Course code: CE118.P13
Lecturer: Ms. Ho Ngoc Diem

Name: Tran Ngoc Anh
Student code: 22520077

ASSIGNMENT 2

(from Principles of Digital Design Reference Book)

Chapter 6 – Sequential Logic

6.16 (Sequential synthesis) Design a counter that counts in the sequence 0, 1, 3, 6, 10, 15 using four (a) D, (b) SR, (c) JK, (d) T flip-flops as memory elements and natural binary encoding.

Covert these number to binary:

0 = 0000, 1 = 0001, 3 = 0010, 6 = 0110, 10 = 1010, 15 = 1111

a) Using D flipflop:

The truth table:

current state				next sate				output			
Q3	Q2	Q1	Q0	Q3’	Q2’	Q1’	Q0’	D3	D2	D1	D0
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	1	0	0	1	1
0	0	1	1	0	1	1	0	0	1	1	0
0	1	1	0	1	0	1	0	1	0	1	0
1	0	1	0	1	1	1	1	1	1	1	1

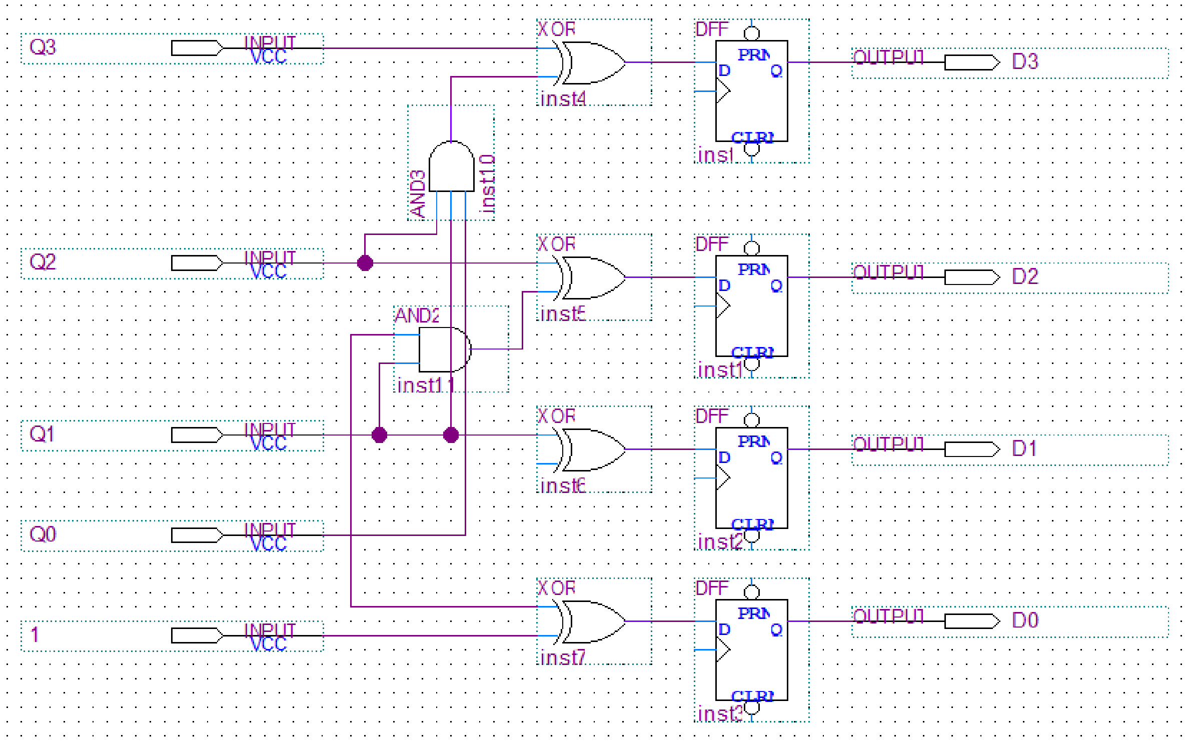
Karnaugh mapping for D flipflop, we have:

D0 = Q0 xor 1

D1 = Q1 xor Q0

D2 = Q2 xor (Q1Q0)

D3 = Q3 xor (Q2Q1Q0)



