****

**Politechnika Poznańska  
Wydział Elektroniki i Telekomunikacji  
Kierunek Elektronika i Telekomunikacja**

**PRACA MAGISTERSKA**

**Temat: Moduł sterownika kart pamięci SD/MMC dla układów FPGA**

*ang. SD/MMC memory card controller for FPGA devices.*

**Autor: inż. Marcin Brach**

**Promotor: dr inż. Olgierd Stankiewicz**

**Poznań, 2019**

Spis treści

[1. Wprowadzenie 3](#_Toc26477109)

[1.1. Cel pracy 3](#_Toc26477110)

[1.2. Zakres pracy i założenia 3](#_Toc26477111)

[2. Komunikacja z kartą pamięci SD 4](#_Toc26477112)

[2.1. Protokół SPI 4](#_Toc26477113)

[2.2. Standard kart pamięci SD i MMC 6](#_Toc26477114)

[2.3. Komendy odsługujące kartę 9](#_Toc26477115)

[2.4. Komunikacja z kartą 11](#_Toc26477116)

[3. Architektura projektowanych modułów 12](#_Toc26477117)

[3.1. Kontroler SPI 12](#_Toc26477118)

[3.2. Moduł sterownika karty pamięci 13](#_Toc26477119)

[3.3. Moduł testujący 16](#_Toc26477120)

[4. Symulacja i testowanie sterownika karty SD 18](#_Toc26477121)

[4.1. Symulacja kontrolera SPI 18](#_Toc26477122)

[4.2. Symulacja sterownika karty pamięci 19](#_Toc26477123)

[4.3. Sposób testowania układu 20](#_Toc26477124)

[4.4. Otrzymane wyniki testów 21](#_Toc26477125)

[5. Podsumowanie 21](#_Toc26477126)

[6. Bibliografia 21](#_Toc26477127)

# Wprowadzenie

Wiele współczesnych aplikacji sprzętowych np. telefony komórkowe, kamery, aparaty fotograficzne, rejestratory wizyjne, odtwarzacze mp3, itp. wymagają przechowywania danych w pamięci masowej. Podczas gdy pojemności pamięci dynamicznych są zbyt małe i nie przechowują informacji po odłączeniu zasilania a używanie dysków twardych jest często niemożliwe ze względu na duże gabaryty i wstrząsy. Z kolei układy scalone z pamięcią flash mają zbyt małe pojemności a karty pamięci Compact Flash są wypierane z rynku dlatego, w wielu zastosowaniach najczęściej stosowanym rozwiązaniem jest wykorzystanie kart pamięci SD (Secure Digital) i MMC (Multi Media Card).

## Cel pracy

Celem pracy jest zaprojektowanie oraz implementacja modułu sterownika obsługi kart pamięci w języku Verilog. W tym celu należy zapoznać się z protokołami komunikacyjnymi oraz standardami kart pamięci. Dokonać implementacji wybranego interfejsu oraz całego modułu dla układu FPGA. Wykonać symulację oraz opracować sposób testowania projektowanego układu z użyciem płytki laboratoryjnej. A do samej implementacji i symulacji użyć oprogramowania Lattice Diamond 3.10 oraz Active HDL 10.5.

## Zakres pracy i założenia

Do zakresu pracy należy:

- Analiza działania protokołów komunikacyjnych dla kart SD/MMC,

- Opracowanie koncepcji działania modułu sterownika,

- Implementacja w języku Verilog,

- Opracowanie środowiska testowego (symulacyjnego i sprzętowego) umożliwiającego weryfikację implementacji,

- Przeprowadzenie badań dotyczących wydajności zapisu i odczytu danych na wybranych modelach kart pamięci.

Na podstawie celu oraz zakresu pracy założyłem że implementacje wykonam dla protokołu SPI (Serial Peripheral Interface) z wykorzystaniem karty micro SD (Secure Digital).

# Komunikacja z kartą pamięci SD

## Protokół SPI

SPI (Serial Peripheral Interface) jest dupleksowym, szybkim, synchronicznym interfejsem szeregowym. Kiedyś używany był do łączenia urządzeń takich jak drukarka, skaner, kamera itp. do komputera osobistego PC (Personal Computer). Dziś został wyparty przez standard USB (Universal Serial Bus) ale w dalszym ciągu jest używany do komunikacji między aplikacjami używającymi monitorów, kard pamięci, czujników itp. Standard ten pozwala na łączenie jednego układu nadrzędnego (Master) z wieloma urządzeniami pobocznymi (Slave) do czego wykorzystywane są cztery linie sygnałowe.



Rysunek 1 Sposób łączenia układów poprzez interfejs SPI.  
(Źródło: http://elm-chan.org/docs/spi\_e.html)

Rysunek 1 przedstawia sposób komunikacji pomiędzy modułami (nadrzędnym i podrzędnym) poprzez interfejs SPI. Obydwa układy połączone są liniami MOSI (Master-Out Slave-In), MISO (Master-In Slave-Out), SCLK (Serial Clock) i SS (Slave Select). Dane z obydwóch 8-bitowych rejestrów przesuwających (Shift register) przesyłane są pomiędzy urządzeniami w takt sygnału zegarowego SCLK. Natomiast sygnał SS pełni rolę wyboru układu podrzędnego (chip select) oraz synchronizacji startu transmisji. Dla układów z transmisją jednokierunkową tj. przetwornik analogowo cyfrowy (ADC) lub przetwornik cyfrowo-analogowy (DAC) druga linia transmisji może być pominięta. Bity danych przesyłane są od najbardziej znaczącego bitu (MSB) do najmniej znaczącego (LSB).



