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

Streszczenie

Problemem podejmowanym w pracy jest komunikacja z kartą pamięci SD (ang. Secure Digital). W celu zbadania działania komunikacji karty pamięci z układami FPGA (ang. Field Programmable Gate Array) dokonano analizy działania kart pamięci oraz protokołów komunikacyjnych. Następnie wybrano protokół komunikacyjny SPI (ang. Serial Peripheral Interface). Zaprojektowano sterownik karty pamięci oraz wykorzystano język Verilog, w którym zrealizowano jego implementację. Praca przedstawia teoretyczne wymagania niezbędne do implementacji, sposób opisu modułu, wyniki działania zaprojektowanego sterownika oraz wyniki przeprowadzonych testów prędkości tj. zapisu i odczytu danych dla wybranych kart pamięci microSD.

Spis treści

1. Wprowadzenie

Wiele współczesnych aplikacji sprzętowych np. telefony komórkowe, kamery, aparaty fotograficzne, rejestratory wizyjne, odtwarzacze mp3, itp. wymagają przechowywania danych w pamięci masowej. 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. Ponadto układy scalone z pamięcią flash mają zbyt małe pojemności a karty pamięci Compact Flash są wypierane z rynku. W następstwie tego w wielu zastosowaniach najczęściej stosowanym rozwiązaniem jest wykorzystanie kart pamięci SD (ang. Secure Digital) i MMC (ang. Multi Media Card).

1.1. Cel pracy

Celem pracy było zaprojektowanie oraz implementacja modułu sterownika obsługi kart pamięci w języku Verilog. W tym celu konieczne było zapoznanie się z protokołami komunikacyjnymi oraz standardami kart pamięci, dokonanie implementacji wybranego interfejsu oraz całego modułu dla układu FPGA, wykonanie symulacji oraz opracowanie sposobu testowania projektowanego układu z użyciem płytki laboratoryjnej. Zwieńczeniem prac było wykonanie testów funkcjonalnych oraz wykonanie pomiaru prędkości dla wybranych kart pamięci. Przy implementacji i symulacji zaprojektowanego modułu wykorzystano środowisko Lattice Diamond 3.10 oraz oprogramowanie Active HDL 10.5.

1.2. Zakres pracy i założenia

Do zakresu pracy należy:

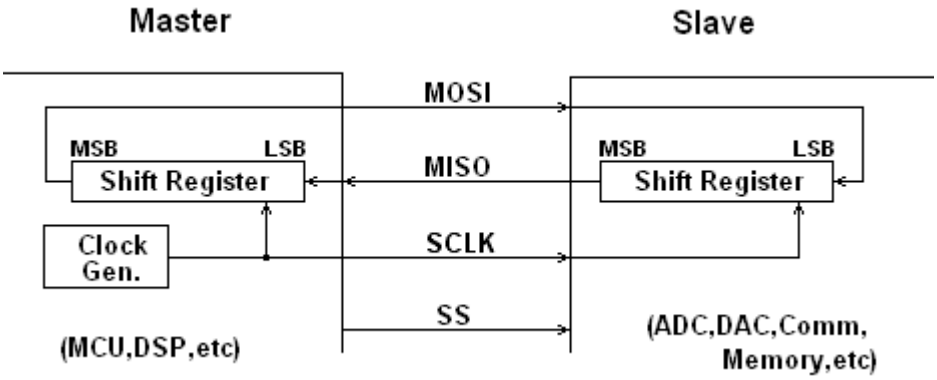
- analiza działania protokołów komunikacyjnych dla kart SD/MMC,
- opracowanie koncepcji działania modułu sterownika,
- implementacja w języku Verilog,
- opracowanie środowiska testowego (symulacyjnego i sprzętowego) umożliwiającego weryfikację implementacji,
- przeprowadzenie badań dotyczących wydajności zapisu i odczytu danych na wybranych modelach kart pamięci.

Na podstawie celu oraz zakresu pracy założono, że implementacja wykorzystywać będzie protokół SPI (ang. Serial Peripheral Interface) z wykorzystaniem karty microSD (ang. Micro Secure Digital), która jest najmniejszych rozmiarów z trzech dostępnych kart pamięci SD (ang. Secure Digital) jednak nie ma to wpływu na interfejs elektryczny.

