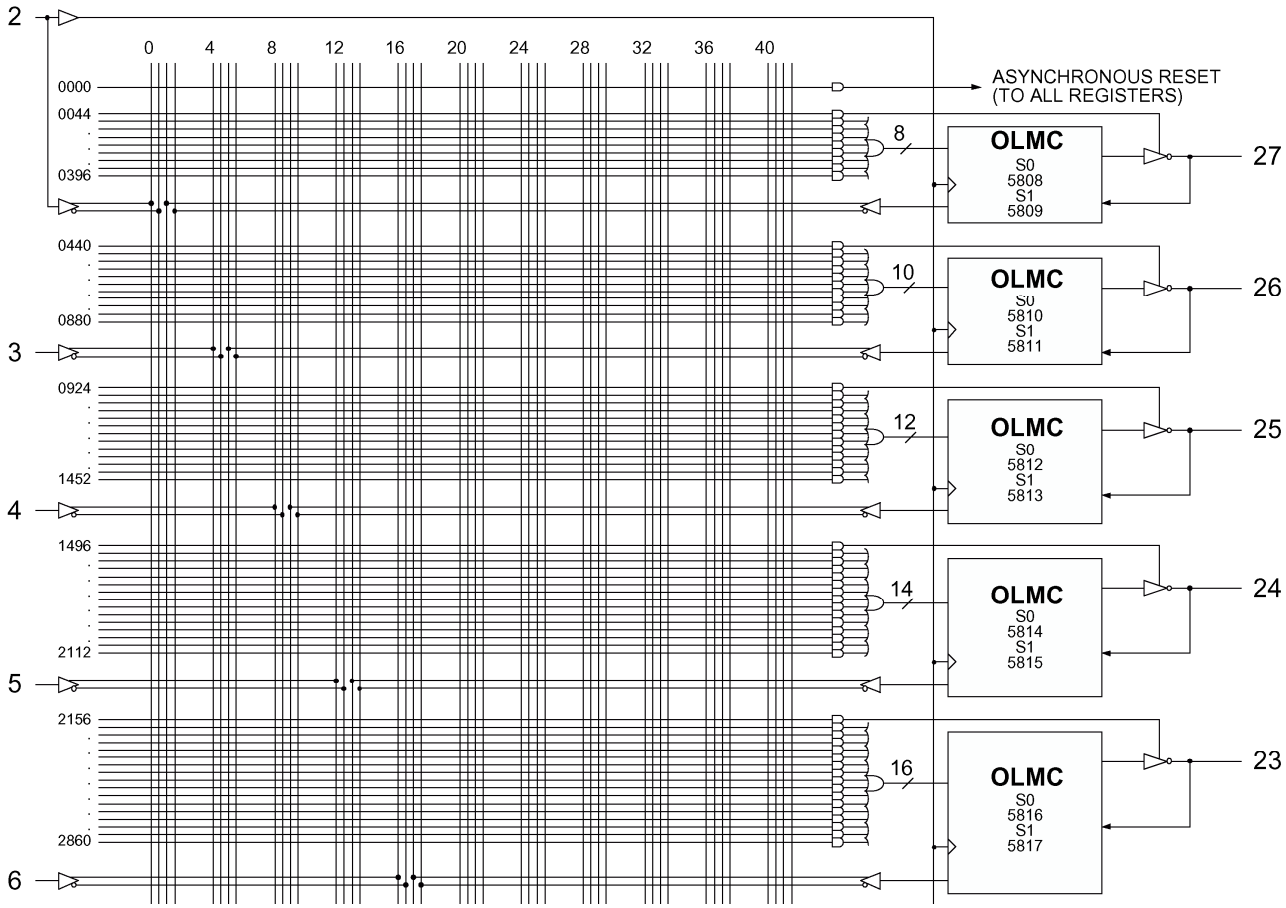
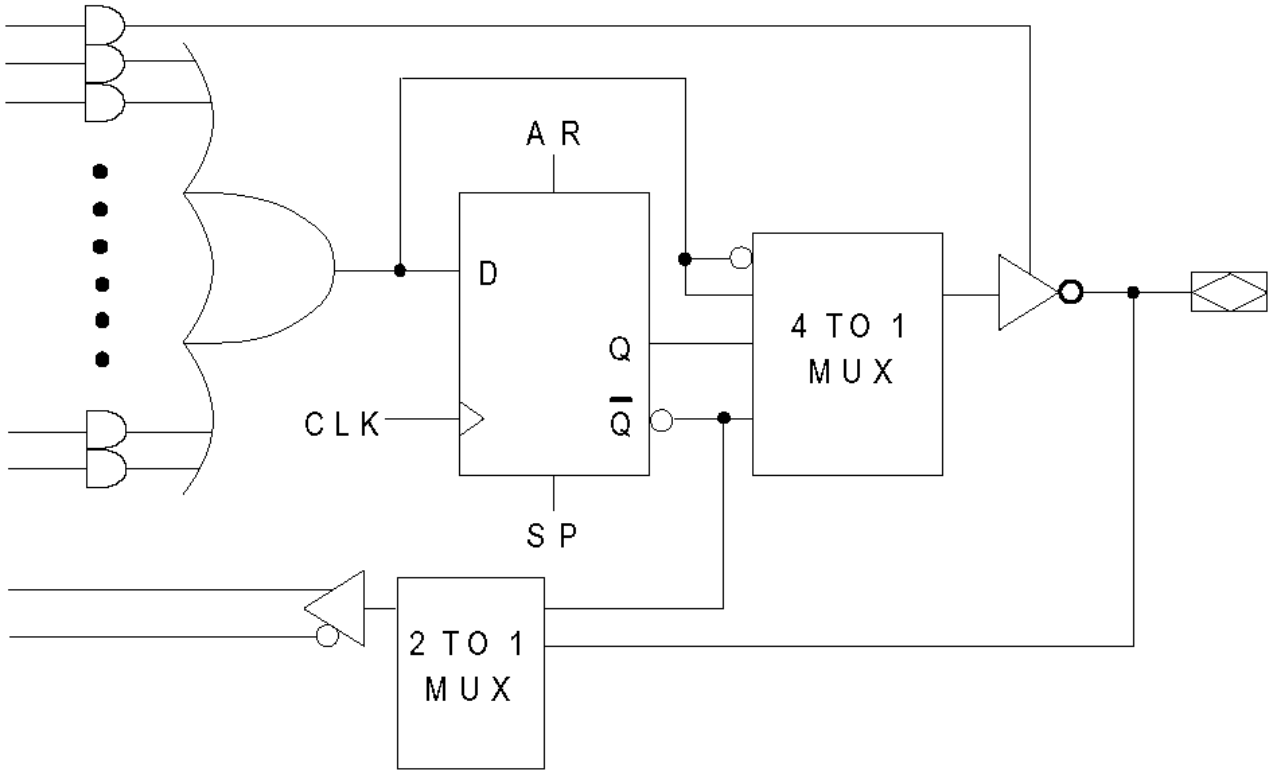
Układy programowalne ispGAL (Lattice Semiconductors)





