Mikrokontrolery i układy FPGA

jako przykłady programowalych układów scalonych

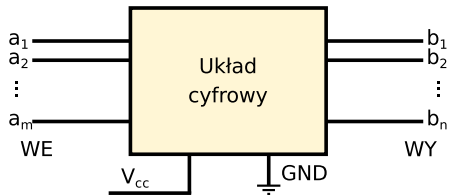
Piotr Podlaski

5 kwietnia 2023

Po co nam programowalne układy scalone?

Załóżmy że chcemy zbudować urządzenie pomiarowe:

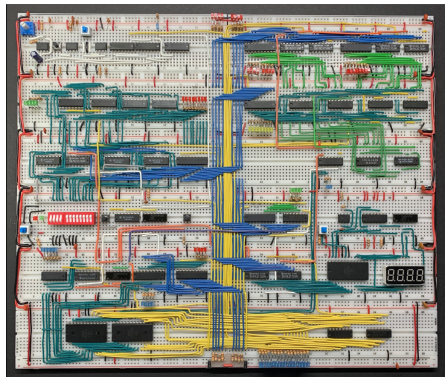
- urządzenie potrzebuje "skomplikowanego" układu cyfrowego
- układ będzie miał dziesiątki (setki) wejść i wyjść
- wewnątrz układu wykorzystamy logikę zarówno kombinacyjną jak i synchroniczną



źródło ilustracji: M. Kuich, slajdy do wykładu "Pracownia Fizyczna i Elektroniczna, 2022"

Po co nam programowalne układy scalone?

- nasze urządzenie projektujemy w oparciu o podstawowe elementy logiczne takie jak:
  - bramki logiczne
  - przerzutniki
  - układy pamięci
  - rejestry
- prototyp układu możemy zbudować na płytce/płytkach stykowych
- wraz ze wzrostem projektu coraz trudniej zapanować nad prototypem na płytkach stykowych



https://www.youtube.com/beneater

Po co nam programowalne układy scalone?

- płytki stykowe mogą zostać zastąpione płytkami drukowanymi (PCB, printed circuit board)
- na płytkach drukowanych możemy w łatwy sposób pomieścić o wiele więcej elementów

https://gigatron.io/

Oba podejścia mają zasadnicze wady:

- przy większych projektach dodawanie nowych funkcjonalności jest kłopotliwe/pracochłonne na etapie prototypowania
- zmiana zachowania układu jest niemożliwa po wyprodukowaniu i zainstalowaniu urządzenia

Po co nam programowalne układy scalone?

Możliwe rozwiązanie:

- cyfrowy układ elektroniczny który można skonfigurować do danego zadania
- to użytkownik decyduje jaka logika odczytuje wejścia układu i steruje jego wyjściami
- pożądaną cechą takiego układu jest możliwość zmiany jego konfiguracji już po zainstalowaniu w docelowym urządzeniu

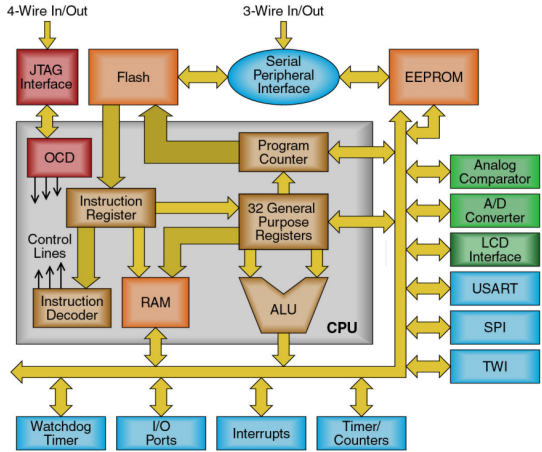
Programowalne układy scalone - podział

Układy scalone które można programować dzielimy na:

- Układy z mikroprocesorem (mikrokontrolery)
  - mózgiem układu jest procesor sekwencyjnie wykonujący operacje (instrukcje)
  - układy te programuje się zazwyczaj w języku C/C++, coraz rzadziej konieczna jest znajomość niskopoziomowego assemblera
  - od kilku lat na rynku są obecne mikrokontrolery które można programować w języku Python
- Programowalne układy logiczne:
  - W układach tego typu konfigurowane są fizyczne połączenia między podstawowymi elementami logicznymi (bramki, przerzutniki, etc.)
  - Przykładem takich układów są układy FPGA (Field Programmable Gate Array) lub CPLD (Complex Programmable Logic Device)
  - Do programowania wykorzystywane są specjalistyczne języki opisu sprzętu (HDL, Hardware Desctiption Language)

Mikrokontrolery - przegląd

- Mikrokontrolery to układy elektroniczne zawierające:
  - procesor
  - pamięć RAM/ROM
  - układ zegarowy
  - porty wejścia/wyjścia
  - interfejsy komunikacyjne (SPI, I^2C, UART)
- Popularne architektury:
  - ARM
  - AVR
  - PIC



