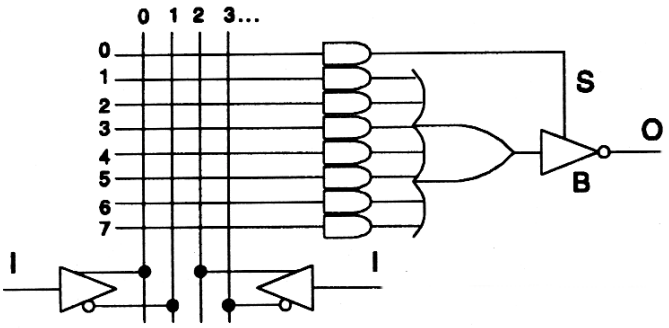
[1] Altera, *Data Sheet*, (http://www.altera.com/html/literature),
[2] T. Łuba, K. Jasiński, B. Zbierzchowski, *Specjalizowane układy cyfrowe w strukturach PLD i FPGA*, Wydawnictwa Komunikacji i Łączności, Warszawa 1997.
[3] Lattice, *Low Density PLDs*, (http://www.latticesemi.com/products/devices/lowplds.html),
[4] Xilinx, *Data Book*, (http://www.xilinx.com/partinfo/databook.htm),
[5] Piotr Dębiec, *Laboratorium Teorii Układów Logicznych i Systemów Mikroprocesorowych - Programowalne Układy Logiczne*,

1. Klasyfikacja układów ASIC (*Application Specific Integrated Circuits*)

a) Układy zamawiane przez użytkownika (*full-custom*),
b) Układy projektowane przez użytkownika (*semi-custom*),
c) Układy programowane przez użytkownika:
- PLD (*Programable Logic Devices*) matryce bramek
  - PAL (*Programable Array Logic*)
  - PLA (*Programable Logic Array*), itp.
- FPGA (*Field Programmable Gate Arrays*) matryce komórek
  - TLU (*Table LookUp*)
  - MUX, itp.

2. Struktura typu PAL – GAL20V8

a) Struktura programowalna PAL,

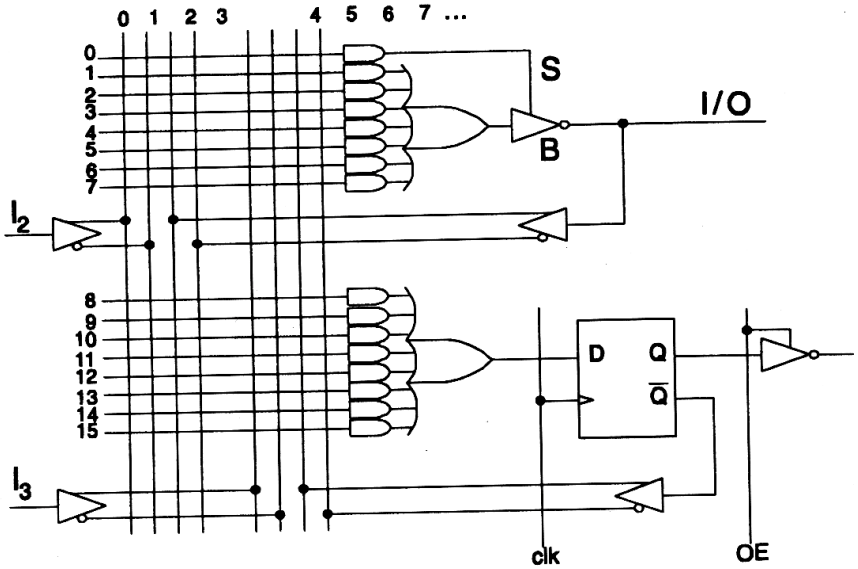


układ kombinacyjny,

uproszczone symbole bramek wzmacniacza i inwertera,

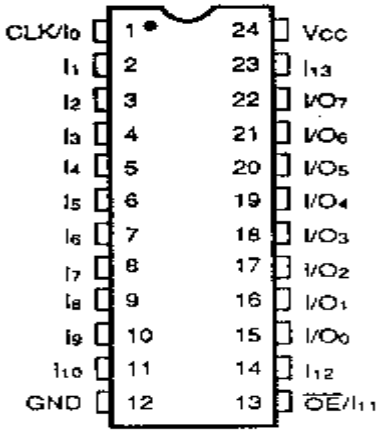
uproszczone symbole bramek wielowejściowych AND

b) Struktura programowalna PAL ze sprzężeniami zwrotnymi i przerzutnikami,



układ sekwencyjny z wejściem taktującym

c) Układ **GAL20V8** firmy Lattice, który jest reprezentantem rodziny układów PLD typu **PAL** (Programmable Array Logic), które posiadają programowalną matrycę **AND** i nieprogramowalną matrycę **OR**.

