KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

„Układy reprogramowalne i SoC”
„Język VHDL (część 3)”

Prezentacja jest współfinansowana przez
Unię Europejską w ramach
Europejskiego Funduszu Społecznego w projekcie pt.

„Innowacyjna dydaktyka bez ograniczeń - zintegrowany rozwój Politechniki Łódzkiej - zarządzanie Uczelnią, nowoczesna oferta edukacyjna i wzmacniania zdolności do zatrudniania osób niepełnosprawnych”

Prezentacja dystrybuowana jest bezpłatnie

Kod sekwencyjny

- Kod VHDL jest współbieżny
- Jedynie wewnątrz procesów, funkcji i procedur kod wykonywany jest sekwencyjnie
  - Wszystkie te bloki są wykonywane współbieżnie z instrukcjami umieszczonymi poza nimi
- Przy pomocy kodu sekwencyjnego można opisywać zarówno układy kombinacyjne, jak i sekwencyjne
- Zmienne mogą być używane jedynie w kodzie sekwencyjnym

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

PROCESS

- Proces to sekwencyjna sekcja kodu VHDL
- Proces jest wykonywany za każdym razem, kiedy sygnał na liście wrażliwościowej ulegnie zmianie

```
[label:] PROCESS (sensitivity list)
   [VARIABLE name type [range] [:= initial_value;]]
BEGIN
   (sequential code)
END PROCESS [label];
```

- Deklaracje zmiennych są opcjonalne.
  - Występuje przed słowem kluczowym BEGIN
- Etykieta jest opcjonalna
  - Poprawia czytelność, ułatwia identyfikację

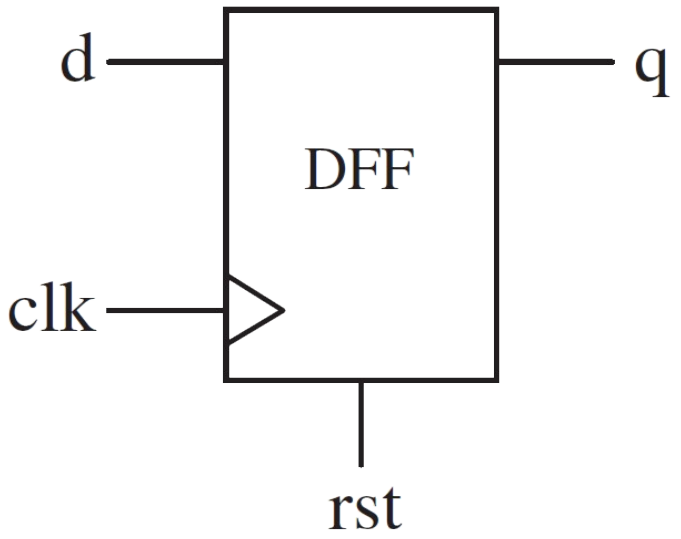
KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Przykład - przerzutnik z asynchronicznym resetem

```
1  ------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------
5  ENTITY dff IS
6     PORT (d, clk, rst: IN STD_LOGIC;
7           q: OUT STD_LOGIC);
8  END dff;
9  ------------------------------------
10 ARCHITECTURE behavior OF dff IS
11 BEGIN
12    PROCESS (clk, rst)
13    BEGIN
14       IF (rst='1') THEN
15          q <= '0';
16       ELSIF (clk'EVENT AND clk='1') THEN
17          q <= d;
18       END IF;
19    END PROCESS;
20 END behavior;
21 ------------------------------------
```



Sygnały i zmienne

- W języku VHDL istnieją dwa sposoby przekazywania zmieniających się wartości: SIGNAL i VARIABLE
  - SIGNAL może być zadeklarowany w części deklaracyjnej ARCHITECTURE i jest widoczny globalnie
  - VARIABLE można zadeklarować tylko wewnątrz kodu sekwencyjnego i nie jest widoczna poza nim
- Uaktualniona wartość zmiennej jest widoczna natychmiast, nowa wartość sygnału jest widoczna dopiero po zakończeniu bieżącego wykonania procesu
- Przypisania do sygnałów wykonuje się przy pomocy operatora '<=', a do zmiennych operatora ':='

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

IF

- Większość ludzi ma naturalną tendencję do używania IF częściej, niż innych instrukcji

```
IF conditions THEN assignments;
ELSIF conditions THEN assignments;
...
ELSE assignments;
END IF;
```

```
IF (x<y) THEN temp:="11111111";
ELSIF (x=y AND w='0') THEN temp:="11110000";
ELSE temp:=(OTHERS =>'0');
```

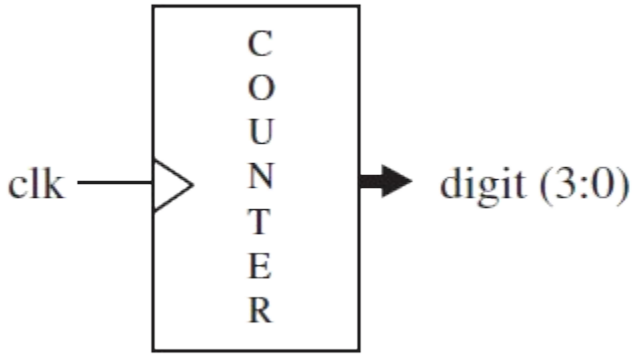
KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

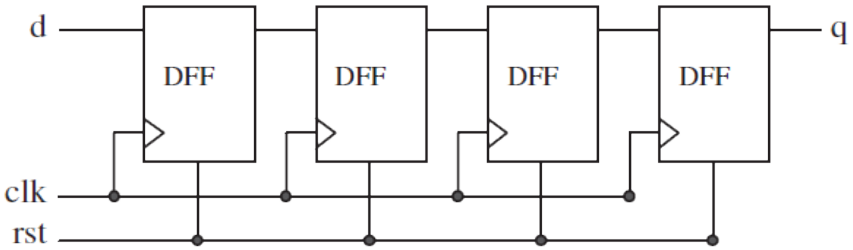
UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Przykład - licznik jednocyfrowy

```
1  ------------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------------
5  ENTITY counter IS
6     PORT (clk : IN STD_LOGIC;
7           digit : OUT INTEGER RANGE 0 TO 9);
8  END counter;
9  ------------------------------------------
10 ARCHITECTURE counter OF counter IS
11 BEGIN
12    count: PROCESS(clk)
13       VARIABLE temp : INTEGER RANGE 0 TO 10;
14    BEGIN
15       IF (clk'EVENT AND clk='1') THEN
16          temp := temp + 1;
17          IF (temp=10) THEN temp := 0;
18          END IF;
19       END IF;
20       digit <= temp;
21    END PROCESS count;
22 END counter;
23 ------------------------------------------
```



