PUF

Dokumentacja

Mateusz Berliński

Opis projektu

Tematem projektu jest obsługa analogowego Joysticka przy pomocy dwóch kanałów ADC modułu PmodAD1. Informacje o położeniu gałki przekazywane są z joysticka poprzez sygnał analogowy, w którym napięcie odpowiada danej pozycji. Każda oś posiada osobne wyprowadzenie analogowe. Wartość napięcia dla każdej osi jest zapisywana jako wartość 12-bitowa przy pomocy dwóch przetworników analogowo-cyfrowych zawartych w PmodAD1. W zależności od wartości 12-bitowych odczytanych z ADC, które odpowiadają odchyleniu gałki zapalają się odpowiednie diody.

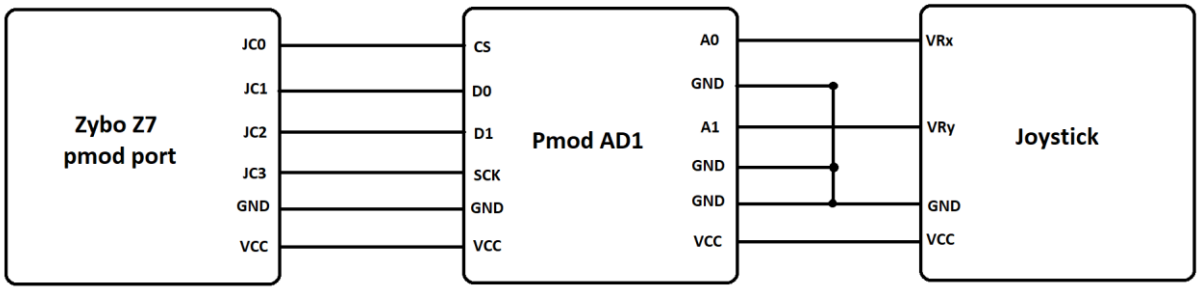
Wstęp teoretyczny

SPI (Serial peripheral interface) to synchroniczny interfejs oparty o komunikacje Master-Slave. Master inicjuje komunikacje, ustawiając na linii Slave Select (SS) stan niski i generuje zegar szeregowy (SCLK) zapewniający synchroniczną komunikację. Master przesyła dane przez linię Master-Out-Slave-In (MOSI) i odbiera dane przez linię Master-In-Slave-Out (MISO).

PmodAD1 komunikuje się z płytą Zybo poprzez protokół komunikacyjny podobny do SPI. Różnica pomiędzy standardowym protokołem SPI, a tym protokołem przejawia się w układzie pinów w tym Pmod. Typowy Interfejs SPI posiada sygnały Slave Select, Master-Out-Slave-In, Master-in-Slave-Out i sygnał zegara szeregowego. Jednak z dwoma przetwornikami ADC na tym chipie obie linie danych (MOSI i MISO) są zaprojektowane do działania tylko jako wyjścia, dzięki czemu są to obie linie danych Master-In-Slave-Out.

PmodAD1 dostarcza 12 bitów informacji do płyty systemowej przez 16 cykli zegara z pierwszymi czterema bitami składający się z czterech wiodących zer i pozostałych 12 bitów reprezentujących 12 bitów danych z MSB jako pierwszym. Pierwsze wiodące zero jest taktowane na opadającym zboczu sygnału CS z taktowanymi wszystkimi kolejnymi bitami na zboczu opadającym sygnału zegara szeregowego.

Schemat blokowy



JC0 – SCLK – zegar szeregowy

JC1 – MISO_0 – dane odbierane z pierwszego kanału PmodAD1

JC2 - MISO_1 – dane odbierane z drugiego kanału PmodAD1

JC3 – SS – wybór slave’a

Schemat PmodAD1:

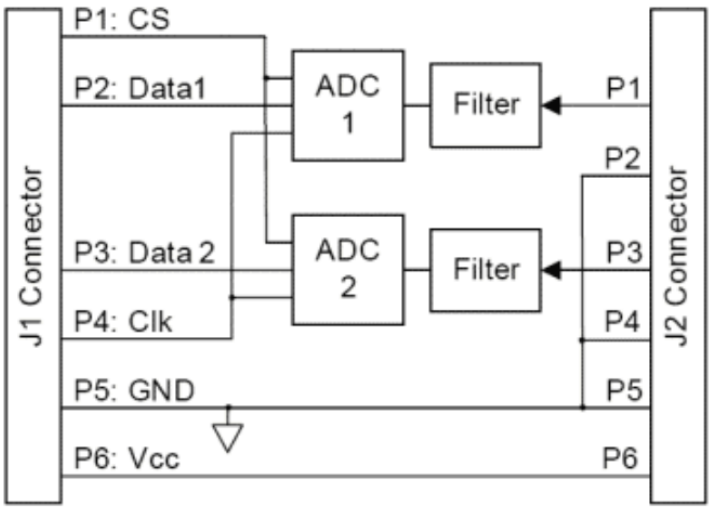
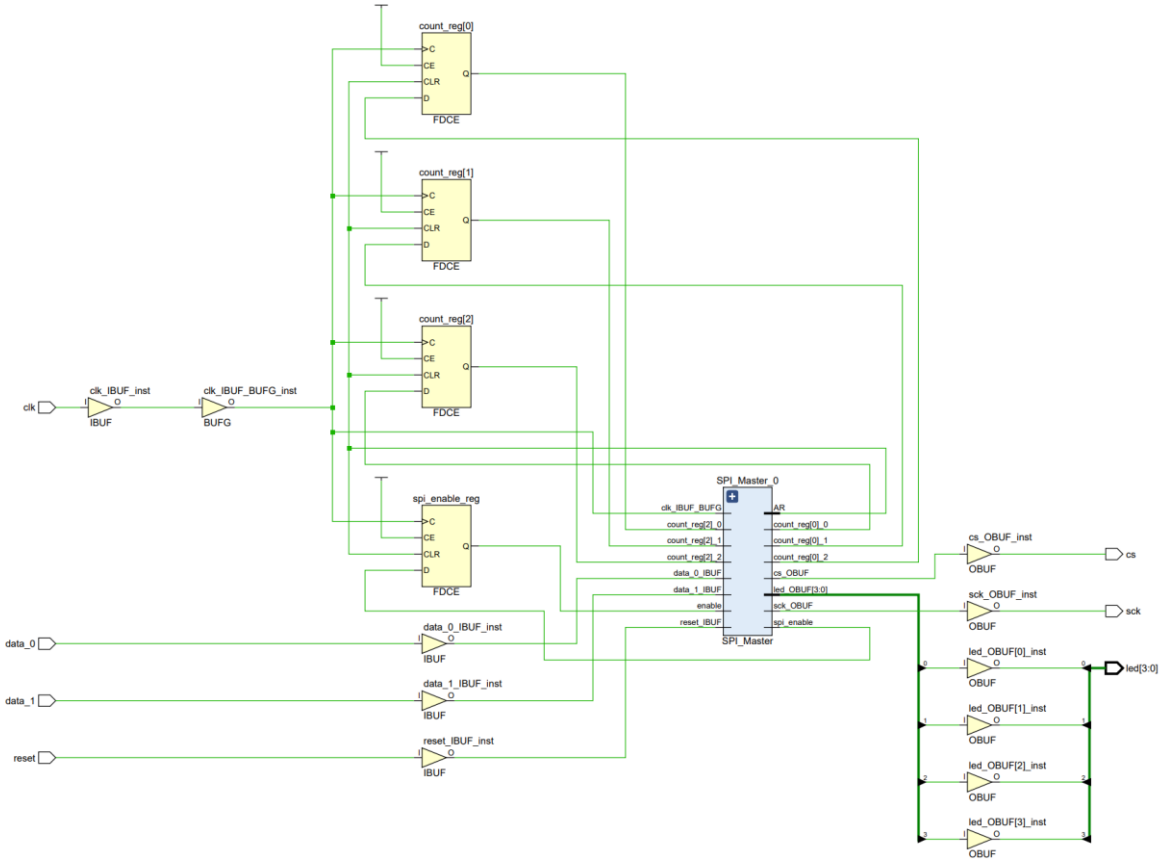


Figure 1. AD1 circuit diagram.

Schemat sprzętowy:



Opis komponentów

PmodAD1:

Główny komponent, który kontroluje działanie urządzenia. Rozpoczyna działanie SPI wstawiając logiczne 1 na enable. Przekazuje pojedyncze bity z data_0 i data_1 do komponentu SPI_Master, a odbiera 16-bitowe wartości z których zapisuje 12 bitów prawidłowych danych. Na podstawie tych wartości zapala diody.

clk – wejście zegara systemowego
reset – wejście przycisku btn0, który po aktywowaniu blokuje działanie
data_0 – wejście danych odbieranych z pierwszego kanału PmodAD1
data_1 – wejście danych odbieranych z pierwszego kanału PmodAD1
sck – bufor zegara szeregowego
cs – bufor wyboru slave'a
led – wyjście pinów led

