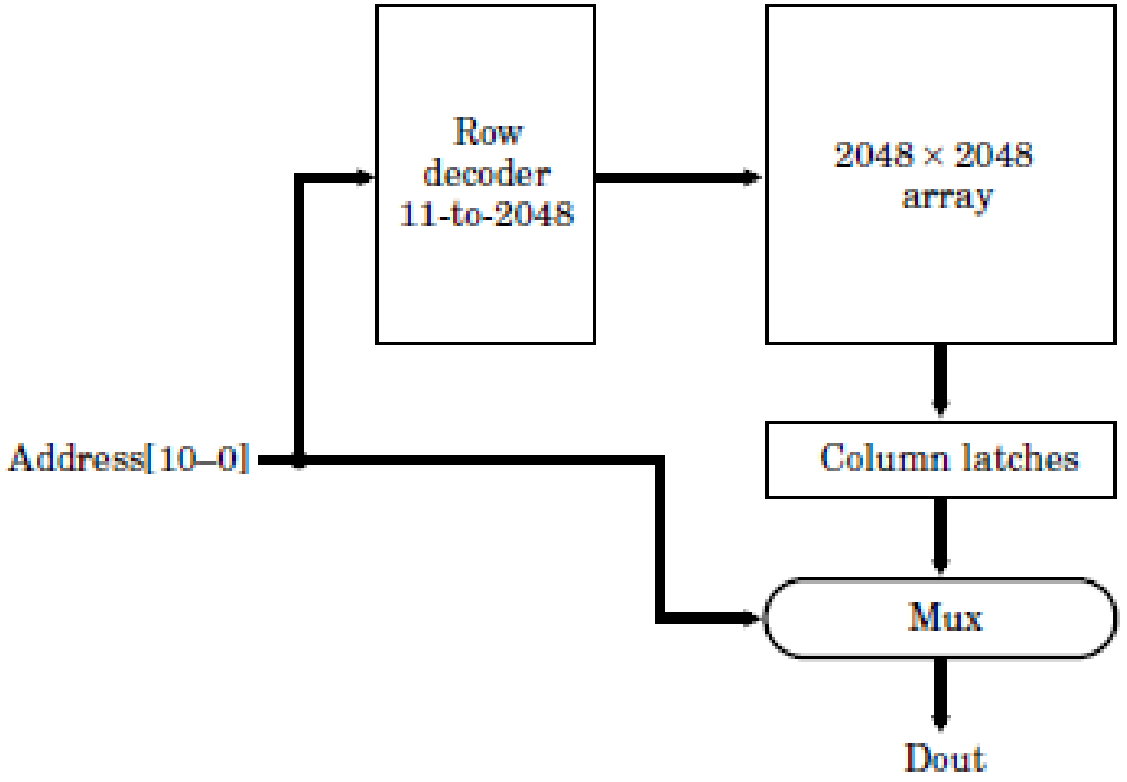


Note :
There are 3 state Buffers tied to each Output
In order to allow The columns To share the Same output Lines

Does this design scale up well?

Large Memories Use a Two Level Addressing Process
Where the Address Bits are split across One Decoder and One Multiplexor (two Level decoding)

Design of 4Mx1 DRAM



All the data from one row
Is sent through the
columns latches

Two Level Decoding:

Address Bits : 11 go into the Decoder
11 bits go into the MUX to select the column
4M = Total 22 Bits

The bits **A3 A2 A1 A0** represent a 4 bit address of a location in memory

---

Label the input bits on the Decoder and the Select Line bits on the Multiplexor with **A3 A2 A1 A0**.

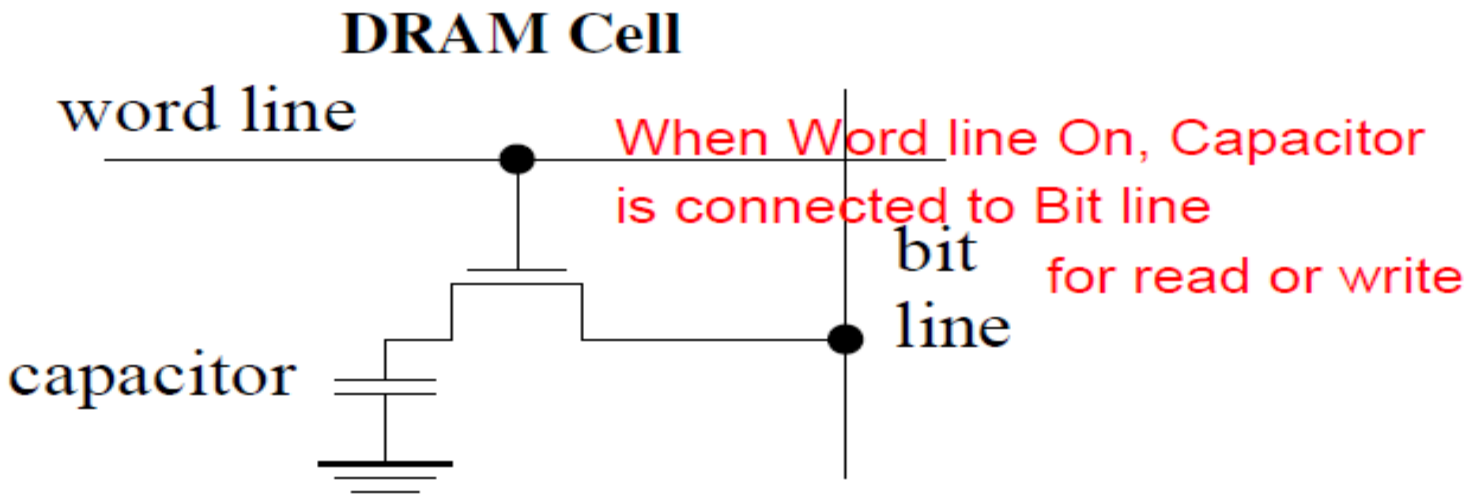
Dynamic RAM

- Our SRAM cell uses a lot of transistors Pair of inverting gates keep data
- A better implementation uses six transistors
- This is still too expensive
- Alternative: use a capacitor to store a charge to represent 1
- Problem: charge leaks away, must be refreshed

Dynamic: Capacitor is storing charge to represent 0 or 1
charge leaks away

Unique about Capacitor: It takes on the charge of whatever it is connected to. Whether that is a battery, or a voltage. It charges to that value and slowly dissipates this charge.