Przykład - rejestr przesuwny




```
1  ------------------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------------------
5  ENTITY shiftreg IS
6     GENERIC (n: INTEGER := 4); -- # of stages
7     PORT (d, clk, rst: IN STD_LOGIC;
8           q: OUT STD_LOGIC);
9 END shiftreg;
10 ------------------------------------------------
11 ARCHITECTURE behavior OF shiftreg IS
12    SIGNAL internal: STD_LOGIC_VECTOR (n-1 DOWNTO 0);
13 BEGIN
14   PROCESS (clk, rst)
15   BEGIN
16      IF (rst='1') THEN
17         internal <= (OTHERS => '0');
18      ELSIF (clk'EVENT AND clk='1') THEN
19         internal <= d & internal(internal'LEFT DOWNTO 1);
20      END IF;
21   END PROCESS;
22   q <= internal(0);
23 END behavior;
24 ------------------------------------------------
```

Instrukcja WAIT

- Jeżeli proces ma listę wrażliwościową, nie można w nim korzystać z instrukcji WAIT
- Są trzy formy instrukcji WAIT:

```
WAIT UNTIL signal_condition;
```

```
WAIT ON signal1 [, signal2, ... ];
```

```
WAIT FOR time;
```

- WAIT FOR używany jest tylko w symulacjach:

```
WAIT FOR 5ns;
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

WAIT UNTIL

- WAIT UNTIL zatrzymuje proces do momentu, kiedy zostanie spełniony warunek związany z pewnym sygnałem

```
PROCESS -- no sensitivity list
BEGIN
   WAIT UNTIL (clk'EVENT AND clk='1');
   IF (rst='1') THEN
      output <= "00000000";
   ELSIF (clk'EVENT AND clk='1') THEN
      output <= input;
   END IF;
END PROCESS;
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

WAIT ON

- WAIT ON może mieć wiele sygnałów na liście argumentów.
- Proces zostanie wstrzymany do momentu, w którym któryś z nich ulegnie zmianie.

```
PROCESS
BEGIN
   WAIT ON clk, rst;
   IF (rst='1') THEN
      output <= "00000000";
   ELSIF (clk'EVENT AND clk='1') THEN
      output <= input;
   END IF;
END PROCESS;
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Przykłady WAIT

```
1  ------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------
5  ENTITY dff IS
6     PORT (d, clk, rst: IN STD_LOGIC;
7           q: OUT STD_LOGIC);
8  END dff;
9  ------------------------------------
10 ARCHITECTURE dff OF dff IS
11 BEGIN
12    PROCESS
13    BEGIN
14       WAIT ON rst, clk;
15       IF (rst='1') THEN
16          q <= '0';
17       ELSIF (clk'EVENT AND clk='1') THEN
18          q <= d;
19      END IF;
20    END PROCESS;
21 END dff;
22 ------------------------------------
```

```
1  ------------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------------
5  ENTITY counter IS
6     PORT (clk : IN STD_LOGIC;
7           digit : OUT INTEGER RANGE 0 TO 9);
8  END counter;
9  ------------------------------------------
10 ARCHITECTURE counter OF counter IS
11 BEGIN
12    PROCESS -- no sensitivity list
13       VARIABLE temp : INTEGER RANGE 0 TO 10;
14    BEGIN
15    WAIT UNTIL (clk'EVENT AND clk='1');
16    temp := temp + 1;
17    IF (temp=10) THEN temp := 0;
18    END IF;
19    digit <= temp;
20    END PROCESS;
21 END counter;
22 ------------------------------------------
```

Instrukcja CASE

- CASE jest kolejną instrukcją, której można używać wyłącznie w kodzie sekwencyjnym

```
CASE identifier IS
   WHEN value => assignments;
   WHEN value => assignments;
   ...
END CASE;
```

```
CASE control IS
   WHEN "00" => x<=a; y<=b;
   WHEN "01" => x<=b; y<=c;
   WHEN OTHERS => x<="0000"; y<="ZZZZ";
END CASE;
```

- Wszystkie permutacje muszą zostać uwzględnione, słowo OTHERS jest tu przydatne
- Kiedy dla jakiegoś przypadku nie chcemy nic robić, używamy słowa kluczowego NULL

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Instrukcja CASE

- Wyrażenie "WHEN value" może przyjąć jedną z trzech form

```
WHEN value -- single value
WHEN value1 to value2 -- range, for enumerated data types
                      -- only
WHEN value1 | value2 |... -- value1 or value2 or ...
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

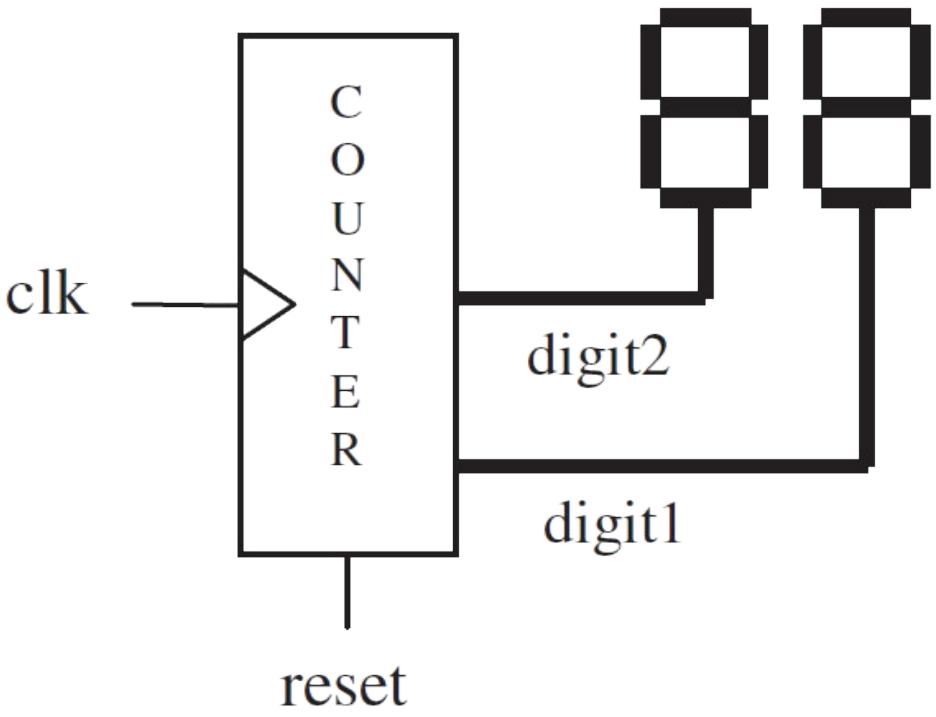
Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

