جامعة حلب في المناطق المحررة
كلية الهندسة المعلوماتية
السنة الرابعة

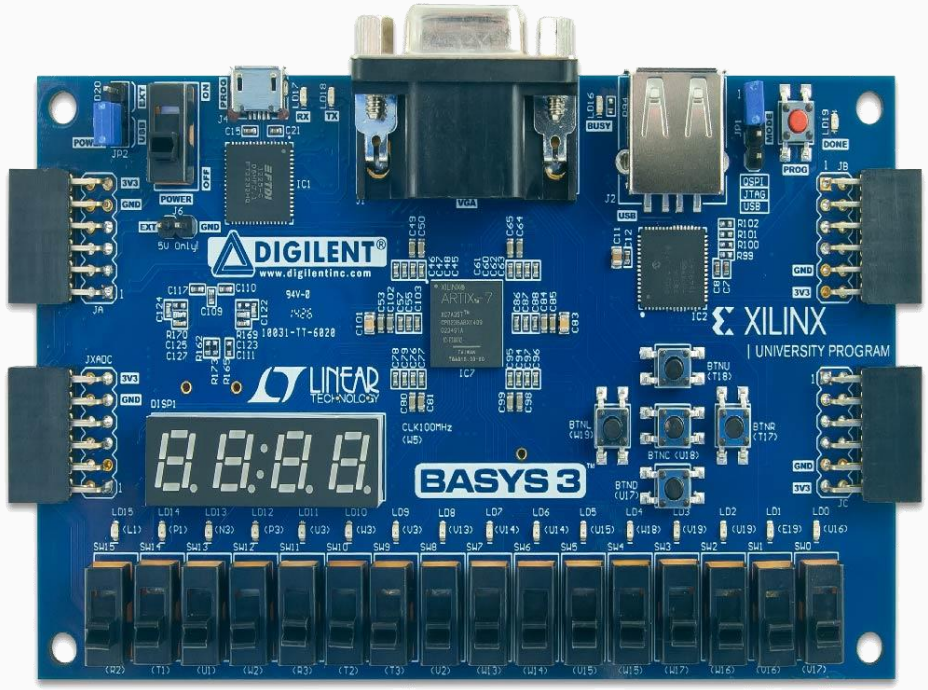
مقرر عملي

بنية وتنظيم الحواسيب2

النواخب والعدادات

د.م. عبد القادر غزال
م. محمد نور بدوي

العام الدراسي: 2023 - 2024



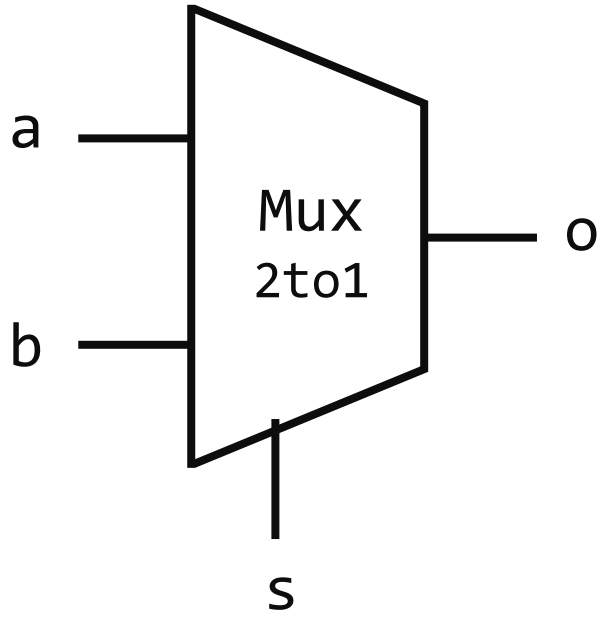
المحاضرة العملية السابعة

النواخب Multiplexer

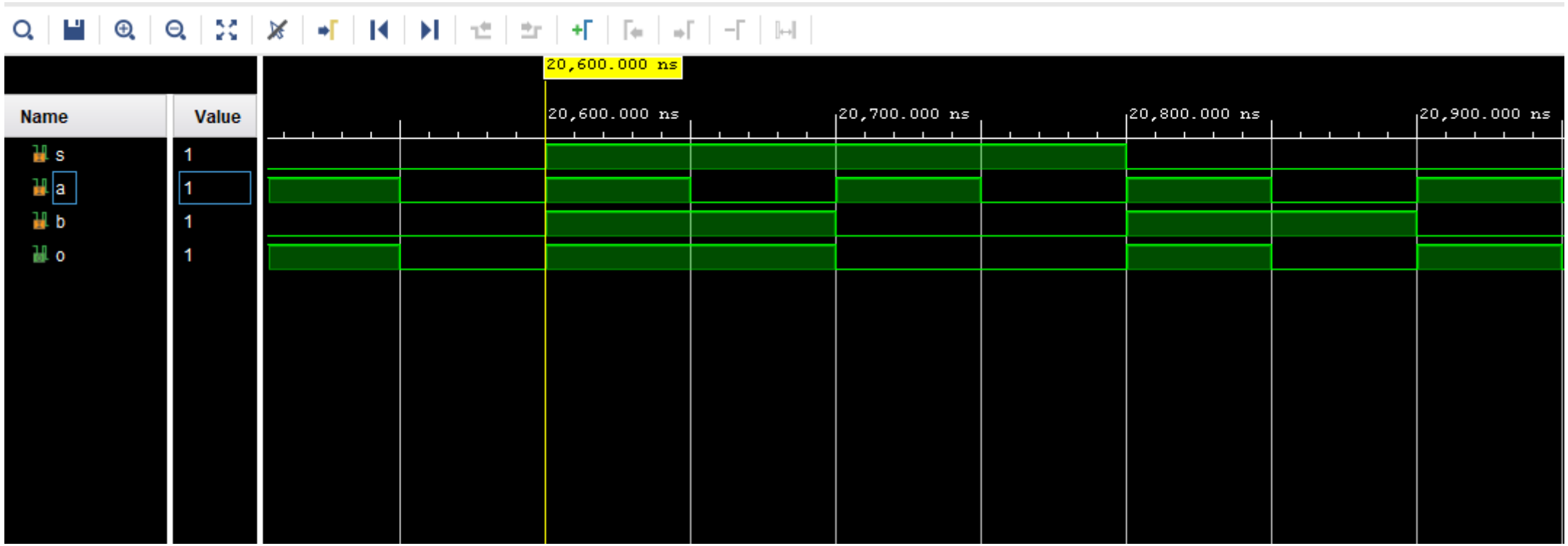
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY mux2to1 IS
    PORT (
        s, a, b : IN STD_LOGIC;
        o : OUT STD_LOGIC);
END mux2to1;
ARCHITECTURE Behavior OF mux2to1 IS
BEGIN
    Mux : PROCESS (s, a, b)
    BEGIN
        IF s = '0' THEN
            o <= a;
        ELSE
            o <= b;
        END IF;
    END PROCESS Mux;
END Behavior;
```

طريقة أخرى

