FSM encoding

as truth table

	input	state	next state	output	
000	0	S_0	S_3	0	110
001	1	S_0	S_1	1	011
010	0	S_1	S_2	1	101
011	1	S_1	S_3	0	110
100	0	S_2	S_3	0	110
101	1	S_2	S_1	1	011
110	0	S_3	S_3	0	110
111	1	S_3	S_3	0	110

input | output

only 3 output-state combi, can be encoded into first 2 bits actualy.

4 states: S_0 00, S_1 01, S_2 10, S_3 11

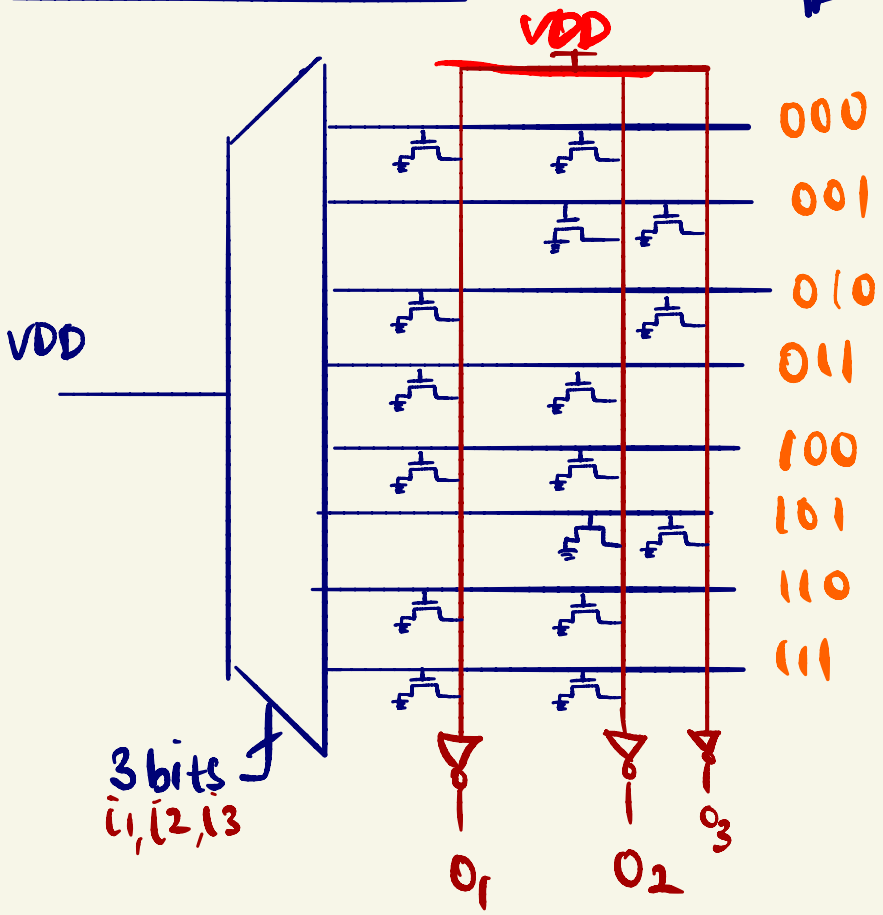
2 inputs: 1, 0

2 outputs: 1, 0

FSM as ROM

implemented as ROM

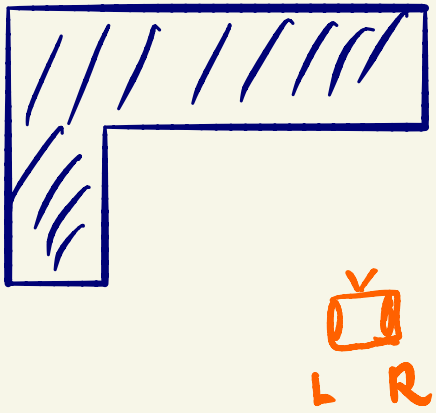
FSM is a truth table



$i_1 i_2 i_3$	$O_1 O_2 O_3$
000	110
001	011
010	101
011	110
100	110
101	011
110	110
111	110