ispGAL22V10 OUTPUT LOGIC MACROCELL (OLMC)

Układy CPLD (Complex Programmable Logic Devices)

Seria XC95xx (XL, XV) XILINX

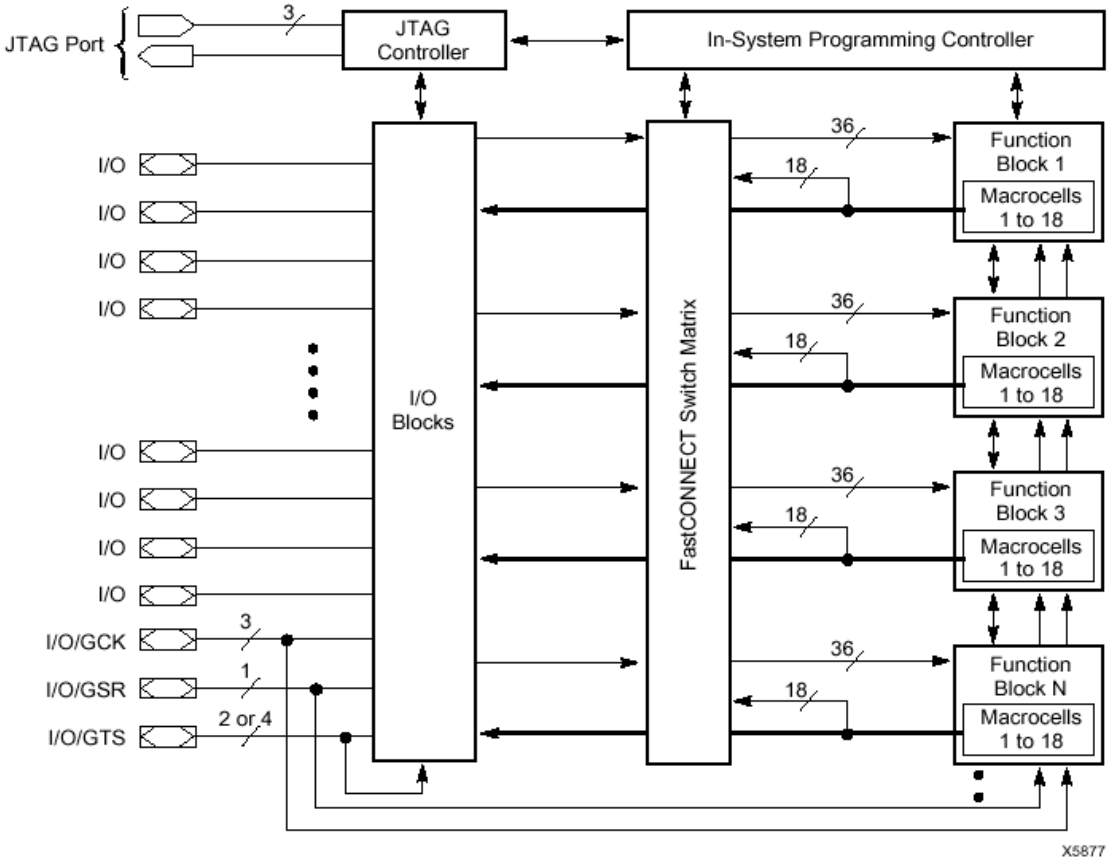