If capacitor connected to 0V, its charge leaks away and is thereby set to 0.



A Single DRAM Bit: One Capacitor One Transistor.

Cheaper But Slower

Transistor will allow us to write a charge or read a charge
To the capacitor

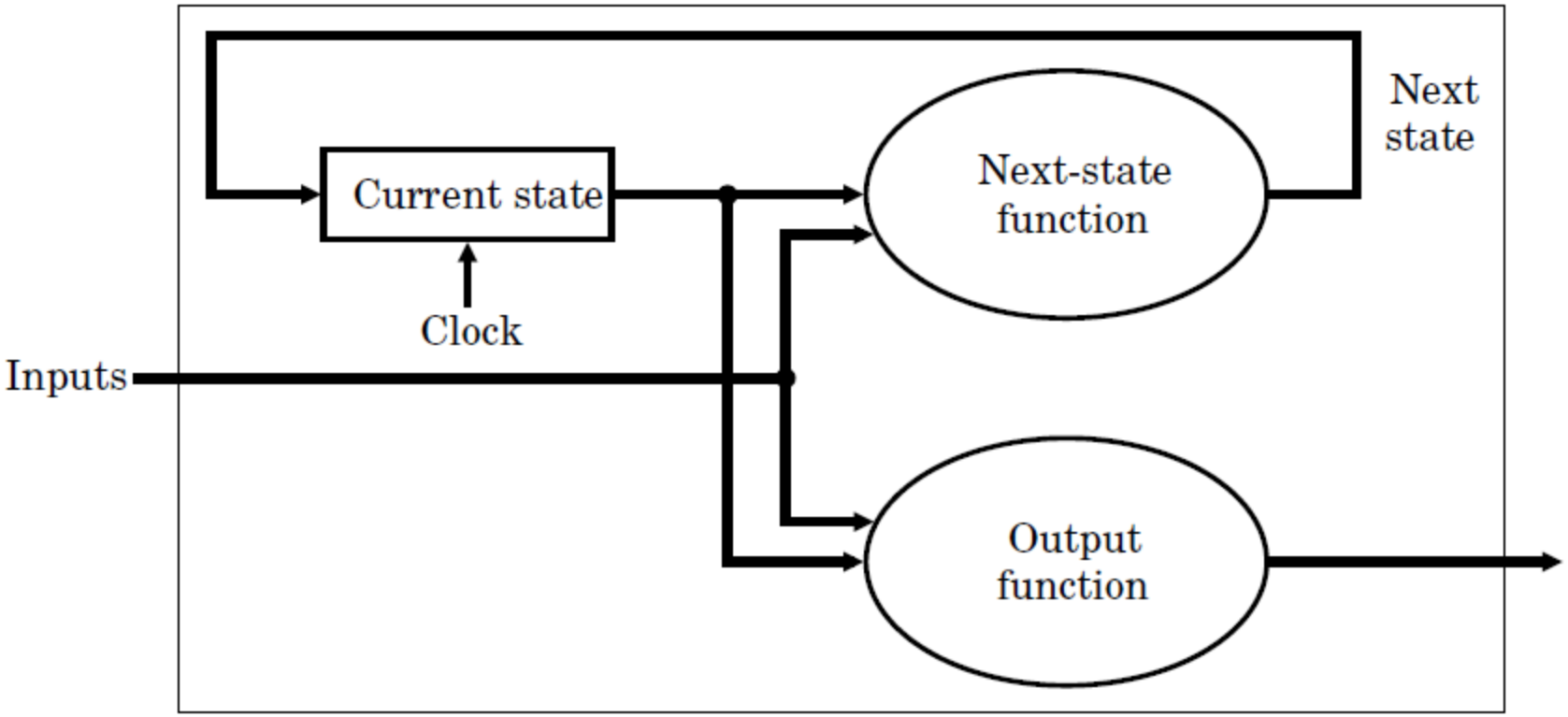
Single Chip Memory Controller : Dedicated to refreshing the charges on the capacitors within DRAM
takes up only 1-2% of active clock cycles. Maximum 4%

DRAM/ SRAM

SRAM: Expensive but Fast CACHES

DRAM: Cheaper But Slower. Always need to refresh the Capacitors
Also a read from a capacitor will dissipate the charge therefore
Read is even slower than write:
Read you need to read the charge and write it back