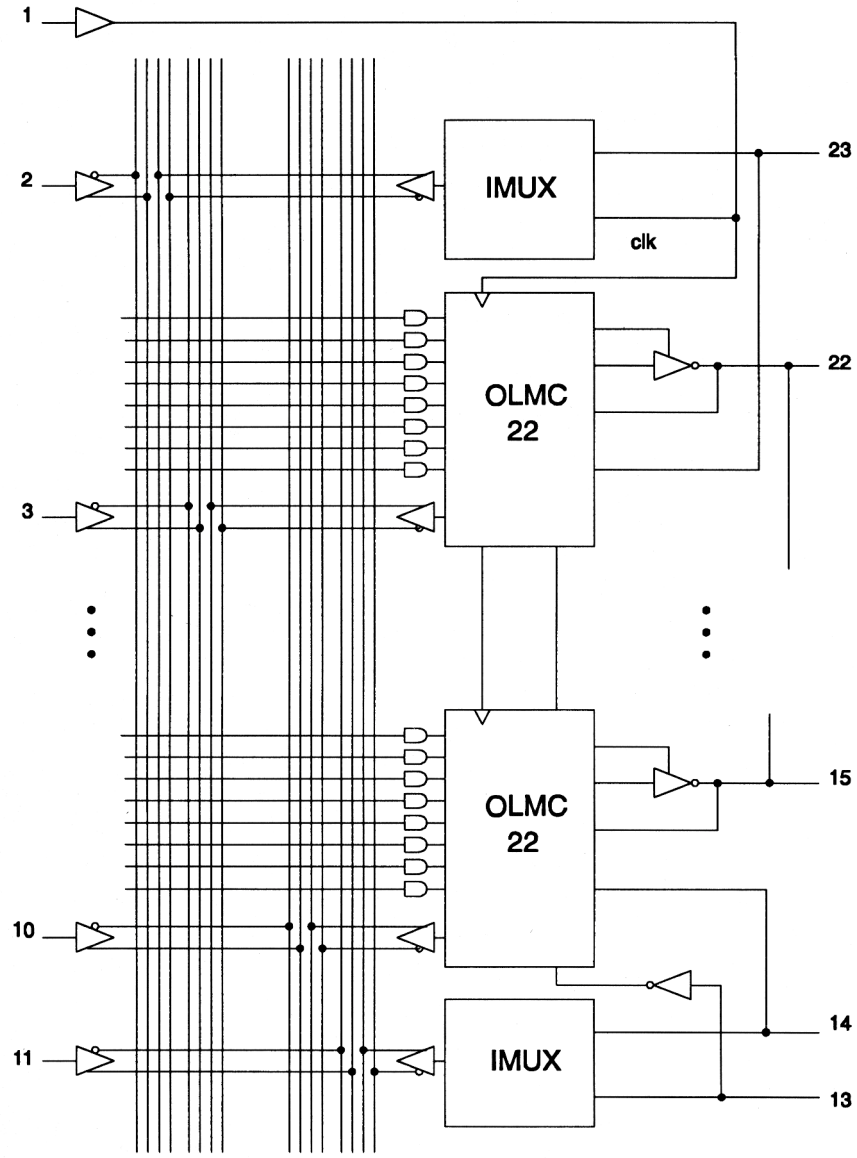


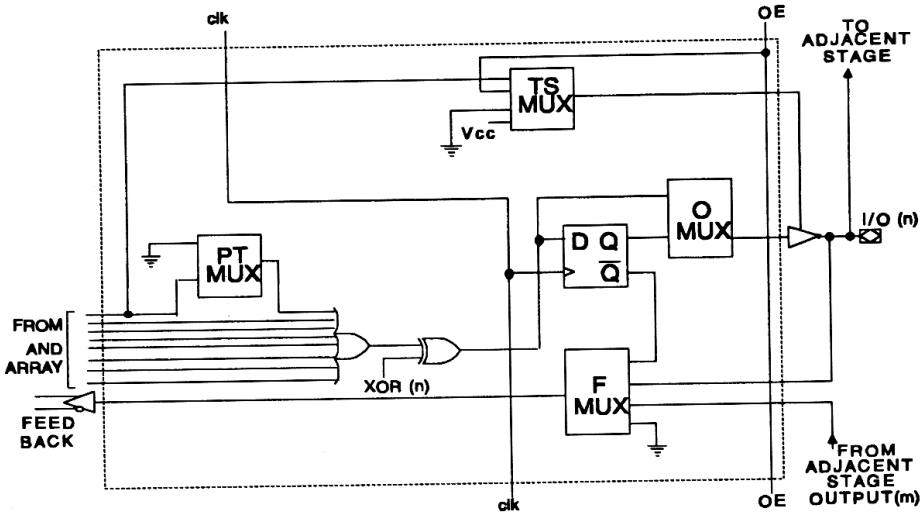
GND - masa
CLK - zegar
I - wejście
I/O - wejście/wyjście
OE - sterowanie buforem
$\mathbf{V_{CC}}$ - zasilanie + 5 V

Najważniejsze cechy układu GAL20V8:

- wykonany w technologii **EE CMOS**, pozwalającej na wielokrotne reprogramowanie układu (gwarantowane 100 cykli)
- zawiera **8** makrokomórek (**MC**), z których każda może zostać indywidualnie skonfigurowana jako wyjście rejestrowe, wyjście kombinacyjne, kombinacyjne wejście/wyjście lub dedykowane wejście (końcówki oznaczone **I/O**)
- wspólne dla wszystkich przerzutników wejście zegarowe (**CLK**) oraz wejście uaktywniające bufor 3-stanowy ($\overline{\mathbf{OE}}$)
- realizacja funkcji boolowskich w postaci sumy iloczynów
- programowalna polaryzacja wyjścia **MC**, pozwalająca na realizację funkcji w postaci afirmacyjnej lub podwójnie zaprzeczonej
- matryca **AND** o rozmiarach **40 x 64**
- zerowanie wszystkich przerzutników po włączeniu zasilania (stan wyjść rejestrowych jest negacją stanu przerzutników)
- możliwość załadowania z końcówek **I/O** pożądanego stanu przerzutników (**Supervoltage Register Preload**)
- możliwość zabezpieczenia przed skopiowaniem zaprogramowanej konfiguracji układu (**Security Bit**)
- możliwość zapisania dodatkowej **64**-bitowej informacji, zawsze dostępnej dla odczytu (**Electronic Signature Word**)