Rysunek 2 Wykorzystanie interfejsu SPI z wieloma modułami Slave.  
(Źródło: https://en.wikipedia.org/wiki/Serial\_Peripheral\_Interface)

Rysunek 2 przedstawia przykładowe podłączenie trzech układów podrzędnych. Zapis lub odczyt następuje po ustawieniu wyjścia układu SPI Master na stan niski. W interfejsie SPI przesunięcie i zatrzaśnięcie się danych podczas zapisu bądź odczytu odbywa się na przeciwnych zboczach zegara. W związku z tym wyróżniamy cztery różne tryby pracy które wynikają z konfiguracji interfejsu SPI po stronie układu nadrzędnego. Tryby te przedstawia tabela 1.

|  |  |
| --- | --- |
| Tryby pracy SPI | Przebiegi czasowe |
| Tryb 0  Zbocze narastające  Zatrzask i przesunięcie  (CPHA=0, CPOL=0) |  |
| Tryb 1  Zbocze narastające  Przesunięcie i zatrzask  (CPHA=1, CPOL=0) |  |
| Tryb 2  Zbocze opadające  Zatrzask i przesunięcie  (CPHA=0, CPOL=1) |  |
| Tryb 3  Zbocze opadające  Przesunięcie i zatrzask  (CPHA=1, CPOL=1) |  |

Tabela 1 Cztery tryby pracy interfejsu SPI.  
(Źródło: http://elm-chan.org/docs/spi\_e.html)

## Standard kart pamięci SD i MMC

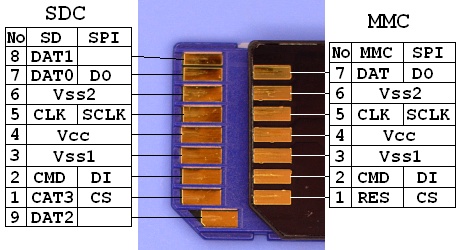
Do najbardziej popularnych standardów kart pamięci należą karty SD (Secure Digital) oraz karty MMC (Multi Media Card). Karta SD została opracowana jako zgodna z większością kart MMC dlatego sprzęt zgodny z kartami SD może w większości przypadków korzystać z MMC. Dostępne są również wersje o zmniejszonych rozmiarach, takie jak RS-MMC, miniSD i microSD, z taką samą funkcjonalnością. Karty te mają wbudowaną pamięć flash i mikrokontroler. Sterowanie tej karty pamięci tj. kasowanie, odczyt, zapis, sterowanie błędami i równoważenie zużycia jest wykonywane bezpośrednio na niej. Dane są przesyłane między kartą pamięci a kontrolerem hosta jako bloki danych w jednostkach 512 bajtów, dlatego karty te można postrzegać jako dyski twarde z punktu widzenia warstw wyższego poziomu (systemu operacyjnego).

Schemat blokowy karty SD pokazano na rys. 3. Składa się ona z 9-stykowego interfejsu (Interface Driver), wewnętrznego układu taktującego (Internal Clock), kontrolera karty (Card Interface Controller), wbudowanych rejestrów karty (OCR, CID, RCA, DSR, CSD, SCR), interfejsu rdzenia pamięci (Memory Card Interface) oraz rdzenia pamięci (Memory Core). 9-pinowy interfejs pozwala na wymianę danych między podłączonym systemem a kontrolerem karty. Kontroler może odczytywać lub zapisywać dane z lub do rdzenia pamięci za pomocą interfejsu rdzenia pamięci. Rejestry wewnętrzne przechowują aktualny stan karty. Kontroler odpowiada na dwa typy żądań użytkowników: kontrolę i dane. Żądania kontrolne konfigurują operację sterownika i umożliwiają dostęp do rejestrów. Natomiast żądania danych służą do odczytu danych lub zapisywania danych w rdzeniu pamięci.

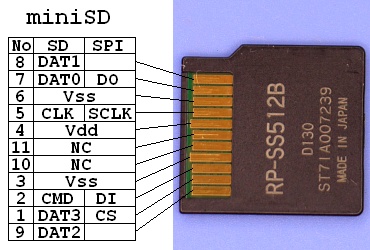


Rysunek 3 Schemat blokowy karty SD.  
(Żródło: http://www.dejazzer.com/ee379/lecture\_notes/lec12\_sd\_card.pdf)

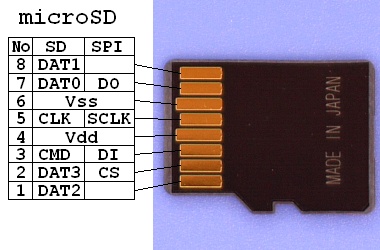
Karta SD posiada 9 pól stykowych czyli o 2 piny więcej niż karta MMC co widać na rysunku 4. Trzy z nich to styki przeznaczone dla zasilania dlatego liczba efektywnych pinów zarezerwowanych do komunikacji z kartą wynosi sześć dla karty SD oraz cztery dla karty MMC. Z tego powodu transfer danych pomiędzy hostem a kartą odbywa się za pomocą synchronicznego interfejsu szeregowego. Zakres napięcia zasilania przechowywany jest w rejestrze OCR który powinien być odczytany i potwierdzony podczas inicjalizacji karty. Jednak napięcie zasilania może być również ustawione na stałe w zakresie 3,0 V do 3,3 V bez potwierdzenia ponieważ wszystkie karty SD i MMC działają w przedziale 2,7 V do 3,6 V. Pobór prądu przez kartę pamięci podczas operacji zapisu może osiągnąć maksymalnie 100 mA dlatego nie należy podłączać większego napięcia niż dopuszczalne (np. 5V) gdyż spowoduje to uszkodzenie karty a układ zasilania karty powinien dostarczać maksymalnie 100mA.



