

# Логичко пројектовање

Предавање 3/10

**Опис MSI комбинационих  
мрежа у VHDL-у**

---

## MSI комбинационе мреже

- Овде се уводи група блокова комбинационе логике која се обично користе у дигиталном дизајну. Како прелазимо у системе који су већи од појединачних гејтова, постоје конвенције именовања које се користе за описивање величине мреже.

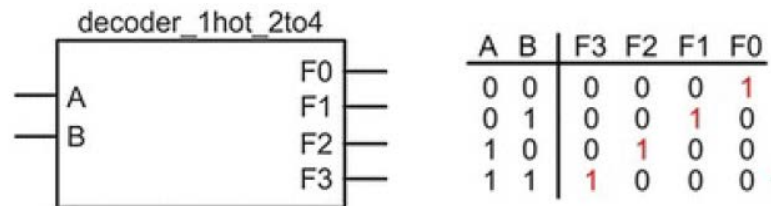
### Commonly Used Names to Describe The Size of Digital Logic

Name	Example	# of Transistors
SSI - Small Scale Integrated Circuits	Individual Gates (NAND, INV)	10's
MSI - Medium Scale Integrated Circuits	Decoders, Multiplexers	100's
LSI - Large Scale Integrated Circuits	Arithmetic Circuits, RAM	1k - 10k
VLSI - Very Large Scale Integrated Circuits	Microprocessors	100k - 1M

## Декодери

- Декодер је комбинационо MSI коло које на улазима узима бинарни код а на излазима даје специфичне вредности тог кода. Код може бити било које врсте или величине.
- Сваки излаз ће бити постављен само за одређени код на улазу. Обзиром да комбинациона логичка кола производе само један излаз, ово значи да ће унутар декодера постојати посебно комбинационо логичко коло за сваки излаз.
- "One-Hot" декодер је коло које има  $n$  улаза и  $2^n$  излаза. Сваки излаз ће имати вредност 1 за један и само један улазни код. Пошто постоји  $2^n$  излаза, увек ће један и само један излаз имати вредност 1 у било ком тренутку.
- Пример "One-Hot" декодера
  - "2-to-4 One-Hot" декодер
  - "3-to-8 One-Hot" декодер
- Пример декодера за 7-сегментни дисплеј