```
WITH s SELECT
    o <= a WHEN '0',
         b WHEN OTHERS;
```

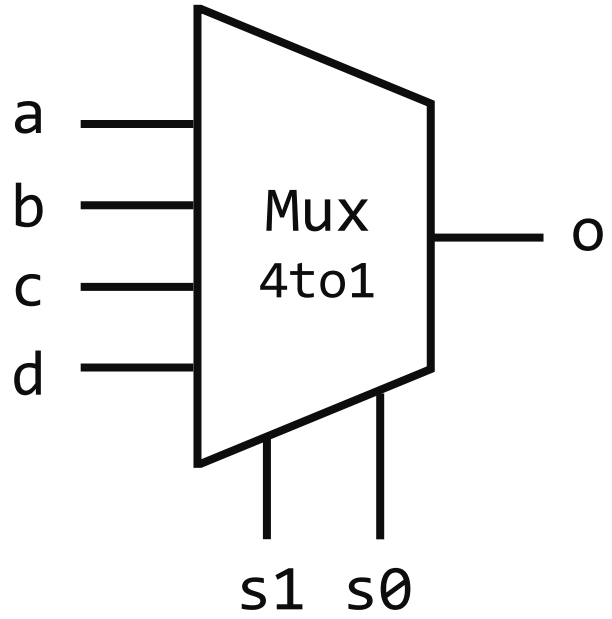


محاكاة التطبيق



النواخب Multiplexer

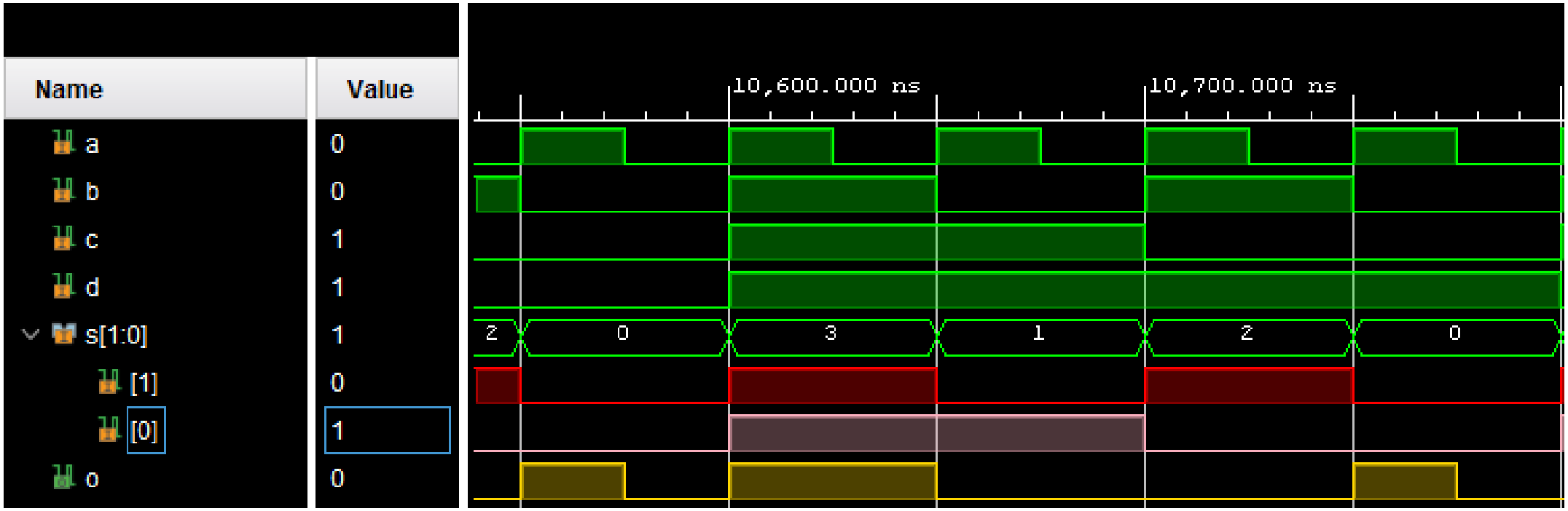
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux4to1 IS
    PORT (
        a, b, c, d : IN STD_LOGIC;
        s : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
        o : OUT STD_LOGIC);
END mux4to1;

ARCHITECTURE Behavior OF mux4to1 IS
BEGIN
    WITH s SELECT
        o <= a WHEN "00",
             b WHEN "01",
             c WHEN "10",
             d WHEN "11";
END Behavior;
```



طريقة أخرى

```
o <= a WHEN s="00" ELSE
     b WHEN s="01" ELSE
     c WHEN s="10" ELSE
     d;
```