Rysunek 4 Opis wyprowadzeń styków dla karty SD po lewej stronie oraz dla karty MMC po prawej.  
(Źródło: http://elm-chan.org/docs/mmc/mmc\_e.html#fsys)



Rysunek 5 Opis wyprowadzeń styków dla karty miniSD.  
(Źródło: http://elm-chan.org/docs/mmc/mmc\_e.html#fsys)



Rysunek 6 Opis wyprowadzeń styków dla karty microSD.  
(Źródło: http://elm-chan.org/docs/mmc/mmc\_e.html#fsys)

## Komendy odsługujące kartę

Komunikacja z kartą SD odbywa się poprzez wysyłanie do niej odpowiednich komend i otrzymywaniu odpowiedzi. Karta może pracować w dwóch trybach: SD oraz SPI. Domyślnym trybem pracy jest tryb SD dlatego aby można było nawiązać komunikację z kartą w trybie SPI należy ją ustawić w ten tryb. Prawidłowa komenda obsługująca kartę SD składa się z 48 bitów i jest przedstawiona na rysunku 7.



Rysunek 7 Format 48-bitowej komendy.  
(Źródło: http://www.dejazzer.com/ee379/lecture\_notes/lec12\_sd\_card.pdf)

Pierwsze dwa najbardziej znaczące bity to bity startu które ostawione są na wartość 01 dla każdej komendy. Następnym polem jest 6-bitowy numer komendy (Command Number) który określa rodzaj wykonywanej operacji na karcie SD. Dalej znajduje się 32-bitowy argument z wartością oraz 7-bitowe pole CRC (Cyclic Redundancy Chceck) oraz bit stopu ustawiony na wartość 1 dla każdej operacji. Wartość CRC jest używana dla weryfikacji integralności otrzymanej odpowiedzi. Domyślnie karta SD ignoruje bity CRC dla większości komend (za wyjątkiem komendy CMD8) chyba że użytkownik zażąda sprawdzenia bitów CRC po każdej wiadomości.



Rysunek 8 Format podstawowej 8-bitowej odpowiedzi na każdą komendę w trybie SPI.  
(Źródło: http://www.dejazzer.com/ee379/lecture\_notes/lec12\_sd\_card.pdf)



Rysunek 9 Format 40-bitowej odpowiedzi.  
(Źródło: http://www.dejazzer.com/ee379/lecture\_notes/lec12\_sd\_card.pdf)

Po każdej wysłanej komendzie karta wysyła odpowiedź która może być jedną z kilku formatów (R1, R1b, R2, R3, R7) w zależności od użytej komendy. Podstawowym formatem jest format R1 który jest 8-bitowy (rysunek 8). Format R1b jest dostępny tylko dla wybranych kart i jest podobny do formatu R1 z tym że posiada dodatkowo sygnał zajętości przesyłanym przez linię danych. Karta może stać się zajęta po otrzymaniu tych poleceń a gdy karta zwróci niezerowy bajt wówczas jest gotowa na kolejne polecenie. Format R2 służy do zwracania statusu wewnętrznych rejestrów karty CID oraz CSD. Format R3 jest natomiast 40-bitowy (rysunek 9) i służy do zwracania rejestru OCR. Format R7 natomiast wykorzystywany jest do przesłania odpowiedzi na polecenie CMD8. Składa się on z pięciu bajtów i zawiera informację o obsłudze dostarczanego napięcia. Format ten zwraca zakres napięć oraz wzorzec kontrolny ustawiony w argumencie. Odpowiedź dla formatu R1 przedstawia rysunek 8 i zawsze rozpoczyna się od zera. Następnie znajdują się znaczniki flag oznaczające: błąd podanego parametru (parameter error), błąd podanego adresu (address error), kasowanie błędu sekwencji (erase sequence error), błąd sumy kontrolnej CRC (CRC error), niepoprawna komenda (illegal command), kasowanie resetu (erase reset), karta w stanie bezczynności (in idle state). W przypadku odpowiedzi 40-bitowej dla formatu R3 po za polami z R1 otrzymujemy dodatkowe pola tj. 4-bitowe pole z wersją (Version), 18-bitowe zarezerwowane pole (Reserved), 4-bitowe pole określające napięcie karty (Voltage) oraz 8-bitowe pole kontrolne (Check Pattern).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nazwa komendy** | **Argument** | **Format odpowiedzi** | **Dane** | **Nazwa skrótowa** | **Opis** |
| CMD0 | None(0) | R1 | Nie | GO\_IDLE\_STATE | Programowy reset karty. |
| CMD1 | None(0) | R1 | Nie | SEND\_OP\_COND | Inicjacja procesu inicjalizacji. |
| ACMD41  (\*) | \*\* | R1 | Nie | APP\_SEND\_OP\_COND | Inicjacja procesu inicjalizacji (tylko dla SD) |
| CMD8 | \*\*\* | R7 | Nie | SEND\_IF\_COND | Sprawdzanie zakresu napięć (tylko dla SD V2) |
| CMD9 | None(0) | R1 | Tak | SEND\_CSD | Odczyt rejestru CSD. |
| CMD10 | None(0) | R1 | Tak | SEND\_CID | Odczyt rejestru CID. |
| CMD12 | None(0) | R1b | Nie | STOP\_TRANSMISSION | Stop odczytu danych. |
| CMD16 | Block  length[31:0] | R1 | Nie | SET\_BLOCKLEN | Zmiana odczytu lub zapisu długości bloku. |
| CMD17 | Address[31:0] | R1 | Tak | READ\_SINGLE\_BLOCK | Odczyt bloku. |
| CMD18 | Address[31:0] | R1 | Tak | READ\_MULTIPLE\_BLOCK | Odczyt wielu bloków. |
| CMD23 | Number of  blocks[15:0] | R1 | Nie | SET\_BLOCK\_COUNT | Definiowanie liczby bloków do transmisji przy następnej komendzie odczytu lub zapisu w trybie wielu bloków (tylko dla standardu MMC). |
| ACMD23  (\*) | Number of  blocks[22:0] | R1 | Nie | SET\_WR\_BLOCK\_ERASE\_COUNT | Definiowanie liczby bloków do skasowania przy następnej komendzie zapisu w trybie multi blokowości (tylko dla SD). |
| CMD24 | Address[31:0] | R1 | Tak | WRITE\_BLOCK | Zapis jednego bloku pamięci. |
| CMD25 | Address[31:0] | R1 | Tak | WRITE\_MULTIPLE\_BLOCK | Zapis wielu bloków. |
| CMD55 | None(0) | R1 | Nie | APP\_CMD | Ładowanie komendy ACMD<n> |
| CMD58 | None(0) | R3 | Nie | READ\_OCR | Odczyt rejestru OCR. |
| \* - komendy ACMD mogą być wykonane tylko po wcześniejszym wysłaniu komendy CMD55.  \*\*: Rsv(0)[31], HCS[30], Rsv(0)[29:0]  \*\*\*: Rsv(0)[31:12], Supply Voltage(1)[11:8], Check Pattern(0xAA)[7:0] | | | | | |