# Пример "2-to-4 One-Hot" декодера (ручна логичка синтеза)

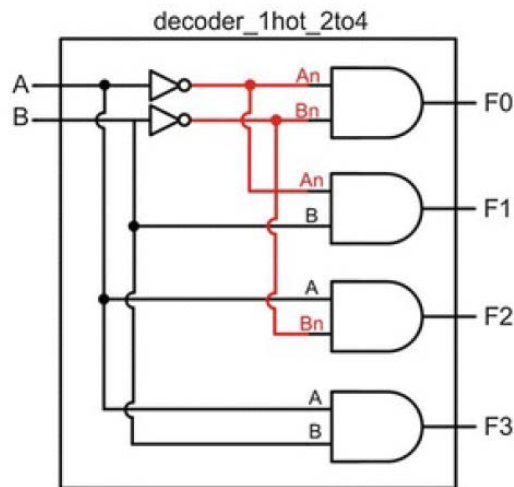


$$F0 = \sum_{A,B}(0) = A' \cdot B'$$

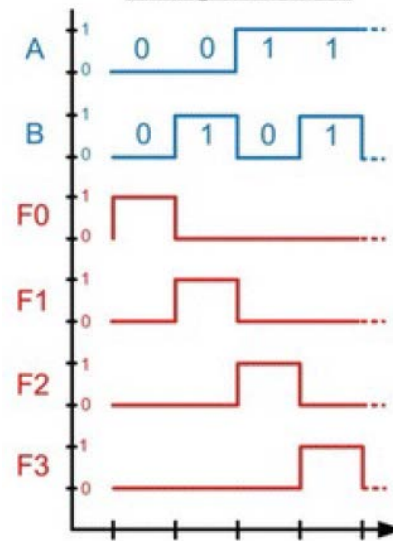
$$F2 = \sum_{A,B}(2) = A \cdot B'$$

$$F1 = \sum_{A,B}(1) = A' \cdot B$$

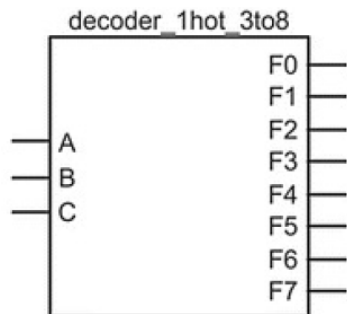
$$F3 = \sum_{A,B}(3) = A \cdot B$$



Timing Waveform



# Пример “3-to-8 One-Hot” декодера (VHDL модел – конкурентна додела сигнала и логички оператори)



A	B	C	F7	F6	F5	F4	F3	F2	F1	F0
0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0	1	0	0
0	1	1	0	0	0	0	1	0	0	0
1	0	0	0	0	0	1	0	0	0	0
1	0	1	0	0	1	0	0	0	0	0
1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0

$$F0 = \sum_{A,B,C}(0) = A'B'C'$$

$$F1 = \sum_{A,B,C}(1) = A'B'C$$

$$F2 = \sum_{A,B,C}(2) = A'B \cdot C'$$

$$F3 = \sum_{A,B,C}(3) = A'B \cdot C$$

$$F4 = \sum_{A,B,C}(4) = A \cdot B'C'$$

$$F5 = \sum_{A,B,C}(5) = A \cdot B'C$$

$$F6 = \sum_{A,B,C}(6) = A \cdot B \cdot C'$$

$$F7 = \sum_{A,B,C}(7) = A \cdot B \cdot C$$

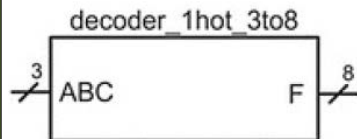
```
entity decoder_1hot_3to8 is
  port (A,B,C          : in bit;
        F0,F1,F2,F3,F4,F5,F6,F7 : out bit);
end entity;

architecture decoder_1hot_3to8_arch of decoder_1hot_3to8 is
begin

  F0 <= (not A) and (not B) and (not C);
  F1 <= (not A) and (not B) and (C);
  F2 <= (not A) and (B)      and (not C);
  F3 <= (not A) and (B)      and (C);
  F4 <= (A)      and (not B) and (not C);
  F5 <= (A)      and (not B) and (C);
  F6 <= (A)      and (B)    and (not C);
  F7 <= (A)      and (B)    and (C);

end architecture;
```

# Пример "3-to-8 One-Hot" декодера (VHDL модел – условна/селекциона додела сигнала)



ABC	F(7)	F(6)	F(5)	F(4)	F(3)	F(2)	F(1)	F(0)
"000"	0	0	0	0	0	0	0	1
"001"	0	0	0	0	0	0	1	0
"010"	0	0	0	0	0	1	0	0
"011"	0	0	0	0	1	0	0	0
"100"	0	0	0	1	0	0	0	0
"101"	0	0	1	0	0	0	0	0
"110"	0	1	0	0	0	0	0	0
"111"	1	0	0	0	0	0	0	0

```
entity decoder_1hot_3to8 is
  port (ABC : in bit_vector(2 downto 0);
        F   : out bit_vector(7 downto 0));
end entity;
```

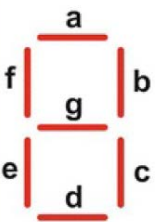








```
architecture decoder_1hot_3to8_arch of decoder_1hot_3to8 is
begin
  F <= "00000001" when (ABC = "000") else
       "00000010" when (ABC = "001") else
       "00000100" when (ABC = "010") else
       "00001000" when (ABC = "011") else
       "00010000" when (ABC = "100") else
       "00100000" when (ABC = "101") else
       "01000000" when (ABC = "110") else
       "10000000" when (ABC = "111");
end architecture;
```

```
architecture decoder_1hot_3to8_arch of decoder_1hot_3to8 is
begin
  with (ABC) select
    F <= "00000001" when "000",
         "00000010" when "001",
         "00000100" when "010",
         "00001000" when "011",
         "00010000" when "100",
         "00100000" when "101",
         "01000000" when "110",
         "10000000" when "111";
end architecture;
```

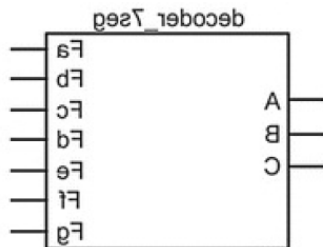
## Декодер за 7-сегментни дисплеј

- Декодер дисплеја са 7 сегмената је коло које се користи за приказ знакова на дисплеју који се обично налазе у дигиталним сатовима и апаратима за домаћинство.
- Дисплеј знакова се обично састоји од 7 појединачних LED диода са ознакама а – g. Улаз у декодер је бинарни еквивалент децималног или хексадецималног знака који треба да се прикаже.
- Изаз декодера је распоред LED диода који ће формирати лик. Декодери са 2 улаза могу приказати знакове од „0“ до „3“. Декодери са 3 улаза могу да прикажу знакове „0“ до „7“. Декодери са 4 улаза могу покретати знакове „0“ до „F“ са малим словима. Хексадецимални знакови су „A, b, c или C, d, E и F“.

