Warszawa, 26.06.2024

Projekt: FIT

Raport

# Opracowanie firmware FPGA dla programowalnego generatora kwadraturowego

Wykonawca: Piotr Zdunek

Data realizacji: od 5/16/2024 do 6/26/2024

## Cel pracy:

Wykonanie oprogramowania FPGA do programowalnego generatora kwadraturowego służącego do kontroli optyki z zastosowaniem przebiegów sin/cos w celu minimalizacji zakłóceń.

## Realizacja pracy:

Projekt oprogramowania z wykorzystaniem modułu ewaluacyjnego CMOD-A7 wraz z symulacjami i testami.

## Wynik pracy:

Link do repozytorium: <https://github.com/elhep/Motor_controller_DIOT>

Nazwa dla generatora używana w projekcie to: **Motion controller (MC)**

Struktura projektu:

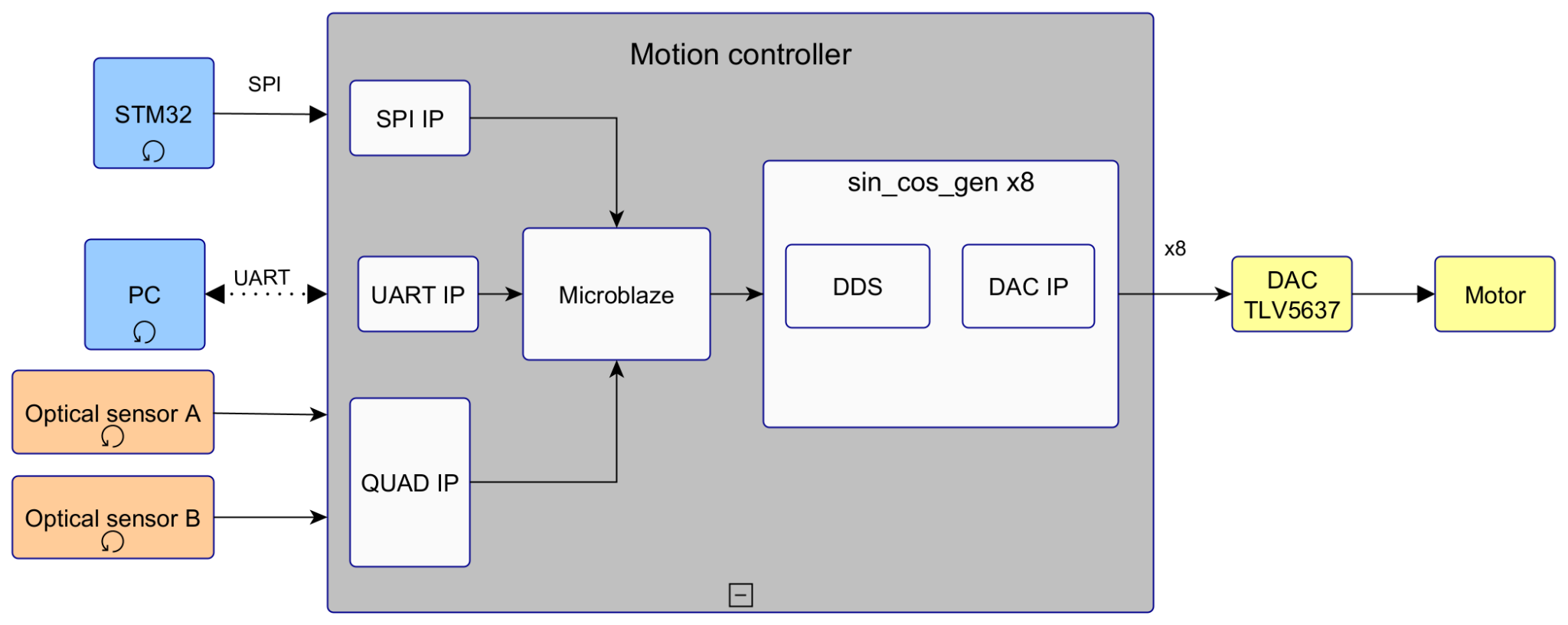
* **HDL** - oprogramowanie FPGA
  + Motion\_controller - główny projekt
  + Tests - projekty testowe
    - dds\_test -> projekt z symulacjami DDS
* **FW** - firmware
  + Vitis - oprogramowanie na softcore microblaze
  + STM32 - oprogramowanie na STM32
* **Doc** - dokumentacja
* **HW** - projekt PCB
* **Tests** - testy