Tabela 2 Lista podstawowych komend obsługiwanych przez kartę pamięci.

## Komunikacja z kartą

Wysyłanie polecenia do karty SD odbywa się gdy linia MOSI jest ustawiona w stan wysoki co oznacza że żadna wiadomość nie została wysłana. Proces wysyłania wiadomości rozpoczyna się od umieszczenia najbardziej znaczącego bitu MSB (most-significant bit) na wyjściu MOSI przy jednoczesnym przełączaniu sygnału zegarowego SCLK z poziomu niskiego na wysoki oraz z wysokiego na niski. Powtórzenie tej procedury aż do najmniej znaczącego bitu LSB (least significant bit) pozwala na przesłanie zawartości całego rejestru do karty SD która składa się na całą komendę.

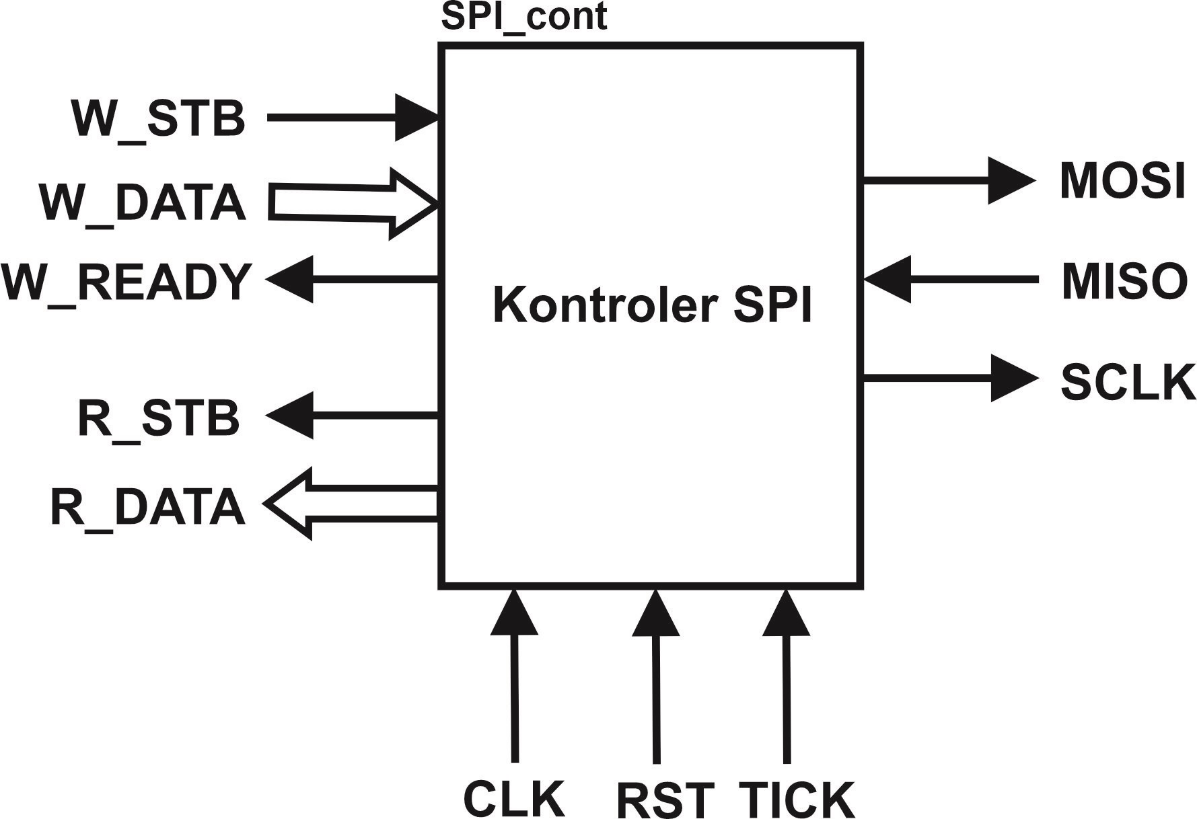
Aby zapewnić poprawne działanie karty SD, zegar taktujący SCLK powinien zawierać się w przedziale 100 – 400 kHz a karta musi być ustawiona w trybie SPI. Żeby ją ustawić w ten tryb linie MOSI oraz CS muszą być ustawione w stan logicznej jedynki przynajmniej przez 74 takty zegarowe. Po tym czasie należy ustawić CS na zero oraz do karty wysłać komendę resetu (CMD0).