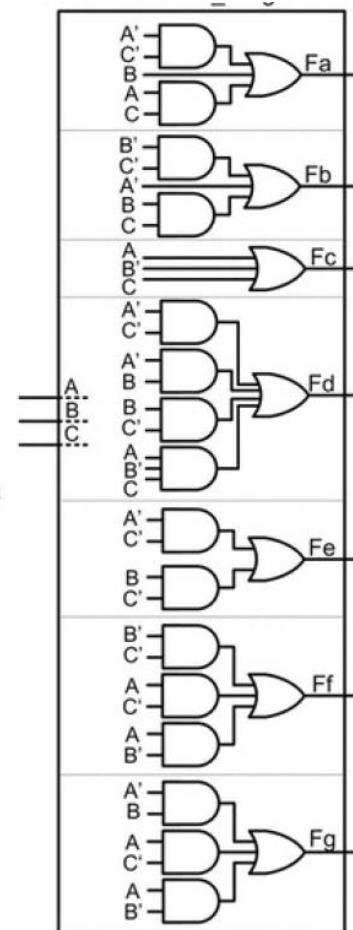
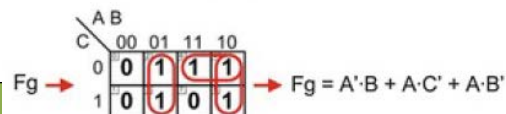
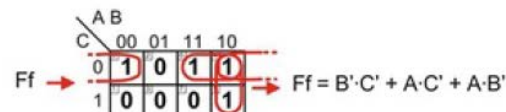
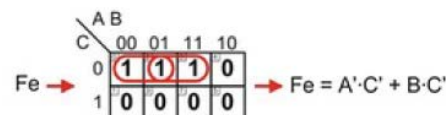
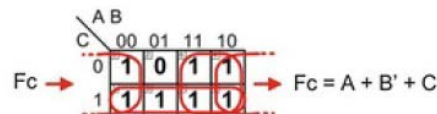
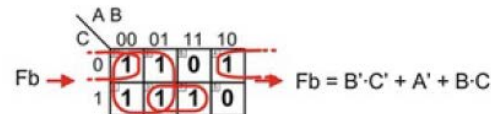
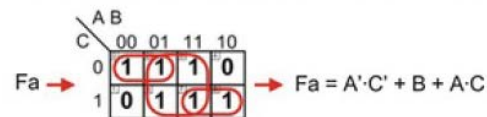
Example: 7-Segment Display Decoder - Truth Table

			A	B	C	F <sub>a</sub> F <sub>b</sub> F <sub>c</sub> F <sub>d</sub> F <sub>e</sub> F <sub>f</sub> F <sub>g</sub>						
<div>LED Labels</div> 	0	0	0		1	1	1	1	1	1	0	
	0	0	1		0	1	1	0	0	0	0	
	0	1	0		1	1	0	1	1	0	1	
	0	1	1		1	1	1	1	0	0	1	
	1	0	0		0	1	1	0	0	1	1	
	1	0	1		1	0	1	1	0	1	1	
	1	1	0		1	0	1	1	1	1	1	
	1	1	1		1	1	1	0	0	0	0	

# Декодер за 7-сегментни дисплеј (ручна логичка синтеза)



A	B	C	Fa	Fb	Fc	Fd	Fe	Ff	Fg
0	0	0	1	1	1	1	1	1	0
0	0	1	0	1	1	0	0	0	0
0	1	0	1	1	0	1	1	0	1
0	1	1	1	1	1	1	0	0	1
1	0	0	0	1	1	0	0	1	1
1	0	1	1	0	1	1	0	1	1
1	1	0	1	0	1	1	1	1	1
1	1	1	1	1	1	0	0	0	0





