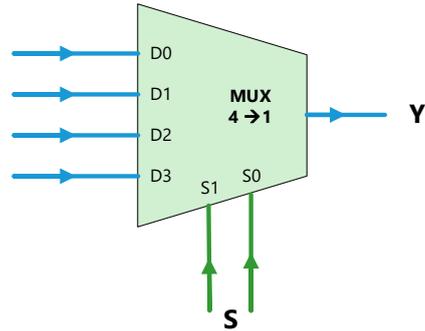


Mux 4 to 1 – מרבב 4 ל-1

s	S1	S0	Y
0	0	0	D0
1	0	1	D1
2	1	0	D2
3	1	1	D3



1. באמצעות פקודת ..select (with S1,S0 מוגדרים כוקטור S)

```

ENTITY mux4to1 IS
    PORT( d0, d1, d2, d3 : IN BIT;
          S : IN bit_vector(1 downto 0);
          Y : OUT BIT);
END mux4to1;

ARCHITECTURE arc_mux OF mux4to1 IS
BEGIN

WITH s SELECT
    Y <= d0 WHEN "00",
        d1 WHEN "01",
        d2 WHEN "10",
        d3 WHEN others;
END arc_mux;

```

2. באמצעות פקודת ..select (with S=S1,S0 מוגדר integer)

```

ENTITY mux4to1 IS
    PORT(d0, d1, d2, d3 : IN BIT;
          s : IN INTEGER RANGE 0 TO 3;
          Y : OUT BIT );
END mux4to1;

ARCHITECTURE arc_mux OF mux4to1 IS
BEGIN

WITH s SELECT
    Y <= d0 WHEN 0,
        d1 WHEN 1,
        d2 WHEN 2,
        d3 WHEN others;
END arc_mux;

```

3. באמצעות פקודת פקודת when else (with S=S1,S0 מוגדר integer)

```

ENTITY mux4to1 IS
    PORT(d0, d1, d2, d3 : IN BIT;
          s : IN INTEGER RANGE 0 TO 3;
          Y : OUT BIT );
END mux4to1;

ARCHITECTURE arc_mux OF mux4to1 IS
BEGIN

    Y <= d0 WHEN s=0 else
        d1 WHEN s=1 else
        d2 WHEN s=2 else
        d3;
END arc_mux;

```

4. באמצעות פקודת ..select with (שימוש בשרשור & ובאות signal)

```

ENTITY mux4to1 IS
    PORT( d0, d1, d2, d3      : IN BIT;
          s0,s1              : IN BIT;
          Y                  : OUT BIT);
END mux4to1;

ARCHITECTURE arc_mux OF mux4to1 IS
    signal sel:bit_vector(1 downto 0);

BEGIN
    sel<=(s1 & s0);

    WITH (sel) SELECT
        Y <=      d0 WHEN "00",
                 d1 WHEN "01",
                 d2 WHEN "10",
                 d3 WHEN others;

END arc_mux;

```

5. באמצעות פקודה אחת (S מוגדר כ-integer ו-D מוגדר כווקטור)

```

ENTITY mux4to1 IS
    PORT(d      : IN BIT_VECTOR(3 downto 0);
          s      : IN INTEGER RANGE 0 TO 3;
          Y      : OUT BIT );
END mux4to1;

ARCHITECTURE arc_mux OF mux4to1 IS
BEGIN
    Y <= d(s);

END arc_mux;

```

6. באמצעות תהליך - Process (פקודת if else)

```

ENTITY mux4to1 IS
    PORT(d      : IN BIT_VECTOR(3 downto 0);
          s      : IN INTEGER RANGE 0 TO 3;
          Y      : OUT BIT );
END mux4to1;

ARCHITECTURE arc_mux OF mux4to1 IS
BEGIN
    PROCESS (d, s)
    BEGIN
        IF      s = 0 THEN Y <= d(0);
        ELSIF  s = 1 THEN Y <= d(1);
        ELSIF  s = 2 THEN Y <= d(2);
        ELSE    Y <= d(3);
        END IF;
    END PROCESS;

END arc_mux;

```

7. באמצעות תהליך - Process (פקודת case)

```

ENTITY mux4to1 IS
    PORT (d      : IN BIT_VECTOR(3 downto 0);
          s      : IN INTEGER RANGE 0 TO 3;
          Y      : OUT BIT );
END mux4to1;

ARCHITECTURE arc_mux OF mux4to1 IS
BEGIN
    PROCESS (d, s)
    BEGIN
        case s is
            when 0      => Y <= d(0);
            when 1      => Y <= d(1);
            when 2      => Y <= d(2);
            when others => Y <= d(3);
        end case;

    END PROCESS;
END arc_mux;