Figure 1: XC9500 Architecture

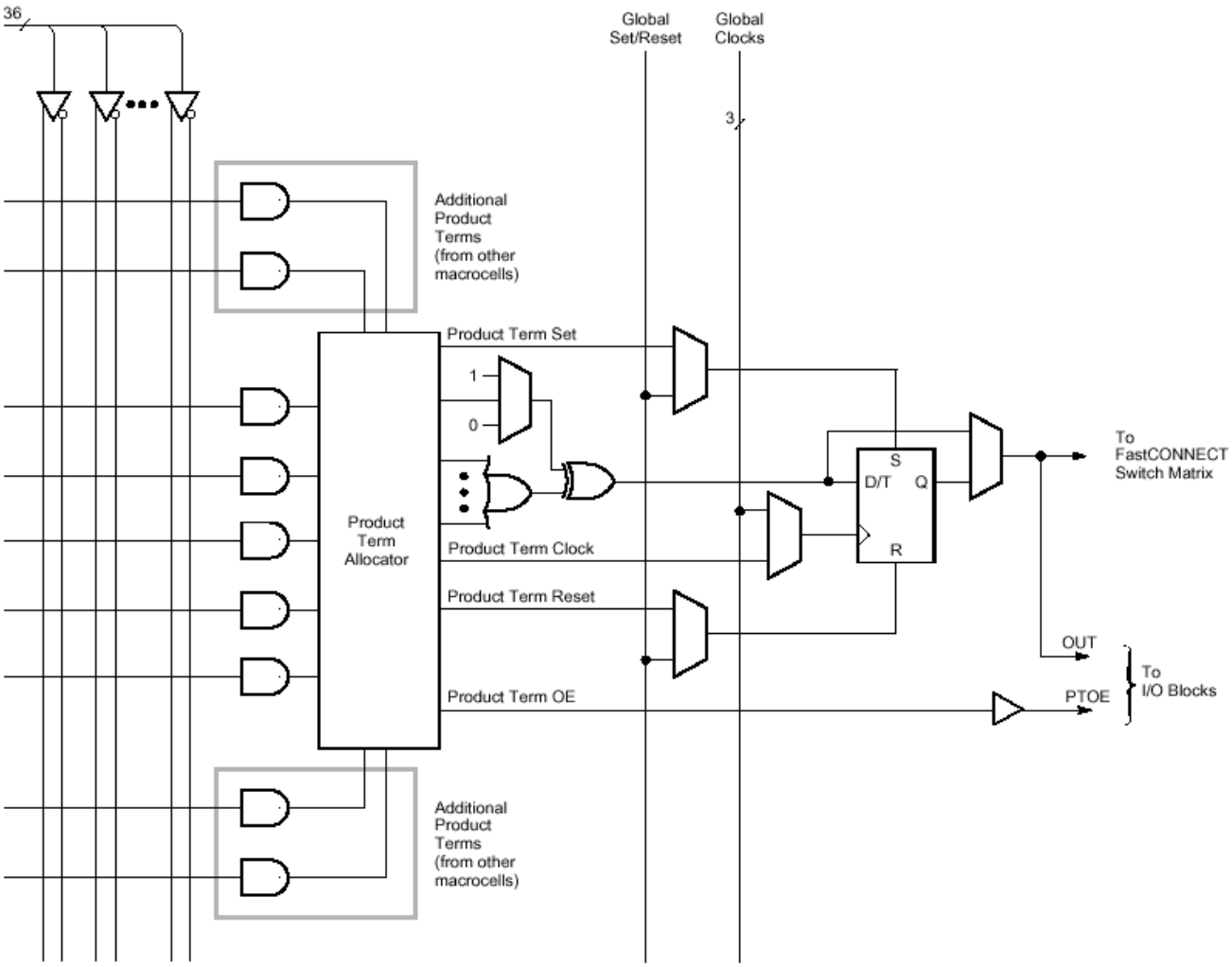
Note: Function Block outputs (indicated by the bold line) drive the I/O Blocks directly.