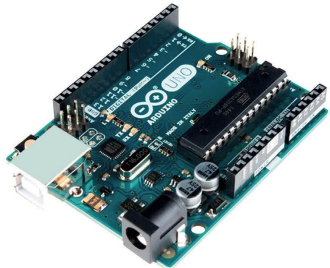
Mikrokontrolery - programowanie

- Większość (jeśli nie wszystkie) współczesnych mikrokontrolerów można programować w języku C/C++
- W przypadku rodziny AVR obsługa pinów i peryferiów wymaga operacji na rejestrach

```
#include <avr/io.h>
#include <util/delay.h>

int main()
{
    DDRA = 1 << PA0;
    while ( true )
    {
        PORTA |= 1 << PA0;
        _delay_ms( 1000 );
        PORTA &= ~(1 << PA0);
        _delay_ms( 1000 );
    }
    return 0;
}
```

Mikrokontrolery - Arduino

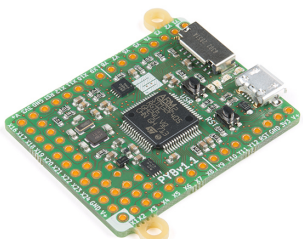
- Arduino to otwarta platforma elektroniczna
- W jego skład wchodzą łatwe w użyciu oprogramowanie i płytki ewaluacyjne
- Pierwotnie oparta o mikrokontroler ATMega 328P
- Duże wsparcie społeczności, wiele modułów rozszerzeń

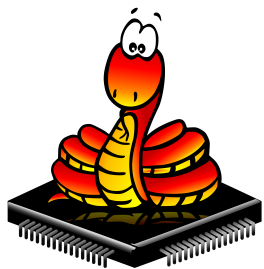


```
int led = 13;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

Mikrokontrolery w Pythone?

- Nowe, bardziej wydajne mikrokontrolery można programować w Pythonie
- Interpreter Pythona działający na tych układach jest napisany w C++
- CircuitPython i MicroPython to odmiany Pythona działające na mikrokontrolerach



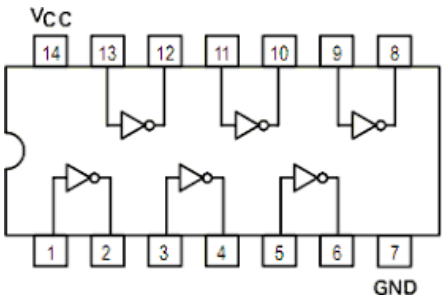


```
import machine as m
import time

led=m.Pin(0, m.Pin.OUT)

while True:
    led.high()
    time.sleep(1.0)
    led.low()
    time.sleep(1.0)
```

Mikrokontroler jako programowalny układ logiczny

Czy mikrokontroler sprawdzi się w roli cyfrowego układu logicznego którego działanie można zmienić, przeprogramować, w dowolnej chwili? Sprawdźmy.

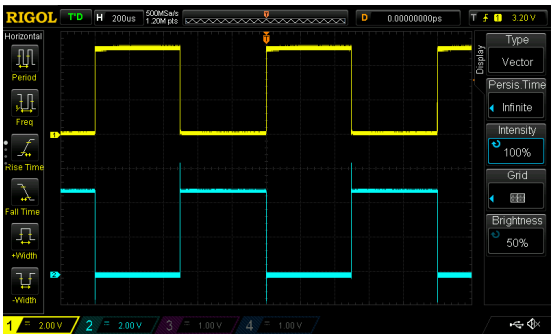
- Zaprogramujmy prosty inwerter: stan na pinie wyjściowym będzie zawsze przeciwny do stanu na wejściu układu
- Aby ocenić jakość naszego inwertera porównamy go z układem 7404, który zawiera fizyczne inwertery



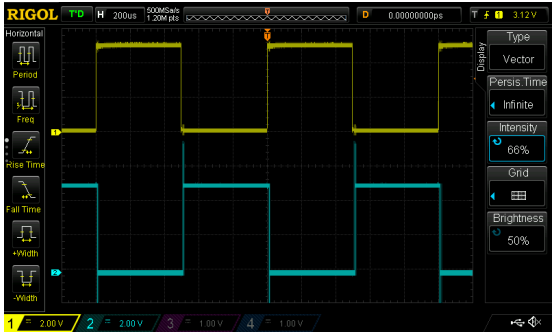