```

8. באמצעות תהליך - Process (לולאת for)

```

ENTITY mux4to1 IS
    PORT (d      : IN BIT_VECTOR(3 downto 0);
          s      : IN INTEGER RANGE 0 TO 3;
          Y      : OUT BIT );
END mux4to1;

ARCHITECTURE arc_mux OF mux4to1 IS
BEGIN
    PROCESS (d, s)
    BEGIN
        For i in 0 to 3 loop
            IF s = i THEN Y <= d(i);
            END IF;
        end loop;
    END PROCESS;
END arc_mux;

```

דלגלג מסוג JKFF

CLK כניסה	J כניסה	K כניסה	Q מוצא
	0	0	שומר מצב
	0	1	0
	1	0	1
	1	1	הופך מצב

1. מוצא מסוג buffer (מוצא ניתן לקריאה)

```

ENTITY jk_ff IS
    PORT ( j, k, clk : IN BIT;
          q          : buffer BIT);
END jk_ff;

ARCHITECTURE arc_jk_ff OF jk_ff IS
BEGIN

    PROCESS (clk)
    BEGIN
        IF clk'EVENT AND clk = '1' THEN

            IF j = '1' and k = '0' THEN q <= '1';
            ELSIF j = '0' and k = '1' THEN q <= '0';
            ELSIF j = '1' and k = '1' THEN q <= not q;
            END IF;
        END IF;
    END PROCESS;
END arc_jk_ff;

```

2. מוצא מסוג out (שימוש באות signal) ושימוש בשרשור ל-JK

```

ENTITY jk_ff IS
    PORT ( j, k, clk : IN BIT;
          q          : OUT BIT);
END jk_ff;

ARCHITECTURE arc_jk_ff OF jk_ff IS
    signal qs: bit;
    signal jk:bit_vector(1 downto 0);

BEGIN
    jk<=j & k;
    PROCESS (clk)
    BEGIN
        IF clk'EVENT AND clk ='1' THEN

            IF jk="10" THEN qs <= '1';
            ELSIF jk="01" THEN qs <= '0';
            ELSIF jk="11" THEN qs <= not qs;
            END IF;
        END IF;
    END PROCESS;
    q<=qs;
END arc_jk_ff;

```

3. שימוש במשתנה variable

```

ENTITY jk_ff IS
    PORT ( j, k, clk : IN BIT;
          q          : OUT BIT);
END jk_ff;

ARCHITECTURE arc_jk_ff OF jk_ff IS

BEGIN

    PROCESS (clk)
        variable qs: bit;
        variable jk:bit_vector(1 downto 0);

    BEGIN
        jk:=j & k;

        IF clk'EVENT AND clk ='1' THEN

            case jk is
                when "01" => qs := '0';
                when "10" => qs := '1';
                when "11" => qs := not qs;
                when others => null;
            end case;

        END IF;
        q<=qs;

    END PROCESS;
END arc_jk_ff;

```

דוגמא מפענח

	A in	B in	C in	F out
0	0	0	0	1
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0

1. באמצעות פקודת `when ... Else`

```
entity my_decoder is
    port (A,B,C : in bit;
          F : out bit);
end my_decoder;

architecture arc_deoder of my_decoder is
    signal ABC: bit_vector(2 downto 0);
begin
    ABC <= A & B & C;
    F <= '1' when ABC="000" or ABC="001" or ABC="100" else
        '0';
end arc_deoder;
```

2. באמצעות פקודת `with select`

```
entity my_decoder is
    port (A,B,C : in bit;
          F : out bit);
end my_decoder;

architecture arc_deoder of my_decoder is
    signal ABC: bit_vector(2 downto 0);
begin
    ABC <= A & B & C;
    WITH (ABC) SELECT

    F <= '1' when "000" | "001" | "100" ,
        '0' when others;
end arc_deoder;
```

3. עם פקודת case בתוך תהליך

```

entity my_decoder is
    port (A,B,C : in bit;
          F : out bit);
end my_decoder;

architecture arc_deoder of my_decoder is
    signal ABC: bit_vector(2 downto 0);
begin
    ABC <= A & B & C;
    process (ABC)
    begin
        case (ABC) is
            when "000" => F <= '1';
            when "001" => F <= '1';
            when "100" => F <= '1';
            when others => F <= '0';
        end case;
    end process ;
end arc_deoder;

```

4. עם פקודת case בתוך תהליך ושימוש באופרטור |

```

architecture arc_deoder of my_decoder is
    signal ABC: bit_vector(2 downto 0);
begin
    ABC <= A & B & C;
    process (ABC)
    begin
        case (ABC) is
            when "000" | "001" | "100" => F <= '1';
            when others => F <= '0';
        end case;
    end process ;
end arc_deoder;