Table 1: XC9500 Device Family

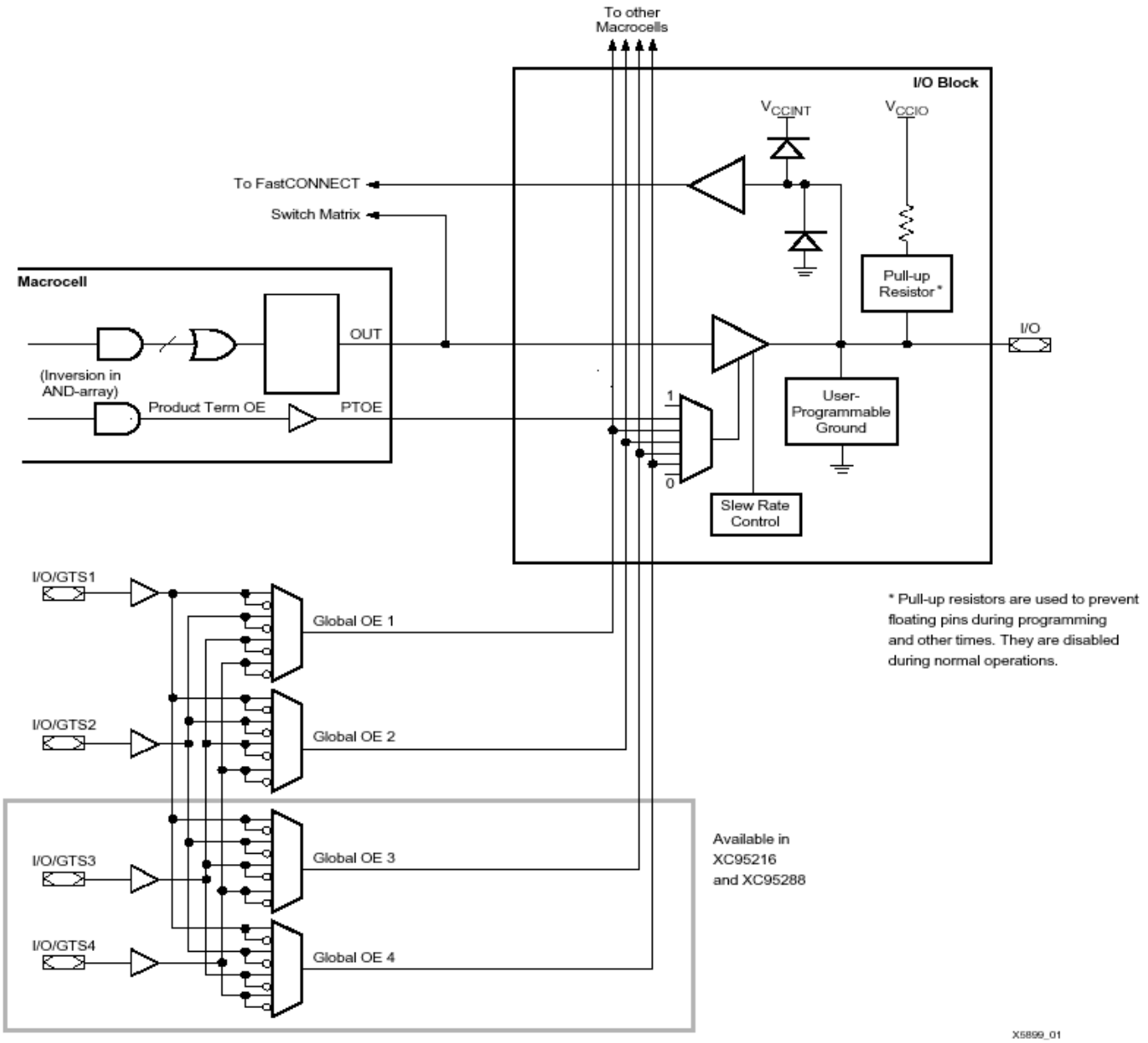
	XC9536	XC9572	XC95108	XC95144	XC95216	XC95288
Macrocells	36	72	108	144	216	288
Usable Gates	800	1,600	2,400	3,200	4,800	6,400
Registers	36	72	108	144	216	288
t_{PD} (ns)	5	7.5	7.5	7.5	10	10
t_{SU} (ns)	3.5	4.5	4.5	4.5	6.0	6.0
t_{CO} (ns)	4.0	4.5	4.5	4.5	6.0	6.0
f_{CNT} (MHz)	100	125	125	125	111.1	111.1
f_{SYSTEM} (MHz)	100	83.3	83.3	83.3	66.7	66.7