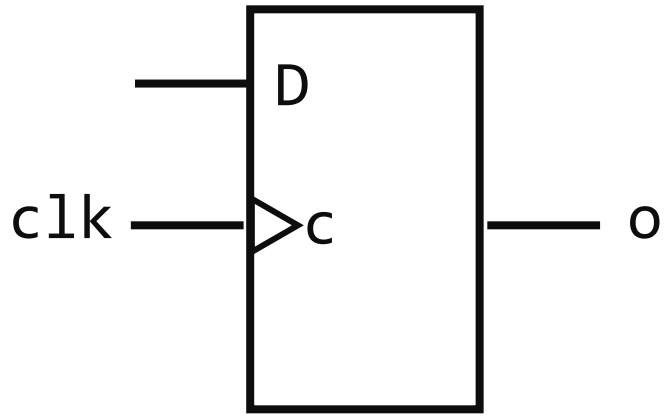
محاكاة التطبيق



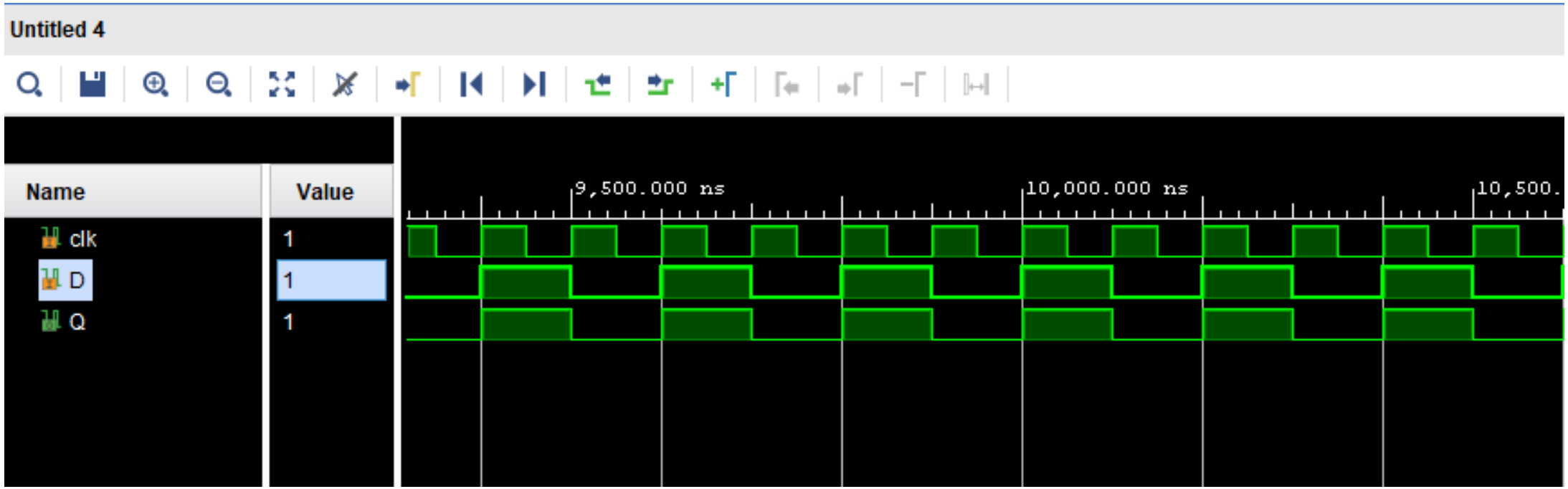
القلابات flip-flop

قلاب D يقدح بالحافة الصاعدة D flip-flop

```
LIBRARY IEEE;
USE IEEE.Std_logic_1164.ALL;
ENTITY D_FF IS
    PORT (
        clk : IN STD_LOGIC;
        D : IN STD_LOGIC;
        Q : OUT STD_LOGIC);
END D_FF;
ARCHITECTURE Behavioral OF D_FF IS
BEGIN
    PROCESS (clk) BEGIN
        IF rising_edge(clk) THEN
         -- falling_edge(CLK)
            Q <= D;
        END IF;
    END PROCESS;
END Behavioral;
```