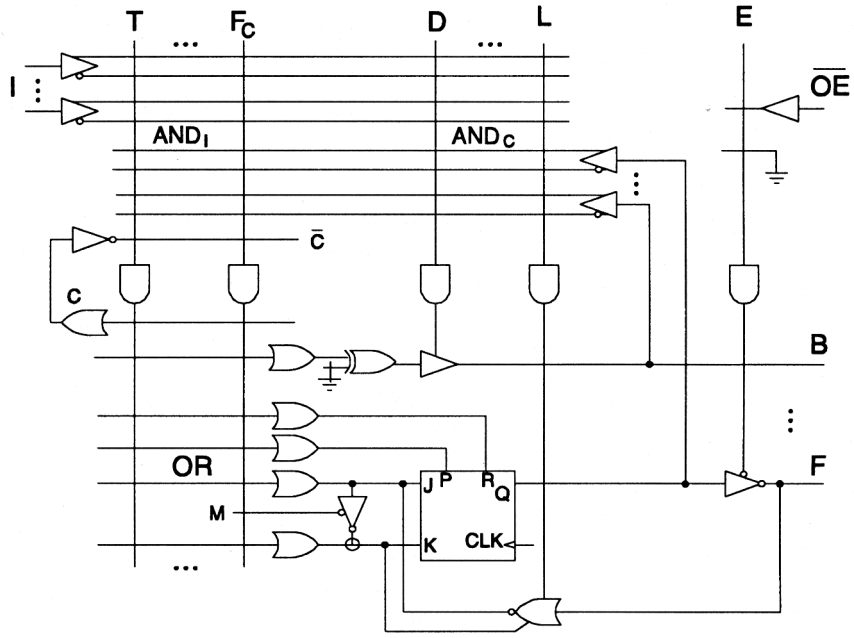
d) Struktura układu GAL20V8,





Schemat makrokomórki wyjściowej (*OLMC - Output Logic MacroCell*)

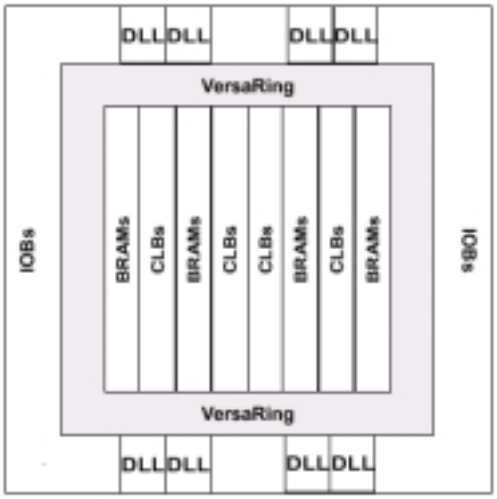
3. Układy typu PLA



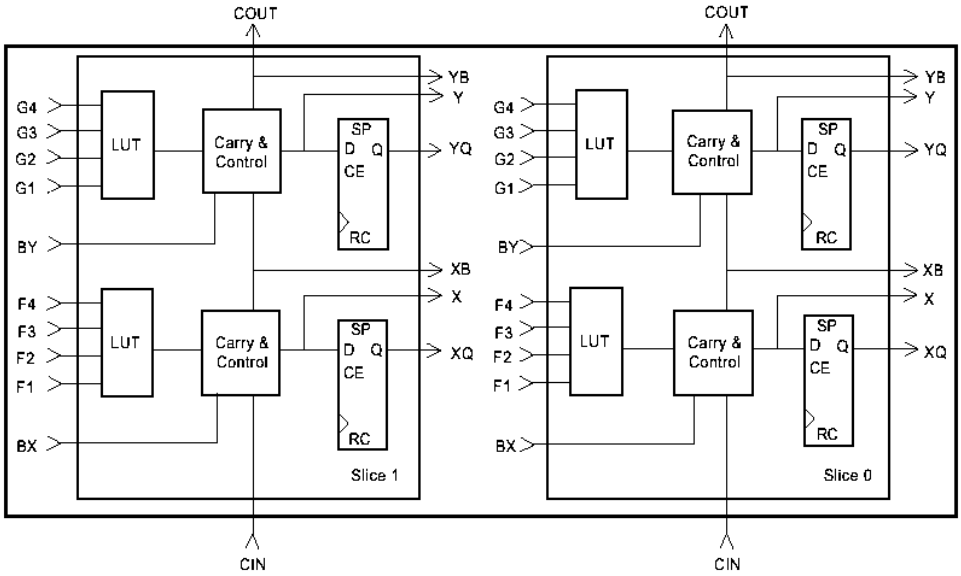
Układ PLS155 (*Programmable Logic Sequencer*)