```
int in  = 10;
int out = 13;
void setup() {
  pinMode(input, INPUT);
  pinMode(out, OUTPUT);
}
void loop() {
  digitalWrite(out, !digitalRead(in));
}
```

Mikrokontroler jako programowalny układ logiczny

Fizyczny inwerter:



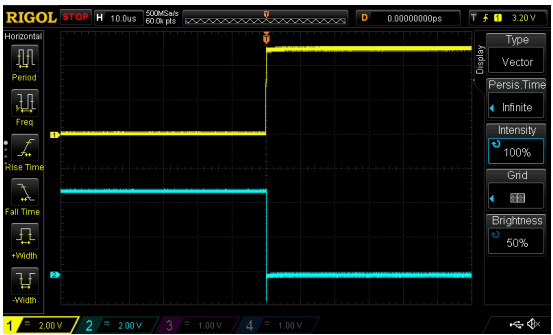
Mikrokontroler:



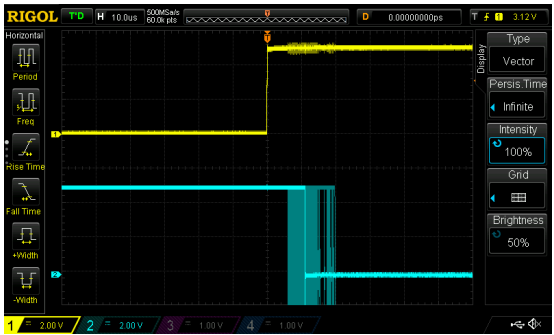
sygnał prostokątny o częstotliwości $f = 1$ kHz,
podstawa czasowa 200 μs na działkę

Mikrokontroler jako programowalny układ logiczny

Fizyczny inwerter:



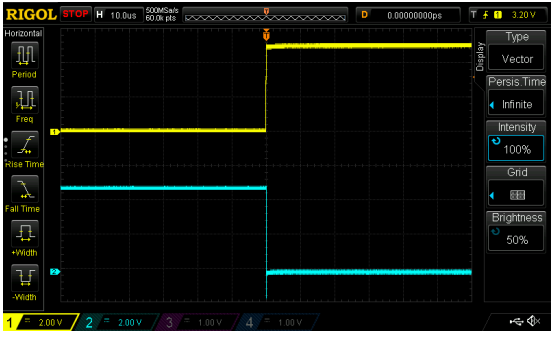
Mikrokontroler:



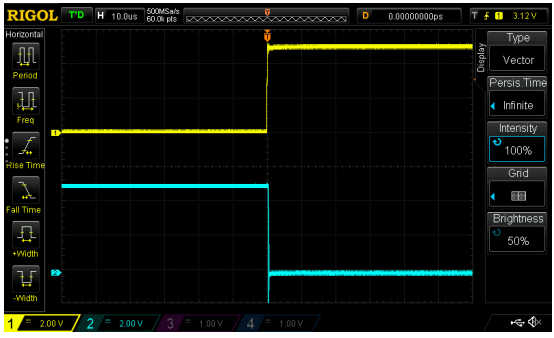
sygnał prostokątny o częstotliwości $f = 1$ kHz,
podstawa czasowa 10 μs na działkę

Mikrokontroler jako programowalny układ logiczny

Fizyczny inwerter:



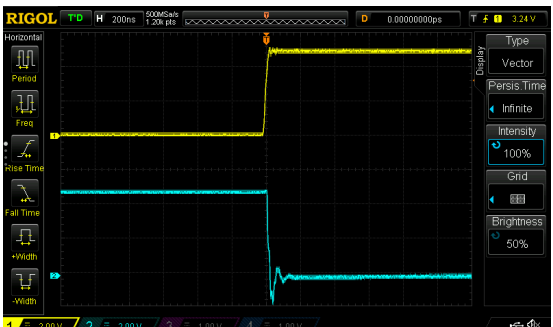
Mikrokontroler (niskopoziomowy kod):



sygnał prostokątny o częstotliwości $f = 1$ kHz,
podstawa czasowa 10 μs na działkę

Mikrokontroler jako programowalny układ logiczny

Fizyczny inwerter:



Mikrokontroler (niskopoziomowy kod):

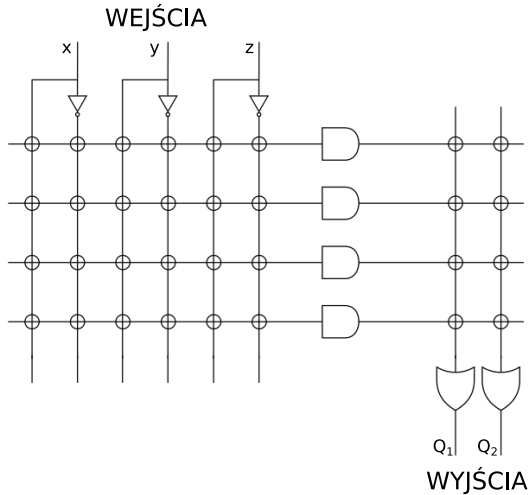
sygnał prostokątny o częstotliwości $f = 1$ kHz,
podstawa czasowa 200 ns na działkę

Mikrokontroler jako programowalny układ logiczny

- Mikrokontrolery mogą zastąpić proste i złożone operacje logiczne jeśli nie zależy nam na szybkości działania i wysokiej precyzji
- stosując niskopoziomowe podejście można poprawić parametry uzyskanego układu, **ALE**: implementacja jakiejkolwiek innej funkcji pogorszy działanie naszego układu logicznego