Note: f_{CNT} = Operating frequency for 16-bit counters
f_{SYSTEM} = Internal operating frequency for general purpose system designs spanning multiple FBs.

Schemat wewnętrzny makroceli



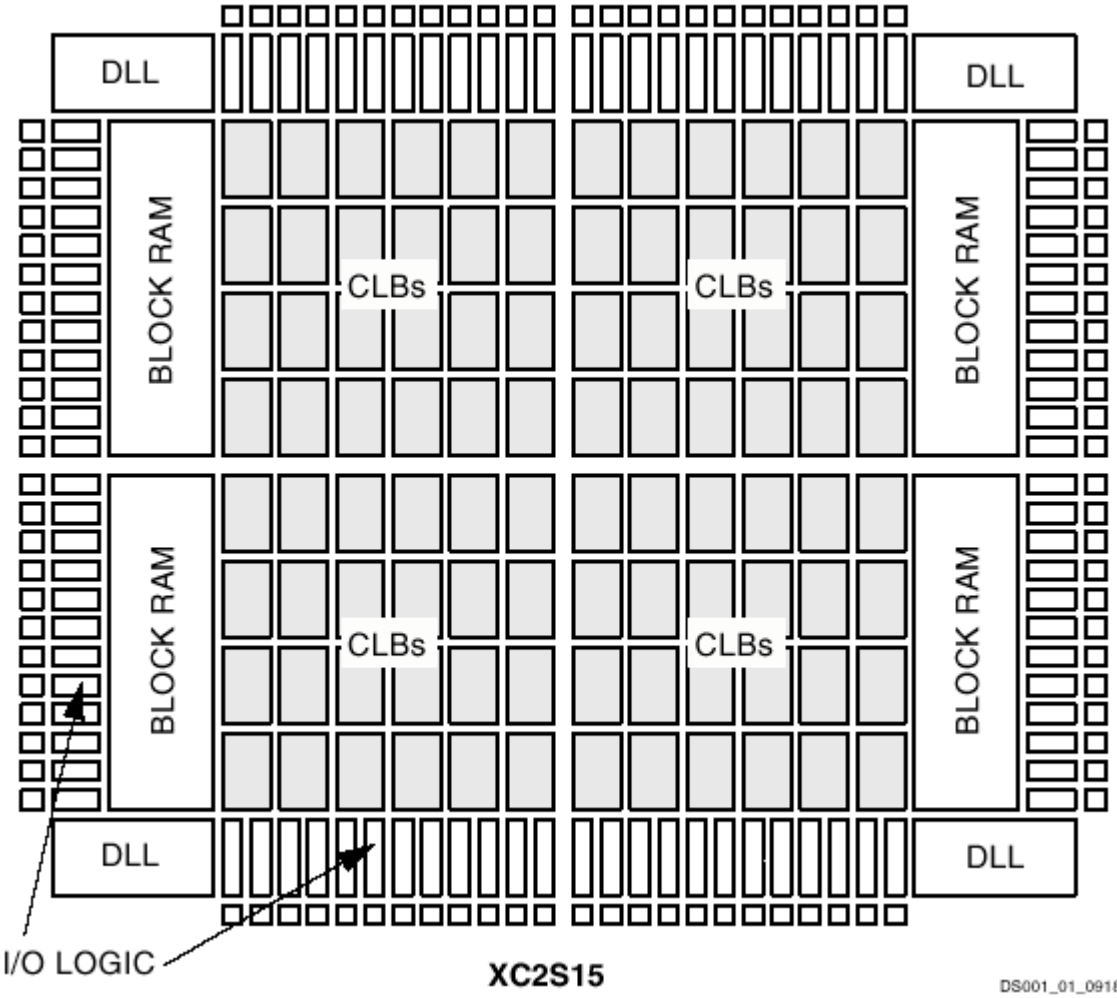
Schemat bloku wejścia/wyjścia (I/O)