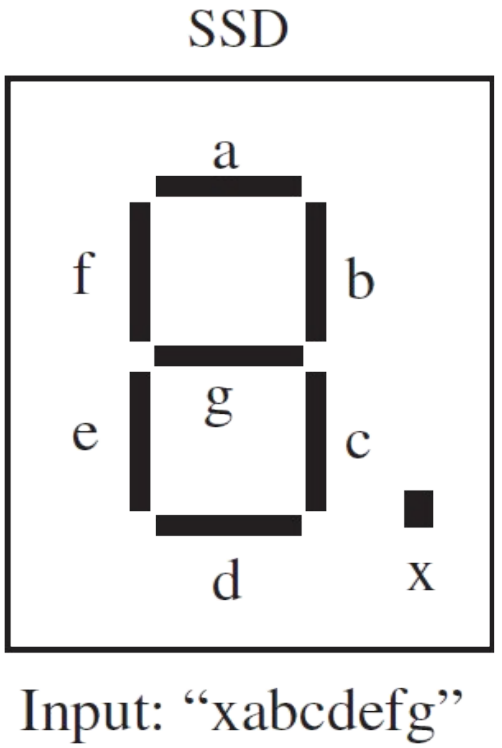
UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Przerzutnik z asynchronicznym resetem

```
1  ---------------------------------------------
2  LIBRARY ieee;                 -- Unnecessary declaration,
3                                -- because
4  USE ieee.std_logic_1164.all; -- BIT was used instead of
5                                -- STD_LOGIC
6  ---------------------------------------------
7  ENTITY dff IS
8     PORT (d, clk, rst: IN BIT;
9           q: OUT BIT);
10 END dff;
11 ---------------------------------------------
12 ARCHITECTURE dff3 OF dff IS
13 BEGIN
14    PROCESS (clk, rst)
15    BEGIN
16       CASE rst IS
17          WHEN '1' => q<='0';
18          WHEN '0' =>
19             IF (clk'EVENT AND clk='1') THEN
20                q <= d;
21             END IF;
22          WHEN OTHERS => NULL; -- Unnecessary, rst is of type
23                               -- BIT
24       END CASE;
25    END PROCESS;
26 END dff3;
27 ---------------------------------------------
```

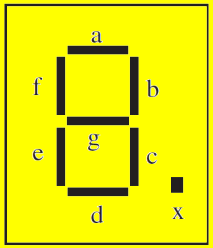
Dwucyfrowy licznik z wyjściem siedmiosegmentowym





Dwucyfrowy licznik z wyjściem siedmiosegmentowym

```
1  ------------------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------------------
5  ENTITY counter IS
6     PORT (clk, reset : IN STD_LOGIC;
7           digit1, digit2
             : OUT STD_LOGIC_VECTOR (6 DOWNTO 0));
8  END counter;
9  ------------------------------------------------
10 ARCHITECTURE counter OF counter IS
11 BEGIN
12       PROCESS(clk, reset)
13          VARIABLE temp1: INTEGER RANGE 0 TO 10;
14          VARIABLE temp2: INTEGER RANGE 0 TO 10;
15       BEGIN
16       ---- counter: ---------------------
17       IF (reset='1') THEN
18          temp1 := 0;
19          temp2 := 0;
20       ELSIF (clk'EVENT AND clk='1') THEN
21          temp1 := temp1 + 1;
22          IF (temp1=10) THEN
23             temp1 := 0;
24             temp2 := temp2 + 1;
25             IF (temp2=10) THEN
26                temp2 := 0;
27             END IF;
28          END IF;
29       END IF;
```



Input: "xabcdefg"

```
30       ---- BCD to SSD conversion: --------
31       CASE temp1 IS
32          WHEN 0 => digit1 <= "1111110"; --7E
33          WHEN 1 => digit1 <= "0110000"; --30
34          WHEN 2 => digit1 <= "1101101"; --6D
35          WHEN 3 => digit1 <= "1111001"; --79
36          WHEN 4 => digit1 <= "0110011"; --33
37          WHEN 5 => digit1 <= "1011011"; --5B
38          WHEN 6 => digit1 <= "1011111"; --5F
39          WHEN 7 => digit1 <= "1110000"; --70
40          WHEN 8 => digit1 <= "1111111"; --7F
41          WHEN 9 => digit1 <= "1111011"; --7B
42          WHEN OTHERS => NULL;
43       END CASE;
44       CASE temp2 IS
45          WHEN 0 => digit2 <= "1111110"; --7E
46          WHEN 1 => digit2 <= "0110000"; --30
47          WHEN 2 => digit2 <= "1101101"; --6D
48          WHEN 3 => digit2 <= "1111001"; --79
49          WHEN 4 => digit2 <= "0110011"; --33
50          WHEN 5 => digit2 <= "1011011"; --5B
51          WHEN 6 => digit2 <= "1011111"; --5F
52          WHEN 7 => digit2 <= "1110000"; --70
53          WHEN 8 => digit2 <= "1111111"; --7F
54          WHEN 9 => digit2 <= "1111011"; --7B
55          WHEN OTHERS => NULL;
56       END CASE;
57    END PROCESS;
58 END counter;
59 ------------------------------------------------
```

LOOP

- Konstrukcja LOOP jest użyteczna, jeżeli chcemy, aby pewien fragment kodu został powtórzony wielokrotnie
- Konstrukcja LOOP może być używana tylko w kodzie sekwencyjnym, a więc tylko wewnątrz procesu, funkcji lub procedury

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Formy konstrukcji LOOP

- Instrukcja LOOP może przyjąć kilka form:
  - FOR/LOOP: pętla powtarzana daną liczbę razy

```
[label:] FOR identifier IN range LOOP
   (sequential statements)
END LOOP [label];
```

  - WHILE/LOOP: pętla powtarzana, gdy warunek jest spełniony

```
[label:] WHILE condition LOOP
   (sequential statements)
END LOOP [label];
```

  - EXIT: instrukcja zakończenia pętli

```
EXIT [label] [WHEN condition];
```

  - NEXT: instrukcja przejścia do kolejnej iteracji

```
NEXT [loop_label] [WHEN condition];
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Przykłady LOOP

```
FOR i IN 0 TO 5 LOOP
   x(i) <= enable AND w(i+2);
   y(0, i) <= w(i);
END LOOP;
```

- Pętla będzie powtarzana dla i zmieniającego się od 0 do 5 - czyli 6 razy
- Oba limity zakresu muszą być statyczne, żeby konstrukcję dało się zsyntezować

