dscal
DIGITAL SYSTEMS & COMPUTER ARCHITECTURE LABORATORY

Εργαστήριο Λογικής Σχεδίασης

VHDL

Βασιλόπουλος Διονύσης

Ε.ΔΙ.Π. Τμήματος Πληροφορικής & Τηλεπικοινωνιών - ΕΚΠΑ

ΤΜΗΜΑ
ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

VHDL – Ακολουθιακά Κυκλώματα

Παράδειγμα

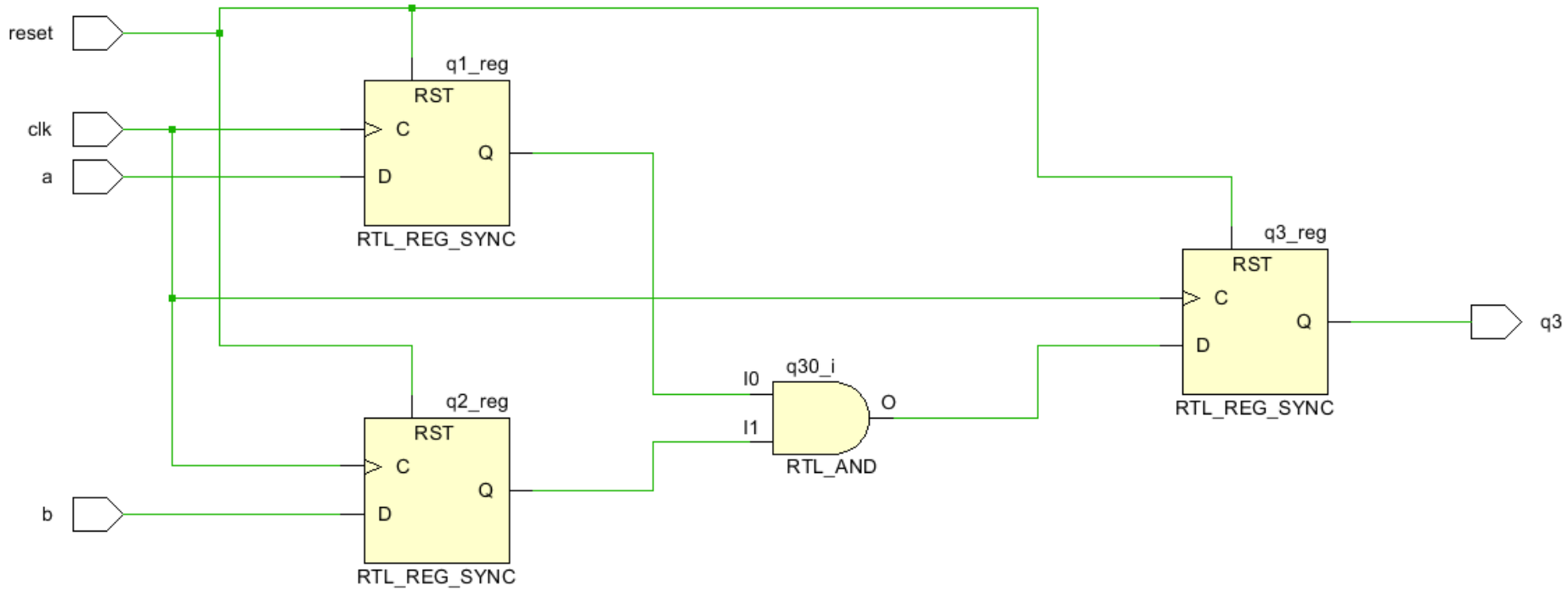
Τι κύκλωμα μοντελοποιεί ο κώδικας;

Πως υπολογίζεται η συχνότητα λειτουργίας του κυκλώματος μετά τη σύνθεση;

```
process (clk) is
begin
   if clk'event and clk='1' then
      if reset = '1' then
         q3<='0';q1<='0';q2<='0';
      else
         q1<=a;
         q2<=b;
         q3<=q1 and q2;
      end if;
   end if;
end process;
```