### Wymagania:

Pierwszym etapem projektu była definicja wymagań dla kontrolera. Ustalone zostały następujące założenia projektu:

* MC musi sterować 8 kanałami podłączonymi do 2-fazowych silników krokowych z wykorzystaniem sygnału sinus/cosinus w celu minimalizacji zakłóceń.
* MC musi generować przebiegi SINUS/COSINUS w zakresie pojedynczych Hz
* MC musi posiadać wejście dekodera kwadraturowego
* MC musi posiadać interfejs SPI do kontroli systemu
* MC musi korzystać z modułu CMOD-A7 [2] jako głównego kontrolera
* MC musi posiadać form factor modułu Magneto

### Diagram blokowy systemu:

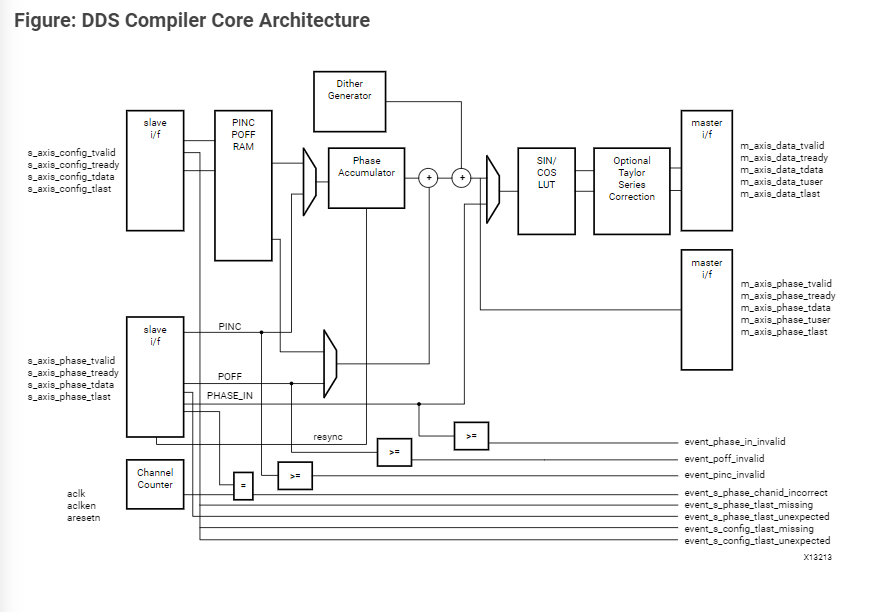


### Opis systemu:

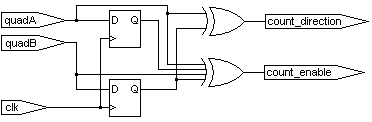
Motion controller składa się z softcore microblaze odpowiedzialnego za komunikację po SPI oraz UART i kontrolę rejestrów bloków generujących sygnał SIN/COS do silników 2-fazowych. Zwykłe sterowniki używają sygnału PWM który zakłóca czułą elektronikę dookoła systemu. Część projektu została wykonana z pomocą generatora blokowego (Block Design) a pozostałe części zostały napisane w języku VHDL. Oprogramowanie na softcore zostało napisane w języku C.

Głównym interfejsem sterującym jest SPI który służy do ustawiania rejestrów kontrolujących wszystkie funkcje systemu. UART został dodany jako interfejs do łatwiejszego podglądu co się aktualnie dzieje w kontrolerze. Docelowo system będzie sterowany za pomocą modułu z mikrokontrolerem STM32.

Każdy z 8 kanałów posiada rejestry do kontroli ustawiania fazy generowanej przez block NCO (Numerically Controlled Oscillator) z pomocą IP Core DDS [1]. Kontrola sygnału wyjściowego odbywa się poprzez ustawianie wartości granicznej do której ma zliczać licznik rewersyjny, który odpowiada za to jak długo generowany jest sygnał SIN/COS. Dane wyjściowe z NCO podane są na wejście IP Core DAC TLV5637 [4] . W konfiguracji DDS wykorzystujemy głównie moduł generacji sinusa/cosinusa.



Dodatkowo kontroler posiada wejście dekodera kwadraturowego do którego podłączone są czujniki położenia typu optycznegoi w zależności od ich stanu inkrementowany jest rejestr który pozwala na adjustowanie optyki. Rejestr ten jak i sygnał resetu licznika są dostępne do odczytu przez interfejs SPI.