```
WHILE (i < 10) LOOP
   WAIT UNTIL clk'EVENT AND clk='1';
   (other statements)
END LOOP;
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Przykłady LOOP

```
FOR i IN data'RANGE LOOP
   CASE data(i) IS
      WHEN '0' => count:=count+1;
      WHEN OTHERS => EXIT;
   END CASE;
END LOOP;
```

```
FOR i IN 0 TO 15 LOOP
   NEXT WHEN i=skip; -- jumps to next iteration
   (...)
END LOOP;
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

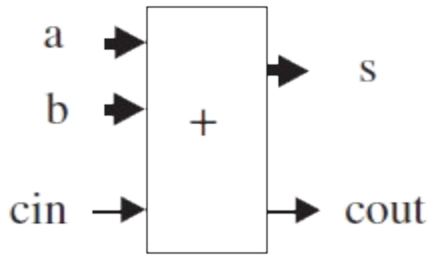
Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

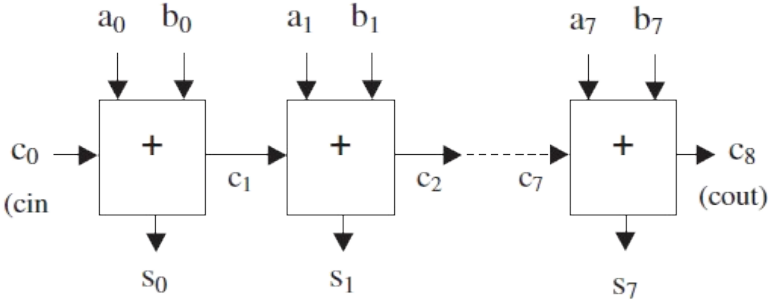
Przykład: sumator

```
1  ----- Solution 1: Generic, with VECTORS --------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ---------------------------------------------
5  ENTITY adder IS
6     GENERIC (length : INTEGER := 8);
7     PORT ( a, b: IN STD_LOGIC_VECTOR (length-1 DOWNTO 0);
8            cin: IN STD_LOGIC;
9            s: OUT STD_LOGIC_VECTOR (length-1 DOWNTO 0);
10           cout: OUT STD_LOGIC);
11 END adder;
12 ---------------------------------------------
13 ARCHITECTURE adder OF adder IS
14 BEGIN
15    PROCESS (a, b, cin)
16       VARIABLE carry : STD_LOGIC_VECTOR (length DOWNTO 0);
17    BEGIN
18       carry(0) := cin;
19       FOR i IN 0 TO length-1 LOOP
20          s(i) <= a(i) XOR b(i) XOR carry(i);
21          carry(i+1) := (a(i) AND b(i)) OR (a(i) AND
22                        carry(i)) OR (b(i) AND carry(i));
23       END LOOP;
24       cout <= carry(length);
25    END PROCESS;
26 END adder;
27 ---------------------------------------------
```

Top level:

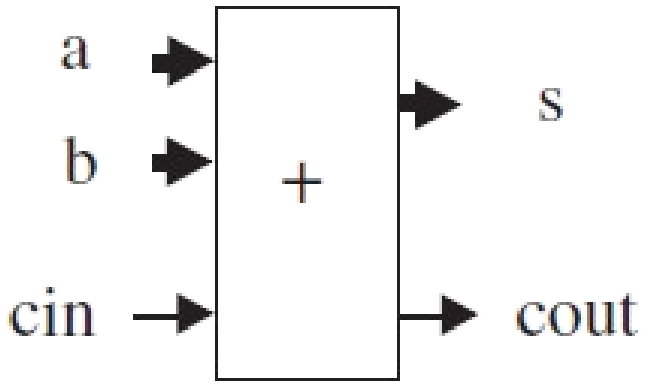


One level below top:



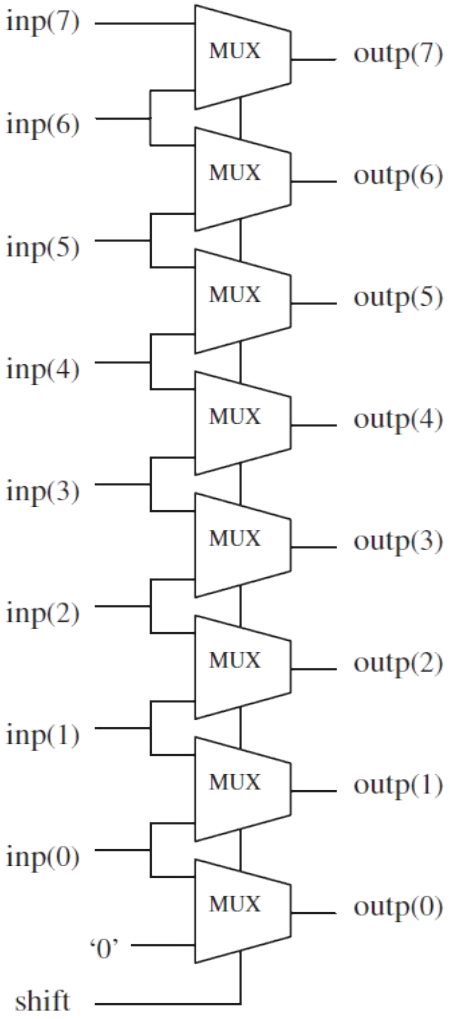
Przykład: sumator

```
1  ---- Solution 2: non-generic, with INTEGERS ----
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ---------------------------------------------
5  ENTITY adder IS
6     PORT ( a, b: IN INTEGER RANGE 0 TO 255;
7            c0: IN STD_LOGIC;
8            s: OUT INTEGER RANGE 0 TO 255;
9            c8: OUT STD_LOGIC);
10 END adder;
11 ---------------------------------------------
12 ARCHITECTURE adder OF adder IS
13 BEGIN
14    PROCESS (a, b, c0)
15       VARIABLE temp : INTEGER RANGE 0 TO 511;
16    BEGIN
17       IF (c0='1') THEN temp:=1;
18       ELSE temp:=0;
19       END IF;
20       temp := a + b + temp;
21       IF (temp > 255) THEN
22          c8 <= '1';
23          temp := temp - 256;
24       ELSE c8 <= '0';
25       END IF;
26       s <= temp;
27   END PROCESS;
28 END adder;
29 ---------------------------------------------
```



Przykład: przesuwnik bitów

```
1  ----------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ----------------------------------------
5  ENTITY barrel IS
6     GENERIC (n: INTEGER := 8);
7     PORT ( inp: IN STD_LOGIC_VECTOR (n-1 DOWNTO 0);
8            shift: IN INTEGER RANGE 0 TO 1;
9            outp: OUT STD_LOGIC_VECTOR (n-1 DOWNTO 0));
10  END barrel;
11 ----------------------------------------
12 ARCHITECTURE RTL OF barrel IS
13    BEGIN
14    PROCESS (inp, shift)
15    BEGIN
16       IF (shift=0) THEN
17          outp <= inp;
18       ELSE
19          outp(0) <= '0';
20          FOR i IN 1 TO inp'HIGH LOOP
21             outp(i) <= inp(i-1);
22          END LOOP;
23       END IF;
24    END PROCESS;
25 END RTL;
26 ----------------------------------------
```



Przykład: licznik wiodących zer