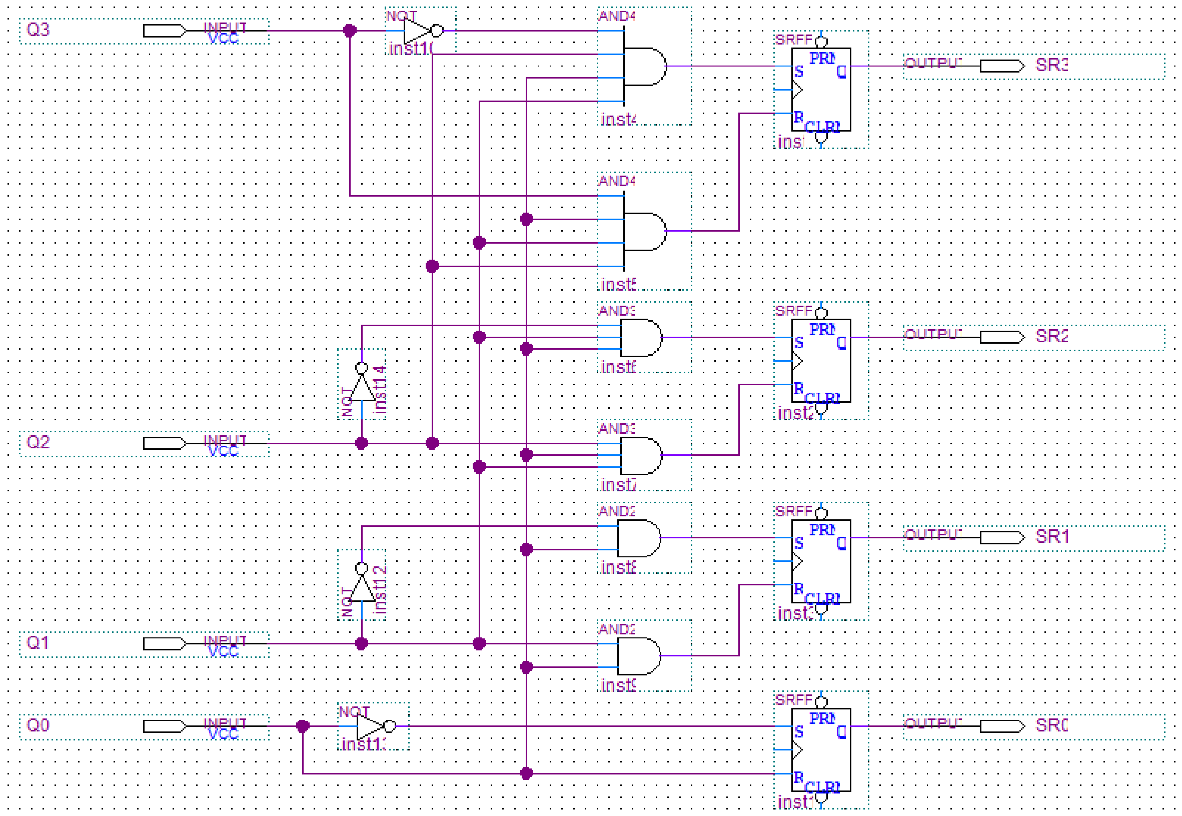
Name: Tran Ngoc Anh
Student code: 22520077

b) Using SR flipflop:

current state				next sate				output							
Q3	Q2	Q1	Q0	Q3’	Q2’	Q1’	Q0’	S3	S2	S1	S0	R3	R2	R1	R0
0	0	0	0	0	0	0	1	0	0	0	1	X	X	X	0
0	0	0	1	0	0	1	1	0	0	1	X	X	X	0	0
0	0	1	1	0	1	1	0	0	1	X	0	X	0	0	1
0	1	1	0	1	0	1	0	1	0	X	0	0	1	0	X
1	0	1	0	1	1	1	1	X	1	X	1	0	0	0	0

Karnaugh mapping for JK flipflop, we have:

S0 = Q0’,	R0 = Q0
S1 = Q0Q1’,	R1 = Q0Q1
S2 = Q2’Q1Q0,	R2 = Q2Q1Q0
S3 = Q3’Q2Q1Q0,	R3 = Q3Q2Q1Q0



c) Using JK flipflop:

current state				next sate				output							
Q3	Q2	Q1	Q0	Q3’	Q2’	Q1’	Q0’	J3	J2	J1	J0	K3	K2	K1	K0
0	0	0	0	0	0	0	1	0	0	0	1	X	X	X	X
0	0	0	1	0	0	1	1	0	0	1	X	X	X	X	0
0	0	1	1	0	1	1	0	0	1	X	X	X	X	0	1
0	1	1	0	1	0	1	0	1	X	X	0	X	1	0	X
1	0	1	0	1	1	1	1	X	1	X	1	0	X	0	X