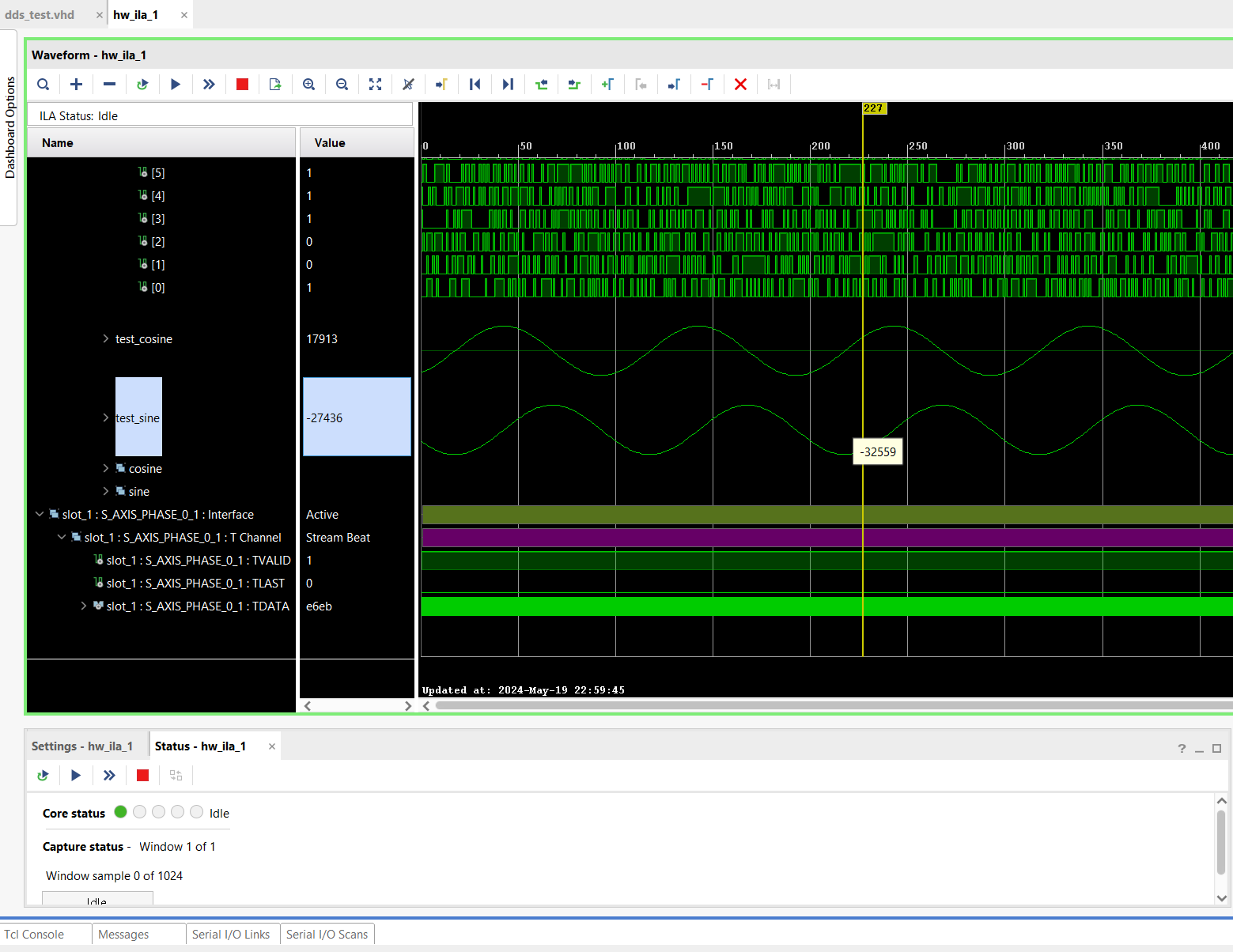


### Symulacje:

W ramach projektu wykonane zostały symulacje działania kanału sterującego silnikiem.