Logika *prawdziwie* programowalna - PAL

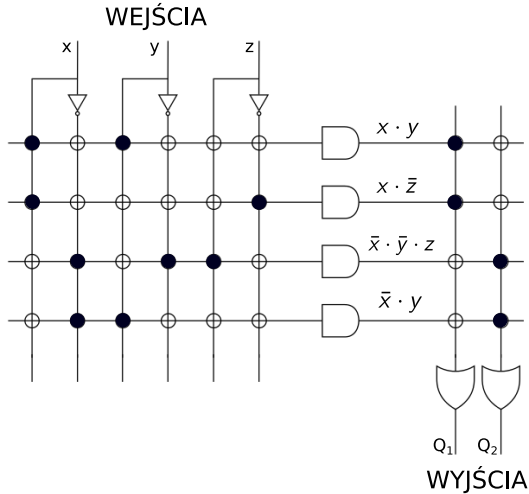
- PAL - Programmable Logic Array
- Macierze bramek z programowalnymi połączeniami
- 'Oczka' na rysunku po prawej oznaczają możliwe miejsca połączeń
- Układ jest programowany przez wykonanie połączeń w wybranych miejscach



Programowalne Macierze Logiczne

Przykład: dwie funkcje logiczne 3-bitowego wejścia układu

- $Q_1 = x \cdot y + x \cdot \bar{z}$
- $Q_2 = \bar{x} \cdot \bar{y} \cdot z + \bar{x} \cdot y$

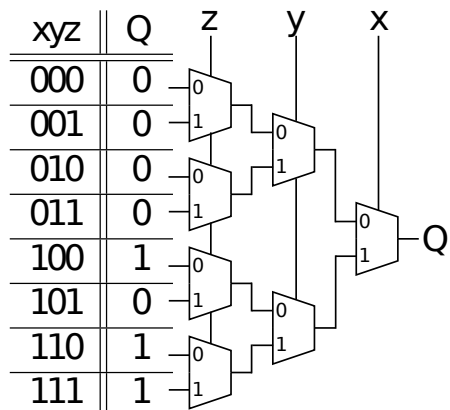


Tablice przeglądowe

$$Q = x \cdot y + x \cdot \bar{z}$$

Tablica przeglądowa (ang. Lookup Table, LUT):

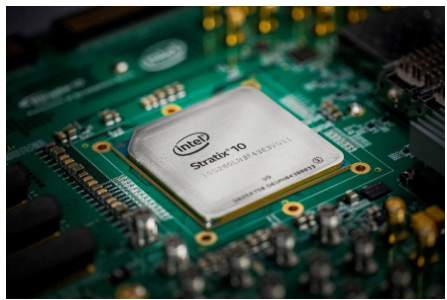
- Komórka pamięci jako kombinacyjny obwód logiczny
- Wejścia układu używane są do adresowania pamięci
- Przy n wejściach potrzebujemy 2^n bitów pamięci żeby opisać dowolny obwód kombinacyjny
- Zachowanie układu konfigurowane jest przez zapis informacji do pamięci



Układy FPGA

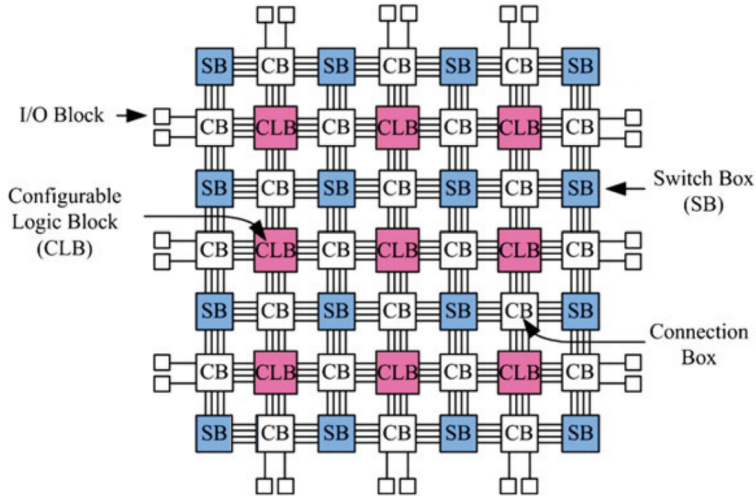
Field Programmable Gate Array -
Bezpośrednio Programowana Macierz Bramek:

- W w pełni konfigurowalne cyfrowe układy scalone
- Konfiguracja układu może być zmieniona w dowolnej chwili - nawet po zainstalowaniu w docelowym produkcie/urządzeniu
- Zazwyczaj konfiguracja jest ulotna - po utracie zasilania układ musi być skonfigurowany od nowa

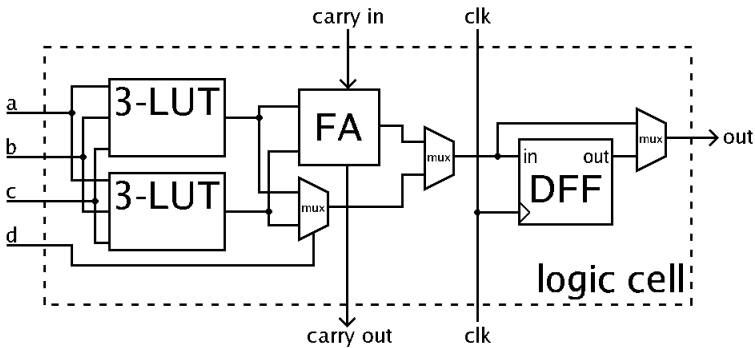