Designing Using Finite-State Machines

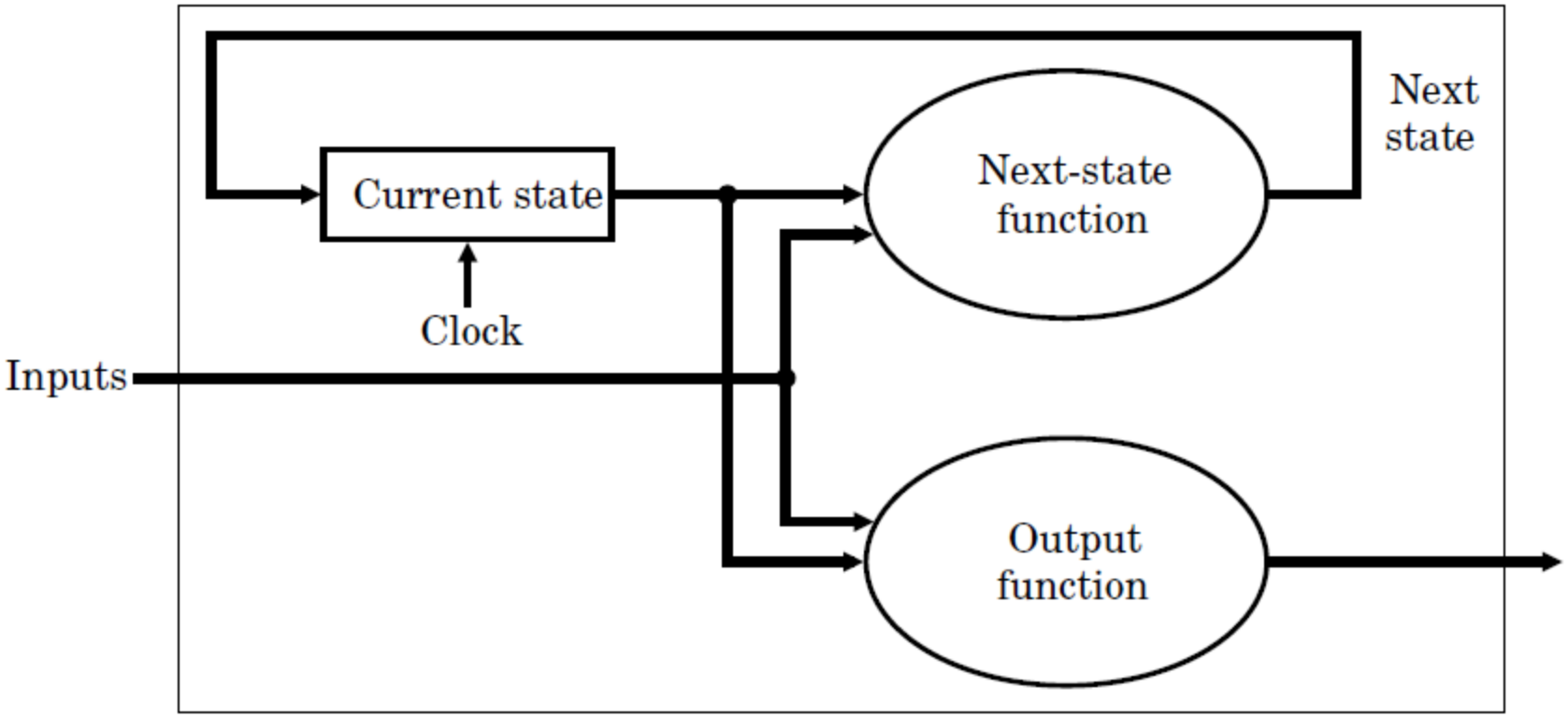


High-level circuit implementation of finite-state machine

Behaviour of
Sequential Systems
Depends on Both the inputs
And the data stored in memory
This will feed back and determine
What the state the system takes
On next.

Cannot simply define
Sequential Logic using
Truth Table

Designing Using Finite-State Machines



High-level circuit implementation of finite-state machine

Instead
Describe Sequential System
With FSM

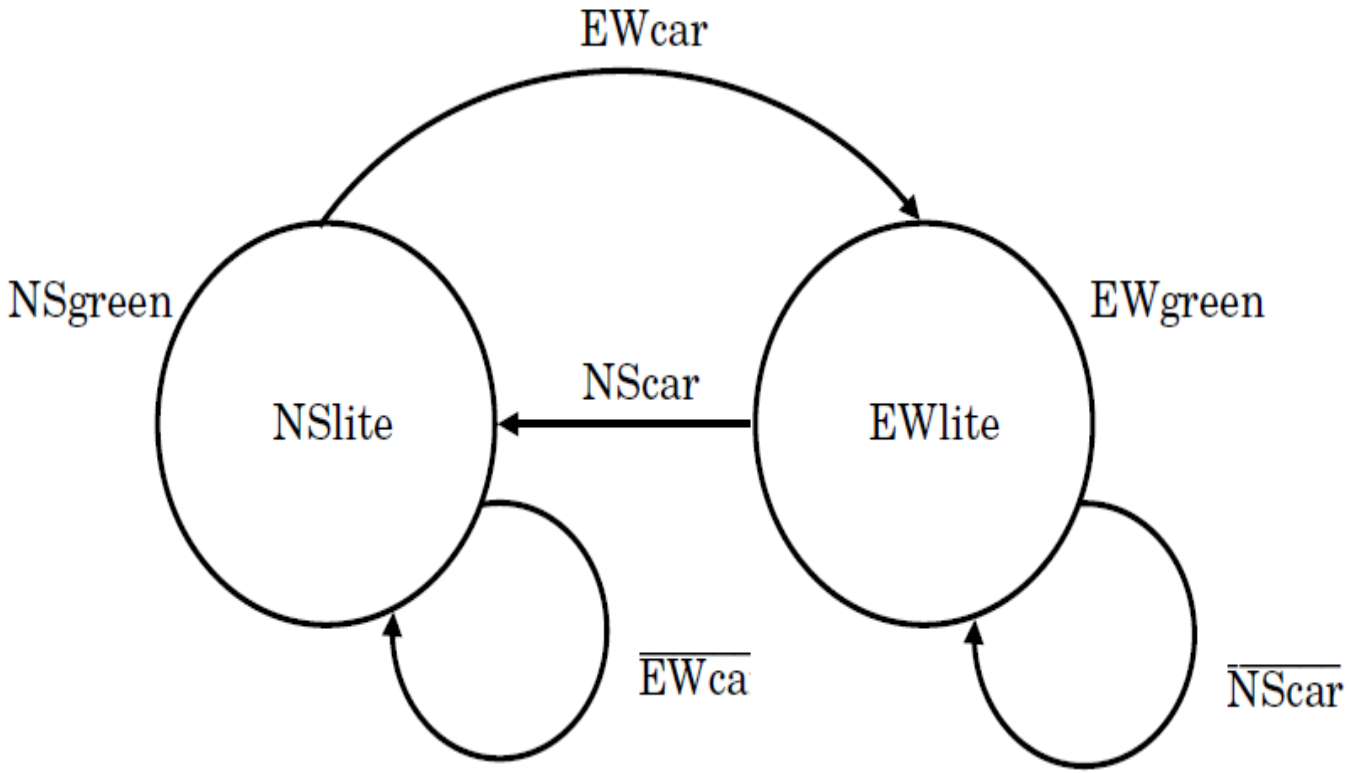
Cannot simply define
Sequential Logic using
Truth Table

Example: Traffic Light

- Output signals: NSlight, EWlight
- Input signals: NScar, EWcar
- State names: NSgreen, EWgreen (no yellow for now)
- Functionality: want light to change only if car is waiting at red light

Otherwise the light should continue to show green in the direction that the last car crossed The intersection

