Computer Architecture Lab Session 3

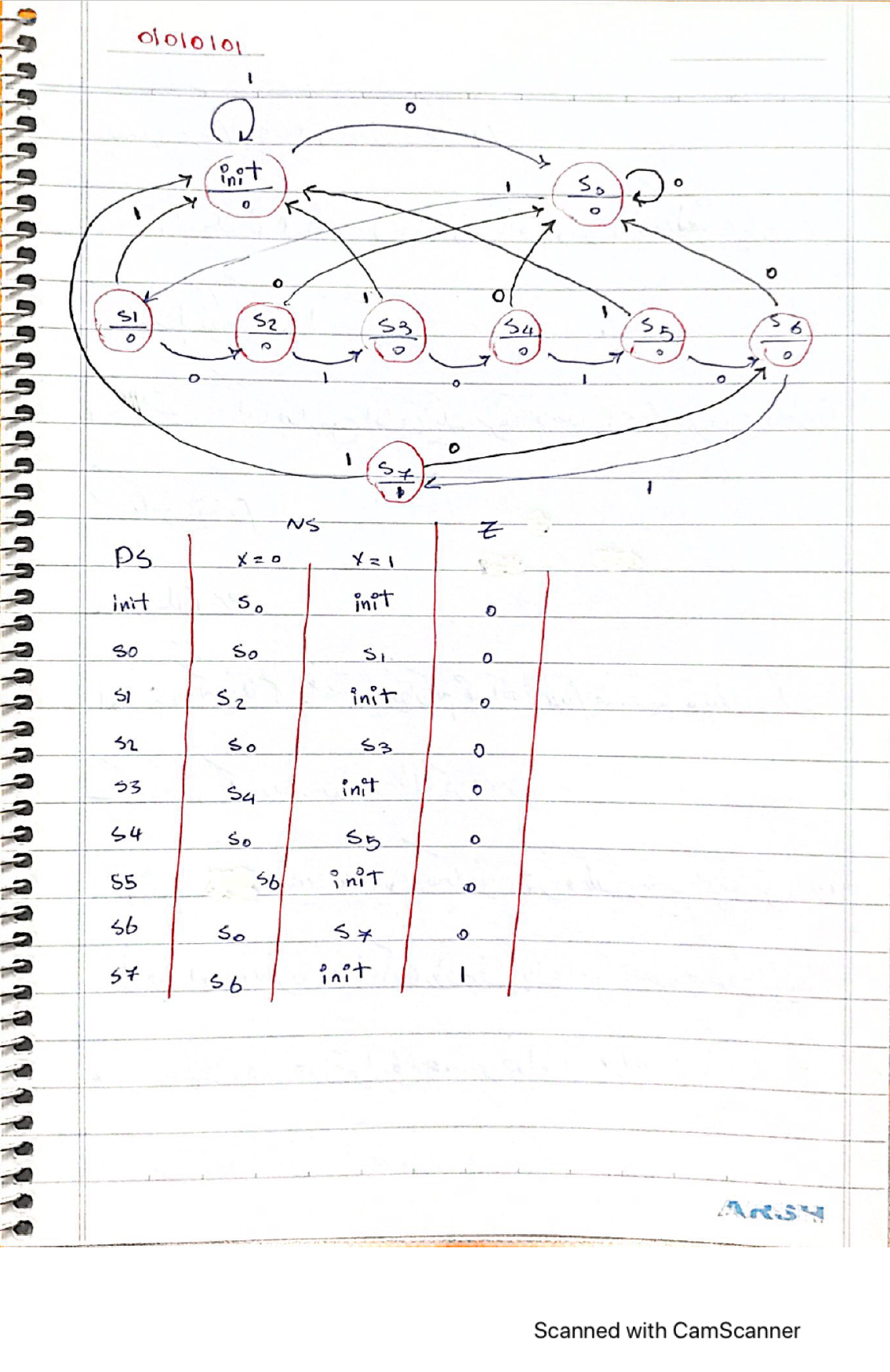
# 01010101 Sequence Detector

# Bardia Ardakanian 9831072

# Ali Asad 9831004

**طراحی یک Sequence Detector با ماشین مور**

برای طراحی یک ماشین مور state‌ها را می‌نویسیم و خروجی‌های هر استیت را هم مشخص می‌کنیم، در نهایت با توجه به ورودی ارتباط بین stateها را مشخص می‌کنیم.



**پیاده سازی یک Sequence Detector با زبان vhdl**

برای نوشتن یک Sequence Detector ، ابتدا موجودیت Detector را تعریف میکنیم و سپس پورت های

ورودی خروجی را تعریف م یکنیم، در نهایت در ساختمان رفتاری Detector را تعریف م یکنیم.