FPGA - Architektura

- Konfigurowalne Bloki Logiczne (CLB)
- Macierz programowalnych połączeń (SB i CB)
- Bloki Wejścia/Wyjścia (I/O Block)

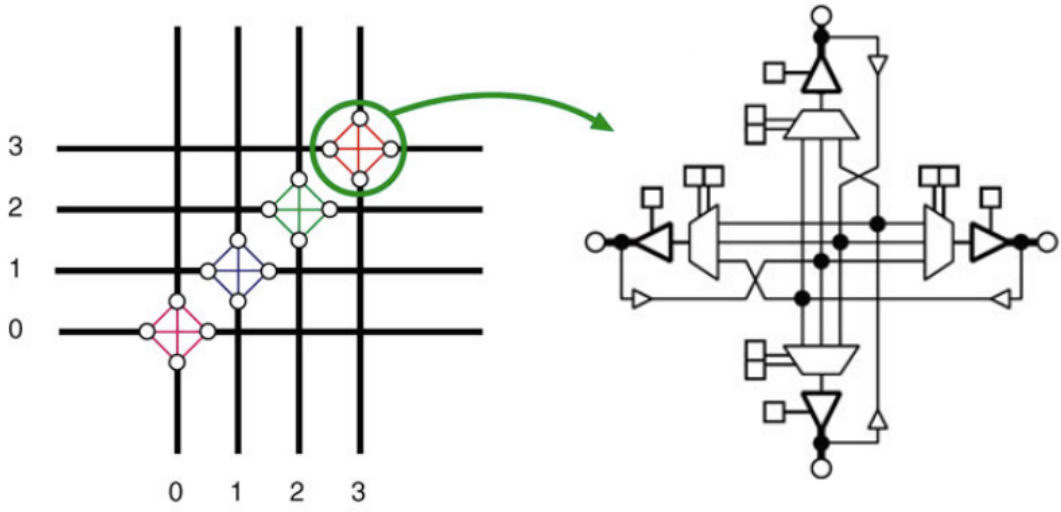


FPGA - CLB



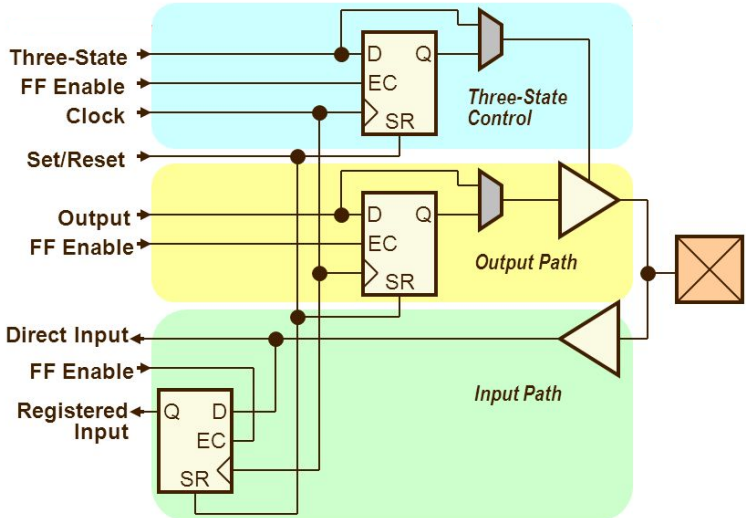
- Tablice Przeglądowe (LUT)
- Sumator z wyjściem/wejściem carry (FA, full adder)
- Przerzutnik typu D (DFF, D flip-flop)

FPGA - Programowalne połączenia



FPGA - Bloki Wejścia/Wyjścia

- Trzy stany wyprowadzeń układu:
  - wejście
  - wyjście
  - wysoka impedancja
- Opcjonalne przerzutniki wejściowe/wyjściowe
- Dodatkowo:
  - różne standardy napięciowe
  - wejścia różnicowe



FPGA - Programowanie

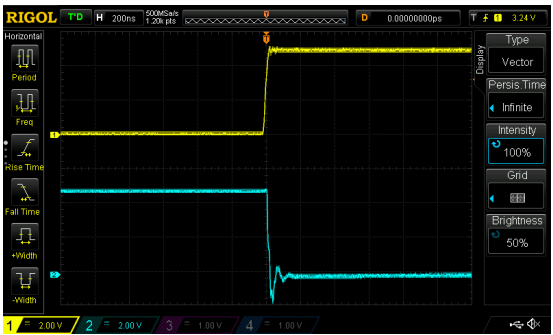
Programowanie FPGA znacząco różni się od programowania mikrokontrolerów czy pisania programów działających na zwykłych komputerach

- Opis funkcji układu za pomocą:
  - schematu logicznego (bramki, przerzutniki, etc.)
  - języka opisu sprzętu np. VHDL - Very High Speed Integrated Circuit Hardware Description Language
- Rozmieszczenie elementów logicznych w CLB układu FPGA
- Wykonanie połączeń pomiędzy blokami logicznymi i blokami wejścia/wyjścia
- Wygenerowanie pliku konfiguracyjnego który może być wgrany do układu FPGA

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

entity inv is
    port (
        input  : in  std_logic;
        output : out std_logic
        );
end inv;

architecture a of inv is
begin
    output <= not input;
end a;
```