```

5. כניסה מסוג integer

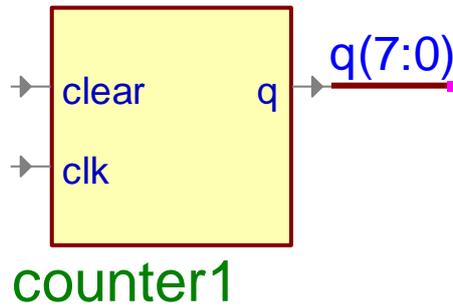
```

entity my_decoder is
    port (ABC : in integer range 0 to 7;
          F : out bit);
end my_decoder;

architecture arc_deoder of my_decoder is
begin
    process (ABC)
    begin
        case (ABC) is
            when 0 to 1 | 4 => F <= '1';
            when others => F <= '0';
        end case;
    end process ;
end arc_deoder;

```

מונה



1. מונה 0 עד 255 עם כניסת איפוס סינכרוני

```

ENTITY counter1 IS
    PORT (clk,clear: IN bit;
          q : BUFFER integer RANGE 0 TO 255);
END counter1;

ARCHITECTURE count OF counter1 IS
BEGIN
    PROCESS (clk)
    BEGIN
        IF clk'event AND clk = '1' THEN
            IF clear = '0' THEN q <= 0;
            ELSIF q < 255 THEN q <= q + 1;
            ELSE q <= 0;
            END IF;
        END IF;
    END PROCESS;
END count;
  
```

2. מונה 0 עד 255 עם כניסת איפוס אסינכרוני

```

ENTITY counter1 IS
    PORT (clk,clear: IN bit;
          q : BUFFER integer RANGE 0 TO 255);
END counter1;

ARCHITECTURE count OF counter1 IS
BEGIN

    PROCESS (clk,clear)
    BEGIN
        IF clear = '0' THEN q <= 0;
        ELSIF clk'event AND clk = '1' THEN
            IF q < 255 THEN q <= q + 1;
            ELSE q <= 0;
            END IF;
        END IF;
    END PROCESS;
END count;
  
```

3. מונה 0 עד 255 (שימוש במשתנה variable , איפוס אסינכרוני)

```

ENTITY counter1 IS
    PORT (clk,clear: IN bit;
          q : OUT    integer RANGE 0 TO 255);
END counter1;

ARCHITECTURE count OF counter1 IS
BEGIN

    PROCESS (clk,clear)
        VARIABLE cnt : integer RANGE 0 TO 255;

    BEGIN
        IF clear = '0' THEN cnt := 0;
        ELSEIF clk'event AND clk = '1' THEN
            IF cnt <255 THEN cnt := cnt + 1;
            ELSE cnt:=0;
            END IF;
        END IF;
        q <= cnt;
    END PROCESS;

END count;

```

4. מונה 0 עד 255 (שימוש במשתנה signal , איפוס אסינכרוני)

```

ENTITY counter1 IS
    PORT (clk,clear: IN bit;
          q : OUT    integer RANGE 0 TO 255);
END counter1;

ARCHITECTURE count OF counter1 IS
    signal cnt : integer RANGE 0 TO 255;
BEGIN

    PROCESS (clk,clear)
    BEGIN
        IF clear = '0' THEN cnt <= 0;
        ELSEIF clk'event AND clk = '1' THEN
            IF cnt <255 THEN cnt <= cnt + 1;
            ELSE cnt <= 0;
            END IF;

        END IF;
    END PROCESS;
    q <= cnt;

END count;

```

5. מונה 0 עד 255 (שימוש בספריה STD_LOGIC_UNSIGNED לשימוש פעולות אריתמטיות עם (VECTOR

```

library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.all;

ENTITY counter1 IS
    PORT (clk, clear: IN STD_LOGIC;
          q : OUT  STD_LOGIC_VECTOR (7 downto 0));
END counter1;

ARCHITECTURE count OF counter1 IS
    signal cnt : STD_LOGIC_VECTOR(7 downto 0);
BEGIN

    PROCESS (clk)
    BEGIN
        IF clear = '0' THEN cnt <= (OTHERS=>'0');
        ELSIF clk'event AND clk = '1' THEN cnt <= cnt + 1;
        END IF;
    END PROCESS;
    q <= cnt;

END count;

```

6. מונה מודולו 200 סופר מ-0 עד 199

```

ENTITY counter200 IS
    PORT ( clk : IN BIT;
          q  : BUFFER INTEGER RANGE 0 TO 255);
END counter200;

ARCHITECTURE count OF counter200 IS
    CONSTANT modulus : INTEGER := 200;
BEGIN
    -- A modulus 200 up counter
    PROCESS (clk)

    BEGIN
        IF (clk'EVENT AND clk = '1') THEN
            IF q < (modulus-1) THEN q<= q+ 1;
            ELSE q<= 0;
            END IF;
        END IF;

    END PROCESS;
END count;

```