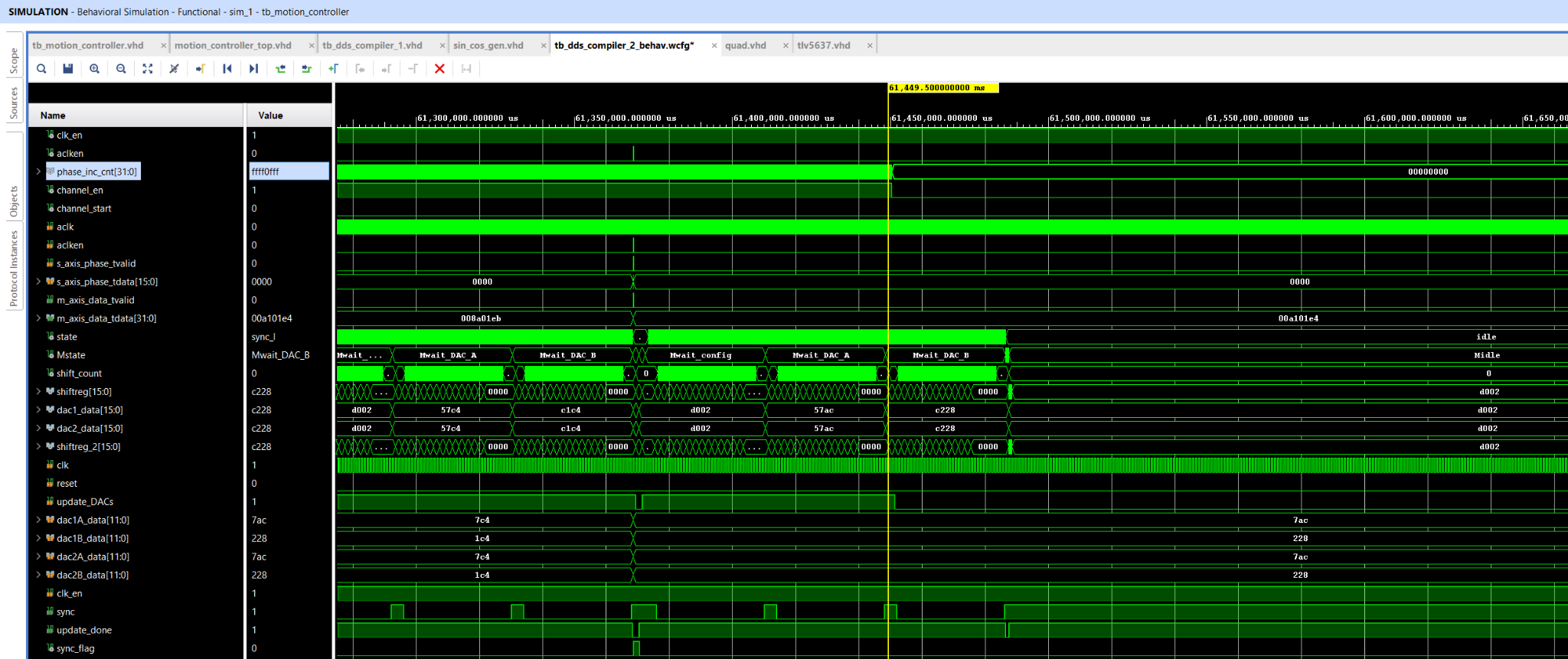
Symulacja generowała odpowiednie sygnały zegarowe oraz sterowała wejściami modułu **sin\_cos\_gen** aby uzyskać sygnał sinusoidalny na wyjściu IP Core DDS oraz odpowiednie dane wejściu do IP Core TLV5637.

Przykładowy przebieg sygnału sinusa na wyjściu DDS o częstotliwości 10 Hz:

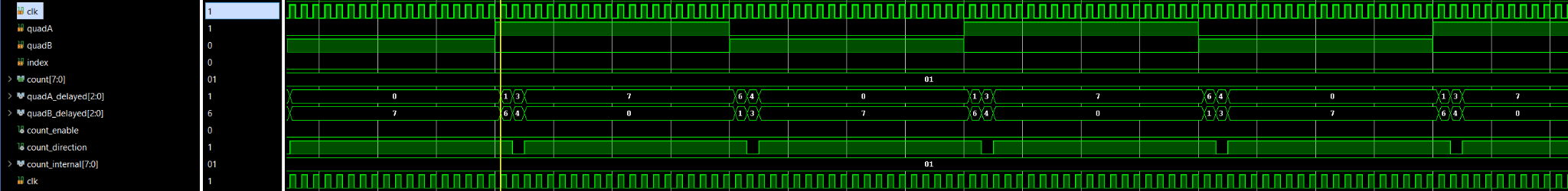


Przykładowa symulacja pokazująca działanie rejestru rewersyjnego, po ustawieniu odpowiedniej wartości w rejestrze sygnał jest generowany aż nie dojdziemy do tej wartości.

