Δοκιμή και Αξιοπιστία Ηλεκτρονικών Κυκλωμάτων

Άσκηση 2:

Υλοποίηση δομικών στοιχείων περιφερειακής σάρωσης

(JTAG boundary scan protocol)

Θεοδώρα Τζιάστα

ΑΜ: 4178

Άσκηση 3.1

Ως πρώτο βήμα της άσκησης υλοποιήθηκαν τα παρακάτω δομικά στοιχεία. Αυτά είναι: ο bypass register (BR), το κελί εντολών (instruction register cell – IR cell) και το κελί περιφερειακής σάρωσης (boundary scan cell - BSC). Για το κάθε ένα από αυτά αντιπαραβάλλεται το προσχέδιο που έχει δημιουργηθεί με τον σχεδιασμό που προκύπτει από το Quartus καθώς και ο κώδικας σε Verilog για το κάθε ένα.

* Bypass register

`**timescale** 1ns/1ps

**module** BR(TDI,CaptureDR,ClockDR,TDO);

**input** TDI,CaptureDR,ClockDR;

**output** TDO;

**wire** TDI,CaptureDR,andtoFF,ClockDR,TDO;

**reg** Q;

**assign** TDO=Q;

**and**(andtoFF,TDI,CaptureDR);

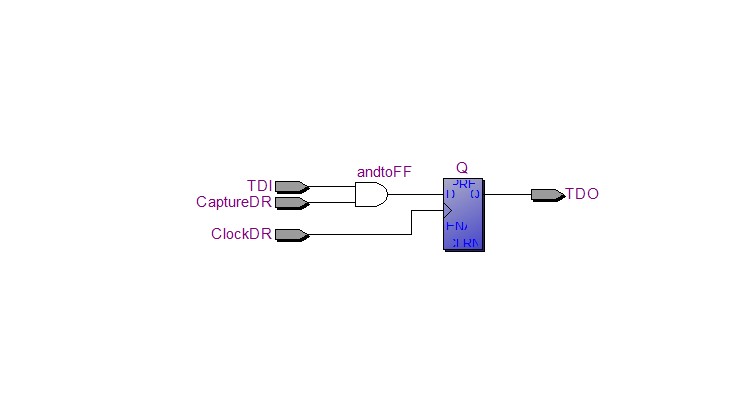
**always** @ (**posedge** ClockDR)

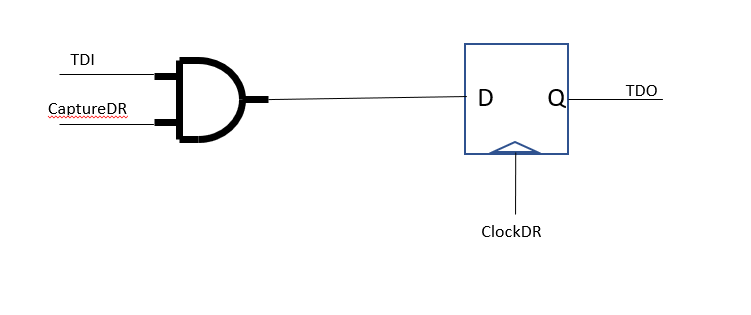
**begin**

Q<=andtoFF;

**end**

**endmodule**





* Instruction Register Cell

`**timescale** 1ns/1ps

**module** IR(Data, TDI, ShiftIR, ClockIR, UpdateIR, ParallelOut, TDO);

**input** Data, TDI, ShiftIR, ClockIR, UpdateIR;

**output** ParallelOut, TDO;

**wire** Data, TDI, ShiftIR, ClockIR, UpdateIR, ParallelOut,TDO;

**reg** SRFFQ, LFFQ;

**wire** DfromMux;

//here we create the Mux

**assign** DfromMux=(ShiftIR)?TDI:Data;

**assign** TDO=SRFFQ;

**assign** ParallelOut=LFFQ;

//here we create the SRFF

**always** @ (**posedge** ClockIR)

**begin**

SRFFQ<=DfromMux;