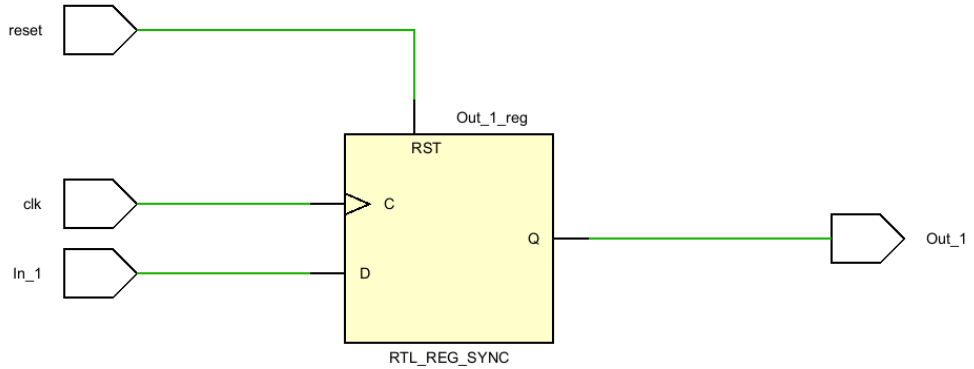
VHDL – Ακολουθιακά Κυκλώματα

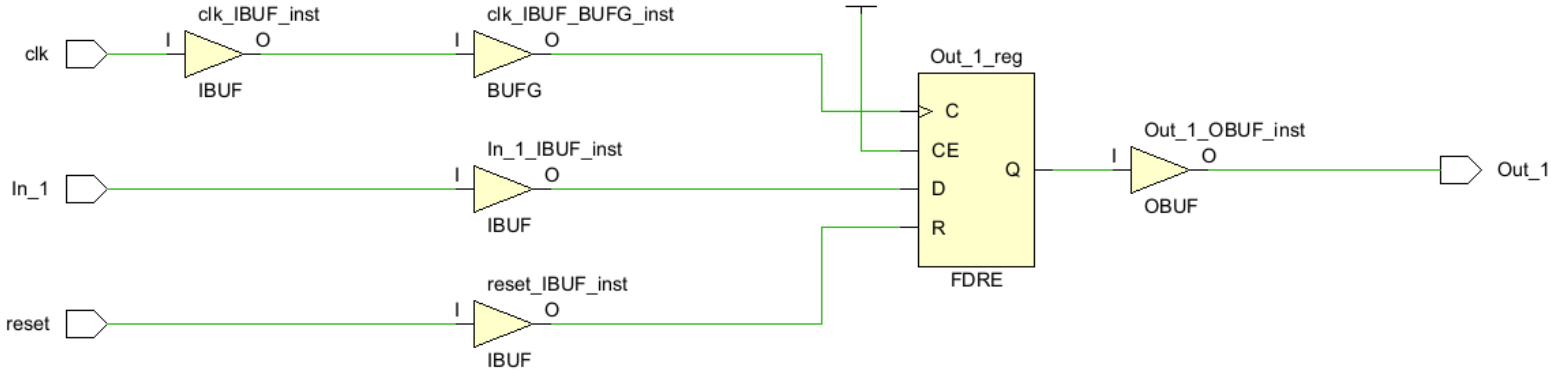
Παράδειγμα



VHDL – Variables

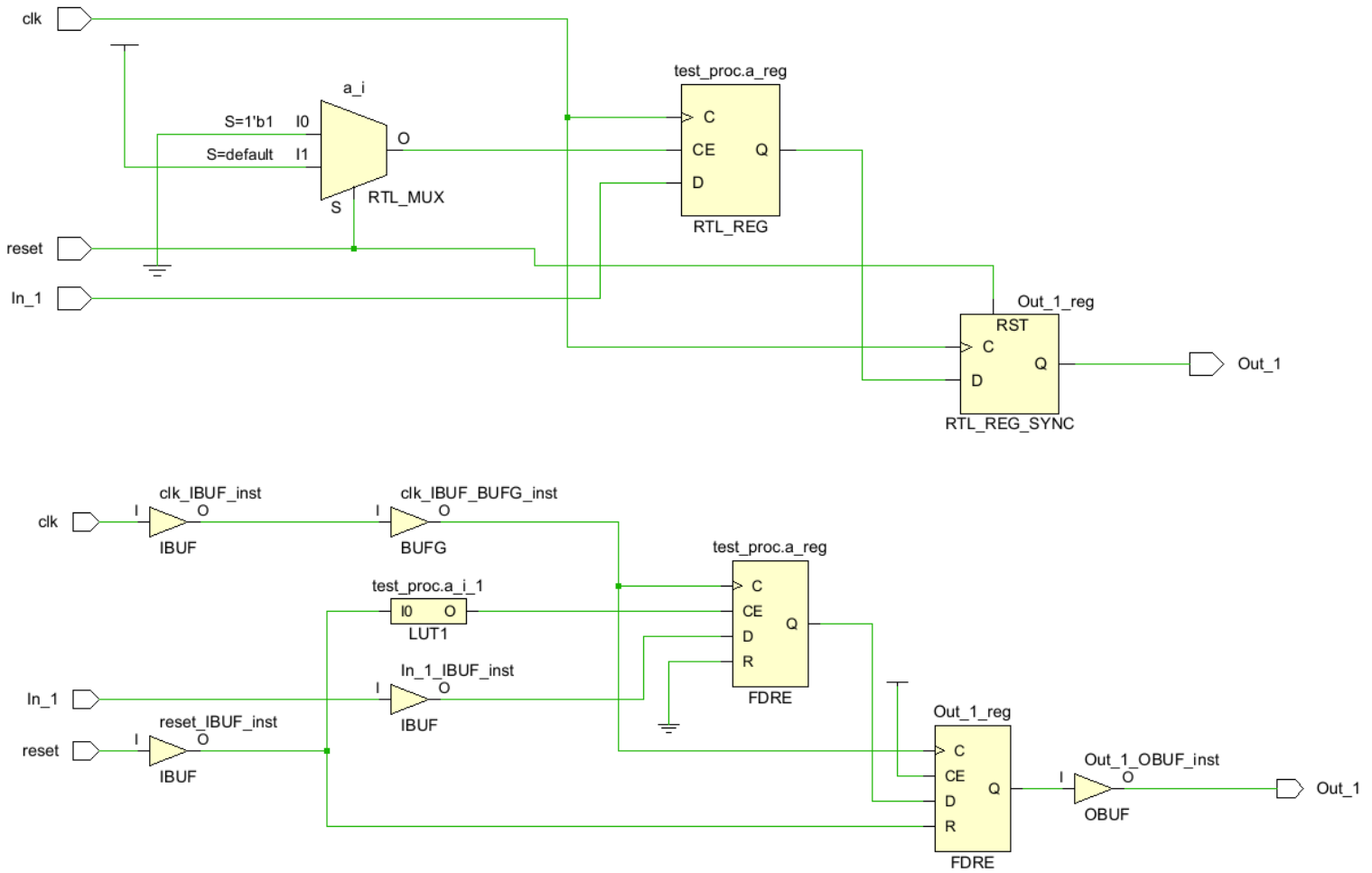
```
test_proc: process (clk) is

variable a: std_logic;

begin

if rising_edge(clk) then
   if reset='1' then
      Out_1<='0';
   else
      a:=In_1;
      Out_1<=a;
   end if;
end if;

end process test_proc;
```