4. FPGA - układy rodziny Virtex firmy Xilinx

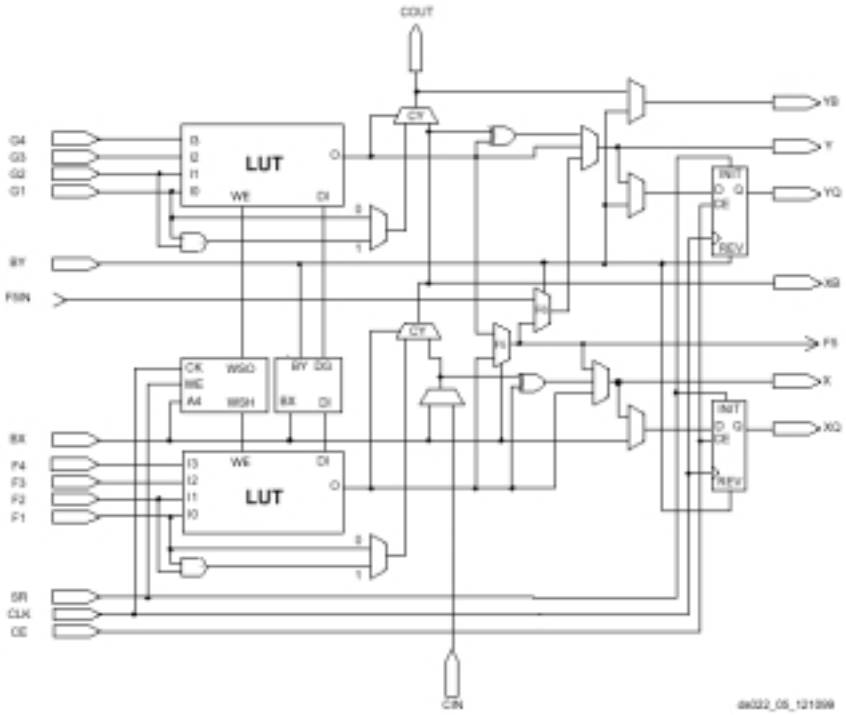
Wiodącymi producentami cyfrowych układów programowalnych są między innymi firmy *Xilinx* i *Altera*. Pierwsza z nich wśród swych produktów oferuje układy (*FPGA – field programmable gate arrays*) rodziny *Virtex*. Składają się one z bloków logicznych wejścia / wyjścia obsługujących wyprowadzenia układu scalonego (*IOB – input / output block*), układów opóźniających (*DLL – Delay-Locked Loops*), matrycy programowalnych bloków logicznych (*CLB – complex programmable logic device*) oraz bloków pamięci RAM z podwójnym układem dostępu do zawartości. Poszczególne bloki logiczne *CLB* układów *Virtex* składają się z dwu komórek logicznych (*LC – logic cell*). Każda z takich komórek zbudowana jest z programowanego obwodu realizującego dowolną funkcję logiczną czterech zmiennych (*LUT – look-up table*), układu przeniesienia oraz rejestru typu D. Odpowiednie zaprogramowanie połączeń w *LC* umożliwia np. realizację jednobitowego sumatora lub układu mnożącego. Struktura połączeń pomiędzy poszczególnymi blokami układu jest również programowana co umożliwia utworzenie wielobitowych sumatorów, układów porównujących dwie wartości binarne, mnożarek, itp.



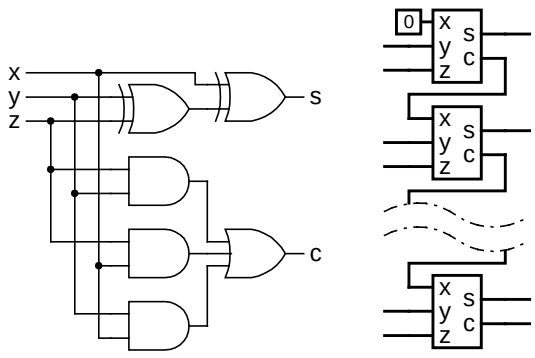
Struktura wewnętrzna układów rodziny Virtex