Gdy tylko karta SD otrzyma komendę rozpoczyna ją procesować. Dla poprawnej odpowiedzi na polecenie karta SD wymaga przełączania sygnału SCLK przez co najmniej 8 taktów i obserwacji wejścia MISO. W tym czasie linia MOSI znajduje się z wysokim poziomie a CS w poziomie niskim. Długość odpowiedzi wynosi zwykle 8-bitów lub 40-bitów w zależności od użytej komendy. Detekcja otrzymywanej wiadomości jest możliwa dzięki temu że każda otrzymywana wiadomość rozpoczyna się od zera. Natomiast gdy karta nie wysyła odpowiedzi linia MISO jest ustawiona jako logiczna jedynka. Warto zauważyć że odpowiedź na każdą komendę jest wysyłane przez kartę kilka cykli SCLK później. Jeśli oczekiwana odpowiedź nie zostanie odebrana w ciągu 16 cykli zegara po wysłaniu polecenia resetu, polecenie to należy wysłać ponownie.

# Architektura projektowanych modułów

## Kontroler SPI

Projektowany kontroler SPI stanowi najmniejszy moduł projektowanego sterownika i odpowiada za transmisję SPI. Transmisja ta odbywa się w trybie 0 gdy przy zboczu narastającym pobierana jest próbka (Sample) a ustawienie wartości (Setup) przy zboczu opadającym. Rysunek 10 przedstawia schemat projektowanego modułu. Do wysłania danych do karty pamięci służy sygnał strobujący W\_STB wraz z 8-bitową magistralą danych W\_DATA która służy do wysłania danych do karty pamięci (na wyjście MOSI) a po zakończeniu transmisji ustawiana jest flaga na wyjściu W\_READY. Dane odpierane z wejścia MISO wystawiane są na 8-bitową magistralę R\_DATA wraz z sygnałem strobującym R\_DATA. Układ posiada ponadto wejście zegarowe CLK wraz z resetem RST oraz sygnał TICK który pełni rolę opóźnionego zegara który służy do wyznaczania częstotliwości SCLK.



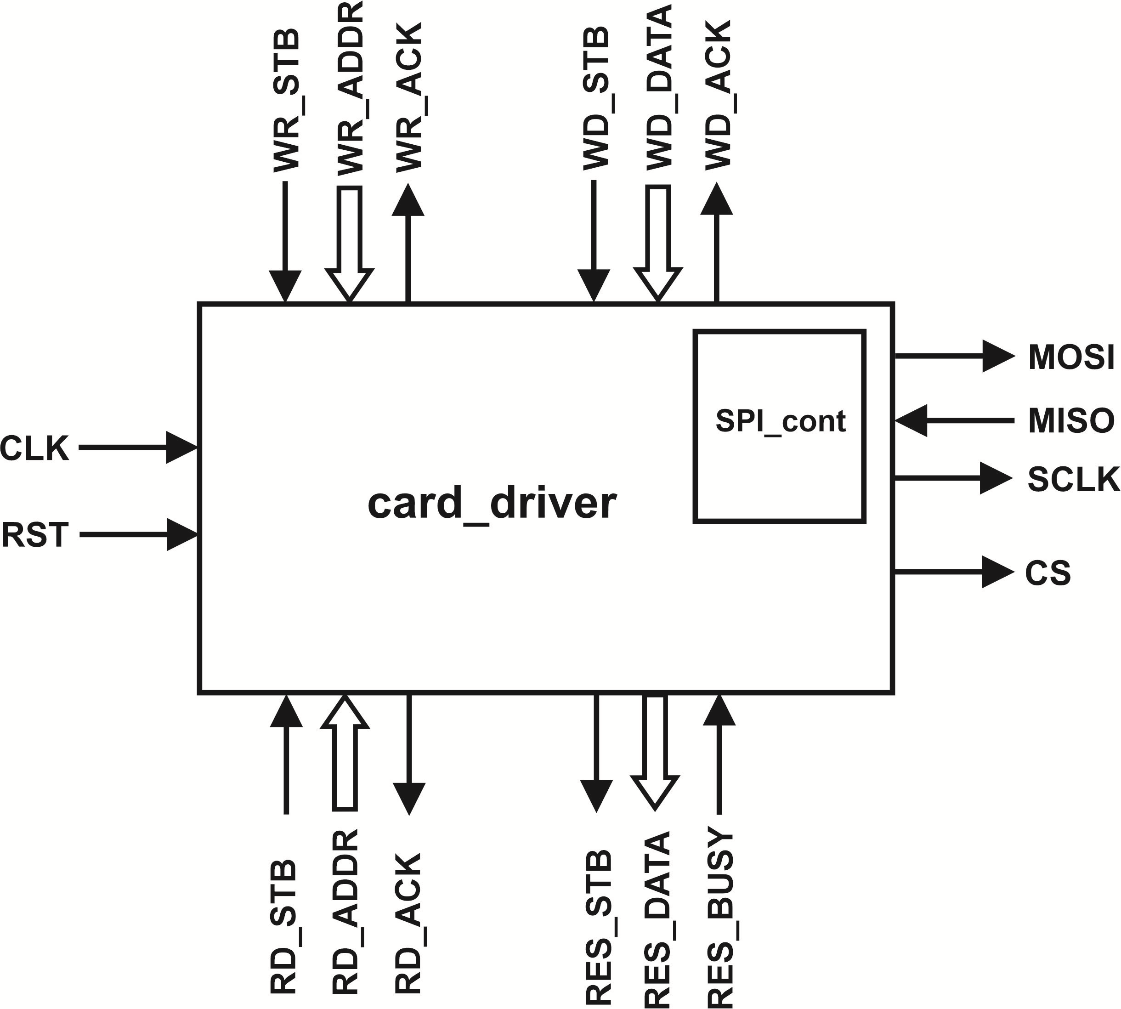
Rysunek 10 Schemat funkcjonalny modułu kontrolera SPI

|  |  |
| --- | --- |
| **Nazwa sygnału:** | **Opis:** |
| **CLK** | Wejście zegarowe |
| **RST** | Wejście resetujące układ (Reset) |
| **TICK** | Wejście opóźnionego zegara |
| **W\_STB** | Wejście strobujące zapis danych |
| **W\_DATA** | 8-bitowa magistarala danych zapisywanych |
| **W\_READY** | Wyjaście potwierdzający przesłanie danych |
| **R\_STB** | Wejście strobujące odczyt |
| **R\_DATA** | 8-bitowa magistrala danych odczytywanych |
| **MOSI** | Wyjście danych |
| **MISO** | Wejście danych |
| **SCLK** | Wyjście zegarowe |