# Декодер за 7-сегментни дисплеј (VHDL модел – конкурентна додела сигнала и логички оператори)

decoder_7seg			A	B	C	Fa	Fb	Fc	Fd	Fe	Ff	Fg
— A — B — C	Fa Fb Fc Fd Fe Ff Fg	— — — — — — —	0	0	0	1	1	1	1	1	1	0
			0	0	1	0	1	1	0	0	0	0
			0	1	0	1	1	0	1	1	0	1
			0	1	1	1	1	1	1	0	0	1
			1	0	0	0	1	1	0	0	1	1
			1	0	1	1	0	1	1	0	1	1
			1	1	0	1	0	1	1	1	1	1
			1	1	1	1	1	1	0	0	0	0

```

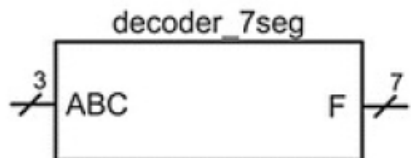
entity decoder_7seg is
    port (A,B,C          : in bit;
          Fa,Fb,Fc,Fd,Fe,Fg,Fg : out bit);
end entity;

architecture decoder_7seg_arch of decoder_7seg is
begin
    Fa <= ((not A) and (not C)) or B or (A and C);
    Fb <= ((not B) and (not C)) or (not A) or (B and C);
    Fc <= A or (not B) or C;
    Fd <= ((not A) and (not C)) or ((not A) and B) or (B and (not C))
        or (A and (not B) and C);
    Fe <= ((not A) and (not C)) or (B and (not C));
    Ff <= ((not B) and (not C)) or (A and (not C)) or (A and (not B));
    Fg <= ((not A) and B) or (A and (not C)) or (A and (not B));

end architecture;

```

# Декодер за 7-сегментни дисплеј (VHDL модел – условна/селекциона додела сигнала)



ABC	a F(6)	b F(5)	c F(4)	d F(3)	e F(2)	f F(1)	g F(0)
"000"	1	1	1	1	1	1	0
"001"	0	1	1	0	0	0	0
"010"	1	1	0	1	1	0	1
"011"	1	1	1	1	0	0	1
"100"	0	1	1	0	0	1	1
"101"	1	0	1	1	0	1	1
"110"	1	0	1	1	1	1	1
"111"	1	1	1	0	0	0	0

```

entity decoder_7seg is
  port (ABC : in bit_vector(2 downto 0);
        F   : out bit_vector(6 downto 0));
end entity;
  
```

```

architecture decoder_7seg_arch of decoder_7seg is
begin
  F <= "1111110" when (ABC = "000") else
        "0110000" when (ABC = "001") else
        "1101101" when (ABC = "010") else
        "1111001" when (ABC = "011") else
        "0110011" when (ABC = "100") else
        "1011011" when (ABC = "101") else
        "1011111" when (ABC = "110") else
        "1110000" when (ABC = "111");
end architecture;
  
```

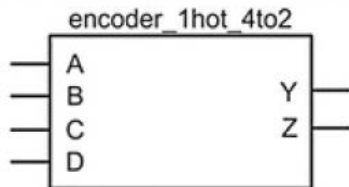
```

architecture decoder_7seg_arch of decoder_7seg is
begin
  with (ABC) select
    F <= "1111110" when "000",
        "0110000" when "001",
        "1101101" when "010",
        "1111001" when "011",
        "0110011" when "100",
        "1011011" when "101",
        "1011111" when "110",
        "1110000" when "111";
end architecture;
  
```

## Кодери

- Кодер функционише супротно у односу на декодер. Појава јединице на одређеном улазном порту одговара јединственом коду на излазном порту.
- "One-Hot" кодер је коло које има  $n$  излаза и  $2^n$  улаза. Излаз је  $n$  – битни бинарни код који одговара појави јединице на један и само један улаз.
- Пример "One-Hot" декодера
  - "4-to-2 One-Hot" декодер

# Пример “4-to-2 One-Hot” кодера (ручна логичка синтеза)



A	B	C	D	Y	Z
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

D =&gt; "00"

C =&gt; "01"

B =&gt; "10"

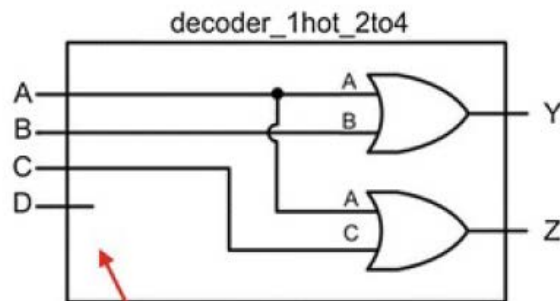
A =&gt; "11"