2. Komunikacja z kartą pamięci SD oraz jej parametry

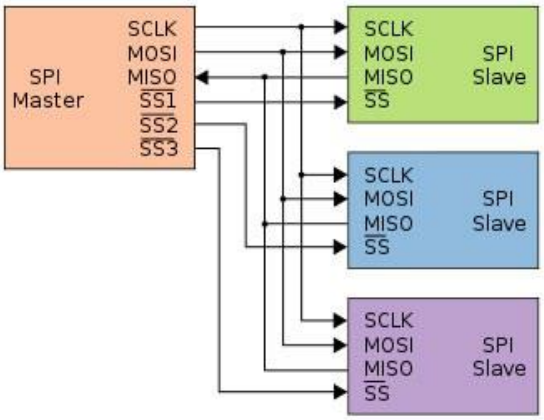
2.1. Protokół SPI

SPI (ang. Serial Peripheral Interface) jest dupleksowym, szybkim, synchronicznym interfejsem szeregowym. Używany jest do łączenia urządzeń takich jak drukarka, skaner, kamera itp. oraz do komunikacji między aplikacjami używającymi monitorów, kart pamięci, czujników itp. w systemach wbudowanych. Interfejs ten pozwala na łączenie jednego układu nadrzędnego (ang. Master) z wieloma urządzeniami podrzędnymi (ang. 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)

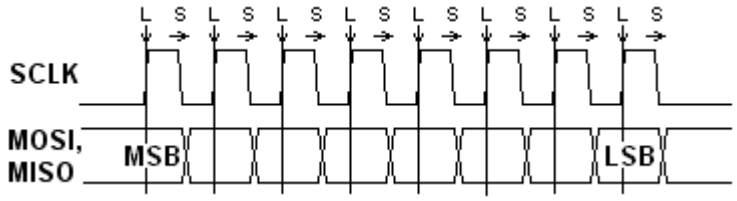
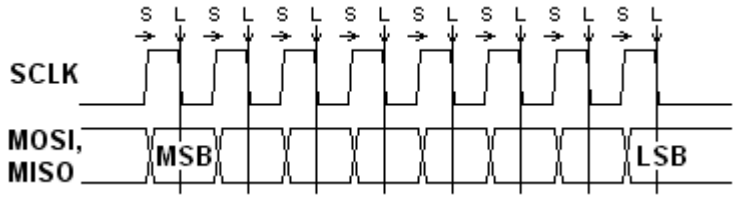
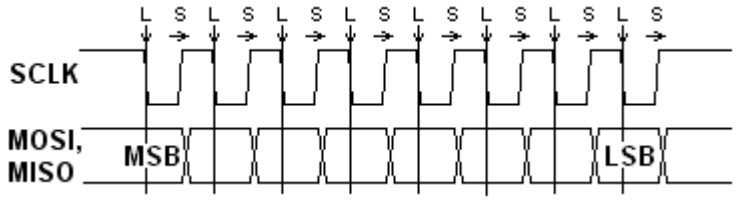
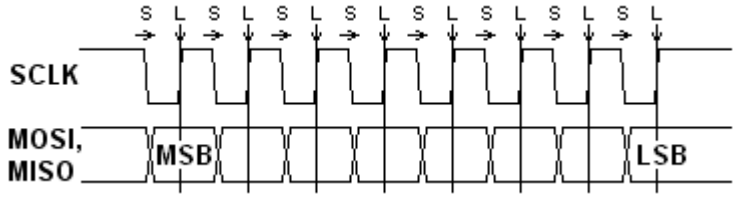
Rysunek 1 przedstawia sposób komunikacji pomiędzy modułami (nadrzędnym i podrzędnym) poprzez interfejs SPI. Obydwa układy połączone są liniami MOSI (ang. Master - Out Slave-In), MISO (ang. Master-In Slave-Out), SCLK (ang. Serial Clock) i SS (ang. Slave Select). Dane z obydwóch 8-bitowych rejestrów przesuwających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



Rysunek 2 Wykorzystanie interfejsu SPI z wieloma modułami podrzędnymi (ang. Slave).
(Źródło: http://www.dejazzer.com/ee379/lecture_notes/lec12_sd_card.pdf)

Rysunek 2 przedstawia przykładowe podłączenie trzech układów podrzędnych. Zapis lub odczyt następuje po ustawieniu wyjścia $\overline{SS1}, \overline{SS2}$ lub $\overline{SS3}$ 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.