|  |
| --- |
| -- VHDL project: VHDL code for Sequence Detector using Moore FSM -- The sequence being detected is "01010101" LIBRARY IEEE; USE IEEE.STD\_LOGIC\_1164.ALL;  ENTITY VHDL\_MOORE\_FSM\_Sequence\_Detector IS     PORT (         clock : IN STD\_LOGIC; --- clock signal         reset : IN STD\_LOGIC; -- reset input         sequence\_in : IN STD\_LOGIC; -- binary sequence input         detector\_out : OUT STD\_LOGIC -- output of the VHDL sequence detector     ); END VHDL\_MOORE\_FSM\_Sequence\_Detector;  ARCHITECTURE Behavioral OF VHDL\_MOORE\_FSM\_Sequence\_Detector IS     TYPE MOORE\_FSM IS (Init, S0, S1, S2, S3, S4, S5, S6, S7);     SIGNAL current\_state, next\_state : MOORE\_FSM;  BEGIN     -- Sequential memory of the VHDL MOORE FSM Sequence Detector     PROCESS (clock, reset)     BEGIN         IF (reset = '1') THEN             current\_state <= Init;         ELSIF (rising\_edge(clock)) THEN             current\_state <= next\_state;         END IF;     END PROCESS;      -- Next state logic of the VHDL MOORE FSM Sequence Detector     -- Combinational logic     PROCESS (current\_state, sequence\_in)     BEGIN         CASE(current\_state) IS             WHEN Init =>                 IF (sequence\_in = '0') THEN                     -- "0"                     next\_state <= S0;                 END IF;             WHEN S0 =>                 IF (sequence\_in = '1') THEN                     -- "01"                     next\_state <= S1;                 END IF;             WHEN S1 =>                 IF (sequence\_in = '0') THEN                     -- "010"                     next\_state <= S2;                 ELSE                     next\_state <= Init;                 END IF;             WHEN S2 =>                 IF (sequence\_in = '1') THEN                     -- "0101"                     next\_state <= S3;                 ELSE                     -- "0"                     next\_state <= S0;                 END IF;             WHEN S3 =>                 IF (sequence\_in = '0') THEN                     -- "01010"                     next\_state <= S4;                 ELSE                     next\_state <= Init;                 END IF;             WHEN S4 =>                 IF (sequence\_in = '1') THEN                     -- "010101"                     next\_state <= S5;                 ELSE                     -- "0"                     next\_state <= S0;                 END IF;             WHEN S5 =>                 IF (sequence\_in = '0') THEN                     -- "0101010"                     next\_state <= S6;                 ELSE                     next\_state <= Init;                 END IF;             WHEN S6 =>                 IF (sequence\_in = '1') THEN                     -- "01010101"                     next\_state <= S7;                 ELSE                     next\_state <= S0;                 END IF;             WHEN S7 =>                 IF (sequence\_in = '0') THEN                     -- "0101010"                     next\_state <= S6;                 ELSE                     next\_state <= Init;                 END IF;         END CASE;     END PROCESS;      -- Output logic of the VHDL MOORE FSM Sequence Detector     PROCESS (current\_state)     BEGIN         CASE current\_state IS             WHEN S7 =>                 detector\_out <= '1';             WHEN OTHERS =>                 detector\_out <= '0';         END CASE;     END PROCESS; END Behavioral; |

**پیاده سازی test bench**

|  |
| --- |
| -- VHDL project: VHDL code for Sequence Detector using Moore FSM -- VHDL testbench for Moore FSM Sequence Detector LIBRARY ieee; USE ieee.std\_logic\_1164.ALL;  ENTITY tb\_VHDL\_Moore\_FSM\_Sequence\_Detector IS END tb\_VHDL\_Moore\_FSM\_Sequence\_Detector;  ARCHITECTURE behavior OF tb\_VHDL\_Moore\_FSM\_Sequence\_Detector IS      -- Component Declaration for the Moore FSM Sequence Detector in VHDL      COMPONENT VHDL\_MOORE\_FSM\_Sequence\_Detector         PORT (             clock : IN STD\_LOGIC;             reset : IN STD\_LOGIC;             sequence\_in : IN STD\_LOGIC;             detector\_out : OUT STD\_LOGIC         );     END COMPONENT;     --Inputs     SIGNAL clock : STD\_LOGIC := '0';     SIGNAL reset : STD\_LOGIC := '0';     SIGNAL sequence\_in : STD\_LOGIC := '0';      --Outputs     SIGNAL detector\_out : STD\_LOGIC;      -- Clock period definitions     CONSTANT clock\_period : TIME := 10 ns;  BEGIN      -- Instantiate the Moore FSM Sequence Detector in VHDL     uut : VHDL\_MOORE\_FSM\_Sequence\_Detector PORT MAP(         clock => clock,         reset => reset,         sequence\_in => sequence\_in,         detector\_out => detector\_out     );      -- Clock process definitions     clock\_process : PROCESS     BEGIN         clock <= '0';         WAIT FOR clock\_period/2;         clock <= '1';         WAIT FOR clock\_period/2;     END PROCESS;     -- Stimulus process     stim\_proc : PROCESS     BEGIN         -- hold reset state for 100 ns.         sequence\_in <= '0';         reset <= '1';         -- Wait 100 ns for global reset to finish         WAIT FOR 30 ns;         reset <= '0';         WAIT FOR 40 ns;         sequence\_in <= '0';         WAIT FOR 10 ns;         sequence\_in <= '1';         WAIT FOR 10 ns;         sequence\_in <= '0';         WAIT FOR 20 ns;         sequence\_in <= '1';         WAIT FOR 20 ns;         sequence\_in <= '0';         WAIT FOR 20 ns;         sequence\_in <= '1';         WAIT FOR 20 ns;         sequence\_in <= '0';         WAIT FOR 20 ns;         sequence\_in <= '1';         WAIT FOR 20 ns;         sequence\_in <= '0';         WAIT FOR 20 ns;         sequence\_in <= '1';         -- insert stimulus here          WAIT;     END PROCESS;  END; |

**Simulation**

تست بنچ نوشته شده را simulate می‌کنیم تا موج های خروجی نمایش داده شود.