Graphical Representation of Traffic Light Controller

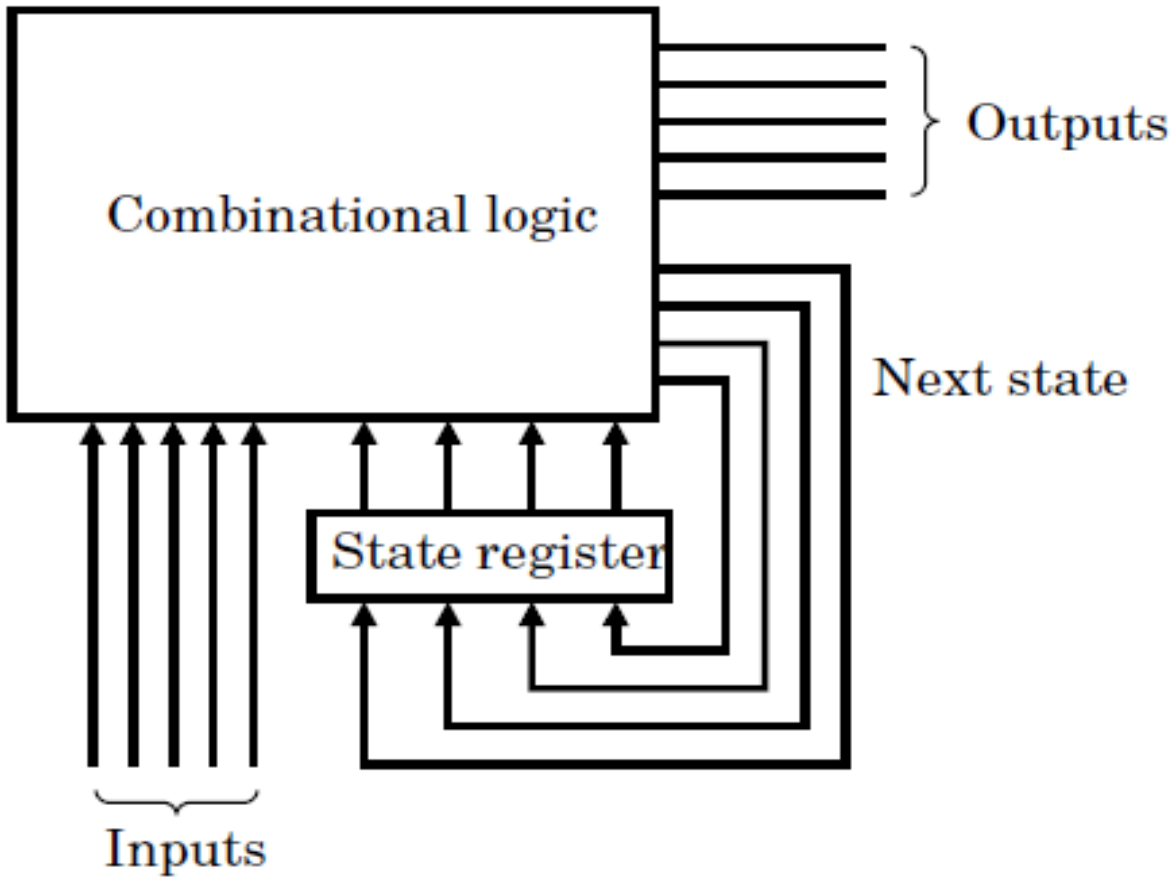


- Names of states outside ovals
- Output in given state inside oval
- Transition arc labelled with Boolean formula of inputs

Within the Nsgreen
State the NSlite (output)
Is asserted.

Therefore Output depends on
Current state
(Moore FSM)