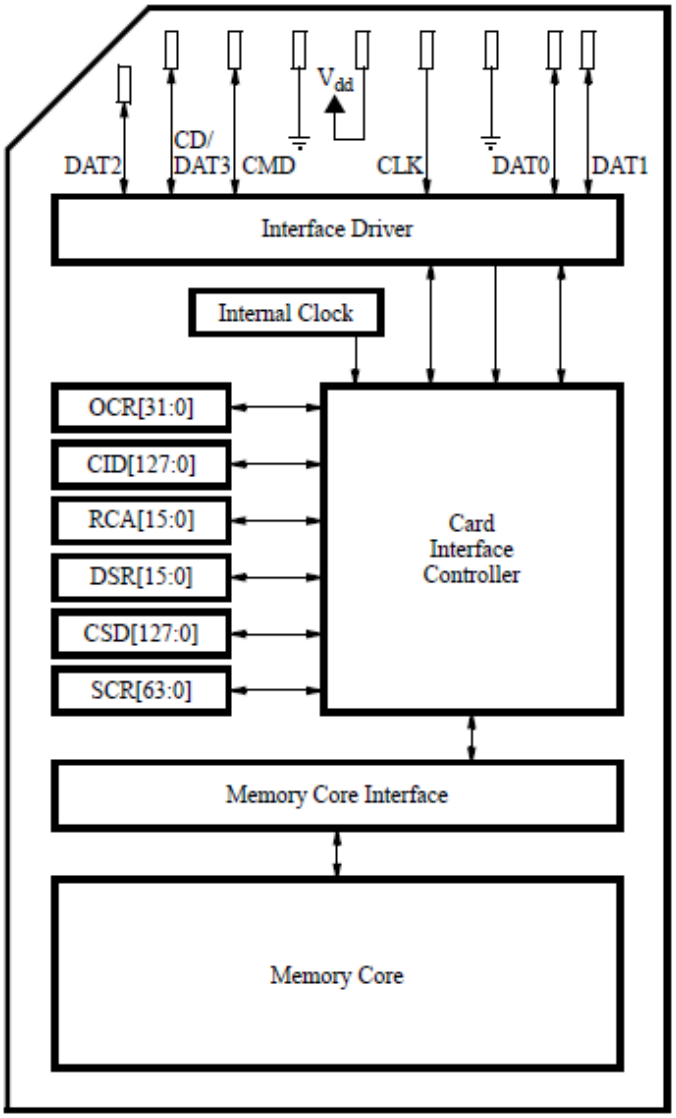
Tabela 1 Cztery tryby pracy interfejsu SPI. (Źródło: http://elm-chan.org/docs/spi_e.html)

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)	

2.2. Karty pamięci SD i MMC

Do głównych przyjętych norm kart pamięci należą karty SD (ang. Secure Digital) oraz karty MMC (ang. 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 (ang. Interface Driver), wewnętrznego układu taktującego (ang. Internal Clock), sterownika karty (ang. Card Interface Controller), wbudowanych rejestrów karty (OCR, CID, RCA, DSR, CSD, SCR), interfejsu rdzenia pamięci (ang. Memory Card Interface) oraz rdzenia pamięci (ang. Memory Core). 9-pinowy interfejs pozwala na wymianę danych między podłączonym systemem a sterownikiem karty. Sterownik 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: sterujące i danych. Żądania sterujące 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.
(Źródło: 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 przedstawiono 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 sterownikiem 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. Napięcie zasilania może być również ustawione na stałe np. na 3,3 V, ponieważ wszystkie karty SD i MMC działają w przedziale 2,7 V do 3,6 V. Maksymalny pobór prądu przez kartę pamięci może osiągnąć nawet 100 mA podczas operacji zapisu.