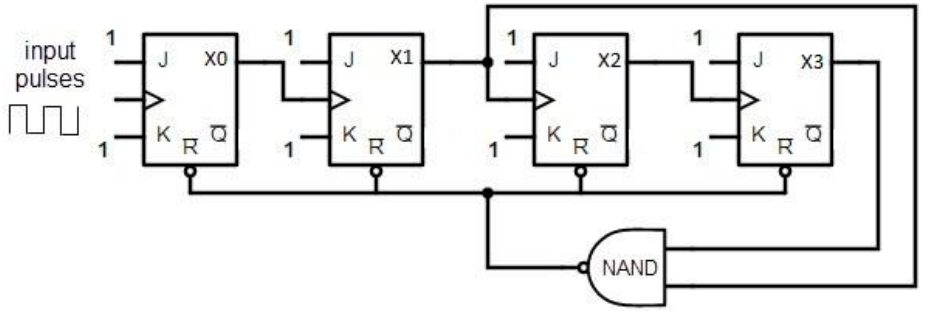


محاكاة التطبيق



العدادات (BCD)

عداد عشري (Binary Coded Decimal) BCD بخانة واحدة، يزداد محتوى العداد بمقدار واحد مع كل حافة صاعدة لنبضات الساعة إلا إذا بلغت قيمة العد 10 عندها ينتقل إلى القيمة 0 .