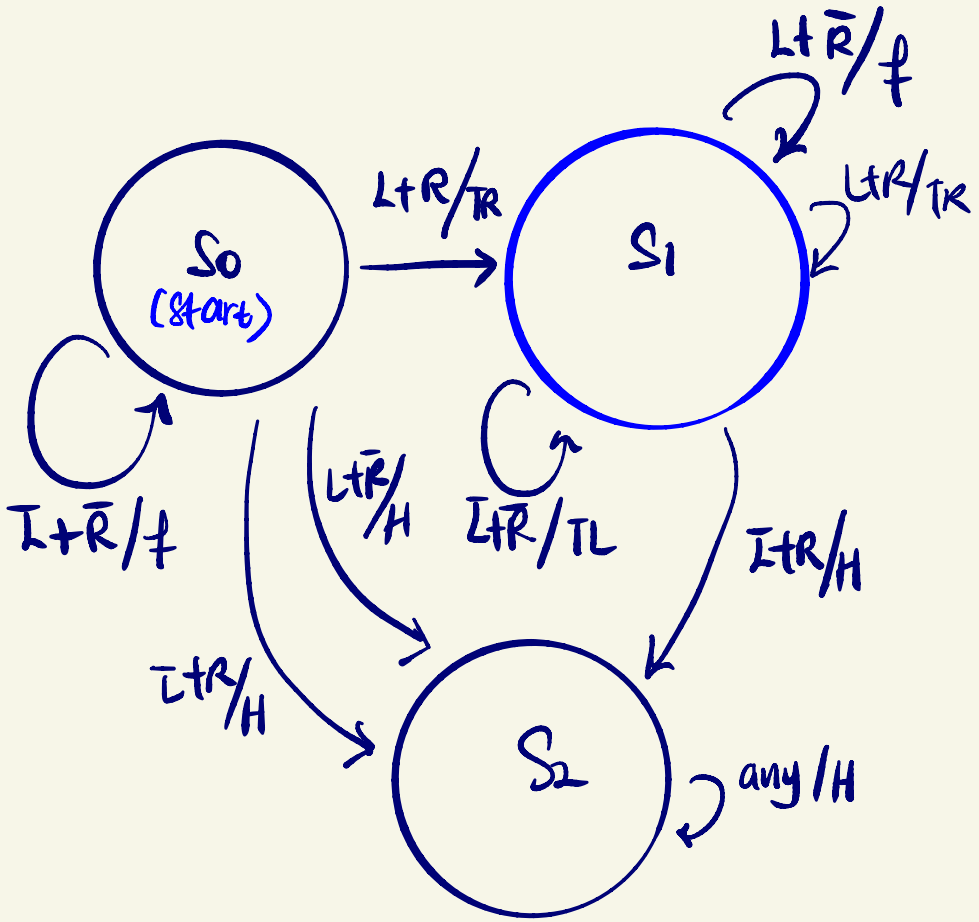
input | output

Another FSM Example

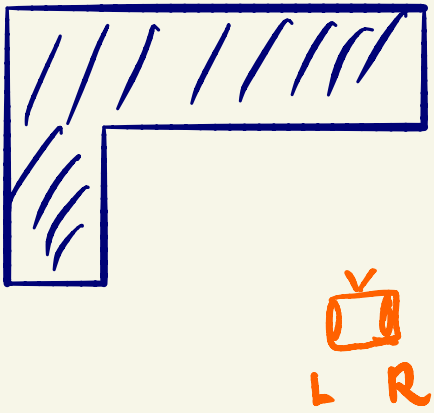


L/R = 0 → white

L/R = 1 → black

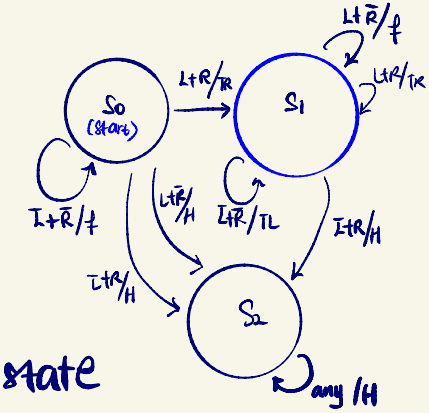


Another FSM Example



L/R = 0 → white

L/R = 1 → black



Input	State	output	state
$L+R$	S_0	TR	S_1
$\bar{L}+\bar{R}$	S_0	F	S_0
$\bar{L}+R$	S_0	H	S_2
$L+\bar{R}$	S_0	H	S_2
$L+R$	S_1	TR	S_1
$\bar{L}+R$	S_1	H	S_2
$\bar{L}+\bar{R}$	S_1	TL	S_1
$L+\bar{R}$	S_1	F	S_1
any	S_2	H	S_2