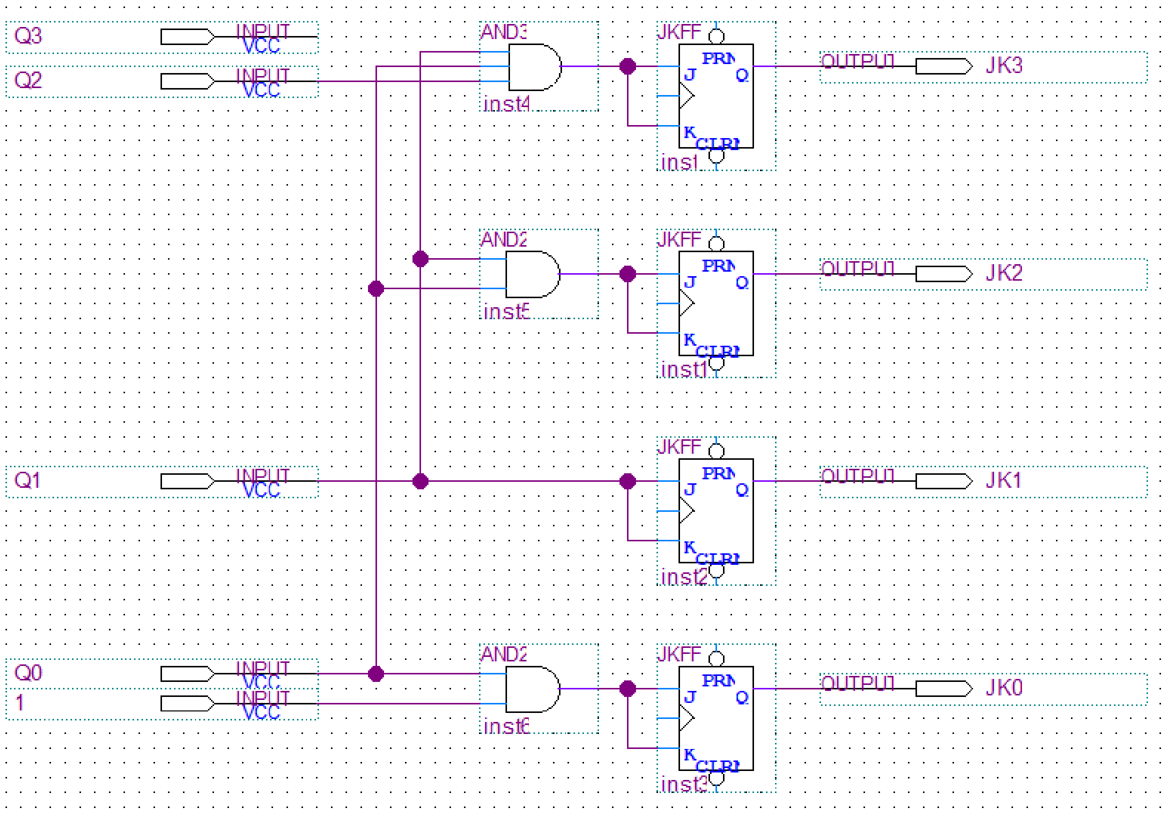
Karnaugh mapping for JK flipflop, we have:

J0 = K0 = 1

J1 = K1 = Q1

J2 = K2 = Q1Q0

J3 = K3 = Q2Q1Q0



d) Using T flipflop:

current state				next sate				output			
Q3	Q2	Q1	Q0	Q3’	Q2’	Q1’	Q0’	T3	T2	T1	T0
0	0	0	0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	1	1	0	0	1	0
0	0	1	1	0	1	1	0	0	1	0	1
0	1	1	0	1	0	1	0	1	1	0	0
1	0	1	0	1	1	1	1	0	1	0	1