Tabela 3 Opis wyprowadzeń sygnałów kontrolera SPI

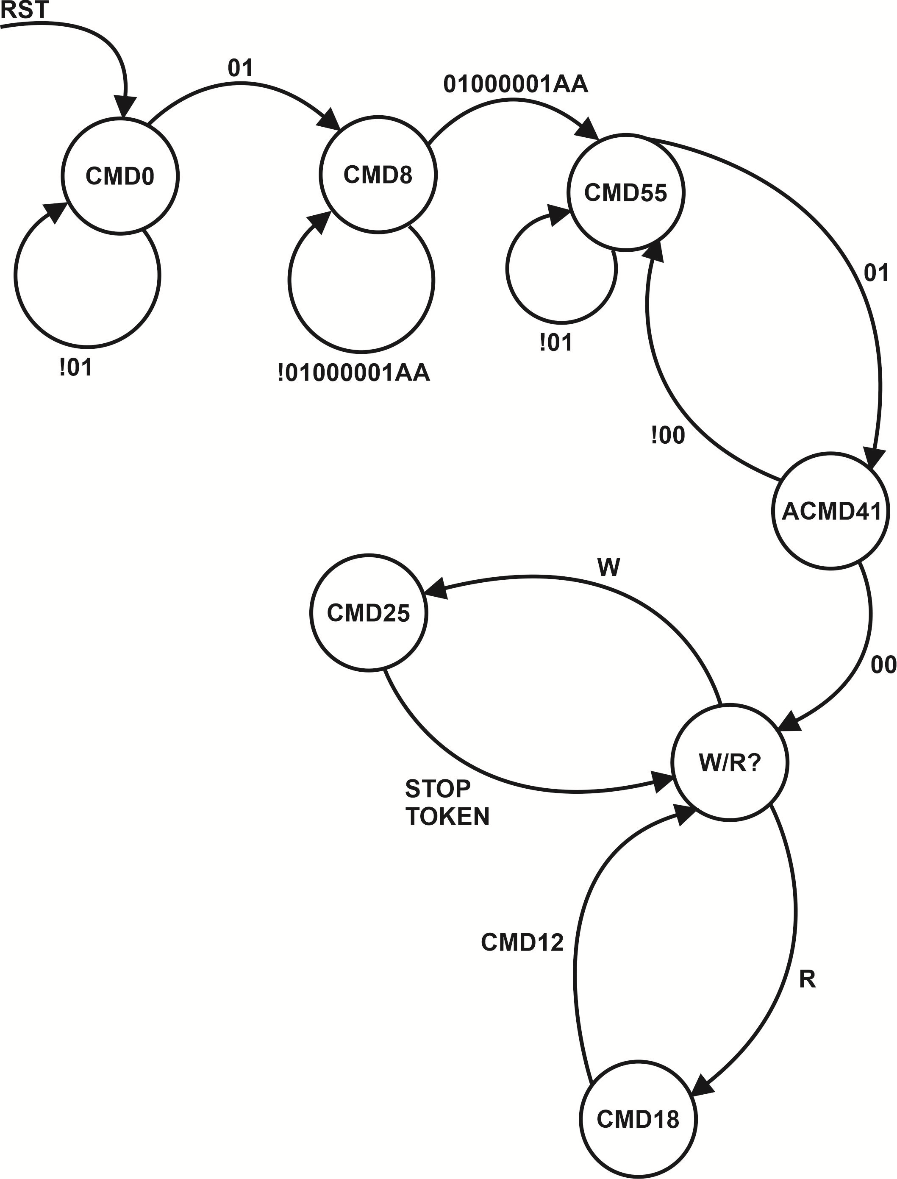
## Moduł sterownika karty pamięci

Sterownik kart pamięci stanowi automat wysyłający komendy inicjalizacji oraz obsługi zapisu i odczytu danych na kartę pamięci. Moduł ten posiada wejście zegarowe CLK, resetujące RST oraz CS (chip select) służące do wyboru układu na którym wykonywane są operacje. Sterownik card\_driver posiada wejście strobujące zapis adresu WR\_STB wraz z 32 bitowym wejściem do zapisu adresu WR\_ADDR oraz wyjściem WR\_ACK które jest ustawiane gdy adres zostanie przesłany do karty pamięci. Po wystawieniu tego sygnału do kontrolera przesyłane są dane a odbywa się to poprzez ustawienie sygnału strobującego WD\_STB przy jednoczesnym podaniu danych na 8-bitowe wejście WD\_DATA danych. Po przesłaniu minimum jednego bloku danych (domyślnie 512 bajtów) ustawiane jest wyjście potwierdzające zapis WD\_ACK. Aby odczytać dane należy ustawić sygnał strobujący odczyt RD\_STB przy jednoczesnym podaniu adresu na 32-bitowe wejście RD\_ADDR. Po zakończeniu odczytu wystawiana jest flaga RD\_ACK. Bezpośrednia komunikacja z kartą pamięci odbywa się poprzez użycie modułu kontrolera SPI (SPI\_cont) przy użyciu linii MOSI, MISO, SCLK.