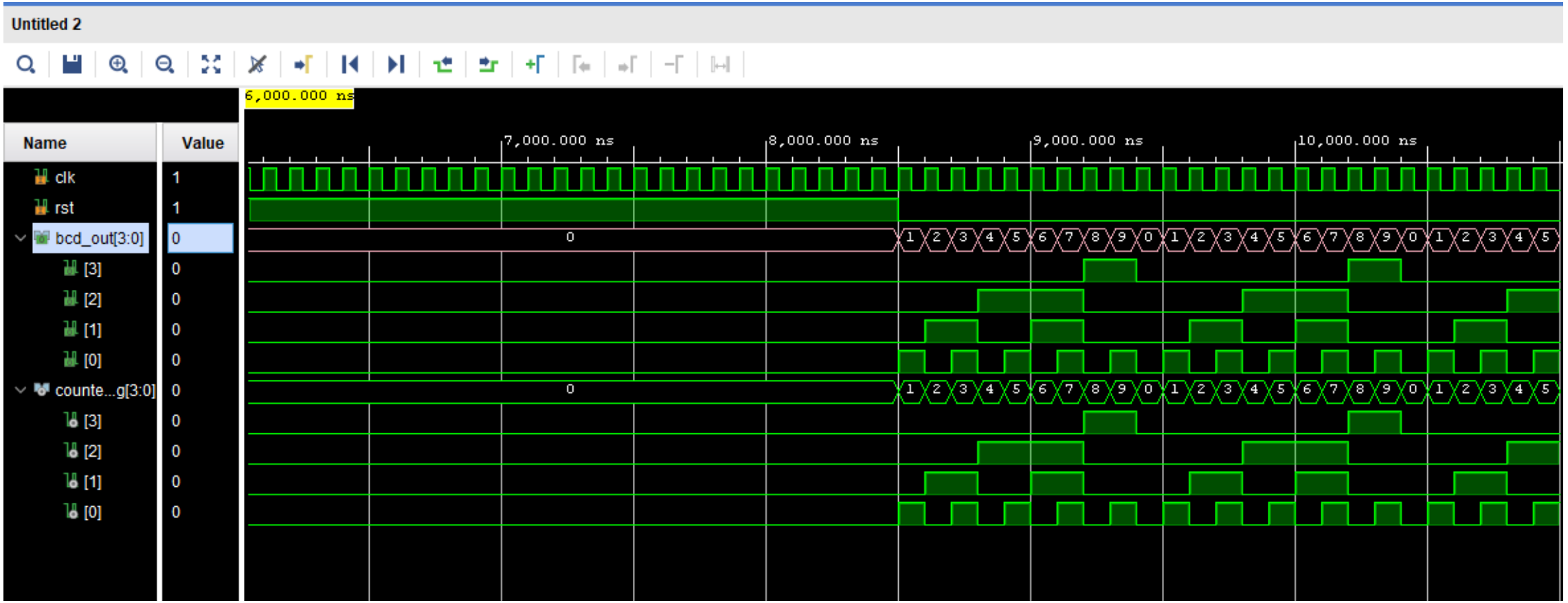



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity BCDCounter is
    Port ( clk : in STD_LOGIC;
           rst : in STD_LOGIC;
           bcd_out : out STD_LOGIC_VECTOR(3 downto 0));
end BCDCounter;

architecture Behavioral of BCDCounter is
    signal counter_reg : STD_LOGIC_VECTOR(3 downto 0) := "0000";
begin
```

العدادات (BCD)

```
    process(clk, rst)
     begin
         if rst = '1' then
             counter_reg <= "0000";
         elsif rising_edge(clk) then
             if counter_reg = "1001" then
                 counter_reg <= "0000";
             else
                 counter_reg <= counter_reg + 1;
             end if;
         end if;
     end process;

     bcd_out <= counter_reg;
end Behavioral;
```

محاكاة التطبيق

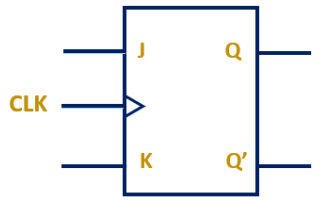


العدادات(jk flip flop)

عداد يقوم بالعد حتى 15 باستخدام قلاب JK بحيث يزداد محتوى العداد بمقدار واحد مع كل حافة صاعدة لنبضات الساعة.

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

entity JKCounter is
    Port ( clk : in STD_LOGIC;
           rst : in STD_LOGIC;
           jk_out : out STD_LOGIC_VECTOR(3 downto 0));
end JKCounter;

architecture Behavioral of JKCounter is
    signal counter_reg : STD_LOGIC_VECTOR(3 downto 0) := "0000";
    signal j, k : STD_LOGIC;
begin
```

