RAPORT

# Lista członków zespołu

1. Gronowski Adrian
2. Klimaszewski Szymon
3. Krupski Konrad Tomasz
4. Lisovyi Ivan

# Nazwy jednostek SPI

## spi\_exe\_unit\_1

Realizacja: Konrad Krupski

## spi\_exe\_unit\_2

Realizacja: Ivan Lisovyi

## spi\_exe\_unit\_3

Realizacja: Szymon Klimaszewski

## spi\_exe\_unit\_4

…

# Decyzje projektowe

## Komunikacja równoległa pomiędzy master – slave

Jako zespół projektowy zdecydowaliśmy, aby komunikacja między master a slave odbywała się w sposób równoległy. Działanie to miało na celu ograniczyć liczbę bitów potrzebnych do komunikacji pomiędzy układami. Jeżeli komunikacja miałby się odbywać w sposób szeregowy to liczba bitów przechowywanych w rejestrze master’a oraz slave’a musiałaby wynosić 80. Dzięki realizacji komunikacji równoległej, ta liczba została ograniczona do 20.

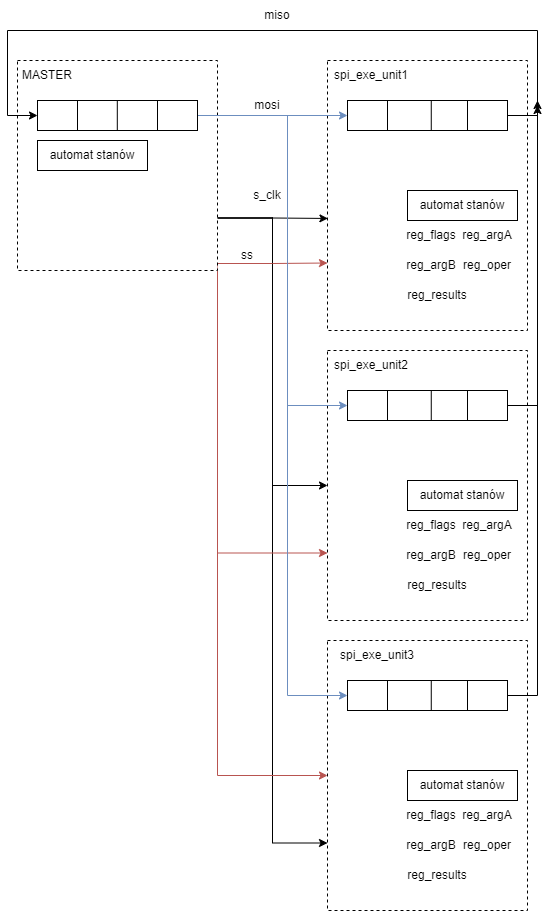
## 20-bitowe rejestry slave’a i master’a

Realizując projekt zdecydowaliśmy się na 20-bitowe rejestry slave‘a i master’a. Wynika to z faktu, że zakładając pracę jednostki exe\_unit na 4 bitach, potrzebujemy 20 bitów na zapisanie informacji o i\_argA, i\_argB, i\_oper oraz o\_result, o\_flags.

## Większa liczba stanów w jednostce master

Jako zespół projektowy zdecydowaliśmy się na zaimplementowanie dodatkowego stanu STATE\_LATE w jednostce master. Stan ten ma na celu opóźnienie przesyłania informacji pomiędzy slave a master. Ponieważ slave zawsze potrzebuje dodatkowego cyklu zegara ze względu na przejście do stanu gdzie wsuwane są rejestry argumentów do exe\_unit’a

## Schemat blokowy



## Przesuwanie bitów w rejestrach master’a i slave’a

Podczas projektu została podjęta decyzja o stronie, w którą dane są przesuwane w master oraz slave. W jednostkach master oraz slave zdecydowaliśmy się, że dane będą przesuwane bitowo w prawą stronę.

## 4-bitowe dane w jednostkach exe\_unit

W celu realizacji wspólnej komunikacji mastera z jednostkami spi\_exe\_unit ustaliliśmy, żeby exe\_unit każdego członka pracowało na danych 4-bitowych. Pozwoliło to na ustalenie ilu bitowe mają być rejestry w masterze oraz slave’ie. Niemniej jednak w celu osiągnięcia pracy na większej ilości bitów należy odpowiednio skonfigurować exe\_unit oraz poszczególne jednostki.

# Specyfikacja Master ’a

## Wejścia

i\_slave\_select

i\_clk – sygnał sterujący zegarem master’a

i\_rst – sygnał, który odpowiada za przywrócenie master’a do stanu początkowego

i\_data – sygnał, który jest wejściem równoległym

i\_send – sygnał pozwalający na wysyłanie danych

i\_miso – master in slave out. Sygnał, który przechowuje dane pochodzące od slave’a

## Wyjścia

o\_data

o\_busy – sygnał mówiący o tym czy master aktualnie zajęty jest komunikacją z slave’em.

o\_mosi - master out slave in. Sygnał, który przechowuje dane wysyłane z master’a do slave’a

o\_sclk – sygnał, który steruje zegarem slave’a

o\_ss – select slave. Sygnał przechowujący wartość, która mówi o tym, z którym slave’em chcemy przeprowadzić komunikację.

## Parametry

BITS – wartość, mówiąca o ilości bitów, na której operuje master

SLAVES\_NUMBER – wartość, mówiąca o ilości modułów slave w procesie komunikacji

STATES\_NUM – wartość, mówiąca o ilość stanów automatu master’a

STATE\_READY – stan, w którym master jest gotowy do rozpoczęcia pracy. W momencie, gdy dostaje on sygnał informujący o potrzebie przesłania informacji przechodzi on do stanu następnego

STATE\_LOAD – stan, w którym master informuję moduł watchdog o ilości cykli potrzebnych do wykonania oraz wybiera, z którym slave’em ma przebiegać komunikacja

STATE\_HIGH – stan odpowiadający za generacje sygnału zegarowego

STATE\_LOW - stan odpowiadający za generacje sygnału zegarowego

STATE\_END – stan mówiący o końcu przesuwania danych w masterze

STATE\_SS – stan w którym wzbudzany jest slave

STATE\_LATE – stan wprowadzający opóźnienie w postaci pojedynczego taktu zegara

## Sygnały pomocnicze

s\_sout\_en – pozwolenie na wpis

s\_sout\_wrt – wpis równoległy bądź szeregowy do shiftera

s\_inter – sygnał mówiący o końcu pracy modułu watchdog

s\_watchdog\_we – sygnał, który przechowuje wartość wprowadzaną do modułu watchdog

s\_bit – sygnał przechowujący wartość w postaci pojedynczego bitu, który jest podawany na wyjście master’a

s\_bit\_in – sygnał przechowujący wartość w postaci pojedynczego bitu, który jest podawany z wejście master’a z konkretnego slave’a

## Instancje modułów

### shifter

Moduł jest odpowiedzialny za przesuwanie bitów otrzymanych z slave’a. Master otzymuje kolejno pojedyncze bity od slave’a i dzięki temu modułowi wartości są odpowiednio przesuwane w rejestrze master’a.

### watchdog

Moduł, które zadaniem jest pobudzenie master’a do pracy na 20-bitowym ciągu. Watchdog ma 20 cykli, dzięki którym możliwe jest zapisanie danych do odpowiednich rejestrów i wykonanie na nich odpowiednich operacji.