SDC

No	SD	SPI
8	DAT1	
7	DAT0	DO
6	Vss2	
5	CLK	SCLK
4	Vcc	
3	Vss1	
2	CMD	DI
1	CAT3	CS
9	DAT2	

MMC

No	MMC	SPI
7	DAT	DO
6	Vss2	
5	CLK	SCLK
4	Vcc	
3	Vss1	
2	CMD	DI
1	RES	CS

Rysunek 4 Opis wyprowadzeń styków dla karty SD oraz dla karty MMC.
(Źródło: http://elm-chan.org/docs/mmc/mmc_e.html)

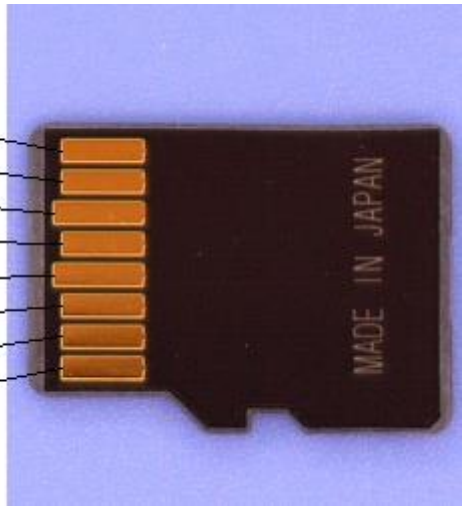
miniSD

No	SD	SPI
8	DAT1	
7	DAT0	DO
6	Vss	
5	CLK	SCLK
4	Vdd	
11	NC	
10	NC	
3	Vss	
2	CMD	DI
1	DAT3	CS
9	DAT2	

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

microSD

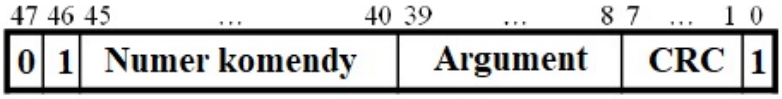
No	SD	SPI
8	DAT1	
7	DAT0	DO
6	Vss	
5	CLK	SCLK
4	Vdd	
3	CMD	DI
2	DAT3	CS
1	DAT2	



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

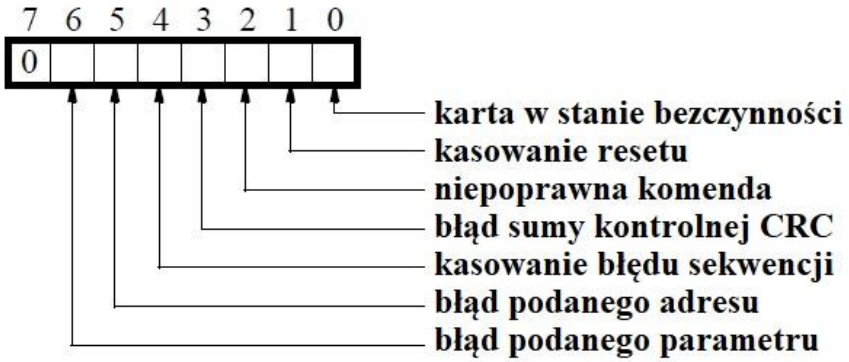
2.3. Opis podstawowych komend

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.

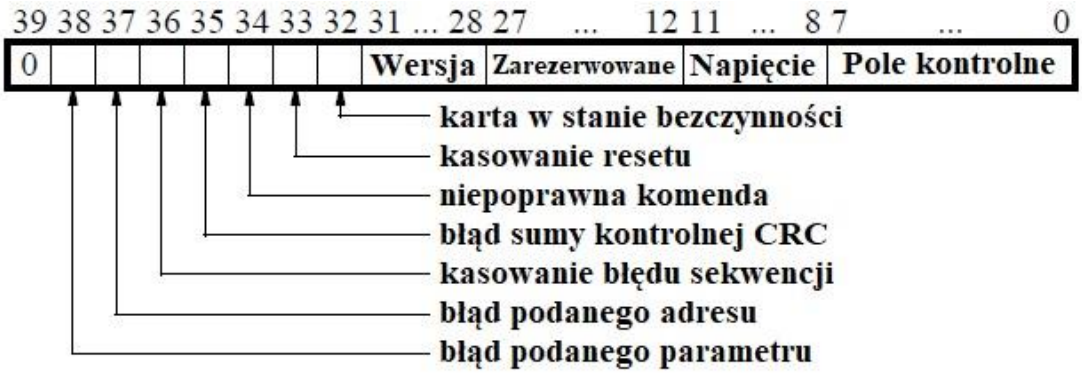


Rysunek 7 Format 48-bitowej komendy.
(Źródło: http://www.dejazzer.com/ee379/lecture_notes/lec12_sd_card.pdf)