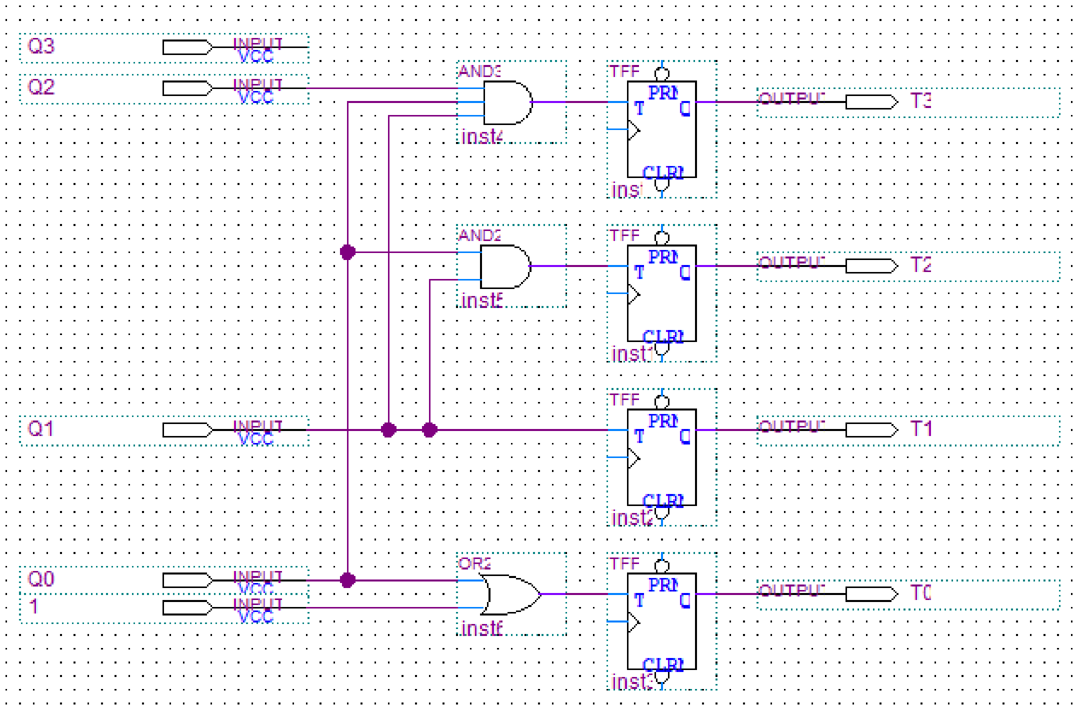
Karnaugh mapping for T flipflop, we have:

T0 = 1

T1 = Q1

T2 = Q1Q0

T3 = Q2Q1Q0

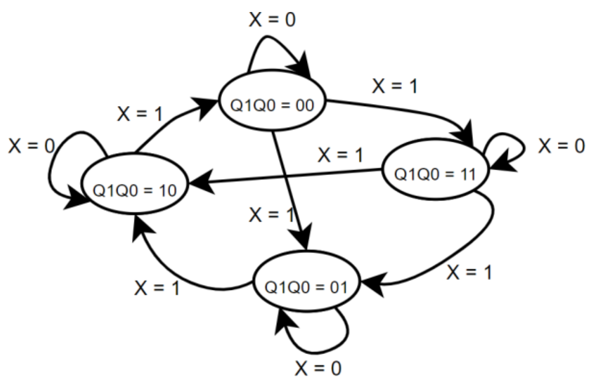


6.18 (Sequential synthesis) Design a parity checker thay counts the number of 1's in the input stream. This checker asserts its output Y if it has received an odd number of 1's on the input X. An asynchronous Reset signal returns the parity checker into its initial state. As storage elements use only (a) D, (b) JK and (c) T flip-flops.

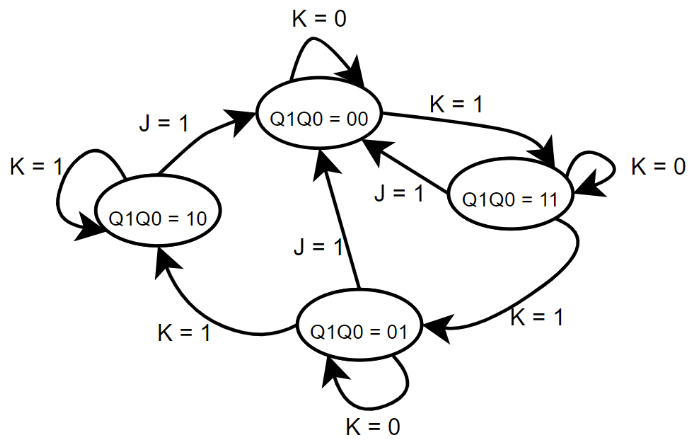
State transition table:

Current state	Input X	Next state	Output Y
S0	0	S0	0
S0	0	S1	1
S1	0	S1	1
S1	1	S0	0

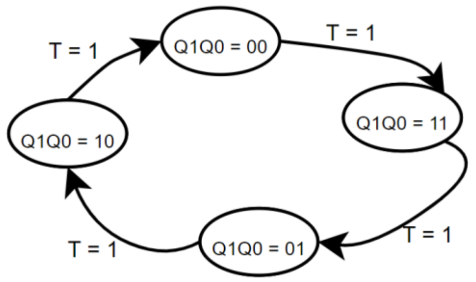
a) Using D flipflop:



b) Using JK flipflop:



c) Using T flipflop:



6.19 (Sequential synthesis) Design a recorgnizer that recognizes an input sequence that has at least three 1's. The recorgnizer has a single input X and a single output Y, in addition to an asynchronous Reset signal. The recognizer sets the output Y to 1 if the input signal X equals 1 at least three clock cycles after Reset was disasserted. For the recognizer described above:

a) Devise the state diagram.

b) Minimize the number of state.

c) Encoded the states to minimize the combinatorial logic.

d) Draw a schematic diagram using D flip-flop.

Define the state as the following:

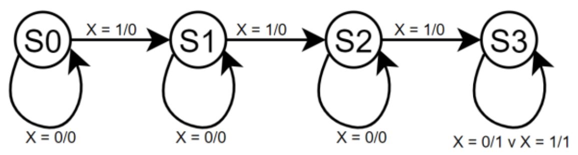
S0: 0 ones received

S1: 1 ones received

S2: 2 ones received

S3: 3 ones received

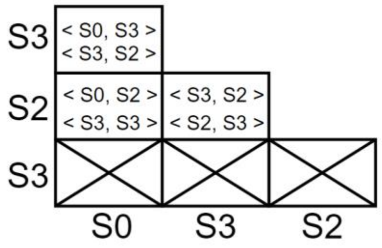
The state diagram is:



Current state	Next state	
	WALK = 0	WALK = 1
S0	S0/0	S1/0
S1	S1/0	S2/0

S2	S2/0	S3/0
S3	S3/1	S3/1

From the state/output table, we can minimize the state number by using implication table:

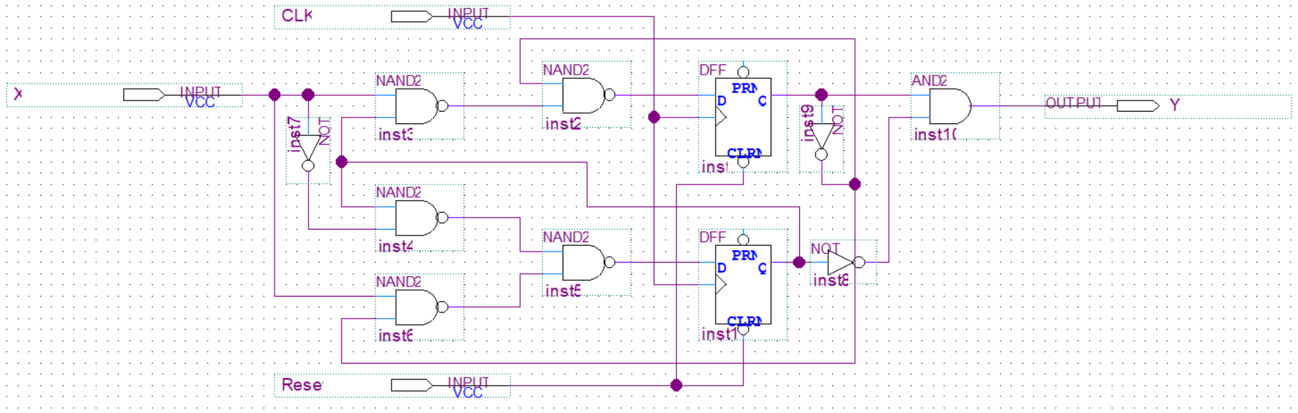


The equations defining the implimentation can be describe as:

D0 = Q0X' + Q1'X

D1 = Q1 + Q0X

Y = Q1Q0'



6.22 (Sequential synthesis) Design a simplified traffic-light controller that switches traffic lights on a crossing where a north-south (NS) street intersects an east-west (EW) street. The input to the controller is the WALK button pushed by pedestrians who want to cross the street. The outputs are two signals NS and EW that control the traffic lights in the NS and EW directions. When NS or EWW are 0, the red light is on, and when they are 1, the green lights is on. Where there are no pedestrians, NS = 0 and EW = 1 for 1 minute, followed by NS = 1 and EW = 0 for 1 minute, and so on. When a WALK button is pushed, NS and EW both become 1 for a minute when the present minute expires. After that the NS and EW signals continue alternating. For these traffic-light controller:

a) Develop a state diagram and a state/output table.

b) Minimize the number of states.

c) Encode the states.

d) Derive a logic schematic.

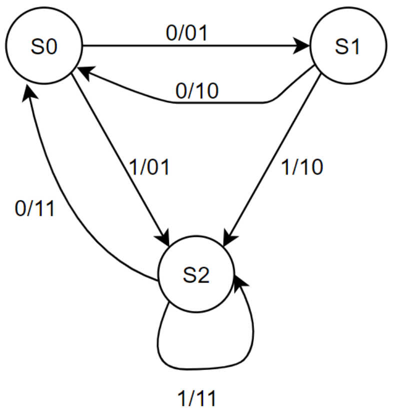
Define the state as the following:

S0: NS = 0, EW = 1

S1: NS = 1, EW = 0

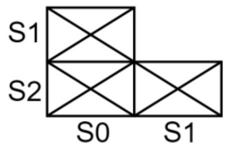
S2: NS = 1, EW = 1

The state diagram is:



Current state	**Next state**	
	WALK = 0	WALK = 1
S0	S1/01	S2/01
S1	S0/10	S2/10
S2	S0/11	S2/11

As you can see below, the number of state is minimal:



Using minimum-bit change strategy, get the minimizing state encoding as follows:

S0 = 00, S1 = 01, S2 = 10

Current state	**Next state**	
	WALK = 0	WALK = 1
00	01/01	10/01
01	00/10	10/10
11	XX/XX	XX/XX
10	00/11	10/11

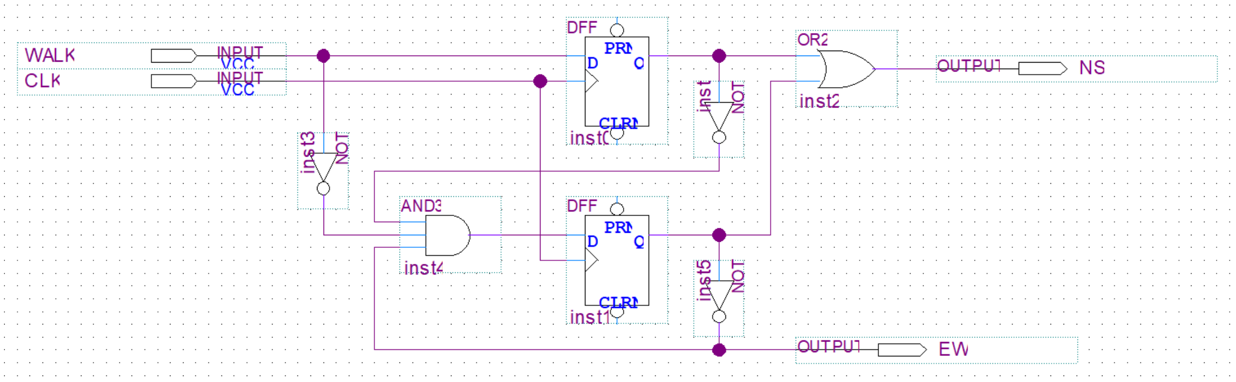
The equations defining the implimentation can be describe as:

D0 = Q1’Q0WALK

D1 = WALK

NS = Q0 + Q1

EW = Q0’


WALK
CLK
INPUT VCC
INPUT VCC
DFF
PRN
D
Q
CLRN
inst0
DFF
PRN
D
Q
CLRN
inst1
inst3
NOT
inst
NOT
inst5
NOT
AND3
inst4
OR2
inst2
OUTPUT
NS
OUTPUT
EW