		Y				
		AB	00	01	11	10
CD	00	X	1	X	1	
	01	0	X	X	X	
	11	X	X	X	X	
	10	0	X	X	X	

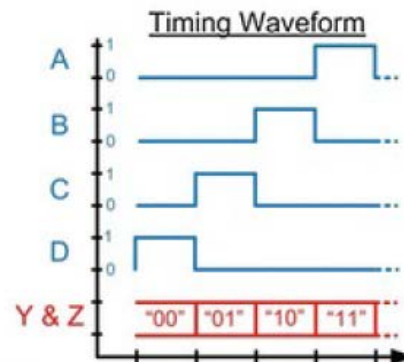
→  $Y = A + B$

		Z				
		AB	00	01	11	10
CD	00	X	0	X	1	
	01	0	X	X	X	
	11	X	X	X	X	
	10	1	X	X	X	

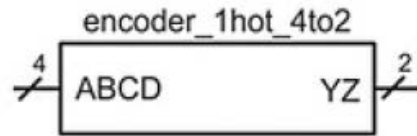
→  $Z = A + C$



Notice that D is not used.



# Пример “4-to-2 One-Hot” кодера (VHDL модел – конкурентна/условна/селекциона додела сигнала)



ABCD	YZ
"0001"	"00"
"0010"	"01"
"0100"	"10"
"1000"	"11"

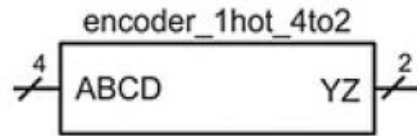
```
entity encoder_1hot_4to2 is
    port (ABCD : in bit_vector(3 downto 0);
          YZ   : out bit_vector(1 downto 0));
end entity;
```

```
architecture encoder_1hot_4to2_arch of encoder_1hot_4to2 is
begin
    YZ(1) <= ABCD(3) or ABCD(2);
    YZ(0) <= ABCD(3) or ABCD(1);
end architecture;
```

```
architecture encoder_1hot_4to2_arch of encoder_1hot_4to2 is
begin
    YZ <= "00" when (ABCD = "0001") else
          "01" when (ABCD = "0010") else
          "10" when (ABCD = "0100") else
          "11" when (ABCD = "1000") else
          "00";
end architecture;
```

```
architecture encoder_1hot_4to2_arch of encoder_1hot_4to2 is
begin
    with (ABCD) select
        YZ <= "00" when "0001",
              "01" when "0010",
              "10" when "0100",
              "11" when "1000",
              "00" when others;
end architecture;
```

# Пример “4-to-2 One-Hot” кодера (VHDL модел – конкурентна/условна/селекциона додела сигнала)



ABCD	YZ
"0001"	"00"
"0010"	"01"
"0100"	"10"
"1000"	"11"

```
entity encoder_1hot_4to2 is
  port (ABCD : in bit_vector(3 downto 0);
        YZ   : out bit_vector(1 downto 0));
end entity;
```

```
architecture encoder_1hot_4to2_arch of encoder_1hot_4to2 is
begin
  YZ(1) <= ABCD(3) or ABCD(2);
  YZ(0) <= ABCD(3) or ABCD(1);
end architecture;
```

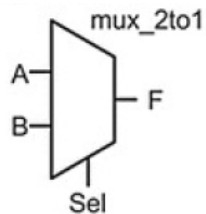
```
architecture encoder_1hot_4to2_arch of encoder_1hot_4to2 is
begin
  YZ <= "00" when (ABCD = "0001") else
        "01" when (ABCD = "0010") else
        "10" when (ABCD = "0100") else
        "11" when (ABCD = "1000") else
        "00";
end architecture;
```

```
architecture encoder_1hot_4to2_arch of encoder_1hot_4to2 is
begin
  with (ABCD) select
    YZ <= "00" when "0001",
          "01" when "0010",
          "10" when "0100",
          "11" when "1000",
          "00" when others;
end architecture;
```

## Мультиплексери

- Мультиплексер је коло које преноси један од својих вишеструких улаза на један излаз на основу селекционих (контролних) улаза.
- Он се може сматрати дигиталним прекидачем.
- Мультиплексер има  $n$  изабраних линија,  $2^n$  улаза и 1 излаз.
- Пример "2-to-1" мултиплексера