Electronic Implementation of Finite-State Controller



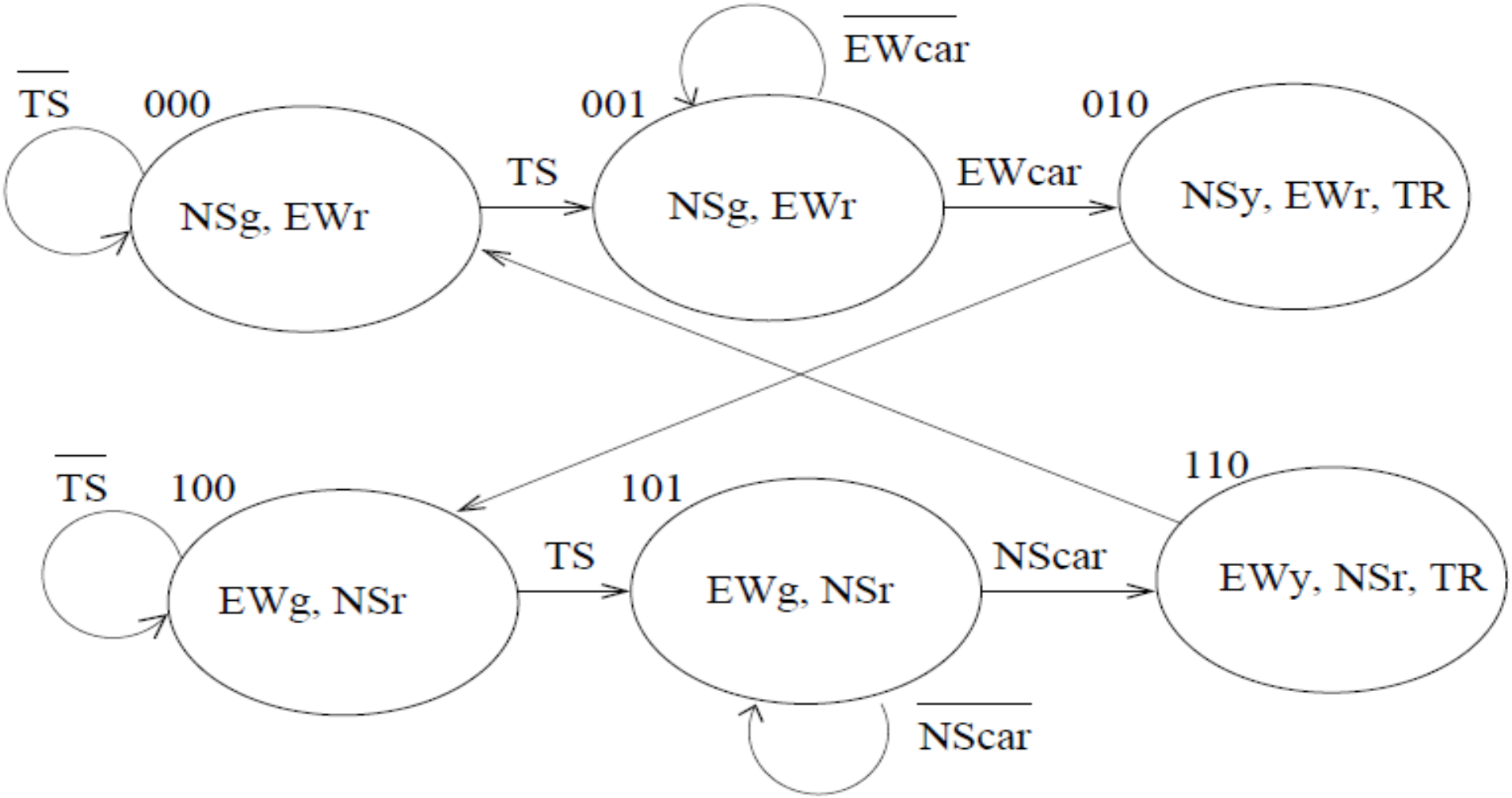
Extending the Traffic-Light Controller

- Add 4-second yellow light
- Assume 0.25Hz clock
- need to add 28-second timer
  - Timer input: TimerReset (TR)
  - Timer output: TimerSignal (TS)
- Behaviour of system
  - Stay green in one direction (red in other direction) until car arrives or 32 seconds elapse, whichever happens last
  - Green turns to yellow for 4 seconds; red in other direction stays
  - Yellow turns to red, red in other direction turns to green

We decide we want green light to stay on 28 seconds

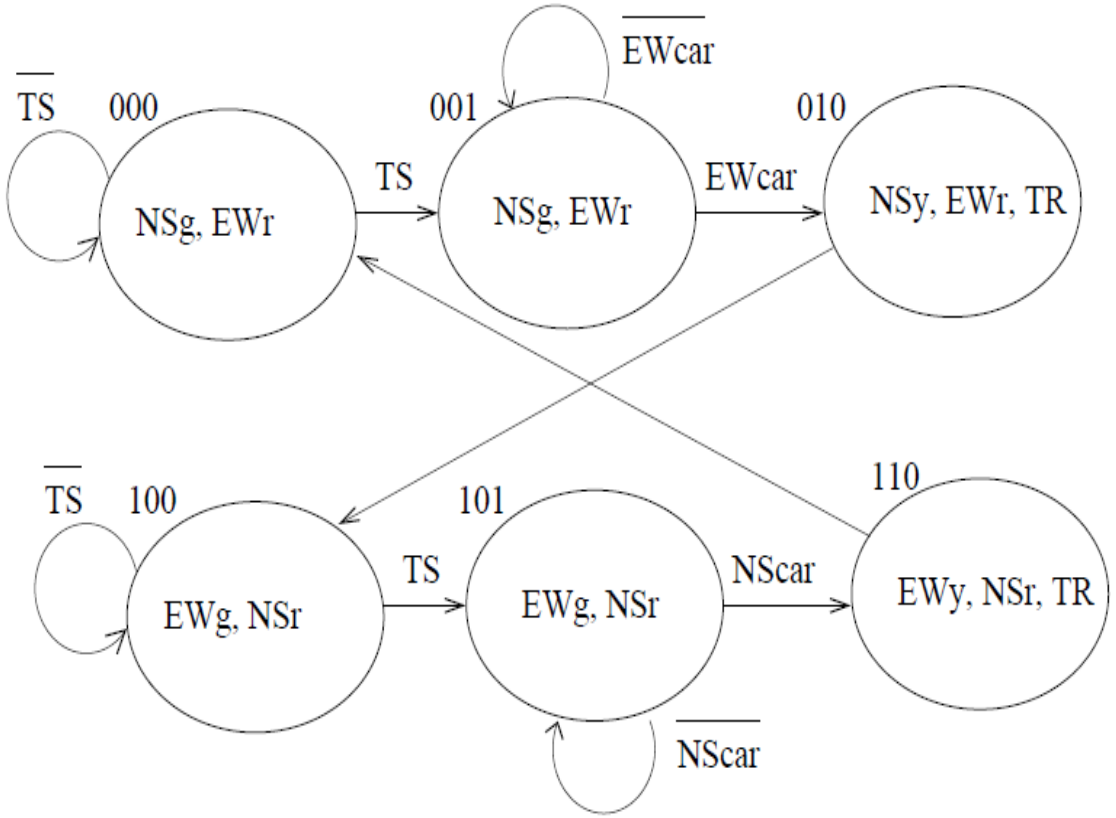
State Diagram of Extended Controller

- Inputs: NScar, EWcar, TS
- Outputs: NSg, NSy, NSr, EWg, EWy, EWr, TR



State Diagram of Extended Controller

- Inputs: NScar, EWcar, TS
- Outputs: NSg, NSy, NSr, EWg, EWy, EWr, TR



Next-State Table for Extended Controller

current state	inputs			next state	current state	inputs			next state
$S_2S_1S_0$	NS-car	EW-car	TS	$S_2'S_1'S_0'$	$S_2S_1S_0$	NS-car	EW-car	TS	$S_2'S_1'S_0'$
0 0 0	X	X	0	0 0 0	1 0 0	X	X	0	1 0 0
0 0 0	X	X	1	0 0 1	1 0 0	X	X	1	1 0 1
0 0 1	X	0	X	0 0 1	1 0 1	0	X	X	1 0 1
0 0 1	X	1	X	0 1 0	1 0 1	1	X	X	1 1 0
0 1 0	X	X	X	1 0 0	1 1 0	X	X	X	0 0 0
0 1 1	X	X	X	X X X	1 1 1	X	X	X	X X X