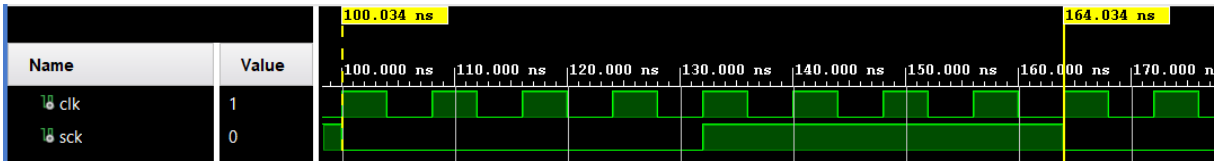
SPI_Master

Komponent, który komunikuje się z pmod. Gdy na enable pojawi się 1 to komponent przypisuje logiczne 1 do sygnału busy, który blokuje enable na czas obieranie danych i przypisuje logiczne 0 na linie CS aktywując adc. Generuje zegar szeregowy do komunikacji. Zapisuje bity odbierane z adc do 16-bitowych wartości i po zakończeniu odbierania wysyła je do komponentu PmodAD1.

clock – wejście zegara systemowego
reset – wejście przycisku btn0, który po aktywowaniu blokuje działanie
enable – wejście sygnału aktywujące SPI
miso_0 – wejście bitów danych z pierwszego adc
miso_1 – wejście bitów danych z drugiego adc
sclk – bufor zegara szeregowego
ss_n – bufor wyboru slave'a
busy – wyjście sygnału blokującego enable na czas odczytu
rx_data_0 – wyjście 16-bitowych wartości z pierwszego kanału adc
rx_data_1 – wyjście 16-bitowych wartości z drugiego kanału adc

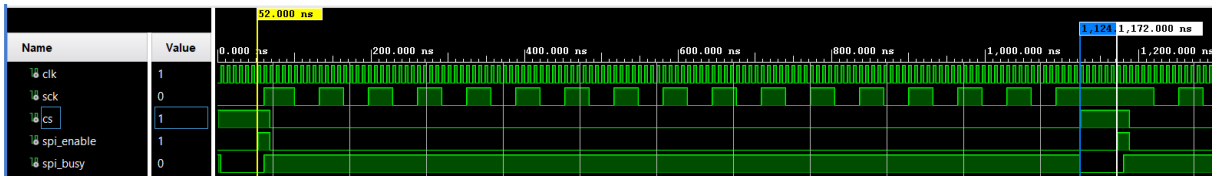
Symulacje

Zegar szeregowy SCK:



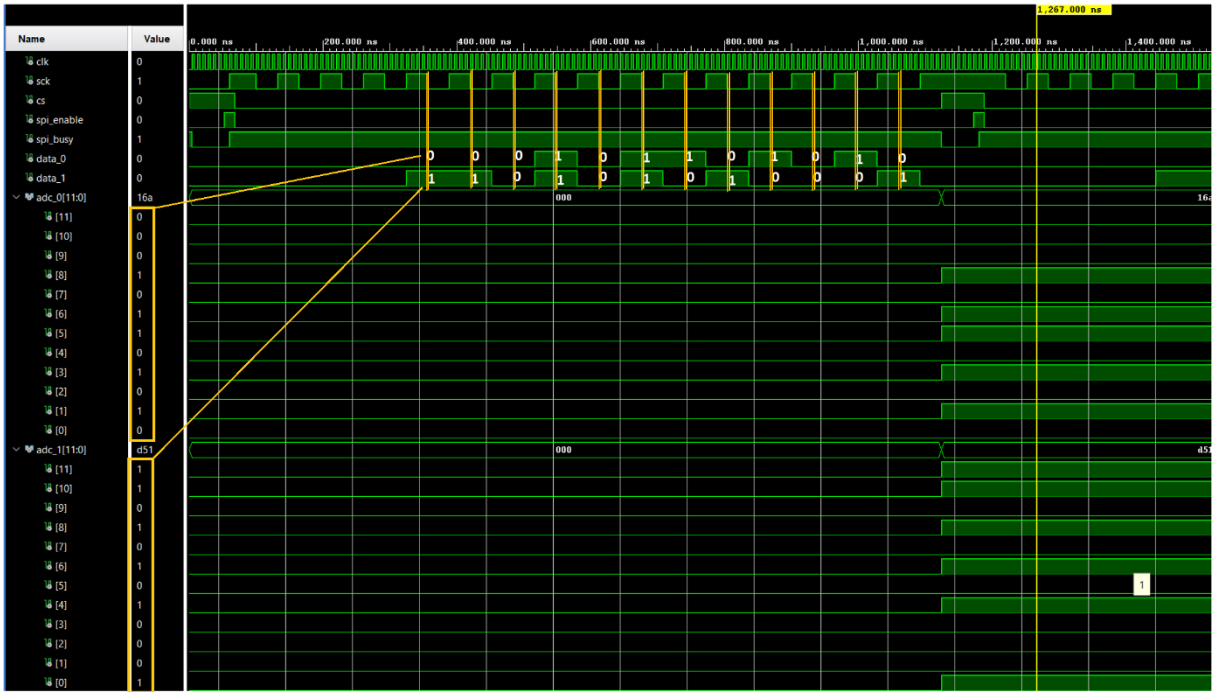
Maksymalna częstotliwość komunikacji z ADC wynosi 20MHz. Dzieląc sygnał clk uzyskałem sck o częstotliwości 15,6MHz (1/64ns).