```
1  ----------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ----------------------------------------
5  ENTITY LeadingZeros IS
6     PORT ( data: IN STD_LOGIC_VECTOR (7 DOWNTO 0);
7            zeros: OUT INTEGER RANGE 0 TO 8);
8  END LeadingZeros;
9  ----------------------------------------
10 ARCHITECTURE behavior OF LeadingZeros IS
11 BEGIN
12    PROCESS (data)
13       VARIABLE count: INTEGER RANGE 0 TO 8;
14    BEGIN
15       count := 0;
16       FOR i IN data'RANGE LOOP
17          CASE data(i) IS
18             WHEN '0' => count := count + 1;
19             WHEN OTHERS => EXIT;
20          END CASE;
21       END LOOP;
22       zeros <= count;
23    END PROCESS;
24 END behavior;
25 ----------------------------------------
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

CASE a IF

- Stosowanie IF może prowadzić w niektórych przypadkach do powstania dekodera priorytetowego
- Nie jest to możliwe w przypadku CASE
- Oba rozwiązania poniżej powinny wygenerować multiplekser

```
---- With IF: --------------
IF (sel="00") THEN x<=a;
ELSIF (sel="01") THEN x<=b;
ELSIF (sel="10") THEN x<=c;
ELSE x<=d;
---- With CASE: ------------
CASE sel IS
   WHEN "00" => x<=a;
   WHEN "01" => x<=b;
   WHEN "10" => x<=c;
   WHEN OTHERS => x<=d;
END CASE;
---------------------------
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

CASE i WHEN

- CASE i WHEN są bardzo podobne
- CASE jest stosowana w kodzie sekwencyjnym, WHEN we współbieżnym

```
---- With WHEN: ----------------
WITH sel SELECT
x <= a WHEN "000",
     b WHEN "001",
     c WHEN "010",
     UNAFFECTED WHEN OTHERS;
---- With CASE: ----------------
CASE sel IS
   WHEN "000" => x<=a;
   WHEN "001" => x<=b;
   WHEN "010" => x<=c;
   WHEN OTHERS => NULL;
END CASE;
-------------------------------
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Niepoprawne użycie sygnału zegarowego

- Zwykle nie da się zsyntezować kodów które zawierają przypisania do tego samego sygnału na obu zboczach zegara (narastającym i opadającym), jak w przykładzie poniżej:

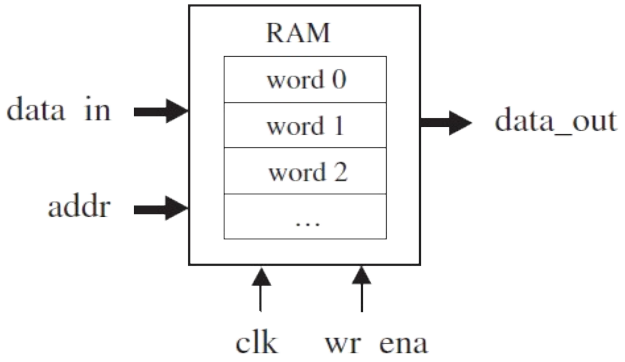
```
PROCESS (clk)
BEGIN
   IF(clk'EVENT AND clk='1') THEN
      counter <= counter + 1;
   ELSIF(clk'EVENT AND clk='0') THEN
      counter <= counter + 1;
   END IF;
   ...
END PROCESS;
```

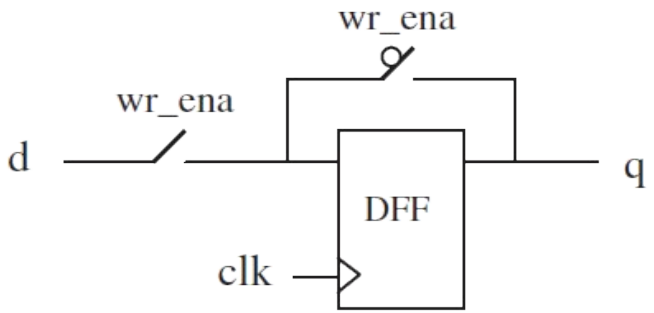
- Można natomiast zastosować dwa osobne liczniki:

```
----------------------
PROCESS (clk)
BEGIN
   IF(clk'EVENT AND clk='1') THEN
      x <= d;
   END IF;
END PROCESS;
----------------------
PROCESS (clk)
BEGIN
   IF(clk'EVENT AND clk='0') THEN
      y <= d;
   END IF;
END PROCESS;
----------------------
```

Przykład: RAM

```
1  ------------------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------------------
5  ENTITY ram IS
6     GENERIC ( bits: INTEGER := 8; -- # of bits per word
7               words: INTEGER := 16); -- # of words in the memory
8     PORT ( wr_ena, clk: IN STD_LOGIC;
9            addr: IN INTEGER RANGE 0 TO words-1;
10           data_in: IN STD_LOGIC_VECTOR (bits-1 DOWNTO 0);
11           data_out: OUT STD_LOGIC_VECTOR (bits-1 DOWNTO 0));
12 END ram;
13 ------------------------------------------------
14 ARCHITECTURE ram OF ram IS
15    TYPE vector_array IS ARRAY (0 TO words-1) OF
16              STD_LOGIC_VECTOR (bits-1 DOWNTO 0);
17    SIGNAL memory: vector_array;
18 BEGIN
19    PROCESS (clk, wr_ena)
20    BEGIN
21       IF (wr_ena='1') THEN
22          IF (clk'EVENT AND clk='1') THEN
23             memory(addr) <= data_in;
24          END IF;
25       END IF;
26    END PROCESS;
27    data_out <= memory(addr);
28 END ram;
29 ------------------------------------------------
```





Kod sekwencyjny a układy kombinacyjne

- Kod sekwencyjny może być użyty do implementacji zarówno układów sekwencyjnych, jak i kombinacyjnych
- Jeżeli chcemy otrzymać kod kombinacyjny, muszą być spełnione dwa warunki
  - Wszystkie sygnały odczytywane w procesie muszą się pojawić na liście wrażliwościowej
  - Wszystkie kombinacje wejść muszą być uwzględnione w kodzie dla każdego z wyjść
- Jeżeli nie spełnimy tych warunków, mogą pojawić się w układzie przerzutniki wyzwalane poziomem