Note unused states, symmetries

Output Table For Extended Controller

- Output table looks like truth table

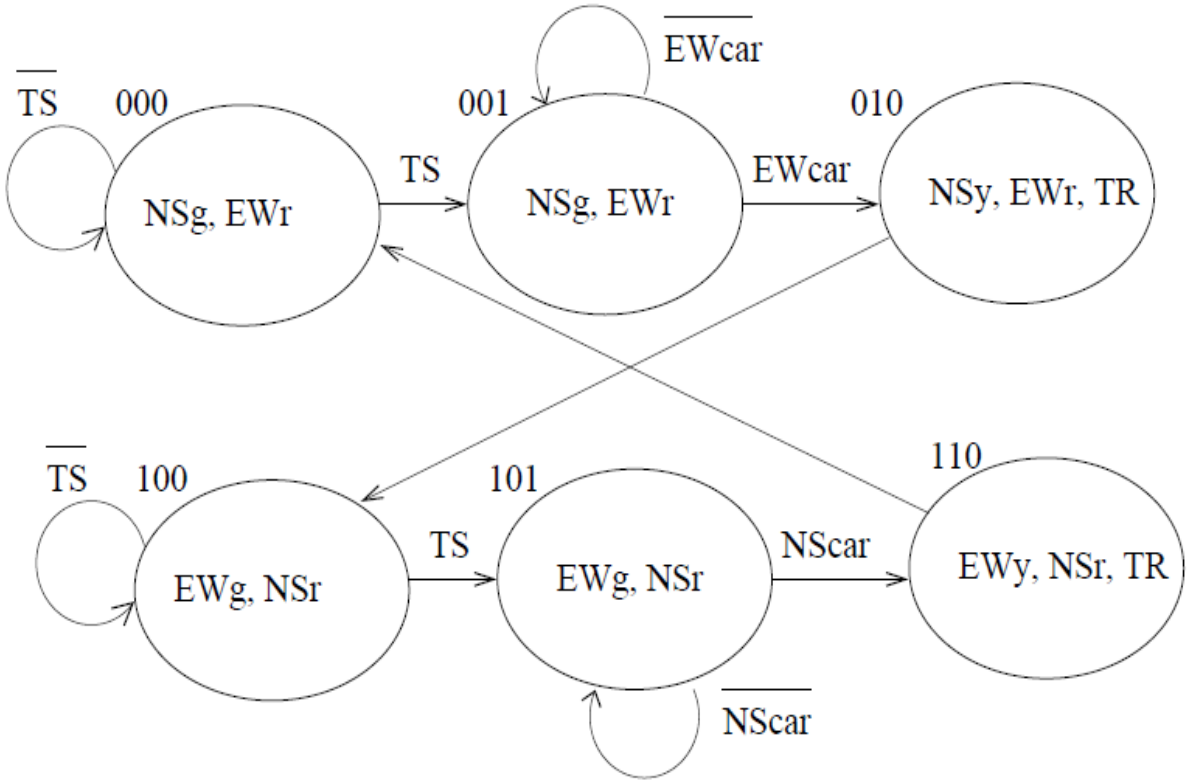
  Inputs are State, Outputs are Outputs

- Traffic light outputs: NSg, NSy, NSr, EWg, EWy, EWr, TR
- If output listed in State, then 1 in output table

  If output not listed in State, then 0 in output table

State Diagram of Extended Controller

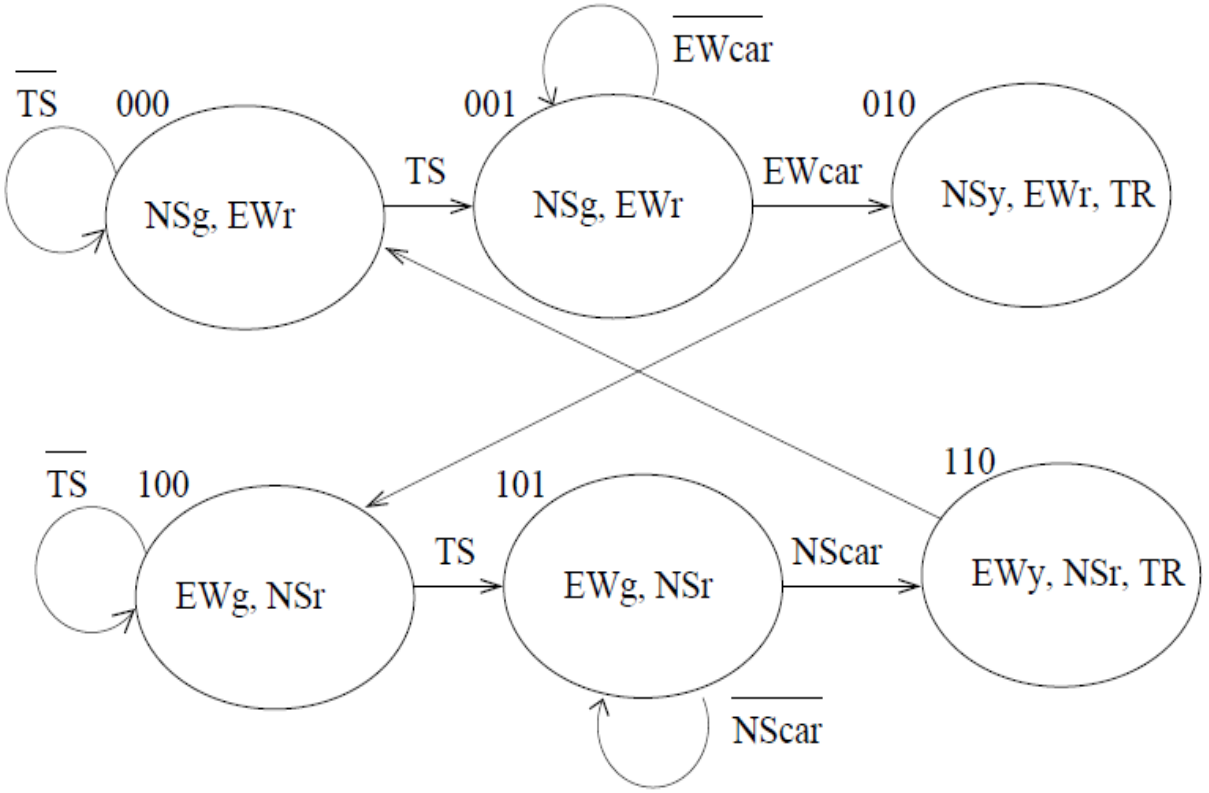
- Inputs: NScar, EWcar, TS
- Outputs: NSg, NSy, NSr, EWg, EWy, EWr, TR



S_2	S_1	S_0	NSg	NSy	NSr	EWg	EWy	EWr	TR
0	0	0	1	0	0	0	0	1	0
0	0	1							
0	1	0							
0	1	1							
1	0	0							
1	0	1							
1	1	0							
1	1	1							