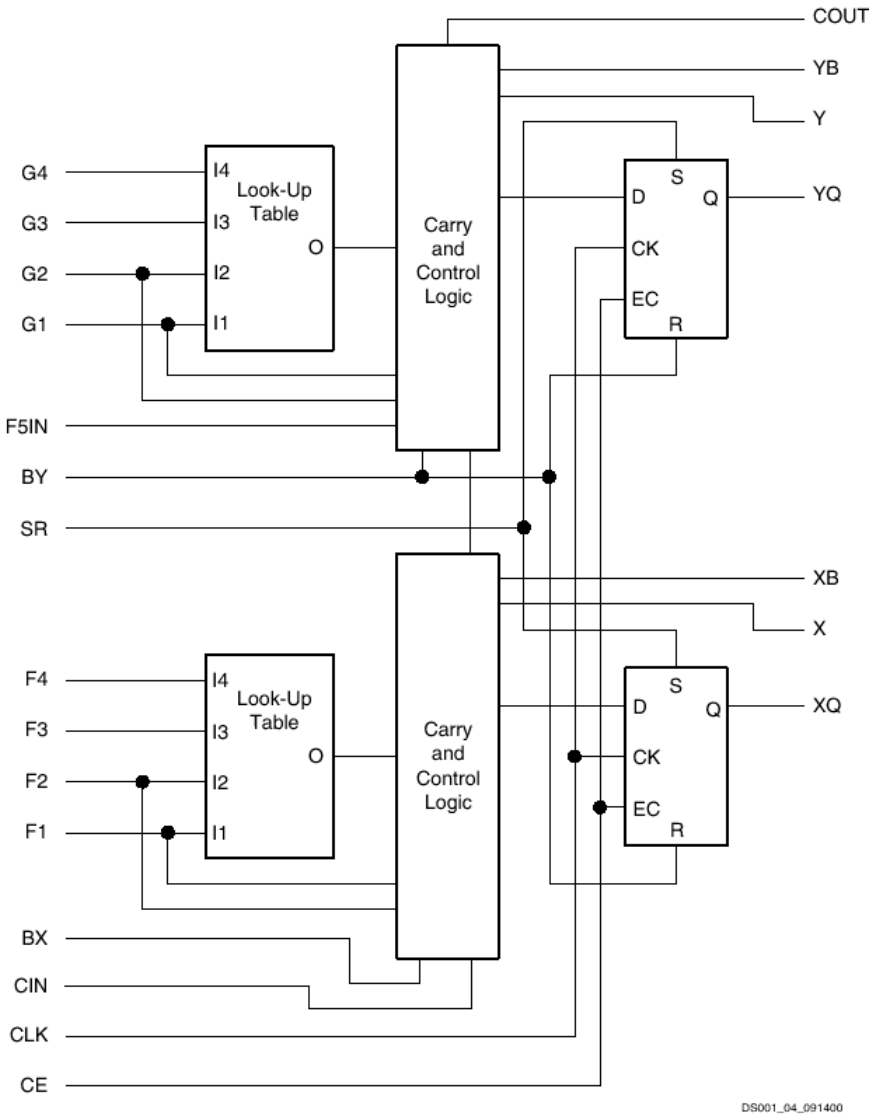
Układy FPGA (Field Programmable Gate Arrays)