VHDL – Variables

```
test_proc: process (clk) is

variable a: std_logic;

begin


if rising_edge(clk) then
   if reset='1' then
      Out_1<='0';
   else
      Out_1<=a;
      a:=In_1;
   end if;
end if;

end process test_proc;
```



VHDL – Assert

Ακολουθιακή και σύγχρονη εντολή (συνήθως εντός simulation)

Ελέγχει εάν μια συνθήκη είναι αληθής. Εάν ΔΕΝ είναι αληθής τότε τυπώνει ένα μήνυμα.

```
assert <condition>;
assert <condition> severity <severity_level>;
assert <condition> report <message_string>;
assert <condition> report <message_string> severity <severity_level>;

condition_to_check <= true;
assert condition_to_check = true  report "Assertion failed! Condition is false." severity error;
    -- If the assert passes, this line will be executed
report "Assertion passed! Condition is true." severity note;
```

Severity levels: note, warning, **error**, **failure**

Scalar type

```
assert i < 5 report "unexpected value. i = " & integer'image(i);
```

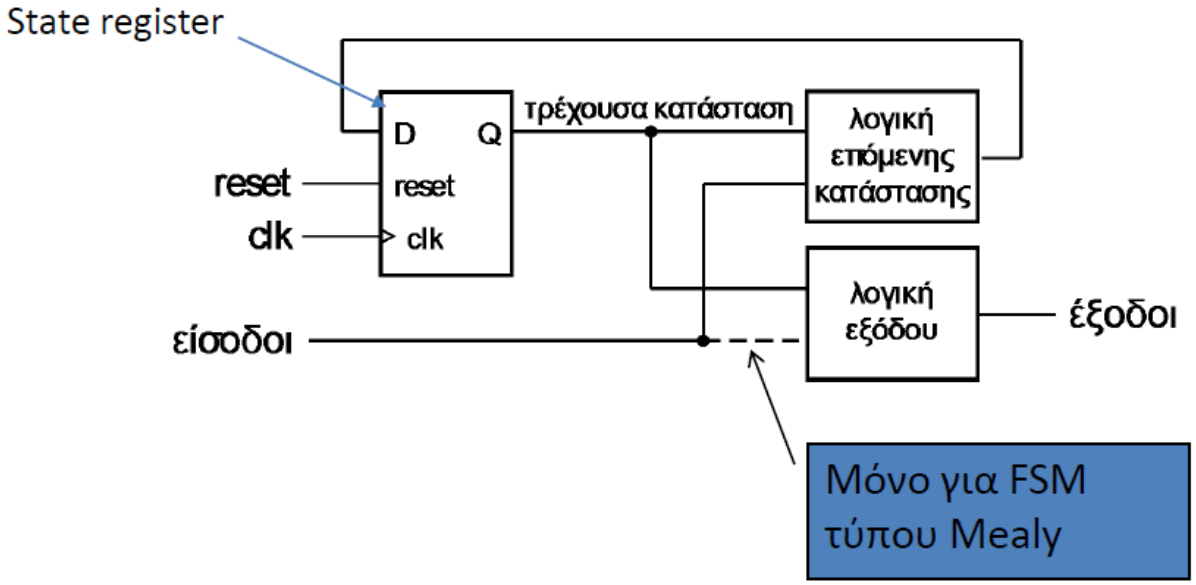
https://peterfab.com/ref/vhdl/vhdl_renerta/mobile/source/vhd00007.htm

VHDL – Finite State Machines

- Χρησιμοποιούνται για να υλοποιήσουν μια ακολουθία ελέγχου
- Μία FSM (Finite-State Machine – Μηχανή Πεπερασμένων καταστάσεων) ορίζεται από
  - σύνολο εισόδων: I
  - σύνολο εξόδων: O
  - σύνολο καταστάσεων: S
  - αρχική κατάσταση: s0 ανήκει στο S
  - συνάρτηση μετάβασης από μία κατάσταση στην επόμενη
  - συνάρτηση εξόδου

VHDL – Finite State Machines

Γενικό διάγραμμα



VHDL – Finite State Machines

Κωδικοποίηση Καταστάσεων

- Κωδικοποιούνται στο δυαδικό
  - Ν καταστάσεις: χρειάζονται τουλάχιστον log2*N* bit
- Η κωδικοποιημένη τιμή χρησιμοποιείται στα κυκλώματα για τις συναρτήσεις μετάβασης και εξόδου
  - η κωδικοποίηση επηρεάζει την πολυπλοκότητα του κυκλώματος
- Είναι δύσκολο να βρεθεί βέλτιστη κωδικοποίηση
  - τα εργαλεία CAD συνήθως κάνουν την κωδικοποίηση
- Η κωδικοποίηση one-hot δουλεύει καλά στα FPGA
- Συχνά χρησιμοποιείται το 000...0 για την κατάσταση αδράνειας
  - μηδενίζει τον καταχωρητή στην κατάσταση αδράνειας