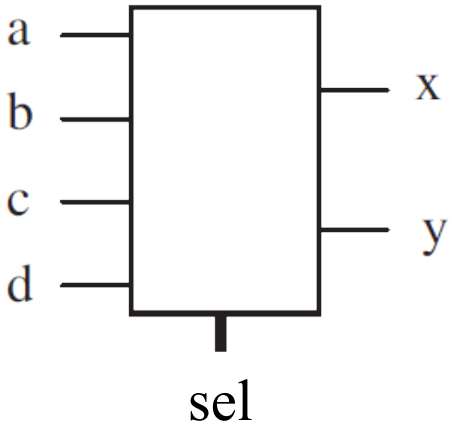
KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

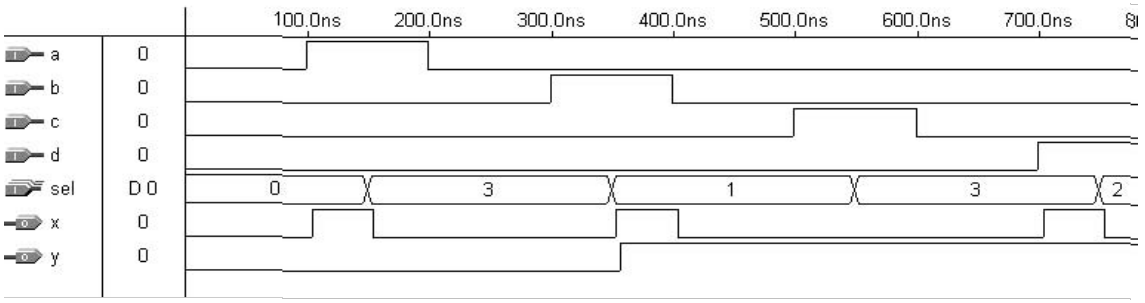
UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Przykład: niepoprawny projekt kombinacyjny

```
1  ------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------
5  ENTITY example IS
6     PORT (a, b, c, d: IN STD_LOGIC;
7           sel: IN INTEGER RANGE 0 TO 3;
8           x, y: OUT STD_LOGIC);
9  END example;
10 ------------------------------------
11 ARCHITECTURE example OF example IS
12 BEGIN
13    PROCESS (a, b, c, d, sel)
14    BEGIN
15       IF (sel=0) THEN
16          x<=a;
17          y<='0';
18       ELSIF (sel=1) THEN
19          x<=b;
20          y<='1';
21       ELSIF (sel=2) THEN
22          x<=c;
23       ELSE
24          x<=d;
25       END IF;
26    END PROCESS;
27 END example;
28 ------------------------------------
```



sel	x y
00	a 0
01	b 1
10	c
11	d

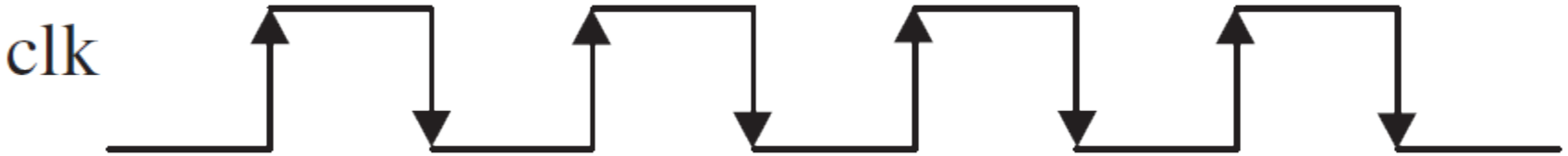


Wersja poprawiona

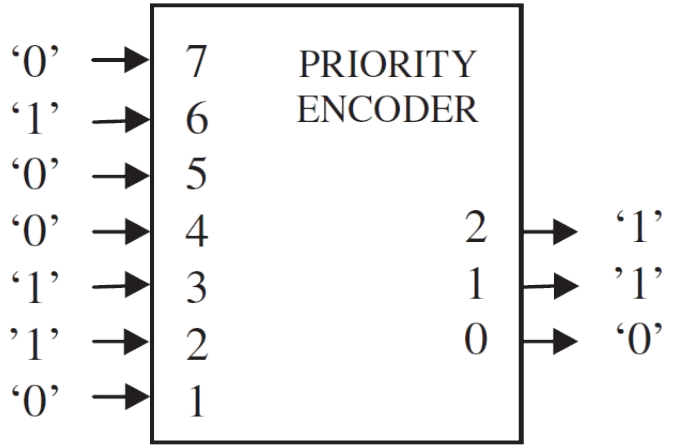
```
1  ------------------------------------
2  LIBRARY ieee;
3  USE ieee.std_logic_1164.all;
4  ------------------------------------
5  ENTITY example IS
6     PORT (a, b, c, d: IN STD_LOGIC;
7           sel: IN INTEGER RANGE 0 TO 3;
8           x, y: OUT STD_LOGIC);
9  END example;
10 ------------------------------------
11 ARCHITECTURE example OF example IS
12 BEGIN
13    PROCESS (a, b, c, d, sel)
14    BEGIN
15       IF (sel=0) THEN
16          x<=a;
17          y<='0';
18       ELSIF (sel=1) THEN
19          x<=b;
20          y<='1';
21       ELSIF (sel=2) THEN
22          x<=c;
23          y<='-';
24       ELSE
25          x<=d;
26          y<='-';
27       END IF;
28    END PROCESS;
29 END example;
30 ------------------------------------
```

Problemy

- Zaprojektować układ zliczający liczbę zboczy zegara (narastających + opadających)



- Zaprojektować koder priorytetowy. Na wyjściu uzyskujemy numer najwyższego aktywnego bitu wejściowego. Stan wyjść "000" oznacza, że żadne z wejść nie jest aktywne.