Another FSM Example

	Input	State	output	state	
0000	L+R	S_0	TR	S_1	0101
1100	$\bar{L}+\bar{R}$	S_0	F	S_0	0000
1000	$\bar{L}+R$	S_0	H	S_2	1110
0100	$L+\bar{R}$	S_0	H	S_2	1110
0001	L+R	S_1	TR	S_1	0101
1001	$\bar{L}+R$	S_1	H	S_2	1110
1101	$\bar{L}+\bar{R}$	S_1	TL	S_1	1001
0101	$L+\bar{R}$	S_1	F	S_1	0001
0010	any	S_2	H	S_2	1110
0110					1110
1010					1110
1110					1110

S_0 00
S_1 01
S_2 10

L+R 00
$L+\bar{R}$ 01
$\bar{L}+R$ 10
$\bar{L}+\bar{R}$ 11

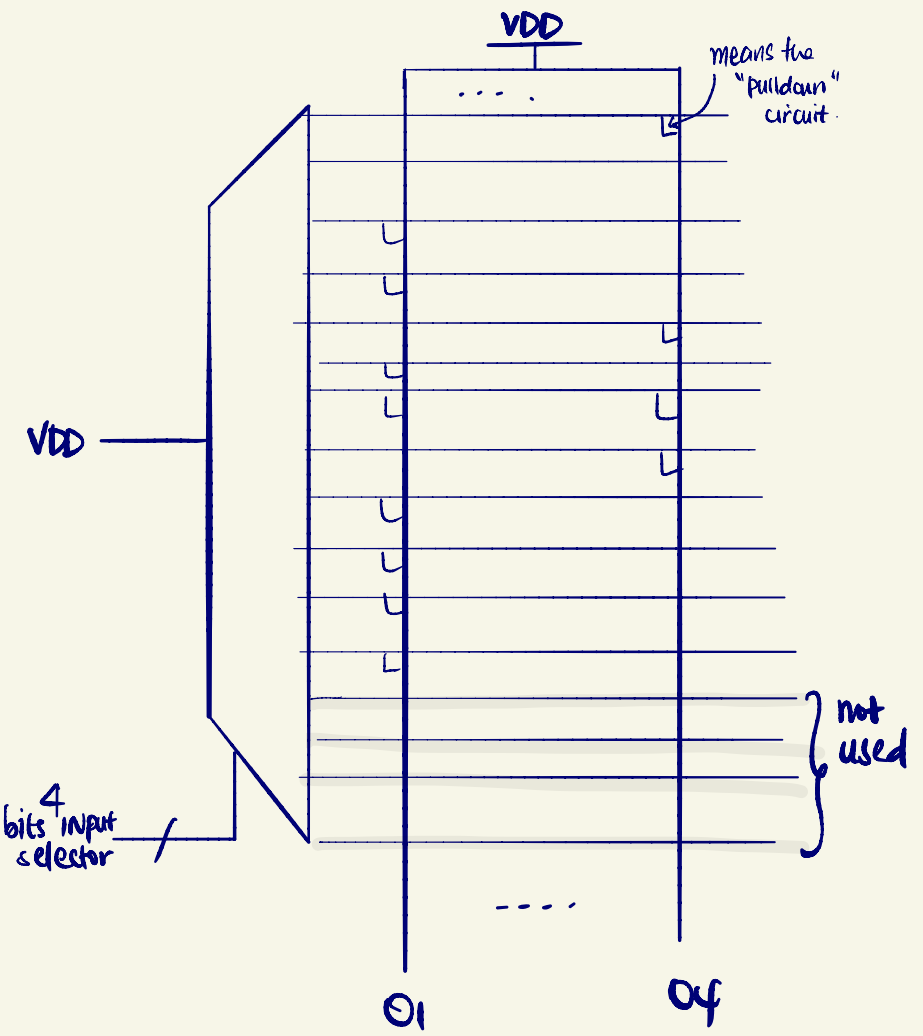
F 00
TR 01
TL 10
H 11

Another FSM Example

4 bits input selector

	Input	State	output state	
0000	L+R	S_0	TR	S_1 0101
1100	$\bar{L}+\bar{R}$	S_0	F	S_0 0000
1000	$\bar{L}+R$	S_0	H	S_2 1110
0100	$L+\bar{R}$	S_0	H	S_2 1110
0001	$L+R$	S_1	TR	S_1 0101
1001	$\bar{L}+R$	S_1	H	S_2 1110
1101	$\bar{L}+\bar{R}$	S_1	TL	S_1 1001
0101	$L+\bar{R}$	S_1	F	S_1 0001
0010	any	S_2	H	S_2 1110
0110				1110
1010				1110
1111				1110