Rysunek 11 Schemat funkcjonalny projektowanego sterownika karty pamięci SD

|  |  |
| --- | --- |
| **Nazwa sygnału:** | **Opis:** |
| **CLK** | Sygnał zegarowy |
| **RST** | Reset |
| **WR\_STB** | Wejście strobujące adres do zapisu danych |
| **WR\_ADDR** | 32-bitowe wejście adresowe do zapisu danych |
| **WR\_ACK** | Wyjście potwierdzające zapis |
| **WD\_STB** | Wejście strobujące zapis danych |
| **WD\_DATA** | 8-bitowe wejście danych do zapisu |
| **WD\_ACK** | Wyjście potwierdzające zapis danych |
| **RD\_STB** | Wejście strobujące adres do odczytu danych |
| **RD\_ADDR** | 32-bitowe wejście adresowe do odczytu danych |
| **RD\_ACK** | Wyjście potwierdzające odczyt |
| **RES\_STB** | Wyjście strobujące zapis danych |
| **RES\_DATA** | 8-bitowe wyjście dla odczytanych danych |
| **RES\_BUSY** | Wejście informujące o zajętości magistrali |
| **MOSI** | Wyjście danych |
| **MISO** | Wejście danych |
| **SCLK** | Wyjście zegarowe |
| **CS** | Sygnał wyjściowy Chip select |



Rysunek 12 Automat przedstawiający zachowanie sterownika karty pamięci

Na rysunku 12 znajduje się automat zgodnie z którym działa projektowany sterownik wysyłając odpowiednie komendy do karty i po otrzymaniu odpowiedzi przechodzi do następnego stanu. Proces inicjalizacji rozpoczyna się od zresetowania karty poprzez wysłanie komendy CMD0 postaci:

01000000 00000000 00000000 00000000 00000000 10010101

która ustawia kartę w stan idle mode i odpowiada 0x01. Następnie wysyłana jest komenda CMD8:

01001000 00000000 00000000 00000001 10101010 10000111

dla której odpowiedź wynosi 0x01000001AA co oznacza że karta SD jest w wersji co najmniej drugiej (ver. 2) i może pracować w przedziale napięcia 2.7 – 3.6 V. Kolejną wysłaną komendą jest CMD55:

01110111 00000000 00000000 00000000 00000000 11111111

Dla której karta odpowiada 0x01 i informuje ona kartę że następną wysłaną komendą będzie komenda specjalna. Po czym wysyłana jest komenda ACMD41:

01101001 01000000 00000000 00000000 00000000 11111111

Jeśli karta odpowie 0x00 oznacza to że proces inicjalizacji przebiegł pomyślnie i karta gotowa jest do wykonania operacji odczytu lub zapisu. Jeśli sygnał strobujący zapis jest aktywny wówczas wysyłana jest komenda CMD25 postaci:

01011001 ADDR[31:0] 11111111

gdzie ADDR[31:0] oznacza adres pod którym jest dokonywany zapis. Koniec tej operacji następuje po wysłaniu żetonu (tokenu) 11111101 co informuje kartę o końcu procesu zapisu. Jeżeli natomiast sygnał strobujący odczyt jest aktywny wtedy wysyłana jest komenda CMD18:

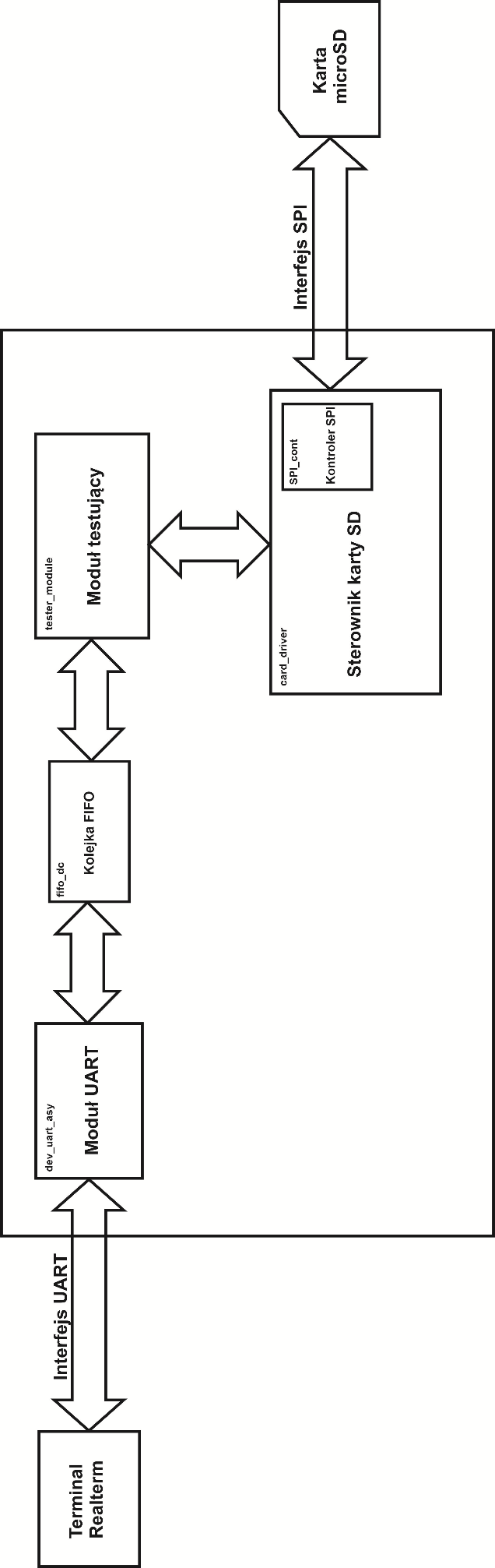
01010010 ADDR[31:0] 11111111

gdzie ADDR[31:0] oznacza adres spod którego dokonywany jest odczyt. Proces odczytu kończy się poprzez wysłanie komendy CMD12:

01001100 00000000 00000000 00000000 00000000 00000000

Warto zauważyć że proces zapisu ma wyższy priorytet niż odczyt dzięki temu zapisujące się dane nie zostaną utracone wskutek żądania odczytu.