Seria SPARTAN2 XILINX

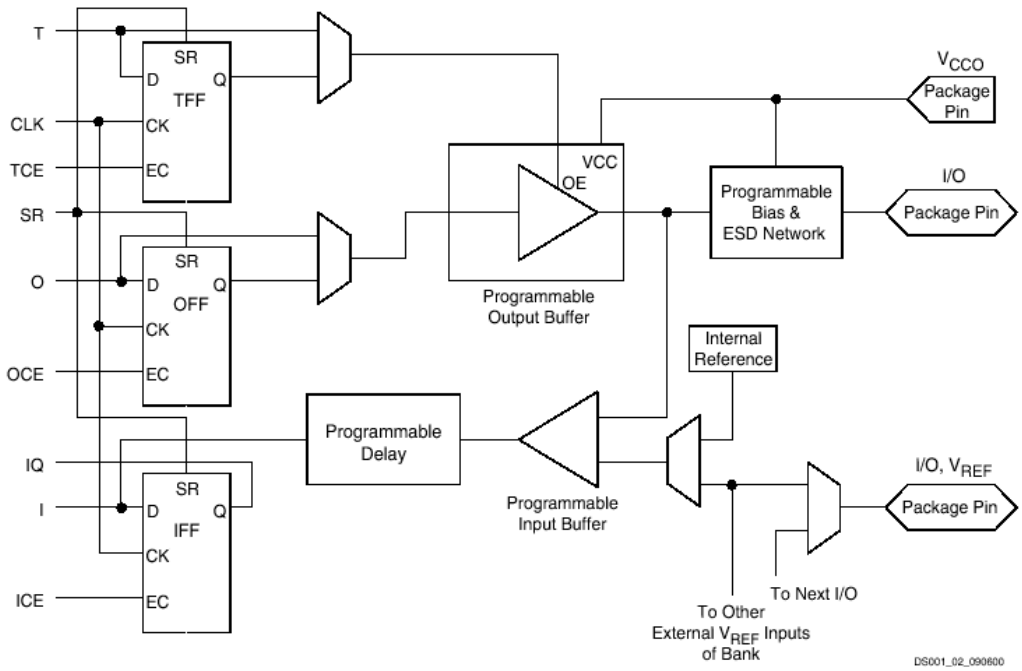
Device	Logic Cells	System Gates (Logic and RAM)	CLB Array (R x C)	Total CLBs	Maximum Available User I/O[1]	Total Distributed RAM Bits	Total Block RAM Bits
XC2S15	432	15,000	8 x 12	96	86	6,144	16K
XC2S30	972	30,000	12 x 18	216	132	13,824	24K
XC2S50	1,728	50,000	16 x 24	384	176	24,576	32K
XC2S100	2,700	100,000	20 x 30	600	196	38,400	40K
XC2S150	3,888	150,000	24 x 36	864	260	55,296	48K
XC2S200	5,292	200,000	28 x 42	1,176	284	75,264	56K