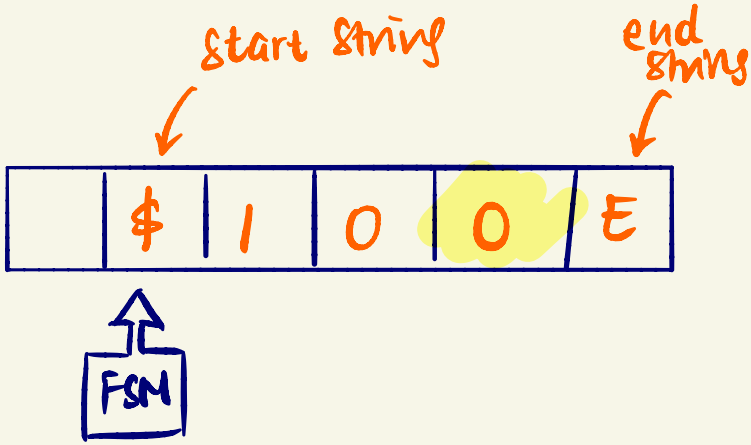
4 output bits



Task: take input N

add 1 in binary

can overwrite $ if overflow



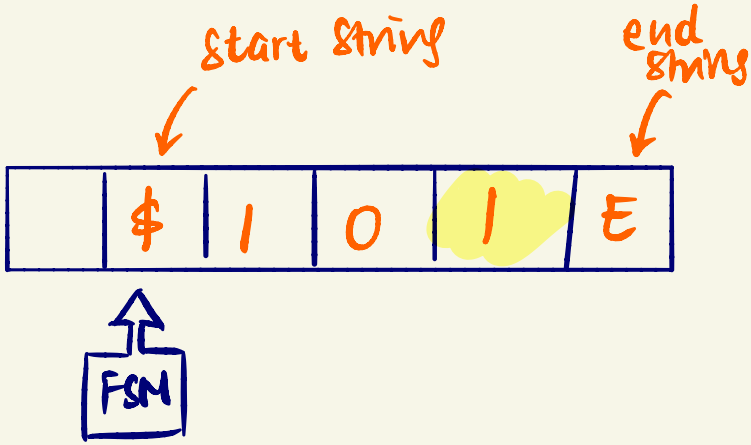
S_{in}	Input	S_{next}	write	move

Case 1

Task: take input N

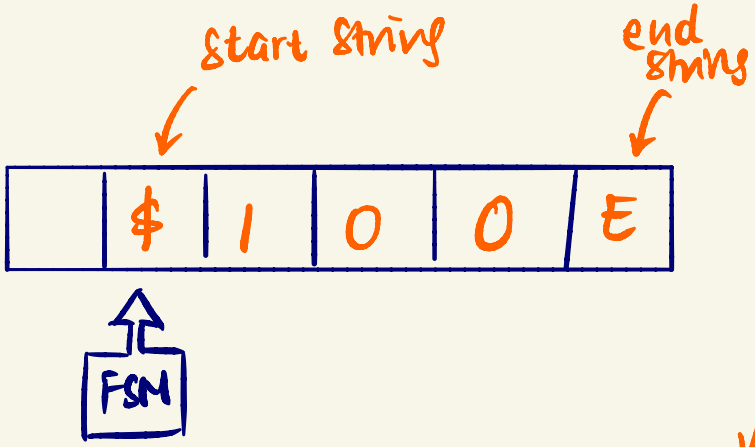
add 1 in binary

can overwrite $ if overflow



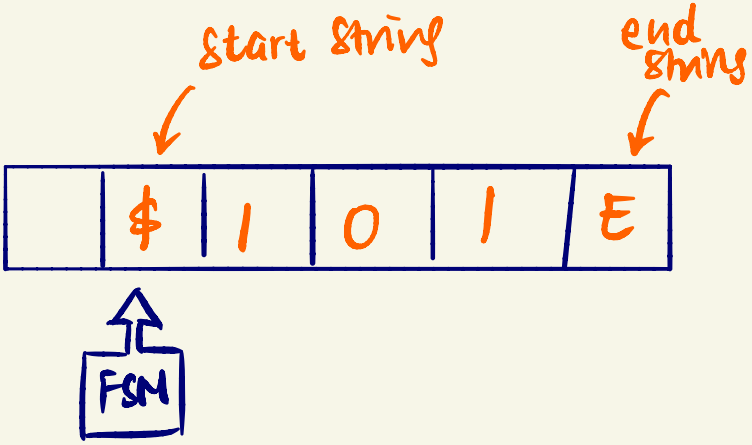
S_{in}	Input	S_{next}	write	move

case 2



S_{in}	Input	S_{next}	write	move
S_0	$	S_R	$	R
S_R	0/1	S_R	0/1	R
S_R	E	S_W	E	L
S_A	0	S_L	1	L
S_L	0/1	S_L	0/1	L
S_L	$	HALT	$	R