**end**

//here we create the LFF

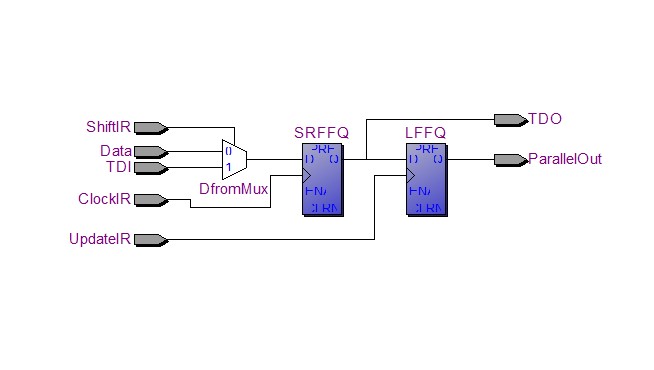
**always** @ (**posedge** UpdateIR)

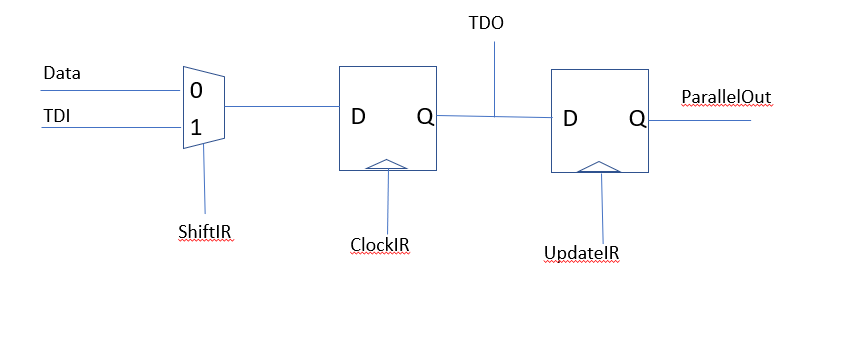
**begin**

LFFQ<=SRFFQ;

**end**

**endmodule**





* Boundary Scan Cell

`**timescale** 1ns/1ps

**module** BSC(DataIn,ShiftIn,ShiftDR,ClockDR,UpdateDR,Mode,ShiftOut,DataOut);

**input** DataIn,ShiftIn,ShiftDR,ClockDR,UpdateDR,Mode;

**output** ShiftOut,DataOut;

**wire** DataIn,ShiftIn,ShiftDR,ClockDR,UpdateDR,Mode,DfromMux,DataOut;

**reg** SRFFQ, LFFQ;

**assign** ShiftOut=SRFFQ;

**assign** D1fromLFFQ=LFFQ;

**assign** DfromMux=(ShiftDR)?ShiftIn:DataIn;

**assign** DataOut=(Mode)?LFFQ:DataIn;

//here we create the SRFF

**always** @ (**posedge** ClockDR)

**begin**

SRFFQ<=DfromMux;

**end**

//here we create the LFF

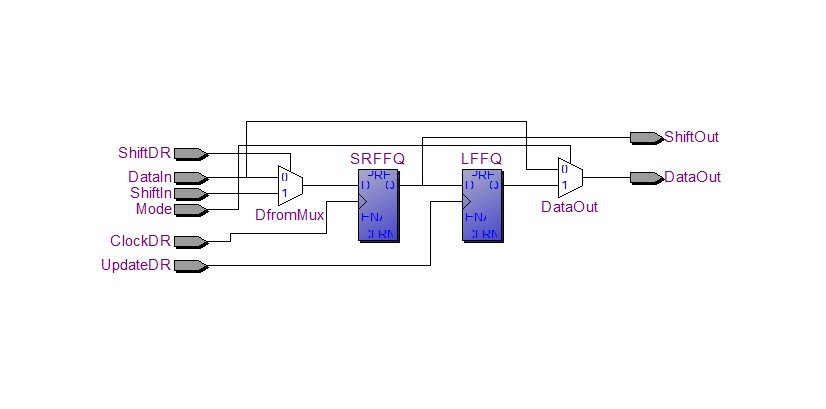
**always** @ (**posedge** UpdateDR)