Symbol



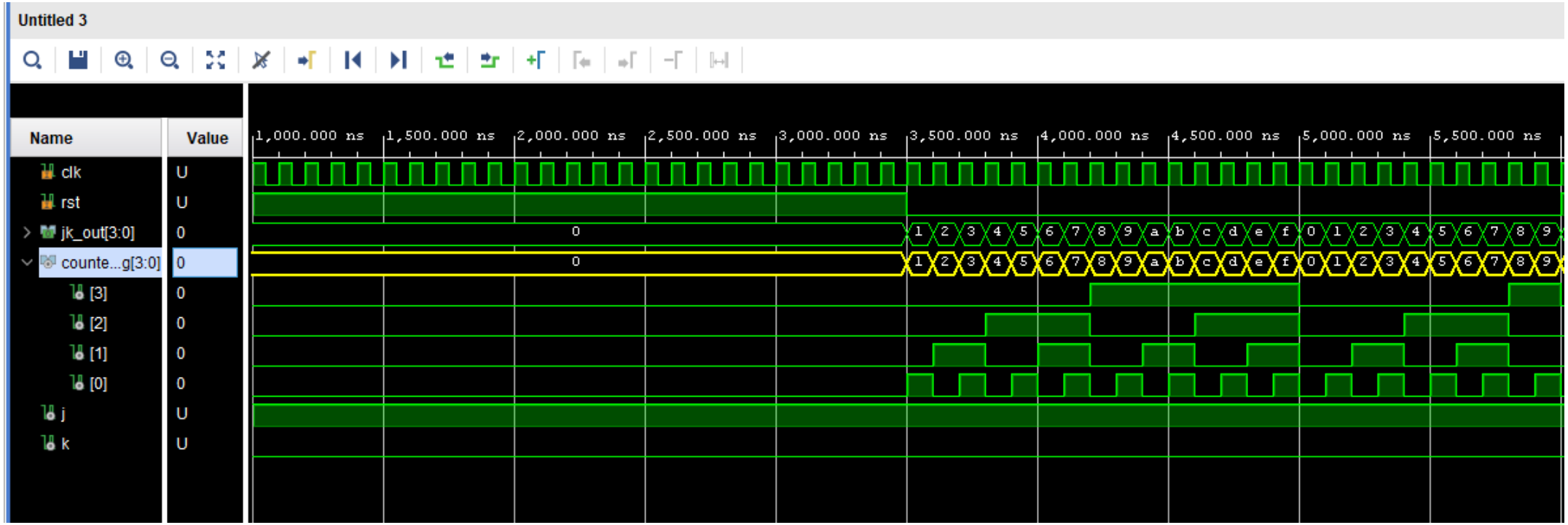
Truth Table

CLK	J	K	Q n+1
↑	0	0	Q n
↑	0	1	0
↑	1	0	1
↑	1	1	Q n'

العدادات(jk flip flop)

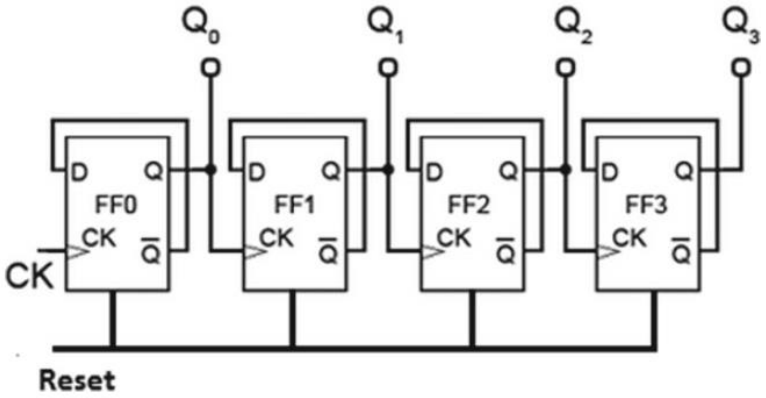
```
    process(clk, rst)
    begin
        if rst = '1' then
            counter_reg <= "0000";
        elsif rising_edge(clk) then
            if counter_reg = "1111" then
                counter_reg <= "0000";
            else
                -- Increment the counter using JK flip-flops
                j <= '1';
                k <= '0';
                counter_reg <= counter_reg + 1;
            end if;
        end if;
    end process;

    jk_out <= counter_reg;
end Behavioral;
```

محاكاة التطبيق



العدادات(D flip flop)