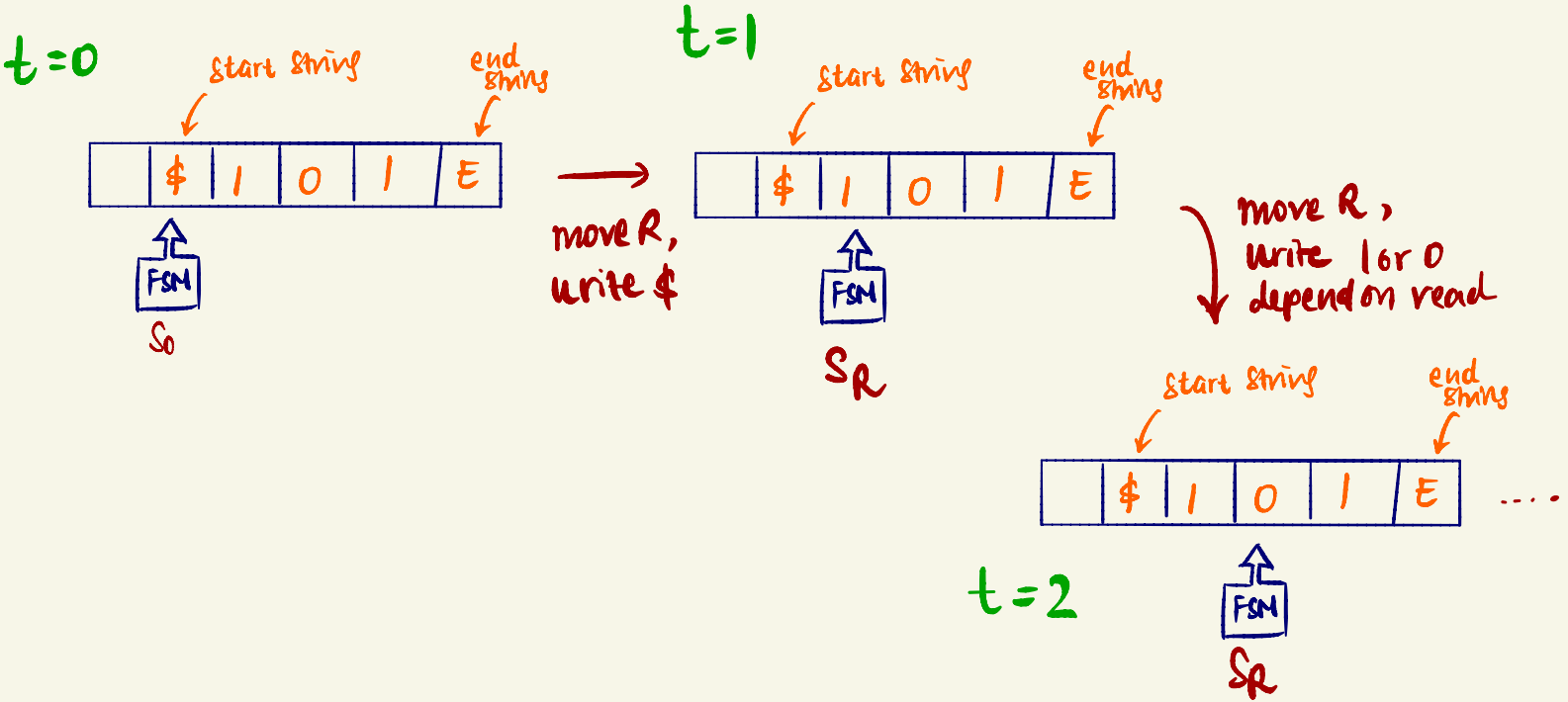
case 1:
LSB is 0



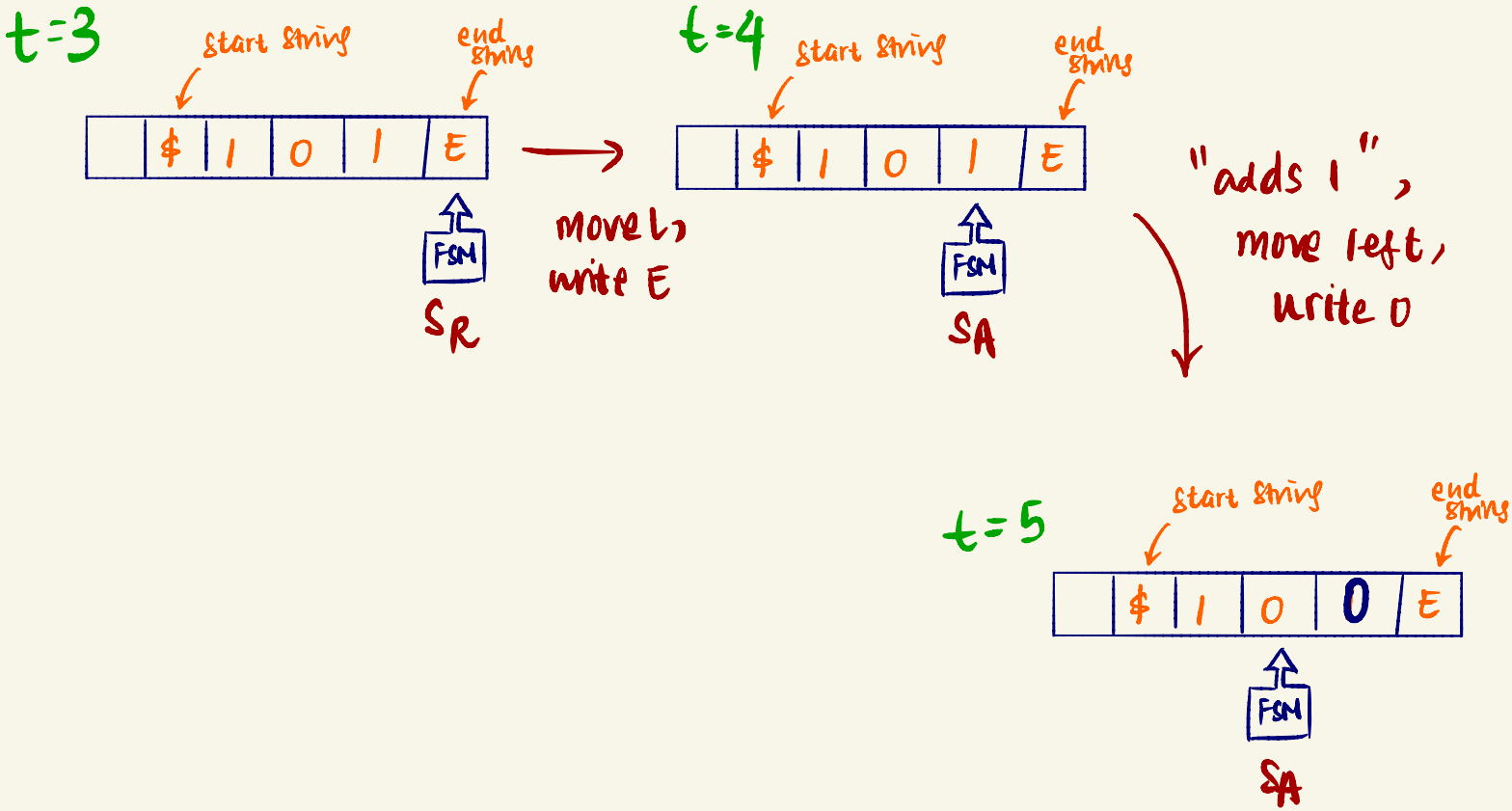
S_{in}	Input	S_{next}	write	move
S_0	$	S_R	$	R
S_R	0/1	S_R	0/1	R
S_R	E	S_A	E	L
S_A	1	S_A	0	L
S_A	$	HALT	1	R