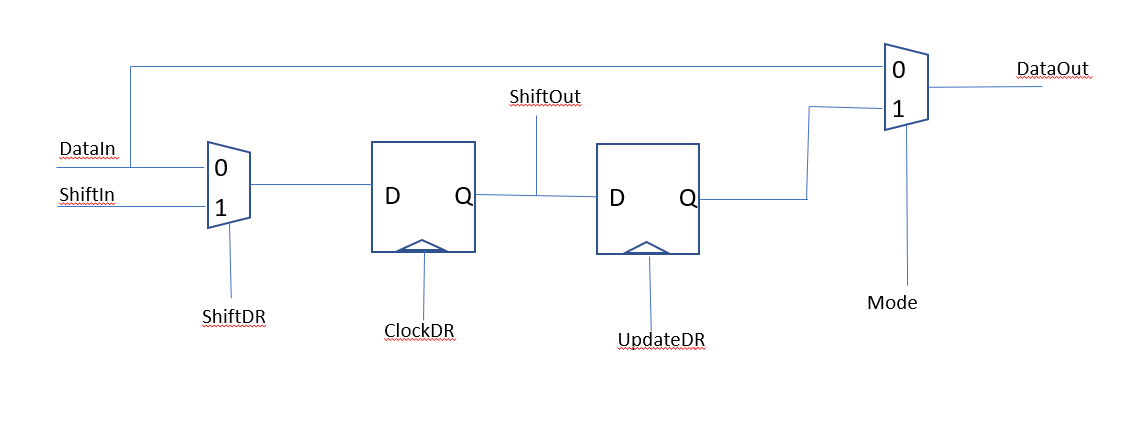
**begin**

LFFQ<=SRFFQ;

**end**

**endmodule**





Στο επόμενο στάδιο της άσκησης δημιουργήθηκε ένα testbench σε Verilog με στο οποίο επαληθεύεται η λειτουργία του BSC κελιού. Οι τιμές των εισόδων DataIn και ShiftIn είναι συμπληρωματικές σε κάθε μια περίπτωση ώστε να επιβεβαιωθεί η προώθηση των αναμενόμενων δεδομένων. Ο κώδικας φαίνεται παρακάτω και ακολουθεί και η κυματομορφή που προκύπτει:

`**timescale** 1ns/1ps

**module** BSC\_tb();

**reg** DataIn,ShiftIn,ShiftDR,ClockDR,Mode;

**wire** ShiftOut,DataOut;

BSC BSCinstance(DataIn,ShiftIn,ShiftDR,ClockDR,ClockDR,Mode,ShiftOut,DataOut);

**initial** **begin**

ClockDR=0;

#20

**forever** **begin**

#10 ClockDR=!ClockDR;

**end**

**end**

**initial** **begin**

Mode=0; // normal mode on

ShiftDR=1;// store in CAP FF shiftin data

**repeat**(4)@(**posedge** ClockDR)

ShiftIn<=0;

DataIn<=1;

ShiftDR=0; // store in CAP FF internal logic data

**repeat**(4)@(**posedge** ClockDR)

ShiftIn<=0;

DataIn<=1;

Mode=1; // test mode on

ShiftDR=1;// store in CAP FF shiftin data

**repeat**(4)@(**posedge** ClockDR)

DataIn<=1;

ShiftIn<=0;

ShiftDR=0;// store in CAP FF internal logic data

**repeat**(4)@(**posedge** ClockDR)

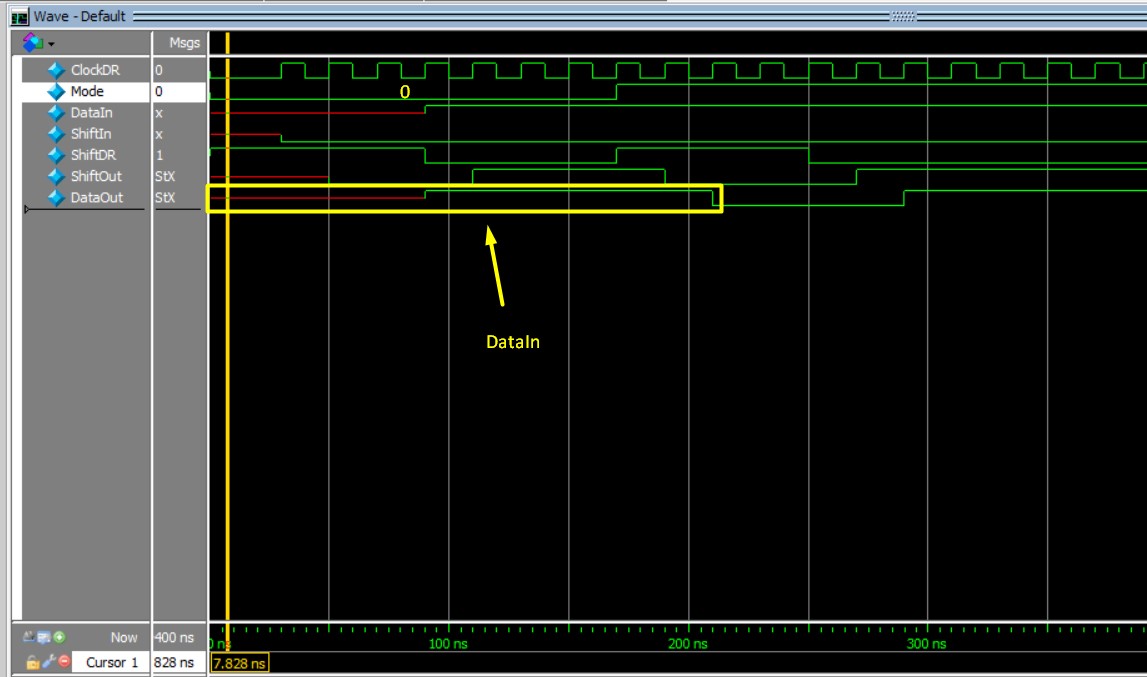
DataIn<=1;