## Moduł testujący



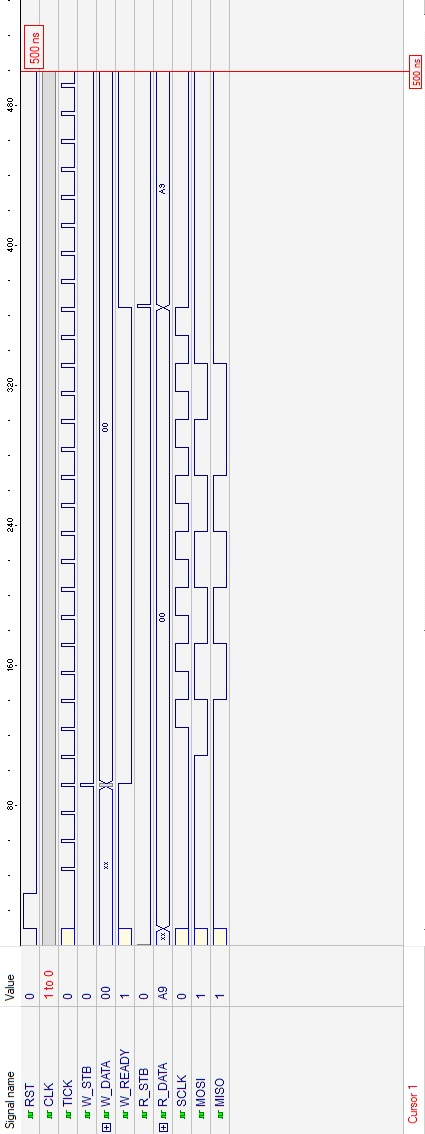
Rysunek 13 Schemat blokowy testowanego modułu sterownika

Aby poprawnie przetestować projektowany układ należało wykorzystać dodatkowe moduły dostępne na uczelni tj. moduł UART i kolejka FIFO oraz stworzyć moduł testujący który wyśle odpowiednie dane, adresy i stany na wejścia projektowanego modułu card\_driver. Rysunek 13 przedstawia schemat połączeń pomiędzy modułami od karty pamięci microSD do terminala na którym obserwowane były wyniki podczas testowania modułu w sprzęcie.

# Symulacja i testowanie sterownika karty SD

## Symulacja kontrolera SPI

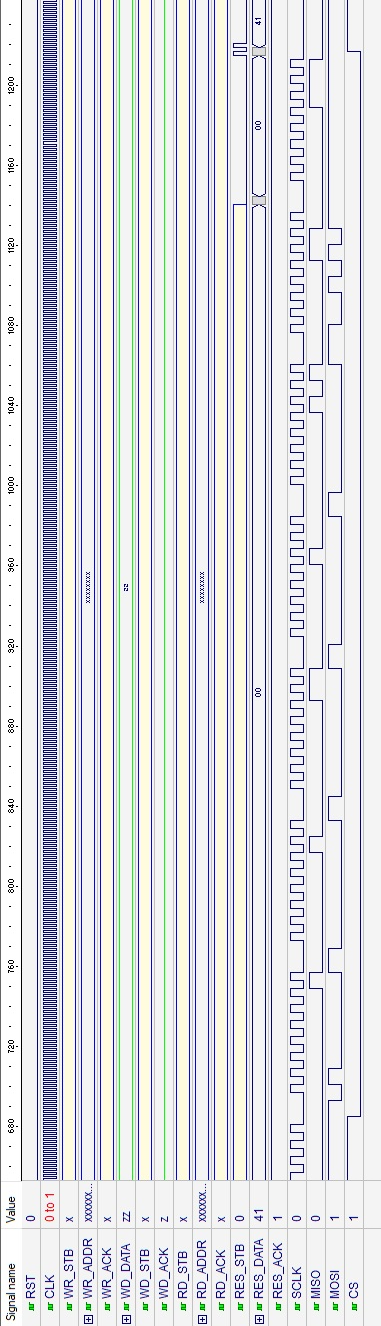
Rysunek 12 przedstawia wyniki zapisu i odczytu jednego bajtu danych poprzez protokół SPI, do zapisu (wysłania) użyto wartości () natomiast do odczytu (odebrania) wartości ().



Rysunek 14 Wyniki symulacji kontrolera SPI

## Symulacja sterownika karty pamięci

Rysunek 14 przedstawia przesyłanie komendy CMD0 do karty pamięci przez wyjście MOSI oraz przykładowe dane odbioru danych na wejście MISO.

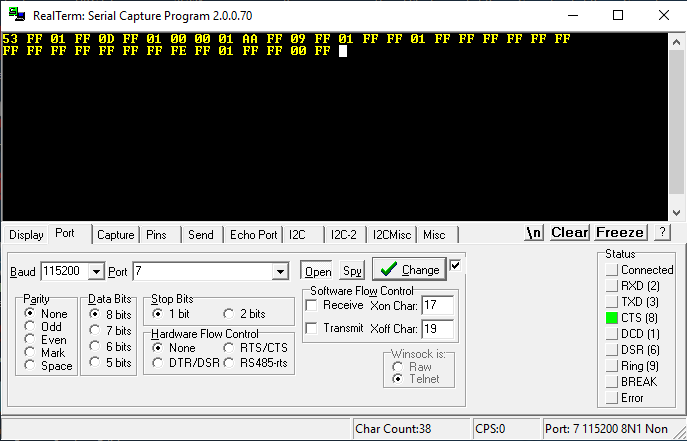


Rysunek 15 Wyniki symulacji przesłania komendy CMD0 i odbioru przykładowych danych

## Sposób testowania układu

Do dalszych testów zastosowano moduł testujący a wyniki obserwowałem na terminalu Realterm.

1. Wyniki procesu inicjalizacji karty pamięci



Rysunek 16 Wartości otrzymane po inicjalizacji karty pamięci

1. Wyniki zapisu danych na karcie
2. Wyniki odczytu danych z karty

## Otrzymane wyniki testów

# Podsumowanie

# Bibliografia