FPGA jako programowalny układ logiczny

Fizyczny inwerter:

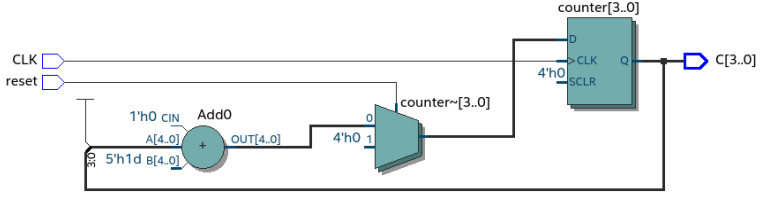


FPGA:

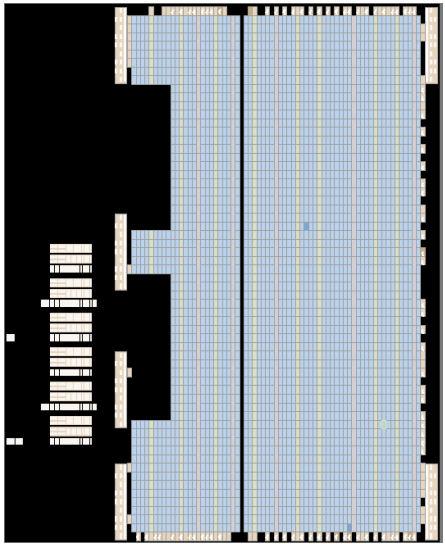
sygnał prostokątny o częstotliwości $f = 1$ kHz,
podstawa czasowa 200 ns na działkę

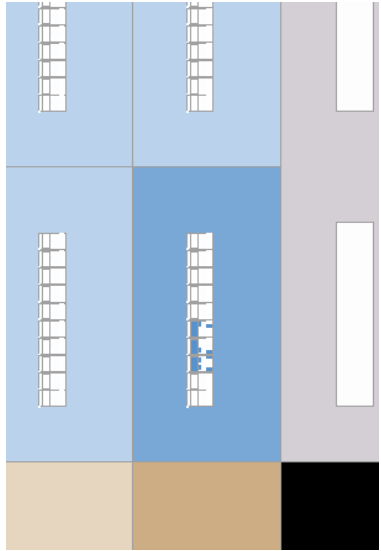
Bardziej skomplikowany przykład - licznik 4-bitowy

```
architecture a of cnt is
    signal counter: unsigned (3 downto 0);
begin
    process (clk) begin
        if rising_edge(clk) then
            if reset = '1' then
                counter <= (others => '0');
            else
                counter <= counter + 1;
            end if;
        end if;
    end process;

    c<= std_logic_vector(counter);
end a;
```

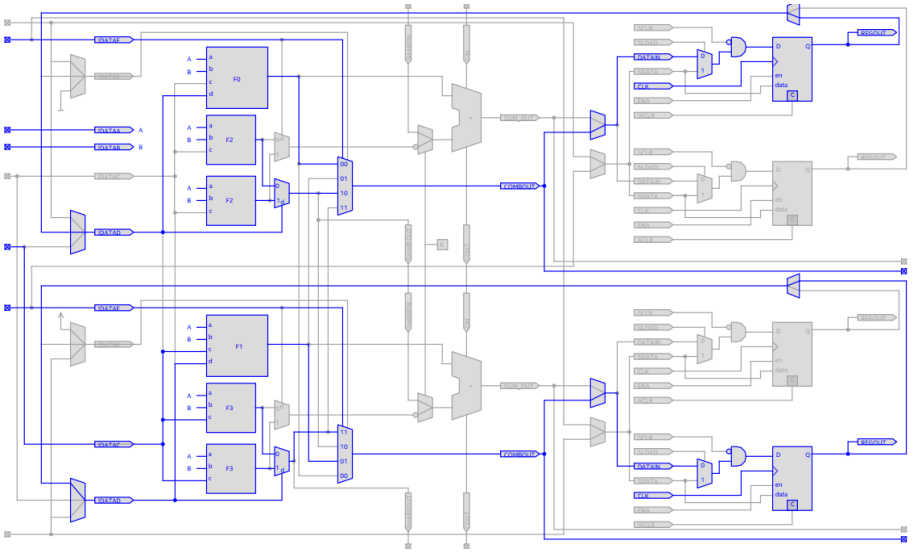


Bardziej skomplikowany przykład - licznik 4-bitowy





Bardziej skomplikowany przykład - licznik 4-bitowy



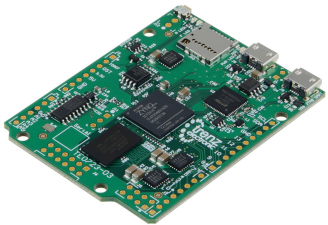
FPGA - dodatkowe funkcjonalności

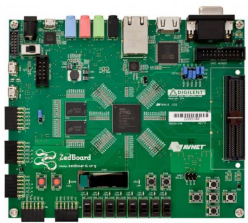
Współczesne układy FPGA wyposazone są w szereg dodatkowych funkcji:

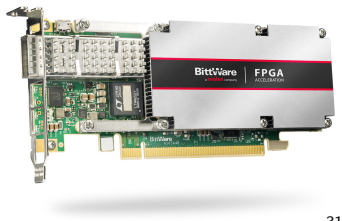
- Rozproszona pamięć RAM wewnątrz układów
- Sprzętowe kontrolery magistrali szeregowych (SPI, I^2C, PCIe)
- Dedykowane bloki do cyfrowego przetwarzania sygnałów (DSP)
- Sprzętowe kontrolery pamięci DDR
- Przekaźniki szybkich łącz szeregowych (dochodzące do 58 Gb/s)
- Wbudowane rdzenie procesora (najczęściej ARM) i uwspólniona szyna pomiędzy procesorem a logiką programowalną
- Dedykowane akceleratory sprzętowe do Machine Learningu