case 2:
LSB is 1

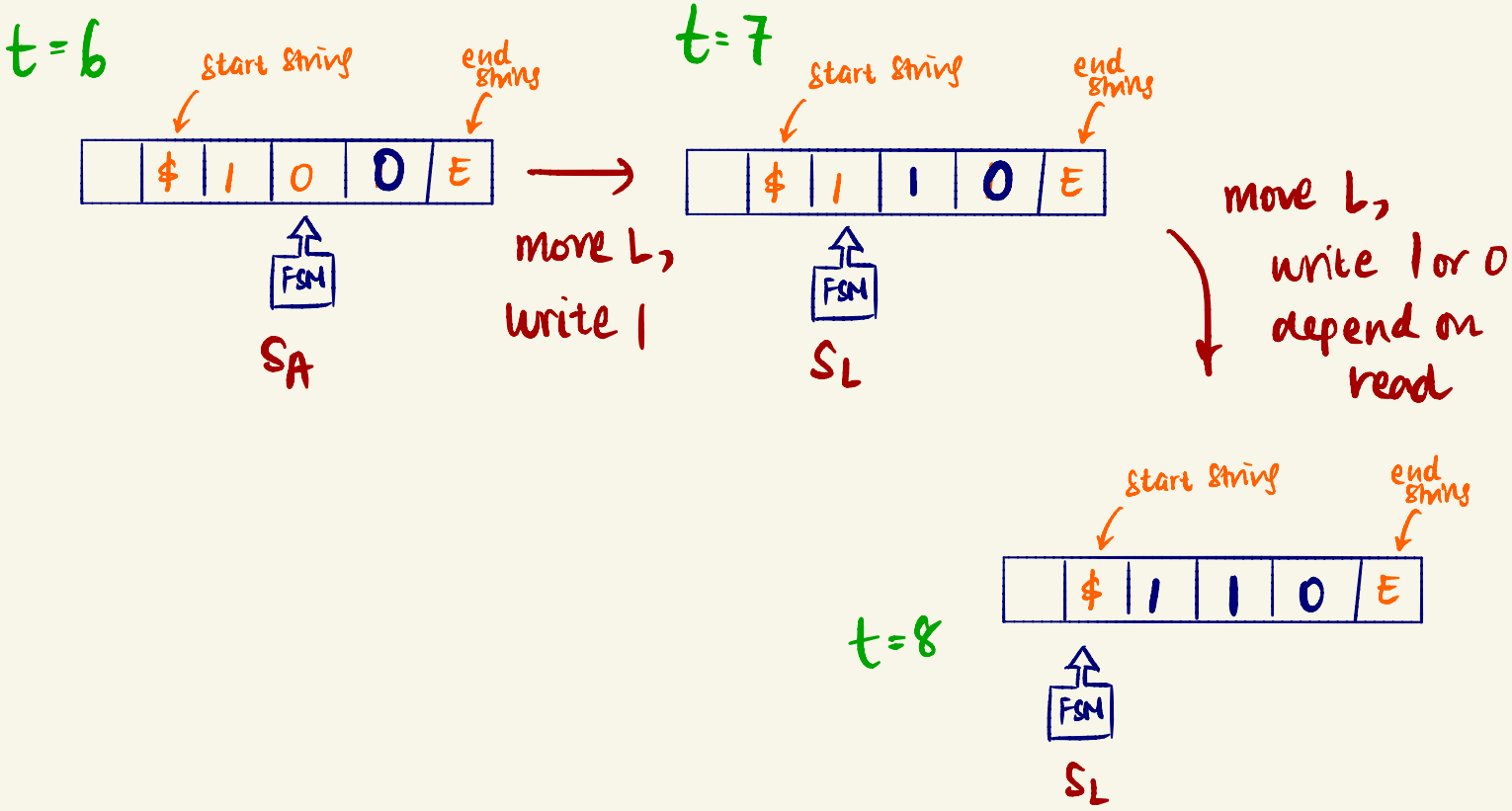
Lets run it !