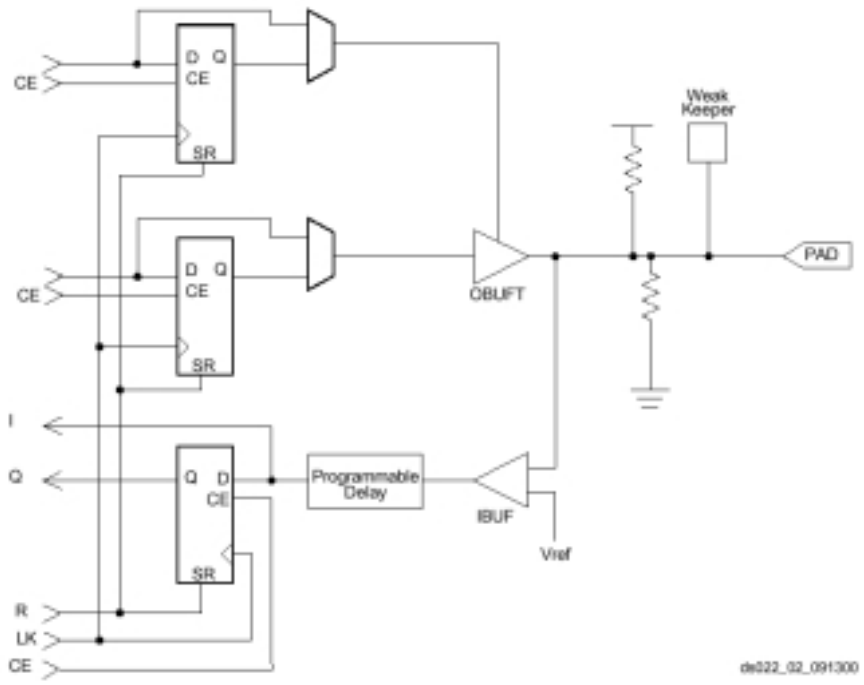
Schemat bloku logicznego CLB składającego się z dwu komórek LC



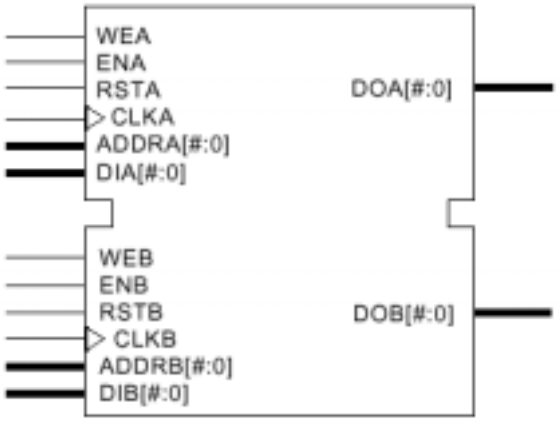
Schemat wewnętrzny komórki LC



Przykład realizacji sumatora liczb binarnych.



Schemat bloku wejść/wyjść (IOB)



Pamięć RAM o podwójnym dostępie

Firma *Altera* wśród swych produktów oferuje układy rodziny *Apex*. Układy te zawierają elementy logiczne (*LE – Logic Elements*), obwody wejścia wyjścia, bufory pamięci *FIFO (First Input First Output)*, pamięć RAM z podwójnym układem dostępu do zawartości i bloki *ESB* (*Embeded System Block*). Elementy logiczne *LE* podobnie jak komórki *LC* układów *Virtex* mogą być tak zaprogramowane by wypełniały funkcje jednobitowych sumatorów i mnożarek. Bloki *ESB* składają się z makrokomórek oraz połączonych z nimi matryc bramek. Układy te podobnie jak w przypadku układów rodziny *XLAT* pozwalają na realizację funkcji binarnych 32 zmiennych.

Firma *Xilinx* podaje, że układ typu XCV3200E należący do rodziny *Virtex* zawiera 16224 bloków CLB i 851968 bitów w bloku pamięci RAM. Układy *Apex* firmy *Altera* mogą zawierać do 51840 elementów *LE*, 442368 bitów pamięci i 3456 makrokomórek (układ typu EP20K1500E). Liczba bramek wykorzystywanych w procesie projektowania w tych układach dochodzi do kilku milionów. Producenci podają, że mogą one być taktowane maksymalnie częstotliwościami przekraczającymi 100 MHz.

5. Projektowanie struktury połączeń układów

Obecnie na rynku jest dostępnych kilka komercyjnych narzędzi syntezy i optymalizacji programowalnych układów cyfrowych. Większość z nich akceptuje opis układów w językach opisu sprzętu, takich jak *Verilog, Abel-HDL* (*Hardware Description Language*) czy *VHDL* (lub jego odmianach).

Programy syntezy i optymalizacji programowalnych układów cyfrowych umożliwiają:

- opis projektu za pomocą równań logicznych, tablic prawdy, schematu przejść, lub dowolnej ich kombinacji,
- kompilację, optymalizację i symulację projektu,
- generację pliku wyjściowego dla programatora,