FPGA - Przegląd

- Na rynku jest obecnie kilku liczących się producentów układów FPGA:
  - Xilinx (niedawno kupiony przez AMD)
  - Intel (dawniej ALTERA)
  - Lattice
  - Microchip
- Ogromna różnorodność:
  - najmniejsze układy: kilkanaście wyprowadzeń, kilkaset bloków logicznych, koszt - kilka złotych za układ
  - największe układy: około tysiąca wyprowadzeń, kilka milionów bloków logicznych, koszt - setki tysięcy złotych za układ
- Dostępnych jest dużo płytek uruchomieniowych:





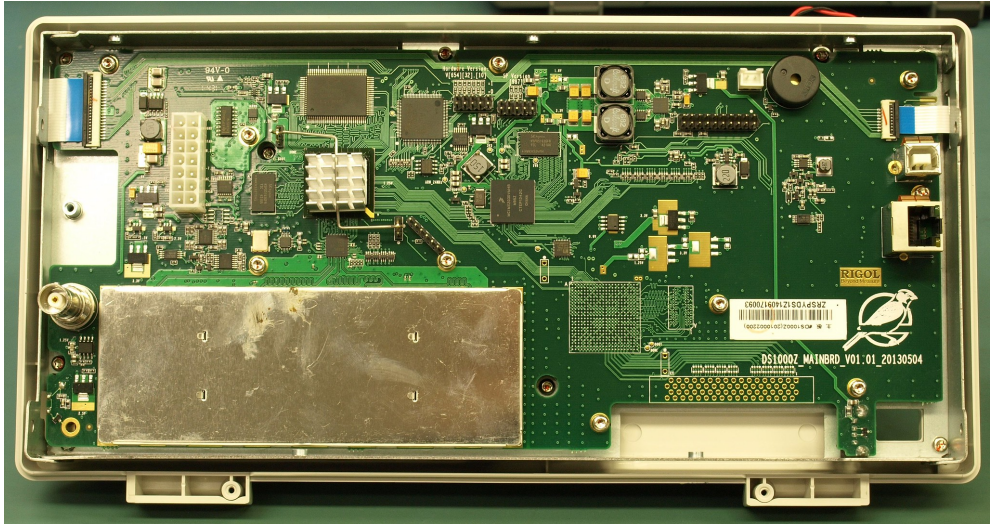


Zastosowania układów FPGA

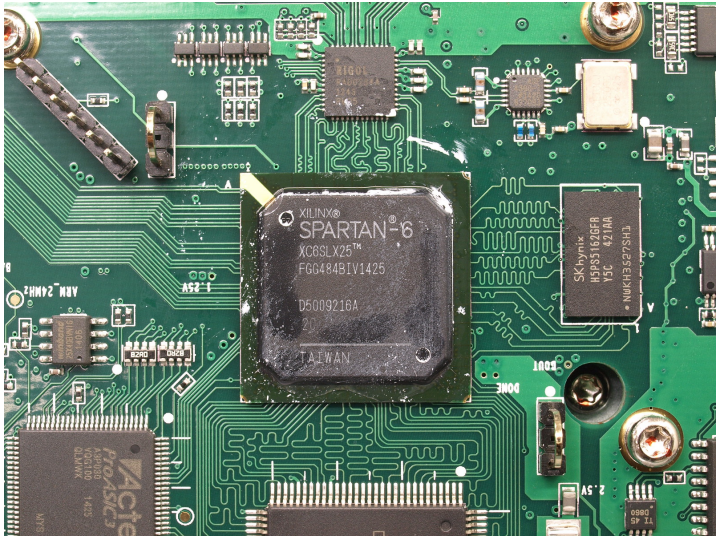
- Akwizycja i przetwarzanie danych w dużych eksperymentach fizycznych
- Aparatura pomiarowa (oscyloskopy, generatory przebiegów, etc.)
- Aparatura medyczna
- Przetwarzanie obrazów i wideo
- Przemysł lotniczy i kosmiczny
- Telekomunikacja