t=0
start string
end string
$
1
0
1
E
FSM
$0
move R,
write $
t=1
start string
end string
$
1
0
1
E
FSM
$R
move R,
write 1 or 0
depend on read
start string
end string
$
1
0
1
E
....
t=2
FSM
$R

Lets run it !

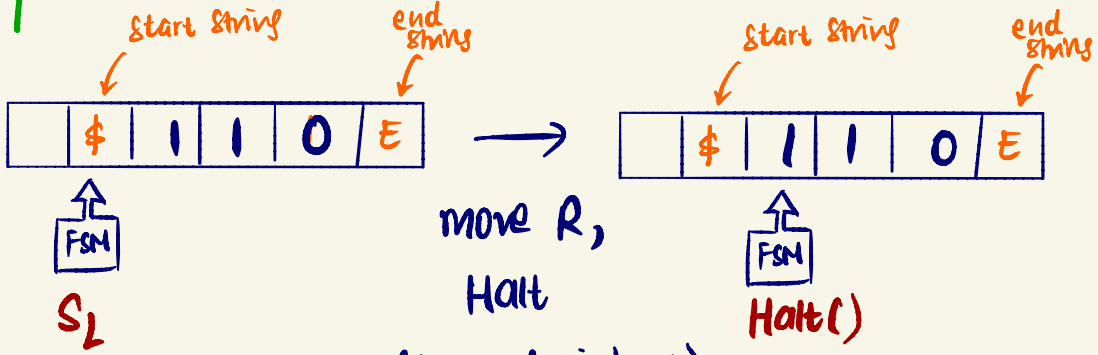

t=3
start string
end string
$
1
0
1
E
FSM
$R
move L,
write E
t=4
start string
end string
$
1
0
1
E
FSM
$A
"adds 1",
move left,
write 0
t=5
start string
end string
$
1
0
0
E
FSM
$A

Lets run it !


t=6
start string
end string
$
1
0
0
E
FSM
$A
move L,
write 1
t=7
start string
end string
$
1
1
0
E
FSM
$L
move L,
write 1 or 0
depend on
read
t=8
start string
end string
$
1
1
0
E
FSM
$L

Lets run it !

t=9



What about other cases?

S_{in}	Input	S_{next}	write	move
S_0	\$	S_R	\$	R
S_R	0/1	S_R	0/1	R
S_R	E	S_A	E	L
S_A	1	S_A	0	L
S_A	\$	HALT	1	R
S_A	0	S_L	1	L
S_L	0/1	S_L	0/1	L
S_L	\$	HALT	\$	R

Case 2: LSB is 1

Case 1: LSB is 0

S_0 with input 1/0 or E?

S_R with input \$?

S_A with input E?

etc...

These will all result in HALT(), basically because it is an invalid input