VHDL – Finite State Machines

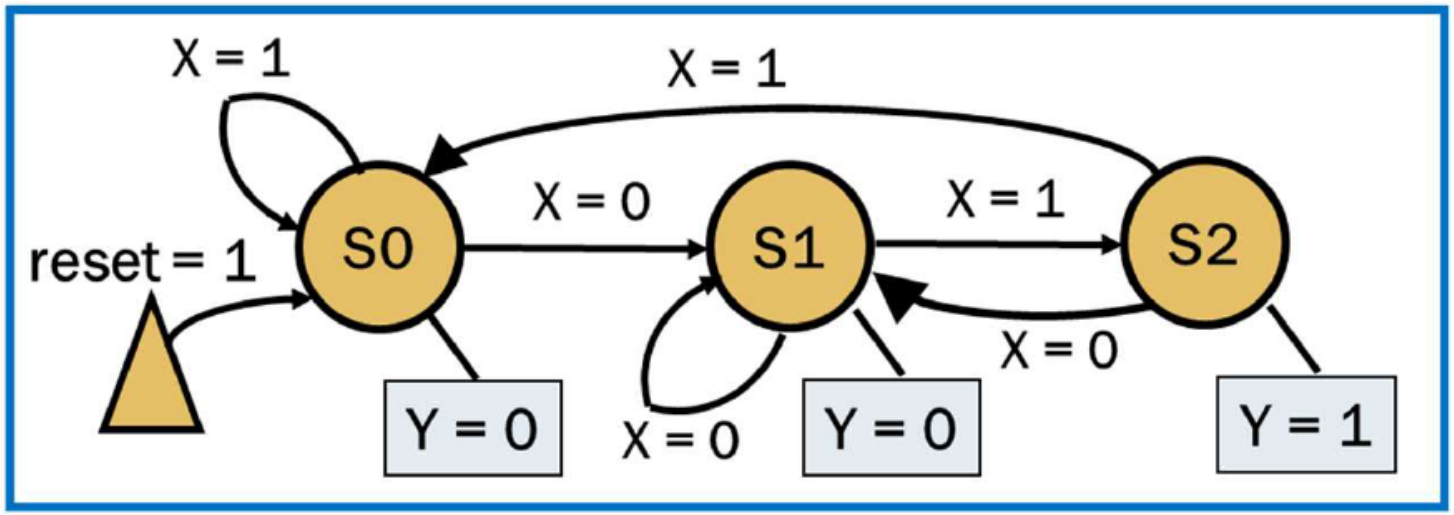
Κωδικοποίηση Καταστάσεων - Type

- Χρησιμοποιούμε έναν τύπο απαρίθμησης για τις τιμές των καταστάσεων
  - αφηρημένο, έτσι αποφεύγουμε να προδιαγράψουμε την κωδικοποίηση

```
type state_type is (zero, edge, one);
signal state_reg, state_next: state_type;
...
```

VHDL – Finite State Machines

Παράδειγμα – Ανίχνευση ακολουθίας 01 - Moore



VHDL – Finite State Machines

Παράδειγμα – Ανίχνευση ακολουθίας 01 - Moore

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity PATTERN_FSM is
   Port ( CLK : in STD_LOGIC;
          RESET : in STD_LOGIC;
          X : in STD_LOGIC;
          Y : out STD_LOGIC);
end PATTERN_FSM;

architecture Behavioral of PATTERN_FSM is

-- state definition
   type FSM_states is (S0, S1, S2);

-- internal signals
   signal current_state, next_state: FSM_states;
   signal X_in : STD_LOGIC; -- Only when there is an INREG
```

ΛΟΓΙΚΗ για RESET

```
begin
-- Optional for synchronization. RESET case
   INREG:  process (CLK)
   begin
      if (CLK = '1' and CLK'event) then
         if (RESET = '1') then X_in <= '1'; -- to trap state S0 during reset
         else X_in <= X;
         end if;
      end if;
   end process;

-- Common process for all FSMs to create state register
   SYNC:  process (CLK)
   begin
      if (CLK = '1' and CLK'event) then
         if (RESET = '1') then current_state <= S0;
         else current_state <= next_state;
         end if;
      end if;
   end process;
```

VHDL – Finite State Machines

Παράδειγμα – Ανίχνευση ακολουθίας 01 - Moore

```
-- Process to create next state logic and output logic
   ASYNC: process (current_state, X_in)  -- Moore
   begin
   -- FSM next state and output initialization
      next_state <= S0;
      Y <= '0';
      case current_state is
         when S0 =>
            if (X_in = '0') then next_state <= S1;
            else next_state <= S0;
            end if;
         when S1 =>
            if (X_in = '1') then next_state <= S2;
            else next_state <= S1;
            end if;
         when S2 => Y <= '1';
            if (X_in = '0') then next_state <= S1;
            else next_state <= S0;
            end if;
         -- fail-safe behavior
         when others => next_state <= S0;
      end case;
   end process;