Budowa CLB

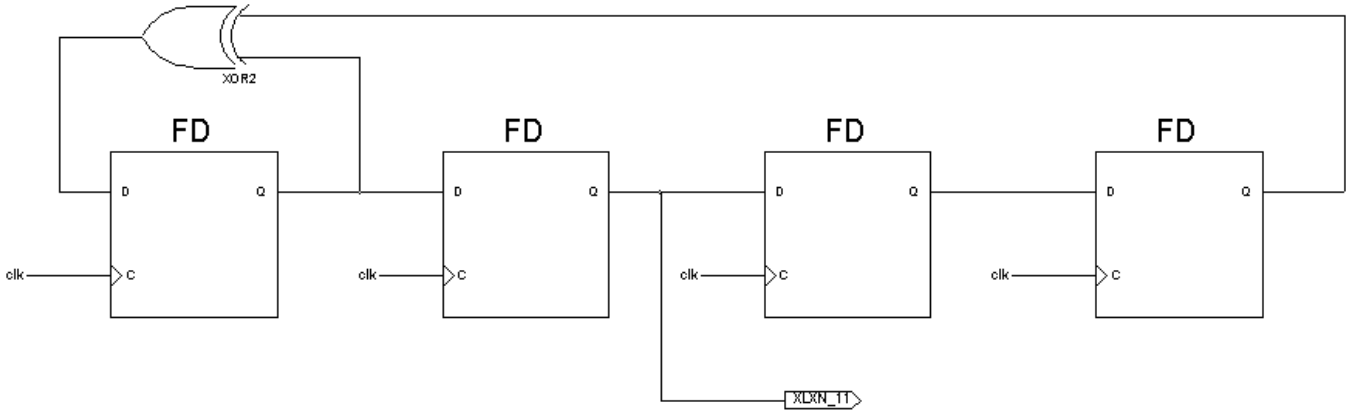


Budowa bloku I/O

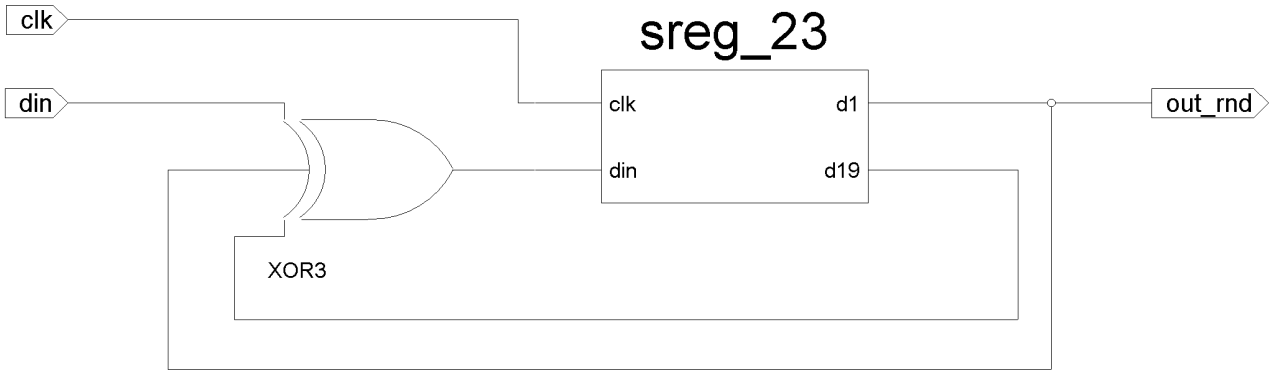


Metody opisu układów logicznych

1. Schemat



2. Schemat hierarchiczny



3. Opis w formie kodu HDL (Hardware Description Language)

ABEL

VHDL

VERILOG

Przykładowy kod w języku VHDL

(23-bitowy rejestr przesuwny)

pliki nagłówkowe bibliotek

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

blok połączeń (interfejs)

```
entity sreg_23 is
    Port ( clk : in std_logic;
           d1 : out std_logic;
           d19 : out std_logic;
           din : in std_logic);
end sreg_23;
```

behawioralny opis arhitektury

```
architecture Behavioral of sreg_23 is
signal reg: STD_LOGIC_VECTOR (22 downto 0)
                            := "10111011111100011110111";
begin

process(clk) begin
     if clk'event and clk = '1' then
          reg(22 downto 0) <= din & reg(22 downto 1);
     end if;
end process;

d19 <= reg(18);
d1 <= reg(0);

end Behavioral;
```

Chociaż ten opis wygląda jak program, to nie jest programem.

Poszczególne instrukcje nie zawsze są wykonywane sekwencyjnie

(dzielnik częstotliwości przez 1728)

PLIKI NAGŁÓWKOWE BIBLIOTEK

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

DEFINICJA POŁĄCZEŃ BLOKU

```
entity div_1728 is
    Port (CLK : in std_logic;
          WY : out std_logic;
          HOR_DIV : out std_logic);
end div_1728;
```

BEHAWIORALNY OPIS ARCHITEKTURY BLOKU

```
architecture Behavioral of div_1728 is
     type state_1728 is range 0 to 1727;
     signal count : state_1728 := 0;

begin
     process (CLK) begin
          if CLK'event and CLK = '1' then
               if count < 1727 then
                    count <= count + 1;
                    WY <= '0';
               else
                    count <= 0;
                    WY <= '1';
               end if;
          end if;
     end process;

     process (CLK) begin
          if CLK'event and CLK = '1' then
               if count < 128 then
                    HOR_DIV <= '1';
               else
                    HOR_DIV <= '0';
               end if;
          end if;
     end process;

end Behavioral;
```

FSM (Finite State Machine) - PRZYKŁAD – STEROWANIE PROSTĄ WINDĄ