Pierwsze dwa najbardziej znaczące bity to bity startu które ostawione są na wartość 01 dla każdej komendy. Następnym polem jest 6-bitowy numer komendy (ang. Command Number) który określa rodzaj wykonywanej operacji na karcie SD. Dalej znajduje się 32-bitowy argument z wartością oraz 7-bitowe pole CRC (ang. Cyclic Redundancy Check) 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 (ang. parameter error), błąd podanego adresu (ang. address error), kasowanie błędu sekwencji (ang. erase sequence error), błąd sumy kontrolnej CRC (ang. CRC error), niepoprawna komenda (ang. illegal command), kasowanie resetu (ang. erase reset), karta w stanie bezczynności (ang. 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 (ang. Version), 18-bitowe zarezerwowane pole (ang. Reserved), 4-bitowe pole określające napięcie karty (ang. Voltage) oraz 8-bitowe pole kontrolne (ang. Check Pattern).

Tabela 2 Lista podstawowych komend obsługiwanych przez kartę pamięci.

Nazwa komendy	**Argument**	**Format odpowiedzi**	**Dane**	**Nazwa skrótowa**	**Opis**
CMD0	Brak(0)	R1	Nie	GO_IDLE_STATE	Programowy reset karty
CMD1	Brak(0)	R1	Nie	SEND_OP_COND	Inicjacja procesu inicjalizacji
ACMD41 (*)	Zarezerwowany bit[31], HCS [30], Zarezerwowane bity [29:0]	R1	Nie	APP_SEND_OP_COND	Inicjacja procesu inicjalizacji (tylko dla SD w wersji 2)
CMD8	Zarezerwowane bity[31:12], Wspierane zasilanie (VHS)[11:8], Pole kontrolne [7:0]	R7	Nie	SEND_IF_COND	Sprawdzanie zakresu napięć (tylko dla SD w wersji 2)
CMD9	Brak(0)	R1	Tak	SEND_CSD	Odczyt rejestru CSD
CMD10	Brak(0)	R1	Tak	SEND_CID	Odczyt rejestru CID
CMD12	Brak(0)	R1b	Nie	STOP_TRANSMISSION	Stop odczytu danych
CMD16	Długość bloku[31:0]	R1	Nie	SET_BLOCKLEN	Zmiana odczytu lub zapisu długości bloku
CMD17	Adres[31:0]	R1	Tak	READ_SINGLE_BLOCK	Odczyt bloku
CMD18	Adres[31:0]	R1	Tak	READ_MULTIPLE_BLOCK	Odczyt wielu bloków
CMD23	Liczba bloków[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 (*)	Liczba bloków[22:0]	R1	Nie	SET_WR_BLOCK_ERASE_COUNT	Definiowanie liczby bloków do skasowania przy następnej komendzie zapisu w trybie wieloblokowym (tylko dla SD)
CMD24	Adres[31:0]	R1	Tak	WRITE_BLOCK	Zapis jednego bloku pamięci
CMD25	Adres[31:0]	R1	Tak	WRITE_MULTIPLE_BLOCK	Zapis wielu bloków
CMD55	Brak(0)	R1	Nie	APP_CMD	Ładowanie komendy ACMD<n>
CMD58	Brak(0)	R3	Nie	READ_OCR	Odczyt rejestru OCR
CMD59	Brak(0)[31:1], Flaga CRC[0]	R1	Nie	CRC_ON_OFF	Włączenie lub wyłączenie obsługi sum kontrolnych CRC 1-włączona,0-wyłączona
* - komendy ACMD mogą być wykonane tylko po wcześniejszym wysłaniu komendy CMD55.					