# Пример “2-to-1 мултиплексера (ручна логичка синтеза)”



Sel	F
0	A
1	B

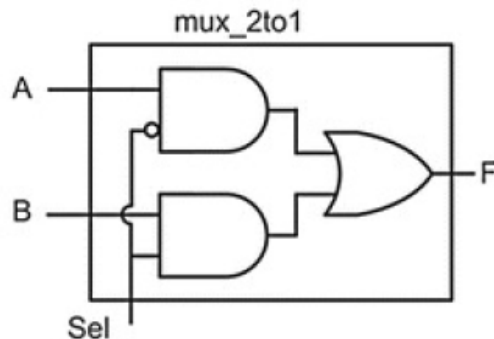
When Sel=0,  
the output is A

When Sel=1,  
the output is B

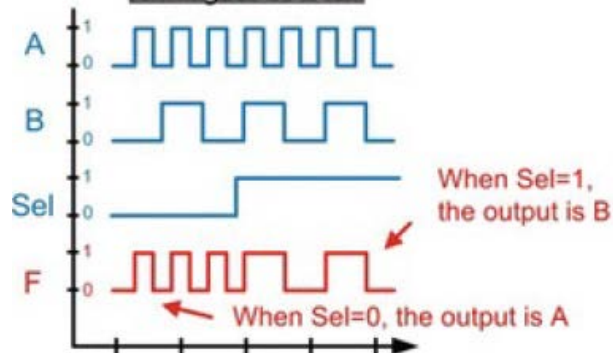
Sel	A	B	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

		Sel A			
		00	01	11	10
B	0	0	1	0	0
	1	0	1	1	1

$$F = \text{Sel}' \cdot A + \text{Sel} \cdot B$$

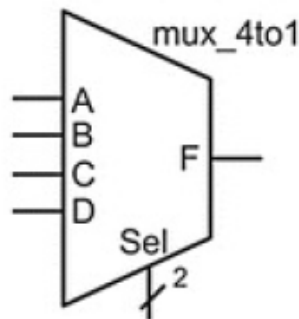


Timing Waveform





## Пример “2-to-1 мултиплексера (VHDL модел – конкурентна/условна/селекциона додела сигнала)



Sel	F
"00"	A
"01"	B
"10"	C
"11"	D

```
entity mux_4to1 is
    port (A,B,C,D : in bit;
          Sel      : in bit_vector(1 downto 0);
          F        : out bit);
end entity;
```

```
architecture mux_4to1_arch of mux_4to1 is
begin
    F <= (A and not Sel(0) and not Sel(1)) or
         (B and not Sel(0) and      Sel(1)) or
         (C and      Sel(0) and not Sel(1)) or
         (D and      Sel(0) and      Sel(1));
end architecture;
```