end Behavioral;
```

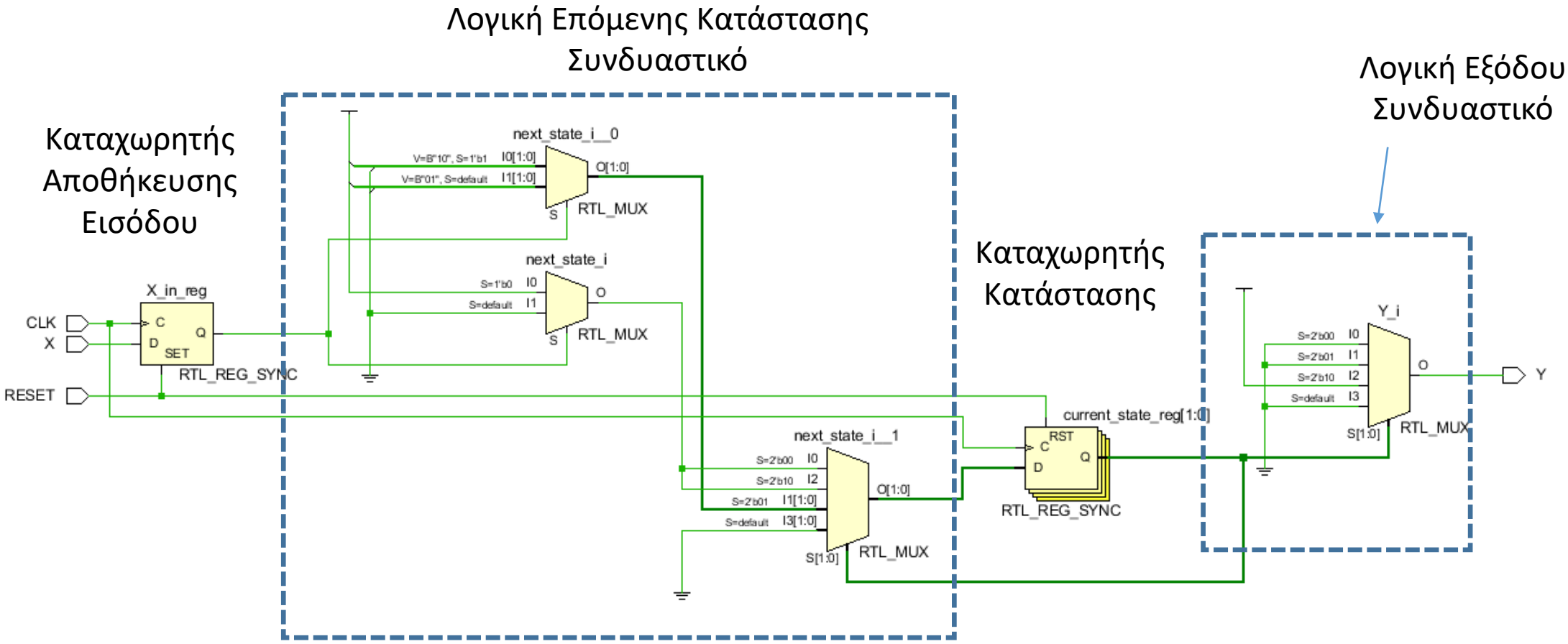
Αρχική τιμή οι τιμές Reset

ΛΟΓΙΚΗ για ΕΠΟΜΕΝΗ ΚΑΤΑΣΤΑΣΗ

ΛΟΓΙΚΗ για ΕΞΟΔΟ

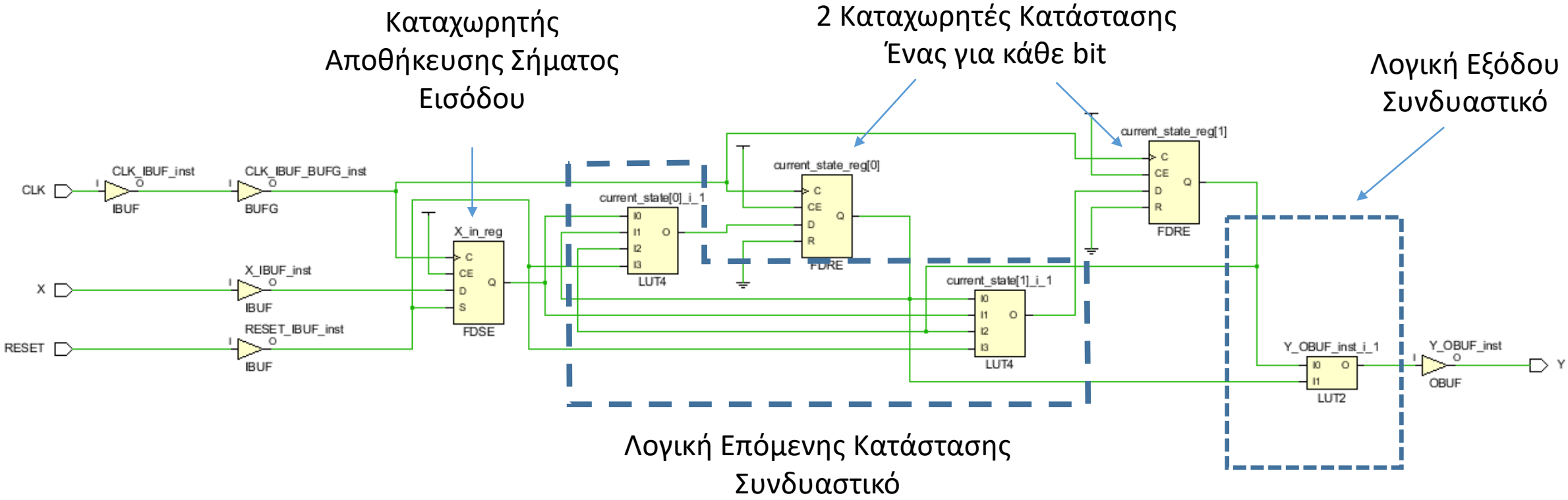
VHDL – Finite State Machines

Παράδειγμα – Ανίχνευση ακολουθίας 01 – Moore - RTL



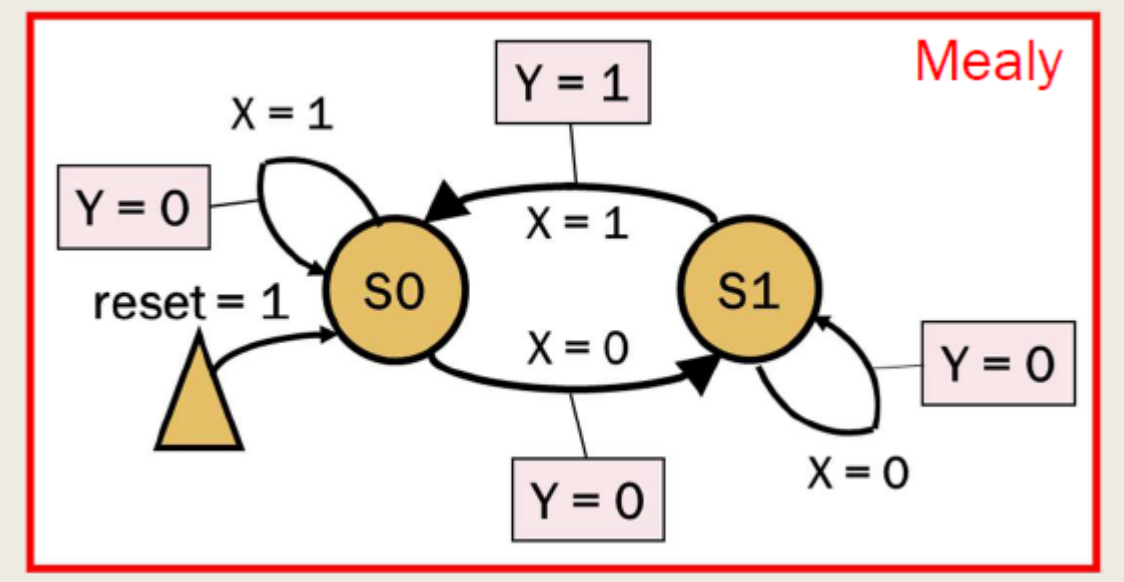
VHDL – Finite State Machines

Παράδειγμα – Ανίχνευση ακολουθίας 01 – Moore - Synthesis



VHDL – Finite State Machines

Παράδειγμα – Ανίχνευση ακολουθίας 01 - Mealy



VHDL – Finite State Machines

Παράδειγμα – Ανίχνευση ακολουθίας 01 - Moore

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity PATTERN_FSM is
   Port ( CLK : in STD_LOGIC;
          RESET : in STD_LOGIC;
          X : in STD_LOGIC;
          Y : out STD_LOGIC);
end PATTERN_FSM;

architecture Behavioral of PATTERN_FSM is

-- state definition
   type FSM_states is (S0, S1);

-- internal signals
   signal current_state, next_state: FSM_states;
   signal X_in : STD_LOGIC; -- Only when there is an INREG
```

```
begin
-- Optional for sychronization
   INREG:  process (CLK)
   begin
      if (CLK = '1' and CLK'event) then
         if (RESET = '1') then X_in <= '1'; -- to trap state S0 during reset
         else X_in <= X;
         end if;
      end if;
   end process;

-- Common process for all FSMs to create state register
   SYNC:  process (CLK)
   begin
      if (CLK = '1' and CLK'event) then
         if (RESET = '1') then current_state <= S0;
         else current_state <= next_state;
         end if;
      end if;
   end process;
```