RIGOL DS1054 - oscyloskop z pracowni

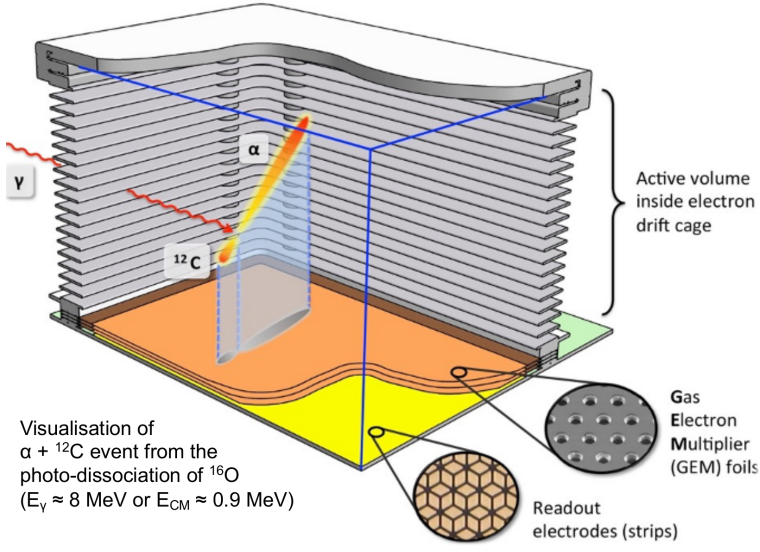
RIGOL DS1054 - oscyloskop z pracowni



RIGOL DS1054 - oscyloskop z pracowni

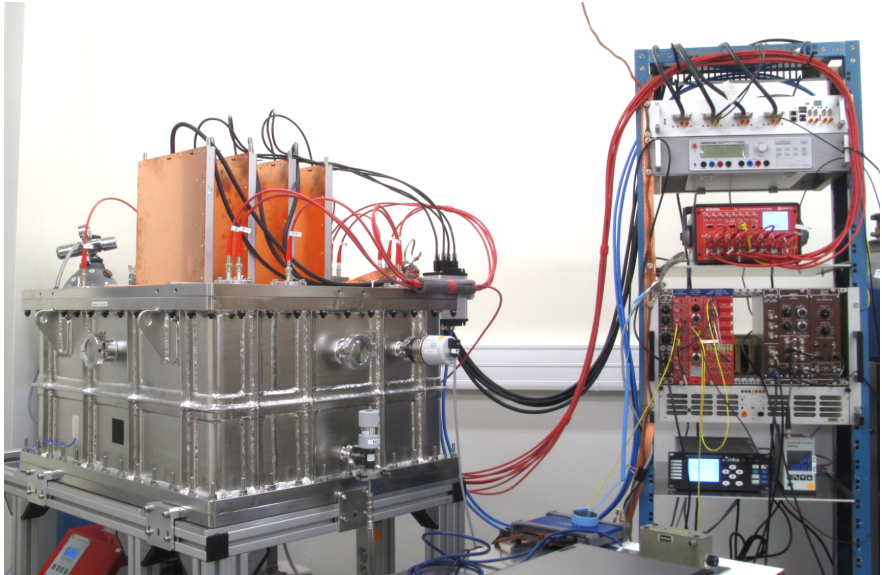


ELITPC - detektor zbudowany na FUWie

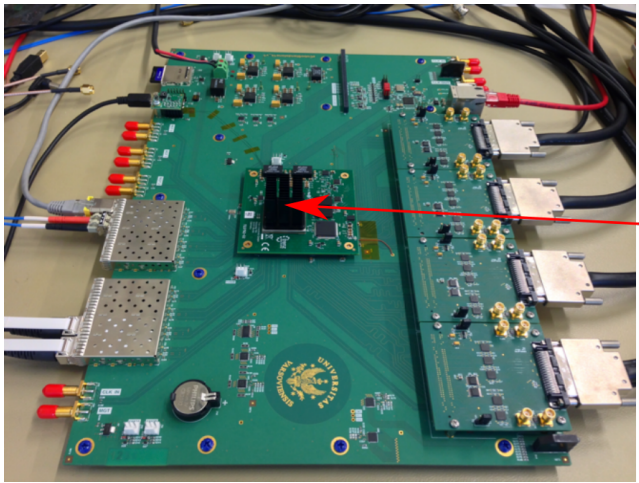


Visualisation of
α + ^{12}C event from the
photo-dissociation of ^{16}O
($E_\gamma \approx 8$ MeV or $E_{CM} \approx 0.9$ MeV)

ELITPC - detektor zbudowany na FUWie

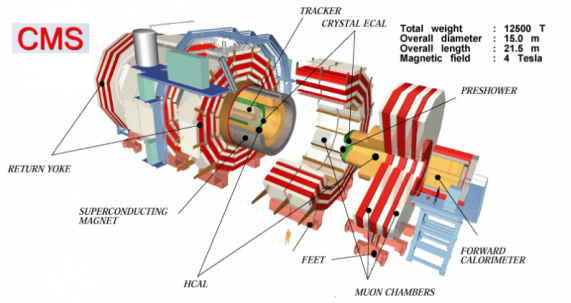


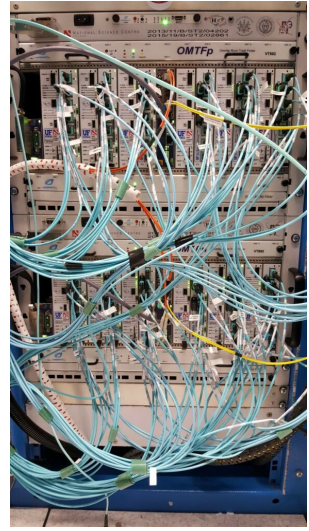
ELITPC - detektor zbudowany na FUWie



Duże eksperymenty wysokich energii w CERN

Układ wyzwalania eksperymentu CMS przy LHC





Selekcja interesujących przypadków:

- oddziaływania występują z częstotliwością 40 MHz
- układ wyzwalania ogranicza tę częstotliwość do 100 kHz
- zdarzenia na dysk zapisywane są z częstotliwością kilku tysięcy na sekundę

Podsumowanie

Układy FPGA:

- to bardzo elastyczne programowalne cyfrowe układy scalone,
- znajdują zastosowanie w wielu dziedzinach nauki/przemysłu
- umożliwiają pełną rekonfigurację
- dostępne są w połączeniu z procesorami w jednym układzie scalonym
- są bardzo użytecznym narzędziem w rękach fizyka