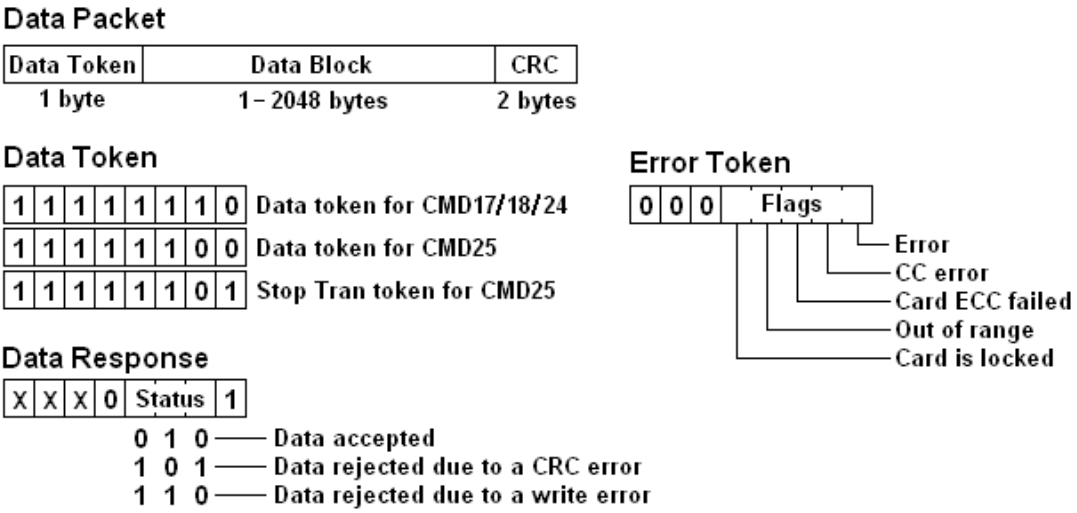
2.4. Komunikacja z kartą

Wysyłanie polecenia do karty SD rozpoczyna 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 (ang. 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 (ang. least significant bit) pozwala na przesłanie zawartości całego rejestru, co powoduje przesłanie całej komendy do karty SD.

Aby zapewnić poprawne działanie karty SD, częstotliwość zegara taktującego SCLK powinna 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).

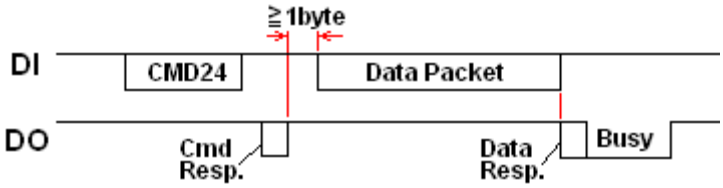
Gdy tylko karta SD otrzyma komendę rozpoczyna ją przetwarzać a następnie wykonywać. Dla poprawnej odpowiedzi na polecenie karta SD wymaga przełączania sygnału SCLK przez co najmniej 8 taktów i obserwacji wejścia MISO. W tym czasie linia MOSI znajduje się z wysokim poziomie a CS w poziomie niskim. Długość odpowiedzi wynosi zwykle 8-bitów lub 40-bitów w zależności od użytej komendy i odpowiedniego dla niej formatu odpowiedzi. 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.

2.5. Transfer danych



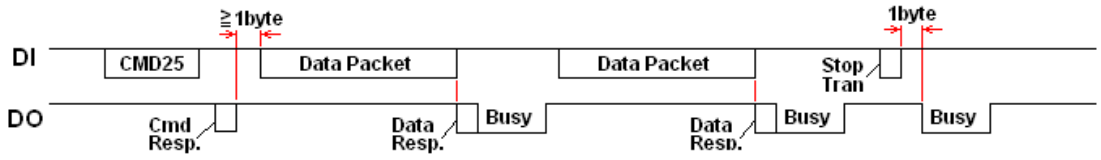
Rysunek 10 Budowa pakietu danych i odpowiedzi dla komend zapisu i odczytu
(Źródło: http://elm-chan.org/docs/mmc/mmc_e.html)

Przesyłane dane (które są zapisywane lub odczytywane) zorganizowane są pakiety. Te z kolei składają się z żetonu (ang. Data Token) który informuje o początku lub końcu transmisji, bloku danych (ang. Data Block) który zawiera przesyłane dane o domyślnej długości 512 bajtów oraz sumy kontrolnej CRC (ang. Cyclic Redundancy Code) która powinna zostać obliczona przed wysłaniem. Rysunek 10 przedstawia budowę pakietu z podziałem na rodzaje żetonów oraz możliwych odpowiedzi.