VHDL – Finite State Machines

Παράδειγμα – Ανίχνευση ακολουθίας 01 - Mealy

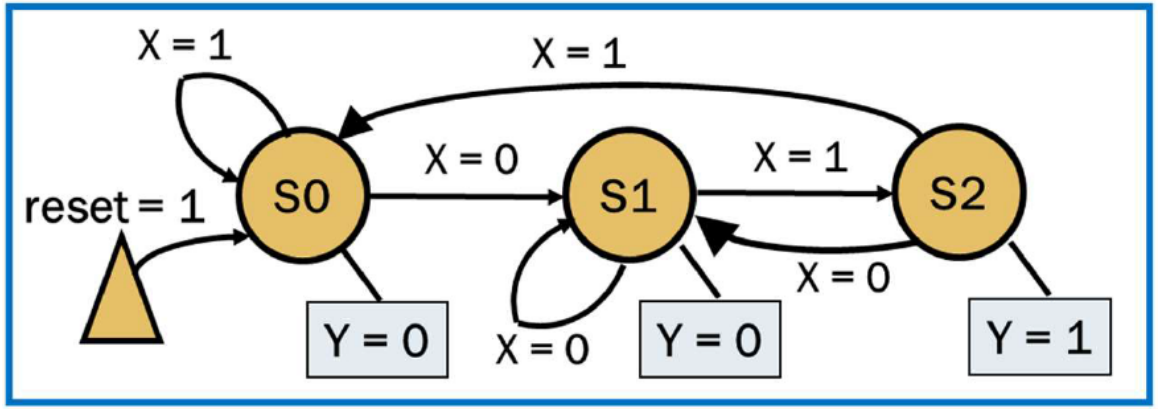
```
-- Process to create next state logic and output logic
   ASYNC: process (current_state, X_in)  -- Moore
   begin
   -- FSM next state and output initialization
      next_state <= S0;
      Y <= '0';
      case current_state is
         when S0 =>
            if (X_in = '0') then next_state <= S1;
            else next_state <= S0;
            end if;
         when S1 =>
            if (X_in = '1') then
                  next_state <= S0;
                  Y<='1';
            else next_state <= S1;
            end if;
-- fail-safe behavior
            when others => next_state <= S0;
      end case;
   end process;
end Behavioral;
```