ShiftIn<=0;

**end**

**endmodule**

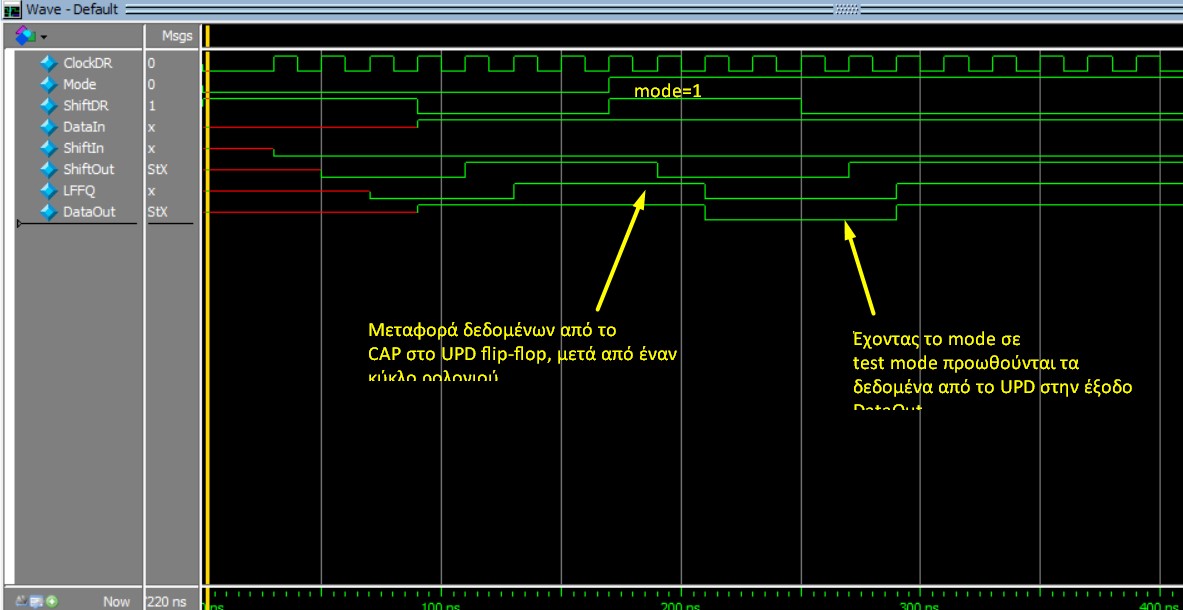
Αρχικά, έχοντας ενεργοποιημένο σε normal mode το mode (Mode=0) για κάποιους κύκλους ρολογιού φαίνεται ότι τα δεδομένα της εισόδου DataIn(internal logic) προωθούνται στην έξοδο DataOut (internal logic). Αυτό επιβεβαιώνεται και από τις κυματομορφές:



Παράλληλα, έχοντας το ShiftDr=1 αποθηκεύονται τα δεδομένα της εισόδου shiftIn στο CAP flip-flop, ενώ όταν το ShiftDr=0 τότε στο CAP flip-flop αποθηκεύονται τα δεδομένα της εισόδου DataIn. Αυτό γίνεται αντιληπτό παρατηρώντας την έξοδο ShiftOut στην παρακάτω κυματομορφή:

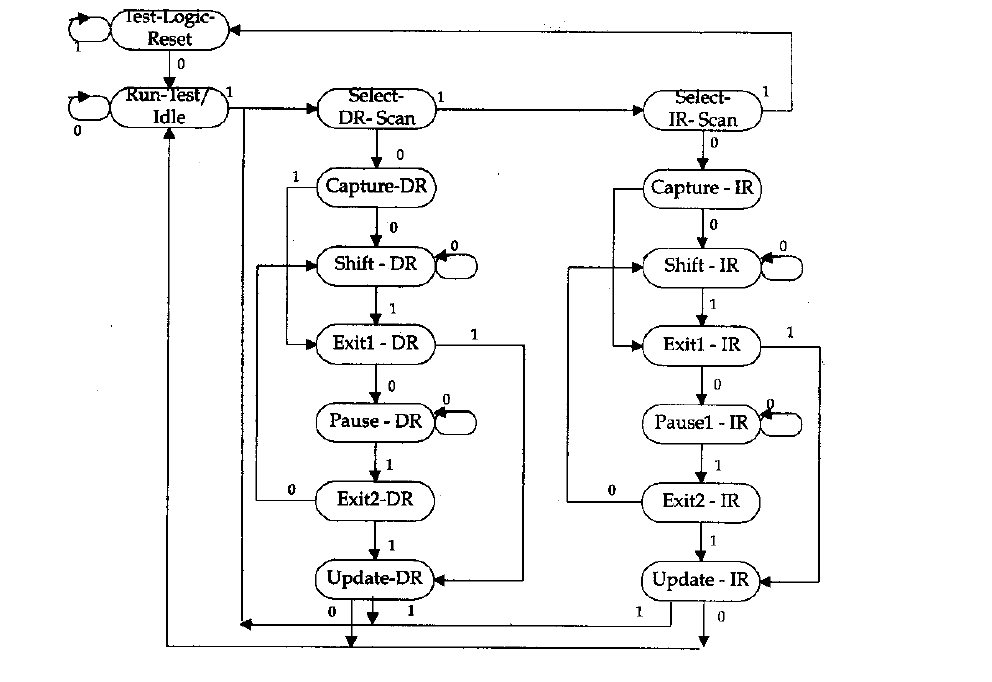


Έπειτα ενεργοποιείται σε test mode το mode (mode=1), έτσι στην έξοδο DataOut προωθούνται τα δεδομένα που βρίσκονται αποθηκευμένα στο UPD flip-flop. Τα παραπάνω φαίνονται στην κυματομορφή που ακολουθεί. Ακόμα στην κυματομορφή επιβεβαιώνεται και η μεταφορά δεδομένων μεταξύ των δύο flip-flop:



Άσκηση 3.2

Στην άσκηση αυτή υλοποιείται το FSM που περιέχεται στον tap ελεγκτή. Ο ελεγκτής περιέχει το πεπερασμένο αυτόματο (finite state machine FSM) του JTAG πρωτοκόλλου. Συγκεκριμένα σε κάθε κύκλο του ρολογιού TCK, το FSM αλλάζει κατάσταση σύμφωνα με την τιμή του σήματος TMS και την τρέχουσα κατάσταση στην οποία βρίσκεται το FSM. O ελεγκτής λειτουργεί σύμφωνα με το σχήμα που ακολουθεί:



Ο κώδικας που υλοποιεί το πεπερασμένο αυτόματο φαίνεται παρακάτω:

`**timescale** 1ns/1ps

**module** tap\_controller(TCK,TMS,TRST,state);

**input** TCK,TMS,TRST;

**output** state;

**wire** TCK,TMS,TRST;

**wire** [3:0] state;

//states

**parameter** [3:0] Test\_Logic\_Reset= 4'b0000;//0

**parameter** [3:0] Run\_Test\_Idle= 4'b1000;//8

**parameter** [3:0] Select\_DR\_Scan= 4'b0001;//1

**parameter** [3:0] Capture\_DR= 4'b0010;//2

**parameter** [3:0] Shift\_DR= 4'b0011;//3

**parameter** [3:0] Exit1\_DR= 4'b0100;//4

**parameter** [3:0] Pause\_DR= 4'b0101;//5

**parameter** [3:0] Exit2\_DR= 4'b0110;//6

**parameter** [3:0] Update\_DR= 4'b0111;//7

**parameter** [3:0] Select\_IR\_Scan= 4'b1001;//9

**parameter** [3:0] Capture\_IR= 4'b1010;//10

**parameter** [3:0] Shift\_IR= 4'b1011;//11

**parameter** [3:0] Exit1\_IR= 4'b1100;//12

**parameter** [3:0] Pause1\_IR= 4'b1101;//13

**parameter** [3:0] Exit2\_IR= 4'b1110;//14

**parameter** [3:0] Update\_IR= 4'b1111;//15

**reg** [3:0] current\_state;

**assign** state=current\_state;

**always** @(**posedge** TCK,**posedge** TRST)

**begin**

//asynchronous reset

**if**(TRST==1'b1) **begin**

current\_state<=Test\_Logic\_Reset;

**end**

**else** **begin**

current\_state<=Test\_Logic\_Reset; //initailize the state

**case**(current\_state)

Test\_Logic\_Reset:

**case**(TMS)

0: current\_state<=Run\_Test\_Idle;

1: current\_state<=Test\_Logic\_Reset;

**endcase**

Run\_Test\_Idle:

**case**(TMS)

0:current\_state<= Run\_Test\_Idle;

1:current\_state<=Select\_DR\_Scan;

**endcase**

Select\_DR\_Scan:

**case**(TMS)

0:current\_state<= Capture\_DR;

1:current\_state<=Select\_IR\_Scan;

**endcase**

Capture\_DR:

**case**(TMS)

0:current\_state<= Shift\_DR;

1:current\_state<=Exit1\_DR;

**endcase**

Shift\_DR:

**case**(TMS)

0:current\_state<= Shift\_DR;

1:current\_state<=Exit1\_DR;

**endcase**

Exit1\_DR:

**case**(TMS)

0:current\_state<= Pause\_DR;

1:current\_state<=Update\_DR;

**endcase**

Pause\_DR:

**case**(TMS)

0:current\_state<= Pause\_DR;

1:current\_state<=Exit2\_DR;

**endcase**

Exit2\_DR:

**case**(TMS)

0:current\_state<= Shift\_DR;

1:current\_state<=Update\_DR;

**endcase**

Update\_DR:

**case**(TMS)

0:current\_state<= Run\_Test\_Idle;

1:current\_state<=Select\_DR\_Scan;

**endcase**

Select\_IR\_Scan:

**case**(TMS)

0:current\_state<= Capture\_IR;

1:current\_state<=Test\_Logic\_Reset;

**endcase**

Capture\_IR:

**case**(TMS)

0:current\_state<= Shift\_IR;

1:current\_state<=Exit1\_IR;

**endcase**

Shift\_IR:

**case**(TMS)

0:current\_state<= Shift\_IR;

1:current\_state<=Exit1\_IR;

**endcase**

Exit1\_IR:

**case**(TMS)

0:current\_state<= Pause1\_IR;

1:current\_state<=Update\_IR;

**endcase**

Pause1\_IR:

**case**(TMS)

0:current\_state<= Pause1\_IR;

1:current\_state<=Exit2\_IR;

**endcase**

Exit2\_IR:

**case**(TMS)

0:current\_state<= Shift\_IR;

1:current\_state<=Update\_IR;

**endcase**

Update\_IR:

**case**(TMS)

0:current\_state<= Run\_Test\_Idle;

1:current\_state<=Select\_DR\_Scan;

**endcase**

**endcase**

**end**

**end**

**endmodule**

Έπειτα, δημιουργείται testbench με το οποίο επιβεβαιώνεται η σωστή λειτουργία του παραπάνω ελεγκτή. Στην είσοδο TMS του ελεγκτή δίνονται ένα προς ένα τα bit του inputVector=10010 ξεκινώντας με το LSB, το δεξιότερο bit που είναι 0. Σύμφωνα με το παραπάνω κώδικα και με το διάγραμμα του ελεγκτή του JTAG το FSM αναμένεται να είναι στην κατάσταση Exit1-DR=0100. Οι καταστάσεις από τις οποίες πρέπει να περάσει το αυτόματο είναι οι εξής:

Test-Logic-Reset Run-Test-Idle Select-DR-Scan Capture-DR Shift-DR Exit-DR

Μόλις το σήμα TRST ενεργοποιηθεί το FSM επιστρέφει στην αρχική του κατάσταση. Τα παραπάνω γίνονται αντιληπτά από την κυματομορφή που ακολουθεί.