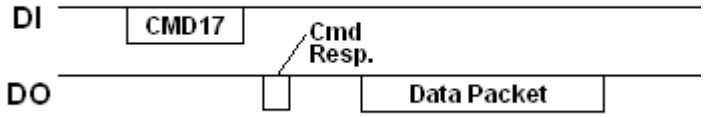
Rysunek 11 Transmisja pakietów dla zapisu pojedynczego bloku.
(Źródło: http://elm-chan.org/docs/mmc/mmc_e.html)

Zapis pojedynczego boku odbywa się za pomocą komendy CMD24. Po jej zaakceptowaniu, karta wysyła odpowiedź (ang. Cmd response) a następnie wysyłany jest blok danych, który jest zapisywany pod adres określony przez komendę. Następnie karta odsyła odpowiedź o pomyślnym zapisaniu danych a kontroler hosta musi zawiesić następne polecenie lub transmisję danych, dopóki karta nie będzie gotowa. Rysunek 11 przedstawia sekwencję przesyłanych ramek.



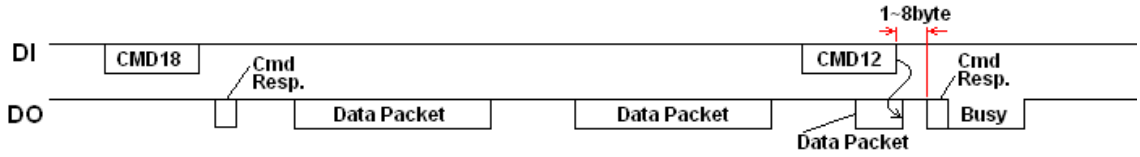
Rysunek 12 Transmisja pakietów w trybie wielu bloków dla zapisu
(Źródło: http://elm-chan.org/docs/mmc/mmc_e.html)

Do zapisu danych w trybie wielu bloków używana jest komenda CMD25 oraz ramka stopu (ang. Stop Tran). Rysunek 12 przedstawia kolejność wysłanych i odbieranych ramek danych. Po wysłaniu komendy i akceptacji jej przez kartę wysyłane są pakiety danych z przerwami na odpowiedź ze strony karty. Zapis danych kończy wysłanie ramki stopu. Liczba bloków może być definiowana przez komendę ACMD23 (po wcześniejszym wysłaniu komendy CMD55).



Rysunek 13 Transmisja ramek danych dla odczytu pojedynczego bloku
(Źródło: http://elm-chan.org/docs/mmc/mmc_e.html)

Odczyt pojedynczego bloku odbywa się poprzez wysłanie komendy CMD17 z adresem bloku (dla kart SD o rozmiarze powyżej 2GB) lub bajtu (dla kart SD o rozmiarze do 2GB). Po otrzymaniu odpowiedzi z potwierdzeniem (ang. Cmd Response) odczytywany jest pakiet danych (ang. Data Packet). Po wykryciu żetonu pakietu odczytywany jest blok danych a następnie suma kontrolna CRC (ang. Cyclic Redundancy Check). Bajty te muszą zostać odebrane nawet jeśli nie będą wykorzystywane przez sterownik. Rysunek 13 przedstawia transmisje ramek dla odczytu jednego bloku danych.



Rysunek 14 Transmisja pakietów w trybie wielu bloków dla odczytu
(Źródło: http://elm-chan.org/docs/mmc/mmc_e.html)

Do odczytu wielu bloków danych z karty pamięci wykorzystywana jest komenda CMD18. Rysunek 14 przedstawia ramki w tym trybie odczytu. Operacja ta jest transmisją otwartą, która kończy się poprzez wysłanie komendy CMD12. Transakcja odczytu inicjowana jest jako wstępnie zdefiniowany transfer wielu bloków, a operacja odczytu jest kończona przy ostatnim transferze bloku, po którym przesyłana jest suma kontrolna CRC (ang. Cyclic Redundancy Check).

2.6. Podział kart i ich kategorie prędkości

Rysunek 15 Logo standardów kart pamięci SD
(1 - karty SD, 2 - karty SDHC, 3 - karty SDXC, 4 - karty SDUC)
(Źródło: https://www.sdcard.org/developers/overview/capacity/index.html)

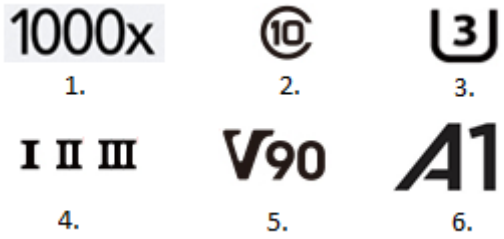
Technologiczny rozwój kart pamięci spowodował opracowanie kilku standardów kart. Obecne znane formaty kart pamięci i ich pojemności to:

SD (ang. Secure Digital) – to karty, dla których maksymalna pojemność wynosi 2GB, a sugerowany system plików to FAT12 lub FAT16.