ΛΟΓΙΚΗ για ΕΞΟΔΟ

VHDL – Finite State Machines

Παράδειγμα – Ανίχνευση ακολουθίας 01 – Moore – Πίνακας Αληθείας

S_{old}	$S_{old}(1)$	$S_{old}(0)$	Xin	$S_{new}(1)$	$S_{new}(0)$	S_{new}	Y
S_0	0	0	0	0	1	S_1	0
S_1	0	1	0	0	1	S_1	0
S_2	1	0	0	0	1	S_1	0
S_3	1	1	0	0	0	S_0	0
S_0	0	0	1	0	0	S_0	0
S_1	0	1	1	1	0	S_2	1
S_2	1	0	1	0	0	S_0	0
S_3	1	1	1	0	0	S_0	0



VHDL – Finite State Machines

Παράδειγμα – Ανίχνευση ακολουθίας 01 – Moore – Πίνακες Αληθείας

Πίνακες KARNAUGH

$S_{new}(1)$

S_{old}\Xin	0	1
00	0	0
01	0	1
11	0	0
10	0	0

$S_{new}(0)$

S_{old}\Xin	0	1
00	1	0
01	1	0
11	0	0
10	1	0

Y

S_{old}\Xin	0	1
00	0	0
01	0	(1)
11	0	0
10	0	0

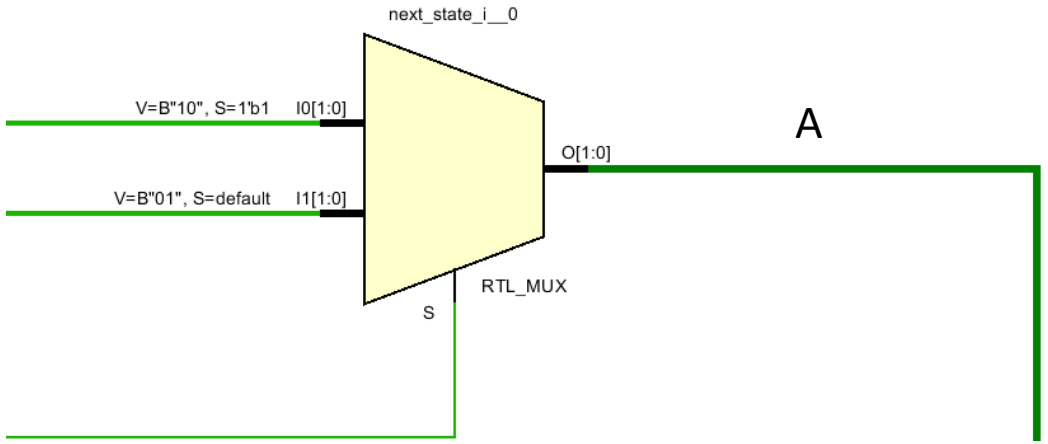
$S_{new}(1) = S_{old}(1)' S_{old}(0)Xin$
$S_{new}(0) = S_{old}(1)'Xin' + S_{old}(1) S_{old}(0)'Xin'$
$Y = S_{old}(1)S_{old}(0)' Xin$