Problemy

- Który kod opisujący przerzutnik wyzwalany zboczem jest poprawny:

```
-------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-------------------------------------
ENTITY dff IS
   PORT ( d, clk, rst: IN BIT;
          q: OUT BIT);
END dff;
----- Solution 1 --------------------
ARCHITECTURE arch1 OF dff IS
BEGIN
   PROCESS (clk, rst)
   BEGIN
      IF (rst='1') THEN
         q <= '0';
      ELSIF (clk'EVENT AND clk='1') THEN
         q <= d;
      END IF;
   END PROCESS;
END arch1;
```

```
----- Solution 2 --------------------
ARCHITECTURE arch2 OF dff IS
BEGIN
   PROCESS (clk)
   BEGIN
      IF (rst='1') THEN
         q <= '0';
      ELSIF (clk'EVENT AND clk='1') THEN
         q <= d;
      END IF;
   END PROCESS;
END arch2;
```

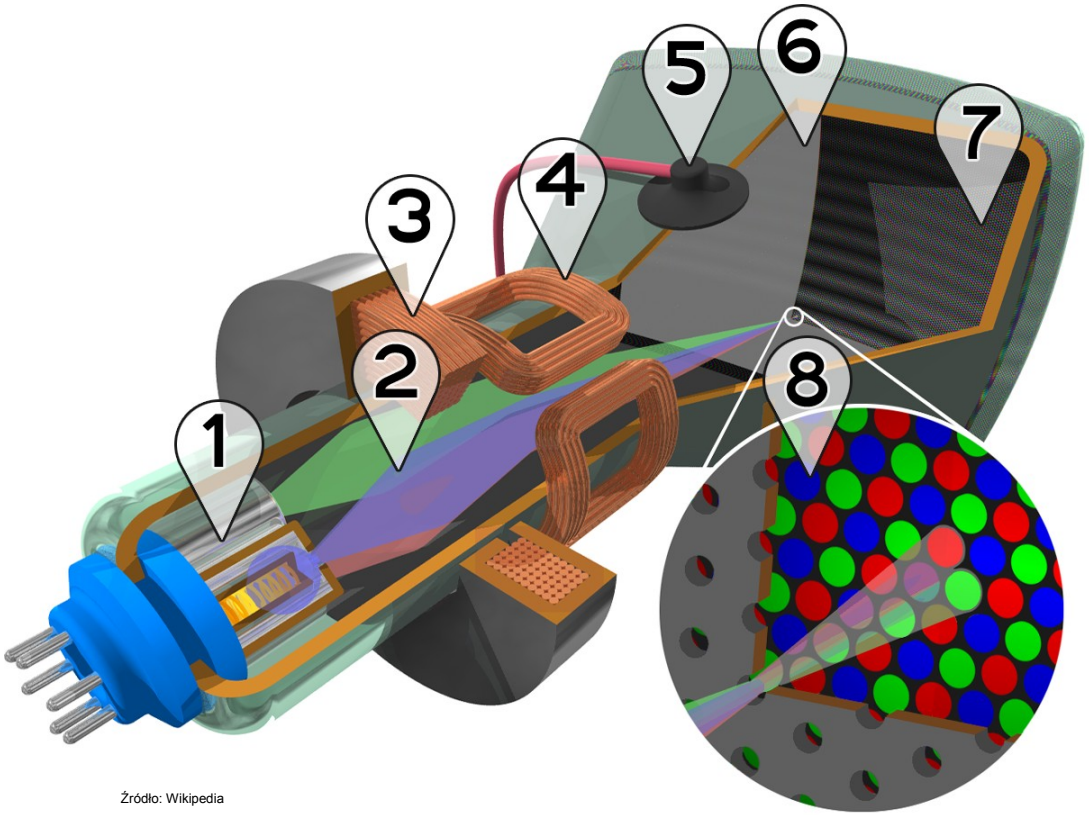
Problemy

- Który kod opisujący przerzutnik wyzwalany zboczem jest poprawny:

```
-------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
-------------------------------------
ENTITY dff IS
   PORT ( d, clk, rst: IN BIT;
          q: OUT BIT);
END dff;
----- Solution 1 --------------------
ARCHITECTURE arch1 OF dff IS
BEGIN
   PROCESS (clk, rst)
   BEGIN
      IF (rst='1') THEN
         q <= '0';
      ELSIF (clk'EVENT AND clk='1') THEN
         q <= d;
      END IF;
   END PROCESS;
END arch1;
```

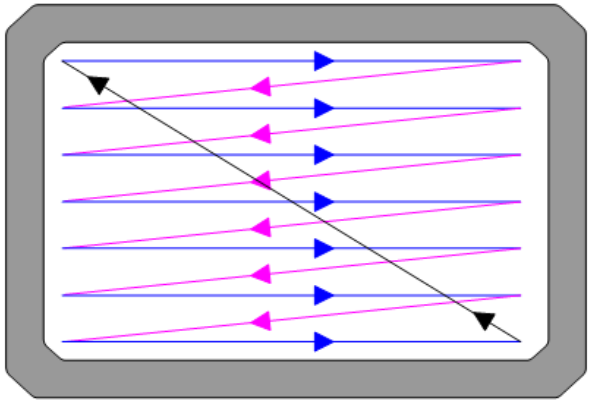
```
----- Solution 2 --------------------
ARCHITECTURE arch2 OF dff IS
BEGIN
   PROCESS (clk)
   BEGIN
      IF (rst='1') THEN
         q <= '0';
      ELSIF (clk'EVENT AND clk='1') THEN
         q <= d;
      END IF;
   END PROCESS;
END arch2;
```