State Diagram of Extended Controller

- Inputs: NScar, EWcar, TS
- Outputs: NSg, NSy, NSr, EWg, EWy, EWr, TR



Current State and Outputs

S_2	S_1	S_0	NSg	NSy	NSr	EWg	EWy	EWr	TR
0	0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	0	0	1	0
0	1	0	0	1	0	0	0	1	1
0	1	1	X	X	X	X	X	X	X
1	0	0	0	0	1	1	0	0	0
1	0	1	0	0	1	1	0	0	0
1	1	0	0	0	1	0	1	0	1
1	1	1	X	X	X	X	X	X	X

Next-State Table for Extended Controller

current state	inputs			next state	current state	inputs			next state
$S_2S_1S_0$	NS-car	EW-car	TS	$S_2'S_1'S_0'$	$S_2S_1S_0$	NS-car	EW-car	TS	$S_2'S_1'S_0'$
0 0 0	X	X	0	0 0 0	1 0 0	X	X	0	1 0 0
0 0 0	X	X	1	0 0 1	1 0 0	X	X	1	1 0 1
0 0 1	X	0	X	0 0 1	1 0 1	0	X	X	1 0 1
0 0 1	X	1	X	0 1 0	1 0 1	1	X	X	1 1 0
0 1 0	X	X	X	1 0 0	1 1 0	X	X	X	0 0 0
0 1 1	X	X	X	X X X	1 1 1	X	X	X	X X X

Note unused states, symmetries

S_2	S_1	S_0	NSg	NSy	NSr	EWg	EWy	EWr	TR
0	0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	0	0	1	0
0	1	0	0	1	0	0	0	1	1
0	1	1	X	X	X	X	X	X	X
1	0	0	0	0	1	1	0	0	0
1	0	1	0	0	1	1	0	0	0
1	1	0	0	0	1	0	1	0	1
1	1	1	X	X	X	X	X	X	X

Next-State Table for Extended Controller

current state	inputs			next state	current state	inputs			next state
$S_2S_1S_0$	NS-car	EW-car	TS	$S_2'S_1'S_0'$	$S_2S_1S_0$	NS-car	EW-car	TS	$S_2'S_1'S_0'$
0 0 0	X	X	0	0 0 0	1 0 0	X	X	0	1 0 0
0 0 0	X	X	1	0 0 1	1 0 0	X	X	1	1 0 1
0 0 1	X	0	X	0 0 1	1 0 1	0	X	X	1 0 1
0 0 1	X	1	X	0 1 0	1 0 1	1	X	X	1 1 0
0 1 0	X	X	X	1 0 0	1 1 0	X	X	X	0 0 0
0 1 1	X	X	X	X X X	1 1 1	X	X	X	X X X

Current state = $S_2S_1S_0$, next state = $S_2'S_1'S_0'$

$$S_0' = \overline{S_1}\,\overline{S_0} \cdot TS + \overline{S_2}\,\overline{S_1}\,S_0 \cdot \overline{EW\,car} + S_2\overline{S_1}\,S_0 \cdot \overline{NS car}$$

$$S_1' = \overline{S_2}\,\overline{S_1}\,S_0 \cdot EW\,car + S_2\overline{S_1}\,S_0 \cdot NS car$$

Next State Functions: Determined by Inputs Only And Current State

Next-State Table for Extended Controller

current state	inputs			next state	current state	inputs			next state
$S_2S_1S_0$	NS-car	EW-car	TS	$S'_2S'_1S'_0$	$S_2S_1S_0$	NS-car	EW-car	TS	$S'_2S'_1S'_0$
0 0 0	X	X	0	0 0 0	1 0 0	X	X	0	1 0 0
0 0 0	X	X	1	0 0 1	1 0 0	X	X	1	1 0 1
0 0 1	X	0	X	0 0 1	1 0 1	0	X	X	1 0 1
0 0 1	X	1	X	0 1 0	1 0 1	1	X	X	1 1 0
0 1 0	X	X	X	1 0 0	1 1 0	X	X	X	0 0 0
0 1 1	X	X	X	X X X	1 1 1	X	X	X	X X X

Current state = $S_2S_1S_0$, next state = $S'_2S'_1S'_0$

$$S'_0 = \overline{S_1}\,\overline{S_0}\cdot TS + \overline{S_2}\,\overline{S_1}\,S_0\cdot\overline{EW\,car} + S_2\overline{S_1}\,S_0\cdot\overline{NScar}$$

$$S'_1 = \overline{S_2}\,\overline{S_1}\,S_0\cdot EW\,car + S_2\overline{S_1}\,S_0\cdot NScar$$

In S'_2: Expanded Output function.

Which of the following is a correct Minterm:

A) $\overline{S_2}S_1S_0NSCar$

B) $S_2S_1\overline{S_0}EWCar$

C) $S_2S_1S_0\overline{TS}$

D) $S_2\overline{S_1}S_0TS$

E) $\overline{S_2S_1S_0}NSCar$

Next-State Table for Extended Controller

current state	inputs			next state	current state	inputs			next state
$S_2S_1S_0$	NS-car	EW-car	TS	$S'_2S'_1S'_0$	$S_2S_1S_0$	NS-car	EW-car	TS	$S'_2S'_1S'_0$
0 0 0	X	X	0	0 0 0	1 0 0	X	X	0	1 0 0
0 0 0	X	X	1	0 0 1	1 0 0	X	X	1	1 0 1
0 0 1	X	0	X	0 0 1	1 0 1	0	X	X	1 0 1
0 0 1	X	1	X	0 1 0	1 0 1	1	X	X	1 1 0
0 1 0	X	X	X	1 0 0	1 1 0	X	X	X	0 0 0
0 1 1	X	X	X	X X X	1 1 1	X	X	X	X X X

Current state = $S_2S_1S_0$, next state = $S'_2S'_1S'_0$

$$S'_0 = \overline{S_1}\,\overline{S_0} \cdot TS + \overline{S_2}\,\overline{S_1}\,S_0 \cdot \overline{EW\,car} + S_2\overline{S_1}\,S_0 \cdot \overline{NScar}$$

$$S'_1 = \overline{S_2}\,\overline{S_1}\,S_0 \cdot EW\,car + S_2\overline{S_1}\,S_0 \cdot NScar$$

In S'_2: Expanded Output function.