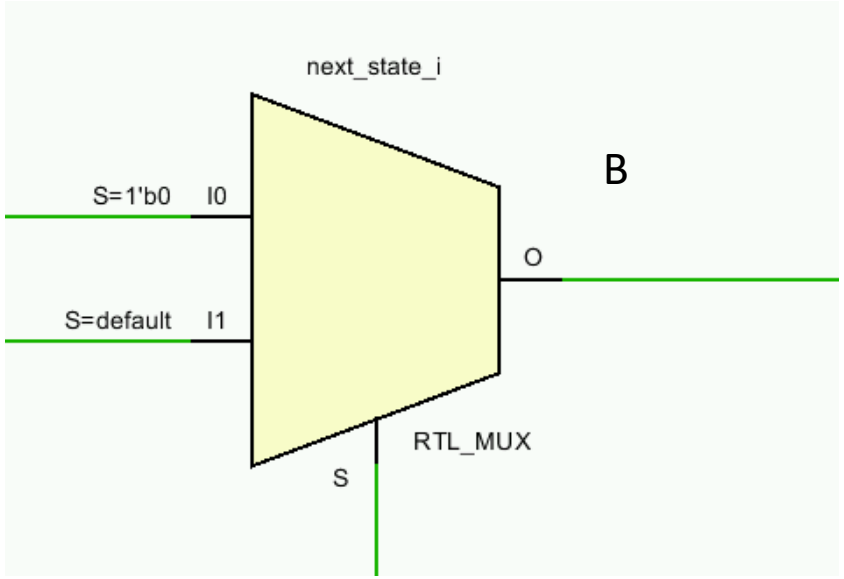
VHDL – Finite State Machines

Παράδειγμα – Ανίχνευση ακολουθίας 01 – Moore – Πίνακες Αληθείας

X_{in}=S	Out	$S1_{new}$
1	10	S_2
0	01	S_1

X_{in}=S	Out	$S1_{new}$
0	1	S_1
1	0	S_0





VHDL – Finite State Machines

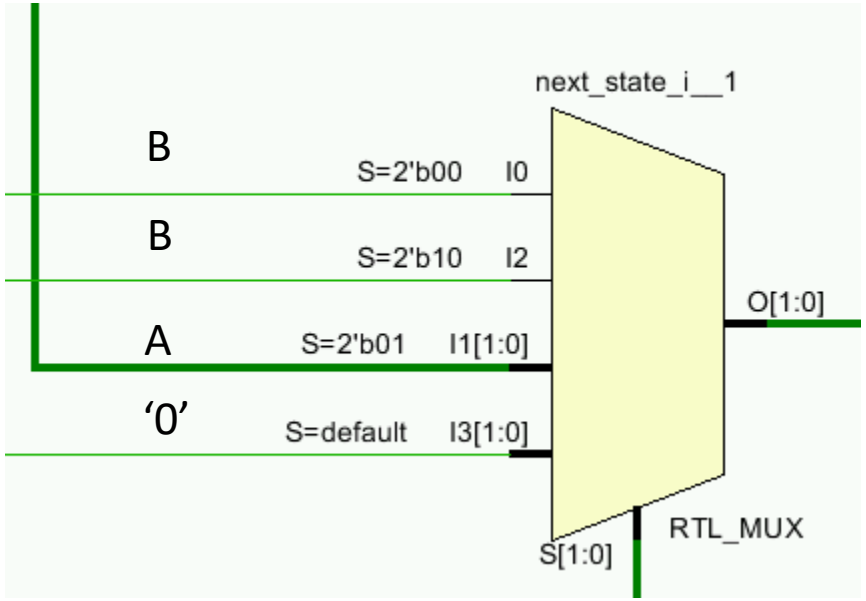
Παράδειγμα – Ανίχνευση ακολουθίας 01 – Moore – Πίνακας Αληθείας

A

X_{in}=S	Out	S_{new}
1	10	S_2
0	01	S_1

B

X_{in}=S	Out	S_{new}
0	01	S_1
1	00	S_0



S_{old}	X_{in}	S_{new}
00(0)	0	S_1
01(1)	0	S_1
10(2)	0	S_1
11(X)	0	S_0
00(0)	1	S_0
01(1)	1	S_2
10(2)	1	S_0
11(X)	1	S_0

Διακόπτες και αποκλυδωνισμός

Οι διακόπτες (switches) και οι πιεστικοί διακόπτες (push-buttons) υποφέρουν από κλυδωνισμό επαφής
– Χρειάζονται περίπου 10ms να σταθεροποιηθούν
– Χρειάζεται να αποκλυδωνίσουμε (debounce) για να αποφύγουμε τις ψεύτικες ενεργοποιήσεις
• Χρησιμοποιούμε ένα διακόπτη μίας θέσης
• Κάνουμε δειγματοληψία στην είσοδο σε διαστήματα μεγαλύτερα από το χρόνο κλυδωνισμού (>10ms)
Στην κάρτα του εργαστηρίου σημαίνει 100000 χτύπους του ρολογιού.
• Αναζητούμε δύο διαδοχικά δείγματα με την ίδια τιμή

VHDL - Αποκλυδωνισμός

Παράδειγμα (1/2)

```
library ieee; use ieee.std_logic_1164.all;
entity debouncer is
port ( clk, reset : in std_logic; -- clk frequency = 100MHz
       pb : in std_logic;
       pb_debounced : out std_logic );
end entity debouncer;

architecture rtl of debouncer is

signal count1000000 : integer range 0 to 999999;
signal clk_100Hz : std_logic;    --10ms
signal pb_sampled : std_logic;

begin
```

VHDL - Αποκλυδωνισμός

Παράδειγμα (2/2)

```
div_100Hz : process (clk, reset) is
begin
if reset = '1' then
        clk_100Hz <= '0';
        count500000 <= 0;
elsif rising_edge(clk) then
        if count1000000 = 999999 then
             count1000000 <= 0;
             clk_100Hz <= '1';
        else
             count1000000 <= count1000000 + 1;
             clk_100Hz <= '0';
        end if;
end if;
end process div_100Hz;
```

```
debounce_pb : process (clk) is
begin
if rising_edge(clk) then
        if clk_100Hz = '1' then
                if pb = pb_sampled then
                        pb_debounced <= pb;
                end if;
                pb_sampled <= pb;
        end if;
end if;

end process debounce_pb;

end architecture rtl;
```

Περίληψη

Harris: Παράγραφοι 3.4, 3.5 και 4.6
Ashenden: Παράγραφοι 4.3 και 4.4

Επίσης οι σύνδεσμοι:
https://stackoverflow.com/questions/21351273/is-it-possible-to-synthesize-vhdl-code-with-variable-in-it
Assert και Report: https://insights.sigasi.com/tech/vhdl-assert-and-report/