Wynik symulacji zawarty w pliku: **tb\_dds\_compiler\_2\_behav.wcfg**

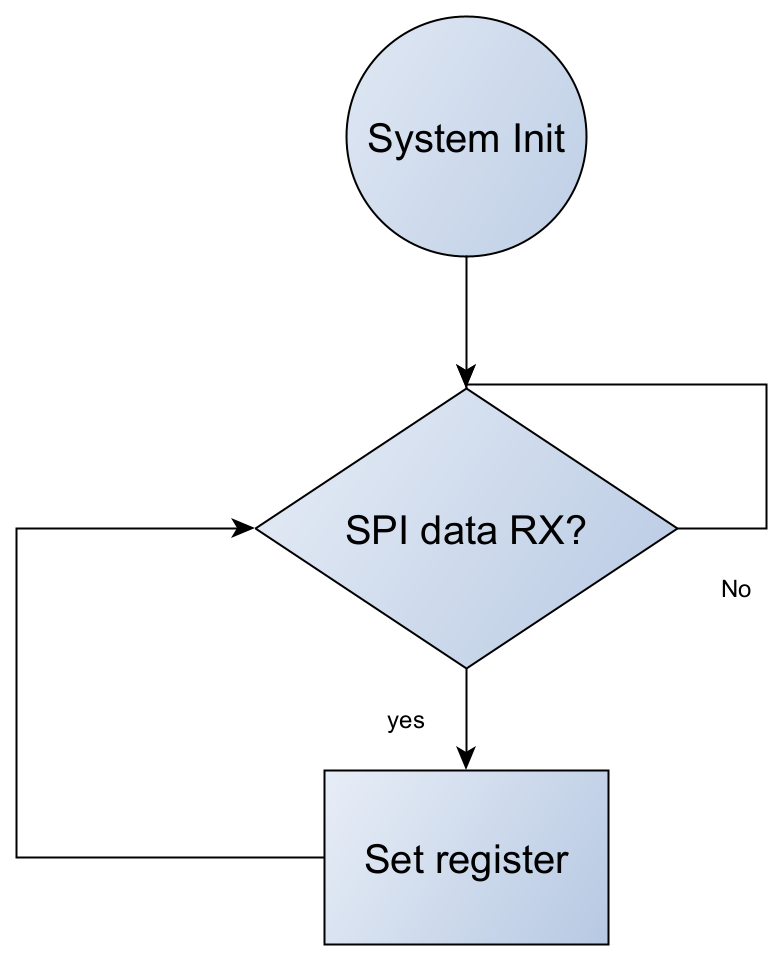


Symulacja dekodera kwadradurowego:



### Oprogramowanie:

Oprogramowanie na softcore microblaze jest oparte o ciągły odczyt interfejsu SPI i ustawianie odpowiednich rejestrów w kanałach wyjściowych. Wszystkie akcje są logowane i dostępne poprzez interfejs UART. Oprogramowanie zostało przygotowane za pomocą IDE Vitis który pozwala na przygotowanie szkieletu projektu na podstawie tego jak zrealizowany jest system cyfrowy. Do typowych IP Core jak np. SPI czy UART dostępne są gotowe biblioteki.



Odczyt interfejsu SPI jest wykonany w trybie pollingu. Komendy wysyłane są w konfiguracji: **[kanał]**, **[numer rejestru], [akcja], [wartość]**. Przykładowo aby ustawić rejestr nr 1 w kanale 1, wartością 0xAA, należy wysłać do kontrolera dane: 0x01, 0x01, 0x01, 0x000000AA, dla uproszczenia zakładamy że rozmiar wartości jest zawsze taki sam i wynosi 32 bitowy, a pozostałe 8 bitowe.

Dostępne numery rejestrów:

* 0x01 - rejestr kontrolny kanału
* 0x02 - rejestr fazy kanału
* 0x03 - rejestr delty kanału

Możliwe akcje:

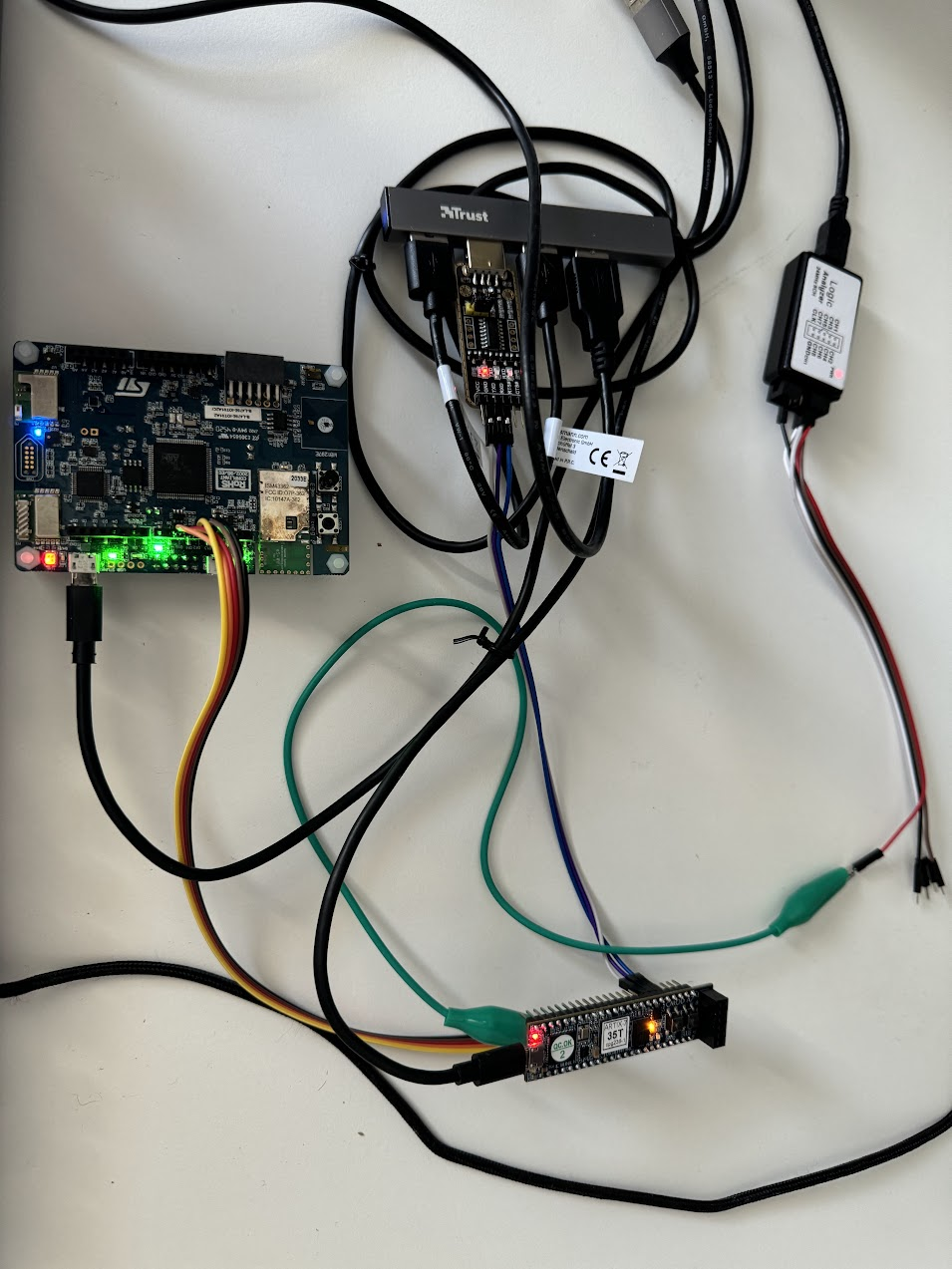
* 0x01 - zapis
* 0x02 - odczyt

Rejestr kontrolny służy wyzwoleniu generacji sinusa.

### Testy:

Testy zostały wykonane z wykorzystaniem modułu CMOD-A7 oraz modułu STM32 L475 [3] oraz ILA (integrated logic analyzer). Moduł STM32 wysyłał komendy do CMOD-A7 które generowały odpowiednie przebiegi na wyjściu modułu. Ze względu na wczesny etap projektu i brak modułów ewaluacyjnych dla przetwornika cyfrowo analogowego TLV5637 zastosowano moduł ILA który pozwala na weryfikacje przebiegów wewnątrz układu FPGA.

Stanowisko testowe:



### Kolejne kroki:

Dalszym etapem projektu będzie realizacja PCB do docelowego urządzenia oraz testy.

Referencje:

1. <https://docs.amd.com/r/en-US/ug958-vivado-sysgen-ref/DDS-Compiler-6.0>
2. <https://digilent.com/reference/programmable-logic/cmod-a7/start>
3. <https://www.st.com/en/evaluation-tools/b-l475e-iot01a.html#documentation>
4. <https://www.ti.com/lit/ds/symlink/tlv5637.pdf?ts=1719410016110&ref_url=https%253A%252F%252Fwww.google.com%252F>