Which of the following is a correct Minterm:

A) $\overline{S_2}S_1S_0$NSCar

B) $S_2S_1\overline{S_0}$EWCar

*** C) $S_2S_1S_0\overline{TS}$

D) $S_2\overline{S_1}S_0$TS

E) $\overline{S_2S_1S_0}$NSCar

Next-State Table for Extended Controller

current state	inputs			next state	current state	inputs			next state
$S_2S_1S_0$	NS-car	EW-car	TS	$S_2'S_1'S_0'$	$S_2S_1S_0$	NS-car	EW-car	TS	$S_2'S_1'S_0'$
0 0 0	X	X	0	0 0 0	1 0 0	X	X	0	1 0 0
0 0 0	X	X	1	0 0 1	1 0 0	X	X	1	1 0 1
0 0 1	X	0	X	0 0 1	1 0 1	0	X	X	1 0 1
0 0 1	X	1	X	0 1 0	1 0 1	1	X	X	1 1 0
0 1 0	X	X	X	1 0 0	1 1 0	X	X	X	0 0 0
0 1 1	X	X	X	X X X	1 1 1	X	X	X	X X X

Current state = $S_2S_1S_0$, next state = $S_2'S_1'S_0'$

$$S_0' = \overline{S_1}\,\overline{S_0}\cdot TS + \overline{S_2}\,\overline{S_1}\,S_0\cdot \overline{EW\,car} + S_2\overline{S_1}\,S_0\cdot \overline{NS car}$$

$$S_1' = \overline{S_2}\,\overline{S_1}\,S_0\cdot EW\,car + S_2\overline{S_1}\,S_0\cdot NS car$$

$$S_2' = \overline{S_2}\,S_1\overline{S_0} + S_2\overline{S_1}$$

Next State Functions: Determined by Inputs And Current State

Next-State Table for Extended Controller

current state	inputs			next state	current state	inputs			next state
$S_2S_1S_0$	NS-car	EW-car	TS	$S'_2S'_1S'_0$	$S_2S_1S_0$	NS-car	EW-car	TS	$S'_2S'_1S'_0$
0 0 0	X	X	0	0 0 0	1 0 0	X	X	0	1 0 0
0 0 0	X	X	1	0 0 1	1 0 0	X	X	1	1 0 1
0 0 1	X	0	X	0 0 1	1 0 1	0	X	X	1 0 1
0 0 1	X	1	X	0 1 0	1 0 1	1	X	X	1 1 0
0 1 0	X	X	X	1 0 0	1 1 0	X	X	X	0 0 0
0 1 1	X	X	X	X X X	1 1 1	X	X	X	X X X

Note unused states, symmetries

S_2	S_1	S_0	NSg	NSy	NSr	EWg	EWy	EWr	TR
0	0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	0	0	1	0
0	1	0	0	1	0	0	0	1	1
0	1	1	X	X	X	X	X	X	X
1	0	0	0	0	1	1	0	0	0
1	0	1	0	0	1	1	0	0	0
1	1	0	0	0	1	0	1	0	1
1	1	1	X	X	X	X	X	X	X

OUTPUT FUNCTIONS BASED ON CURRENT STATE

NSg = ?

OUTPUT FUNCTIONS

S_2	S_1	S_0	NSg	NSy	NSr	EWg	EWy	EWr	TR
0	0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	0	0	1	0
0	1	0	0	1	0	0	0	1	1
0	1	1	X	X	X	X	X	X	X
1	0	0	0	0	1	1	0	0	0
1	0	1	0	0	1	1	0	0	0
1	1	0	0	0	1	0	1	0	1
1	1	1	X	X	X	X	X	X	X

$$NSg = \overline{S_2}\,\overline{S_1},\ EWg = S_2\overline{S_1}$$

$$NSy = \overline{S_2}\,S_1\overline{S_0},\ EWy = S_2S_1\overline{S_0}$$

$$NSr = S_2,\ EWr = \overline{S_2}$$

OUTPUT FUNCTIONS

S_2	S_1	S_0	NSg	NSy	NSr	EWg	EWy	EWr	TR
0	0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	0	0	1	0
0	1	0	0	1	0	0	0	1	1
0	1	1	X	X	X	X	X	X	X
1	0	0	0	0	1	1	0	0	0
1	0	1	0	0	1	1	0	0	0
1	1	0	0	0	1	0	1	0	1
1	1	1	X	X	X	X	X	X	X

$$NSg = \overline{S_2}\,\overline{S_1},\ EWg = S_2\overline{S_1}$$

$$NSy = \overline{S_2}\,S_1\overline{S_0},\ EWy = S_2S_1\overline{S_0}$$

$$NSr = S_2,\ EWr = \overline{S_2}$$

$$TR = S_1\overline{S_0}$$

FSM PROBLEM : Train Station
Similar to LRT system – Rapid Transit

Given a Problem Definition: We derive the FSM

Specifications: Train going between stations, picking and dropping passengers

***Inputs:** B (Buzz-button Pushed to Stop)

S (Train station is coming up yes or no)

P (Passengers waiting to enter the train)

***States:** That the train can be in. What are possibilities

***Outputs:** **C** :Train Doors Closed(C=1) Doors Closed (C=0)

R :Train is Running(R=1) or not Running(R=0)

Description:

*Train will keep running. It will receive a signal 'S' that a station is coming up.

*The Train will only stop at the station if there is also a B signal where a passenger pushed the buzzer **OR** a P signal indicating that passengers are waiting at the upcoming station.

*S will always turn high before, B or P.

*Otherwise, if only S is true, the train will keep running

*Once stopped at the station the Doors are open and the doors remain open as long as passengers keep entering.

*As soon as there are no more passengers- the train continues to a Running state again.

Extending Train Station FSM

***Inputs:** B (Button Pushed to Stop)

S (Train station is coming up yes or no)

P (Passengers waiting to enter the train)

***Outputs:** **C** :Train Doors Open or Closed

R :Train is Running or not Running

Train Must Go Through 3 states when getting to a Station

***First Cycle Door Closed, 2nd Cycle Door Open while passengers coming in**

***Third Cycle Standing at Station but door Closed**