Okno zbierania danych:



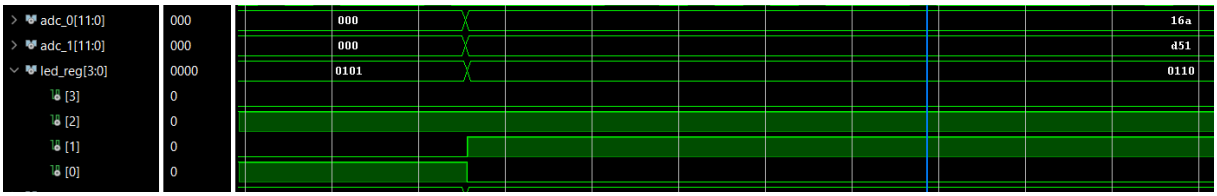
Gdy pojawia się 1 na fladze enable to do ADC zaczyna być wysyłany sygnał sck, na fladze busy pojawia się 1 i trwa przez 16 cykli zegara sck, a cs zmienia wartość na 0 i aktywuje zbieranie danych. Po 16 cyklach sck program czeka 50ns na zakończenie konwersji i rozpoczyna nowy proces zbierania danych. Częstotliwość próbkowania wynosi 0.9 milionów próbek na sekundę (1/1120ns)

Odbieranie danych:



Sygnały data_0 i data_1 symulują przykładowe dane odbierane z ADC. Stan sygnałów jest odczytywany przy opadającym zboczu sck i zapisywany w buforze. Po odebraniu wszystkich bitów gotowe wartości 12-bitowe zapisywane są w adc_0 i adc_1.

Zapalanie odpowiednich diod:



Przeliczając na liczby całkowite adc_0 i adc_1 przechowują liczby z zakresu 0 – 4095, które w naszym przypadku odpowiadają odchyleniu gałki joysticka w 2 kierunkach. Diody led0 i led2 zapalone są, gdy jeszcze w adc_1 i adc_0 nie ma wartości odczytanych z ADC (są same zera), ponieważ takie dane płytka czyta jako odchylenie gałki joysticka. Po odczytaniu danych w adc_1 jest wartość 362 led0 pozostaje z zapalony, a w adc_0 pojawia się wartość 3409 więc zapala się led1 i gaśnie led0.

Wartości adc_0 i adc_1 dla których zapalone są diody:

Led0 – adc_1 < 1024

Led1 – adc_1 > 3071

Led2 – adc_0 < 1024

Led3 – adc_0 > 3071

Test działania joysticka:

```
1
2  import machine
3  import utime
4
5  analog_value1 = machine.ADC(28)
6  analog_value2 = machine.ADC(27)
7
8
9  while True:
10     reading1 = analog_value1.read_u16()
11     voltage1 = reading1*3.3/65536
12     print("ADC1:",reading1,"= ",voltage1,"V")
13     utime.sleep(0.5)
14     reading2 = analog_value2.read_u16()
15     voltage2 = reading2*3.3/65536
16     print("ADC2:",reading2,"= ",voltage2,"V")
17     utime.sleep(0.5)
```

```
Shell
ADC2: 34104 =  1.717273 V
ADC1: 65311 =  3.28867 V
ADC2: 34296 =  1.726941 V
ADC1: 65311 =  3.28867 V
ADC2: 33640 =  1.693909 V
ADC1: 65311 =  3.28867 V
ADC2: 34024 =  1.713245 V
ADC1: 33832 =  1.703577 V
ADC2: 33816 =  1.702771 V
ADC1: 288 =  0.01450195 V
ADC2: 33800 =  1.701965 V
ADC1: 288 =  0.01450195 V
ADC2: 33736 =  1.698743 V
ADC1: 272 =  0.01369629 V
```

Przetestowałem działanie joysticka z pomocą dwóch ADC z Raspberry pi pico i zasilaniem 3.3V. Tak jak zakładano odchylanie gałki joysticka zmienia poziom napięcia w zakresie 0-3.3V na pinach odpowiadających każdej z osi.

Wnioski

Moduł analogowo-cyfrowy PmodAD1 umożliwia odczyt wartości napięcia z różnych źródeł i przetwarza je na cyfrowe wartości. Poprzez odczyt wartości na odpowiednich kanałach, można monitorować pozycję joysticka. Zastosowanie analogowego joysticka otwiera możliwości interakcji z użytkownikiem. Można wykorzystać odczytane wartości napięcia z joysticka do sterowania różnymi funkcjami projektu. Projekt ten wymaga integracji sprzętu i programowania, a symulacje są ważne do osiągnięcia pożądanych rezultatów.