Budowa kineskopu

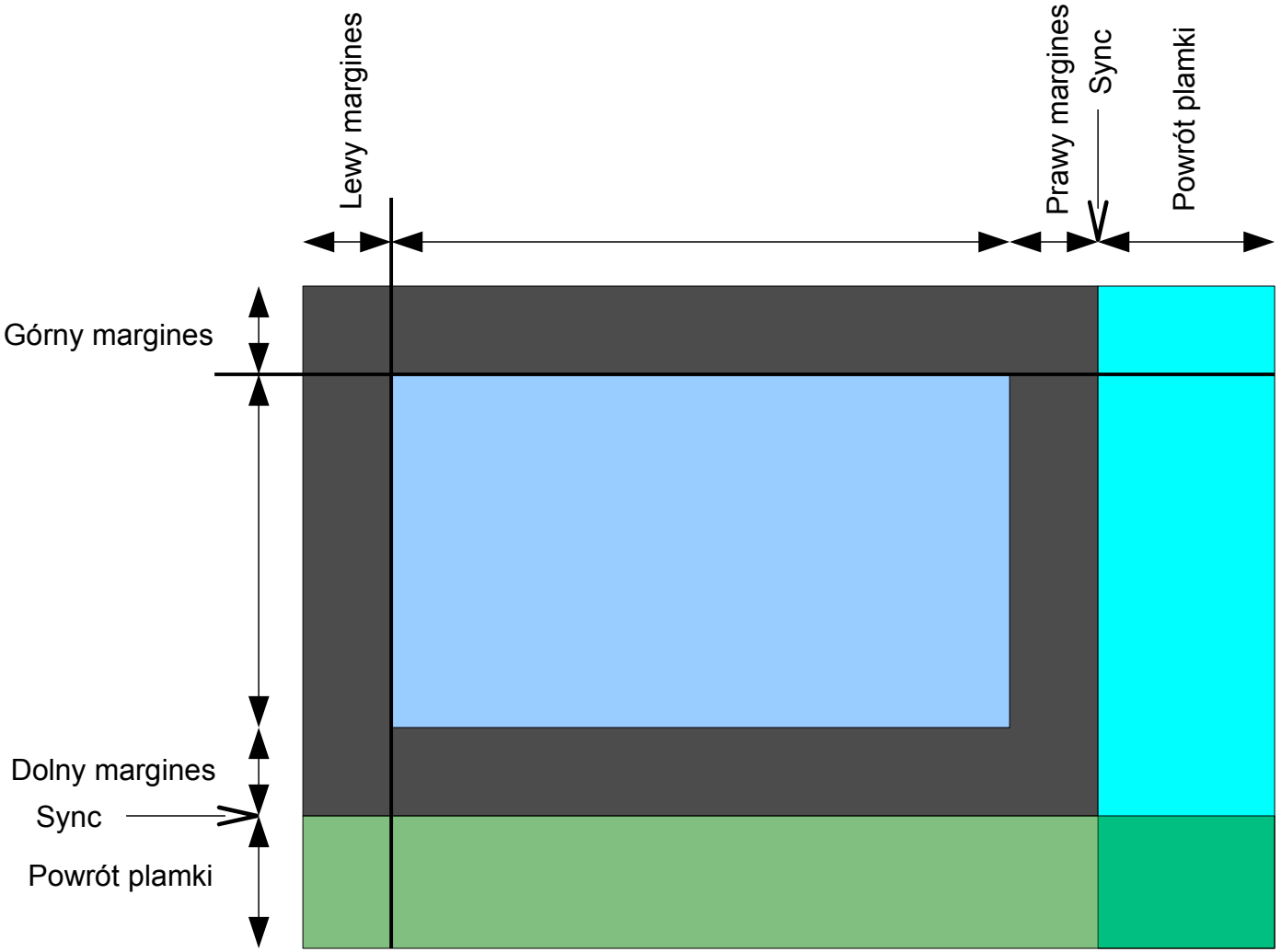


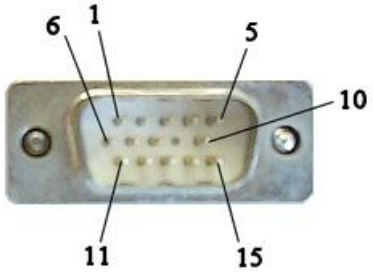
Źródło: Wikipedia

1 – trzy działa elektronowe (katoda)
2 – wiązki elektronów
3 – cewka ogniskująca
4 – cewki odchylające
5 – przyłącze anody
6 – maska separująca wiązki
7 – luminofor
8 – powiększenie fragmentu luminoforu



Zależności czasowe w siatce obrazowej





Pin #	Signal Name
1	Red
2	Green
3	Blue
4	No Connect
5	Ground
6	Ground
7	Ground
8	Ground
9	No Connect
10	Ground
11	No Connect
12	DDC DAT
13	Horizontal Synchronization
14	Vertical Synchronization
15	DDC Clock

Sterownik monitora CRT

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
USE IEEE.STD_LOGIC_ARITH.all;
USE IEEE.STD_LOGIC_UNSIGNED.all;

-- VGA Video Sync generation

ENTITY VGA_SYNC IS
   PORT(clock_25Mhz, red, green, blue: IN STD_LOGIC;
        red_out, green_out, blue_out,
        horiz_sync_out, vert_sync_out: OUT STD_LOGIC;
        pixel_row, pixel_column : OUT STD_LOGIC_VECTOR(9 DOWNTO 0));
END VGA_SYNC;
```

Sterownik monitora CRT (c.d.)

```
ARCHITECTURE a OF VGA_SYNC IS
  SIGNAL horiz_sync, vert_sync : STD_LOGIC;
  SIGNAL video_on, video_on_v, video_on_h : STD_LOGIC;
  SIGNAL h_count, v_count : STD_LOGIC_VECTOR(9 DOWNTO 0);
BEGIN
-- video_on is high only when RGB data is being displayed
  video_on <= video_on_H AND video_on_V;
  horiz_sync_out    <= horiz_sync;
  vert_sync_out     <= vert_sync;
  pixel_column <= h_count;
  pixel_row <= v_count;
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Sterownik monitora CRT (c.d.)

```
-- Generate Horizontal and Vertical Timing Signals for Video Signal
  PROCESS
  BEGIN
     WAIT UNTIL (clock_25Mhz'EVENT) AND (clock_25Mhz='1');
    -- H_count counts pixels (640 + extra time for sync signals)
    --
    --  Horiz_sync  ------------------------------------__________--------
    --  H_count       0                 640             659       755    799
    --

      IF (h_count = 799) THEN
         h_count <= "0000000000";
      ELSE
         h_count <= h_count + 1;
      END IF;

      -- Generate Horizontal Sync Signal using H_count
      if(h_count = 756) then
         horiz_sync <= '1';
      end if;
      if(h_count = 659) then
         horiz_sync <= '0';
      end if;
```

Sterownik monitora CRT (c.d.)

```
--V_count counts rows of pixels (480 + extra time for sync signals)
--
--  Vert_sync      ----------------------------------_______------------
--  V_count         0                            480    493-494          524
--
IF (v_count = 524) AND (h_count = 799) THEN
   v_count <= "0000000000";
ELSIF (h_count = 799) THEN
   v_count <= v_count + 1;
END IF;
-- Generate Vertical Sync Signal using V_count
if (v_count = 493) then
   vert_sync <= '0';
end if;
if (v_count = 495) then
   vert_sync <= '1';
end if;
-- Generate Video on Screen Signals for Pixel Data
if(h_count = 639) then
   video_on_h <= '0';
end if;
if (h_count = 799) then
  video_on_h <= '1';
  if(v_count = 479) then
     video_on_v <= '0';
  end if;
  if(v_count = 524) then
     video_on_v <= '1';
  end if;
end if;
```

Sterownik monitora CRT (c.d.)

```
      -- Put all video signals through DFFs to eliminate
      -- any logic delays that can cause a blurry image

      red_out <= red AND video_on;
      green_out <= green AND video_on;
      blue_out    <= blue AND video_on;

   END PROCESS;
END a;
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

„Układy reprogramowalne i SoC”
„Język VHDL (część 3)”

Prezentacja jest współfinansowana przez
Unię Europejską w ramach
Europejskiego Funduszu Społecznego w projekcie pt.

„Innowacyjna dydaktyka bez ograniczeń - zintegrowany rozwój Politechniki Łódzkiej - zarządzanie Uczelnią, nowoczesna oferta edukacyjna i wzmacniania zdolności do zatrudniania osób niepełnosprawnych”

Prezentacja dystrybuowana jest bezpłatnie