اكتب كود برمجي يقوم بمحاكاة هذه الدارة

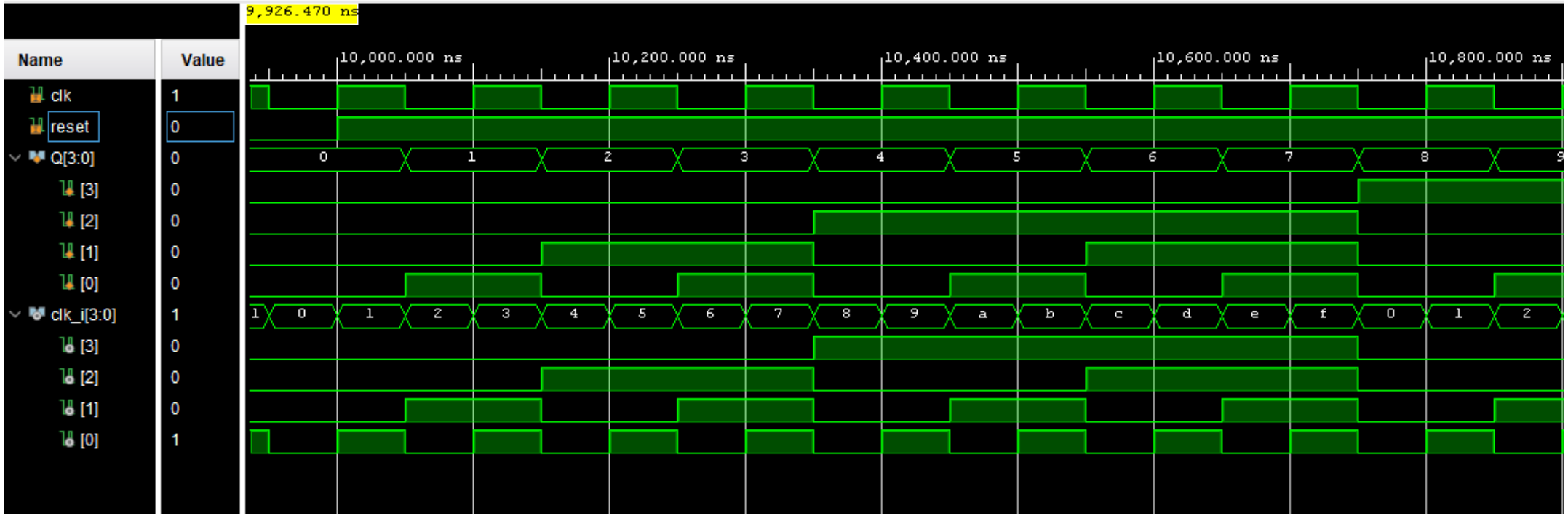



```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;
ENTITY D_CNT IS
    PORT (

        clk : IN STD_LOGIC;
        reset : IN STD_LOGIC;
        Q : BUFFER STD_LOGIC_VECTOR(3 DOWNTO 0)

    );

END D_CNT;

ARCHITECTURE Behavior OF D_CNT IS
    SIGNAL clk_i : STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN
```

العدادات(D flip flop)

```
    -- clock
    clk_i(0) <= clk;
    clk_i(1) <= Q(0);
    clk_i(2) <= Q(1);
    clk_i(3) <= Q(2);

    -- flip_flop

    gen : FOR i IN 0 TO 3 GENERATE
        dff : PROCESS (reset, clk_i)
        BEGIN
            IF (reset = '1') THEN
                Q(i) <= '0';
            ELSIF (clk_i(i)'event AND clk_i(i) = '0' AND reset = '0') THEN
                Q(i) <= NOT Q(i);
            END IF;
        END PROCESS;
    END GENERATE;
END Behavior;
```

محاكاة التطبيق



شرح بعض التعليمات

```
wait until (clk'event and clk = '1');
```

هي تعليمة انتظار تستخدم في الحلقات (loops) أو العمليات (processes) للتأكد من حدوث حدث على إشارة الساعة clk

clk'event هذا هو الجزء الذي يتحقق من حدوث حدث (event) على إشارة الساعة clk

يعني أن التعليمة ستنتظر حتى يحدث تغيير في قيمة إشارة الساعة clk

clk = '1' يتحقق هذا الجزء من التعليمة من أن قيمة إشارة الساعة clk أصبحت '1'، مما يعني أنه تم اكتشاف جبهة صاعدة (rising edge).

```
IF rising_edge(clk) THEN
```

عندما تكون إشارة الساعة صاعدة (من منخفضة إلى مرتفعة)، يتم تنفيذ الكود الموجود داخل الجزء المتسلسل من التعليمة. تستخدم لتفادي مشاكل التوقيت والقيم المتغيرة في نفس الوقت.

انتهت المحاضرة