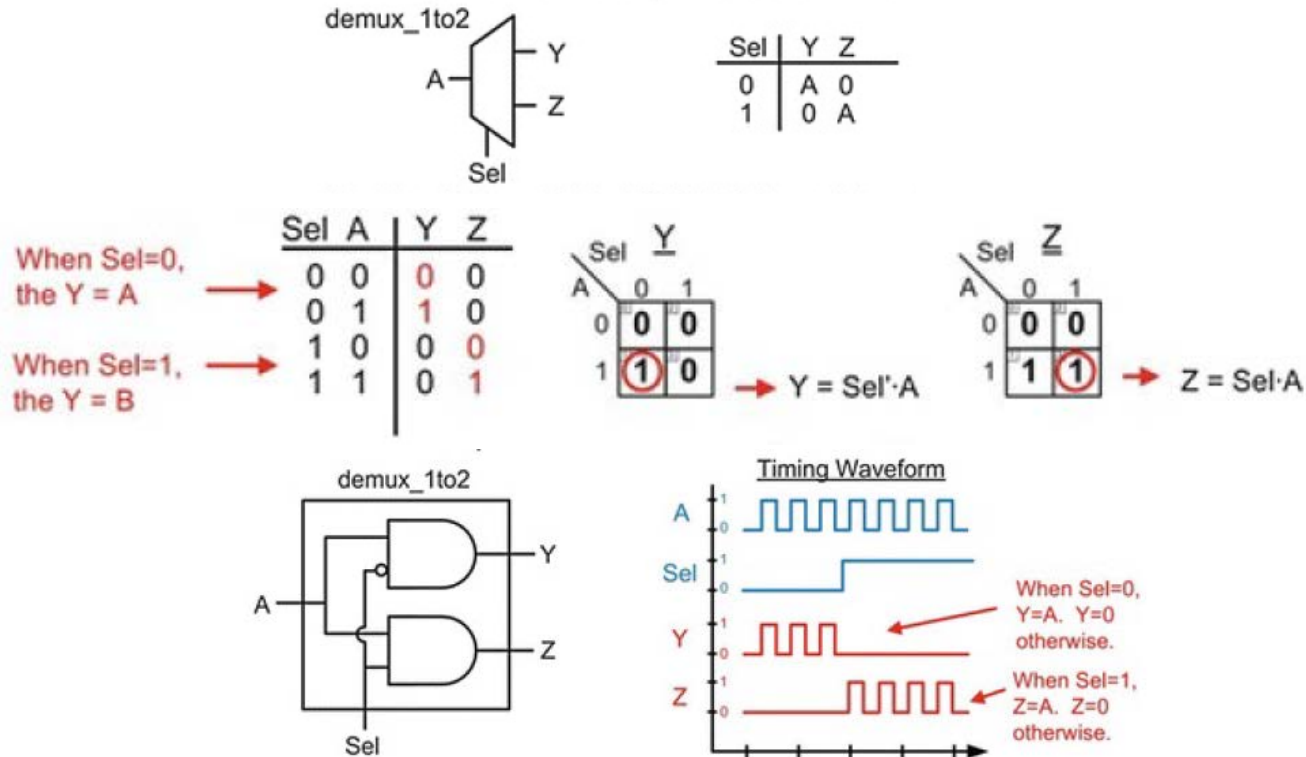
```
architecture mux_4to1_arch of mux_4to1 is
begin
    F <= A when (Sel = "00") else
         B when (Sel = "01") else
         C when (Sel = "10") else
         D when (Sel = "11");
end architecture;
```

```
architecture mux_4to1_arch of mux_4to1 is
begin
    with (Sel) select
        F <= A when "00",
             B when "01",
             C when "10",
             D when "11";
end architecture;
```

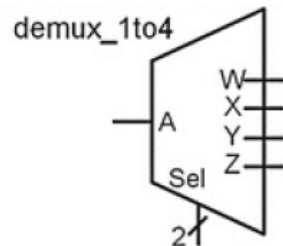
## Демултиплексери

- Демултиплексери функционишу у комплементарном моду у односу на мултиплексере.
- Демултиплексер има 1 улаз који се усмерава на 1 од својих вишеструких излаза.
- Израз који је активан диктира се на основу селекционих (контролних) улаза.
- Демултиплексер има  $n$  изабраних линија које усмерава улаз на један од својих  $2^n$  излаза. Када излаз није изабран, на излазу је логичка 0.
- Пример "1-to-2" демултиплексер

# Пример “1-to-2 демультиплексера (ручна логичка синтеза)”



# Пример “1-to-2 демултиплексера (VHDL модел – конкурентна/условна/селекциона додела сигнала)



Sel	W	X	Y	Z
"00"	A	0	0	0
"01"	0	A	0	0
"10"	0	0	A	0
"11"	0	0	0	A

```
entity demux_1to4 is
  port (A      : in  bit;
        Sel    : in  bit_vector(1 downto 0);
        W,X,Y,Z : out bit);
end entity;
```

```
architecture demux_1to4_arch of demux_1to4 is
begin
  W <= A and not Sel(0) and not Sel(1);
  X <= A and not Sel(0) and      Sel(1);
  Y <= A and      Sel(0) and not Sel(1);
  Z <= A and      Sel(0) and      Sel(1);
end architecture;
```

```
architecture demux_1to4_arch of demux_1to4 is
begin
  W <= A when (Sel = "00") else '0';
  X <= A when (Sel = "01") else '0';
  Y <= A when (Sel = "10") else '0';
  Z <= A when (Sel = "11") else '0';
end architecture;
```

```
architecture demux_1to4_arch of demux_1to4 is
begin
  with (Sel) select
    W <= A when "00", '0' when others;

  with (Sel) select
    X <= A when "01", '0' when others;

  with (Sel) select
    Y <= A when "10", '0' when others;

  with (Sel) select
    Z <= A when "11", '0' when others;
end architecture;
```