SDHC (ang. Secure Digital High Capacity) – to nowsze karty niż SD, które dostępne są o pojemności 2GB - 32GB. Ich Sugerowany system plików to FAT 32.

SDXC (ang. Secure Digital Extended Capacity) – to najnowsze z dostępnych obecnie kart pamięci, dla których maksymalna pojemność może wynosi 32GB - 2TB. Sugerowany system plików to exFAT.

SDUC (ang. Secure Digital Ultra Capacity) – to standard w fazie prototypów a jego planowana pojemność ma wynosić 2TB – 128 TB. Sugerowany system plików to exFAT.



Rysunek 16 Sześć podstawowych symboli opisujących kartę pamięci SD
(1 - Symbol X, 2 - Symbol C, 3 - Symbol U, 4 - Symbol I, 5 - Symbol V, 6 - Symbol A)
(Źródło: https://www.sdcard.org/developers/overview/index.html
https://www.lexar.com/products/memory-cards/)

Oprócz pojemności prędkość to najważniejsza cecha definiująca kartę pamięci. Do jej opisu powstało aż sześć odrębnych kategorii (dla zapisu lub odczytu). Są to:

Symbol X - wielokrotność odczytu płyty audio CD. Kategoria ta pochodzi od standardowej prędkości odczytu płyt audio, wynoszącej 150 KB/s. A symbol X jest mnożnikiem 150 KB/s. Np. 1000x oznacza, że karta czyta dane z maksymalną prędkością 150 MB/s (1000 x 150 KB/s = 150 000 KB/s = 150 MB/s). Oznaczenie to stosuje głównie firma Lexar w swoich kartach.

Symbol C - klasa wydajności zapisu. Klasa ta odnosi się do minimalnej stałej prędkości zapisu na kartę w megabajtach na sekundę. Wskazuje ona wartość, poniżej której osiągi karty nie mają prawa spadać. Stała wartość jest ważna zwłaszcza dla zapisu filmów lub seryjnych zdjęć, gdyż podczas tych aktywności nośnik jest permanentnie obciążony przez dłuższy i nieprzerwany odcinek czasu. Np. karta C4 posiada minimalną prędkość 4 MB/s a karta C10 minimalną prędkość 10 MB/s.

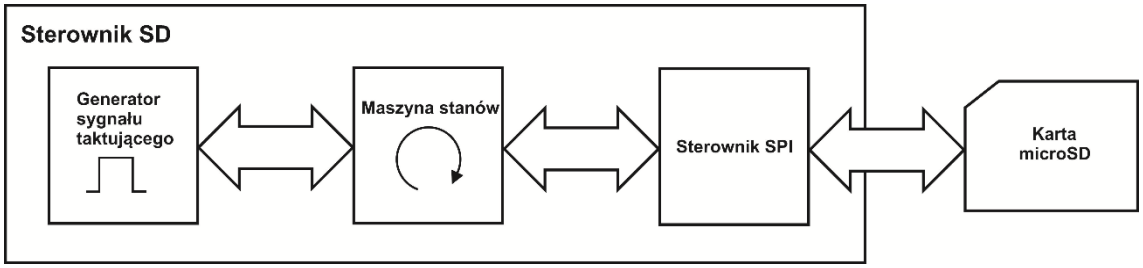
Symbol U - nowsza klasa wydajności zapisu (ang. UHS Class). Jest następstwem klas C (Class). UHS (ang. Ultra High Speed) pozwala na osiągnięcie lepszych wartości odczytu i zapisu. Symbol U oznacza więc nie tylko klasę prędkości, ale także obecność nowszego standardu łączności karty. Łączność UHS to cecha kart SDHC oraz SDXC. Np. karta U1 gwarantuje stałą minimalną prędkość zapisu 10 MB/s, a karta U3 gwarantuje stałą minimalną prędkość zapisu 30 MB/s.

Symbol I, II, III - jest ulepszeniem standardu UHS który wprowadza podział w oparciu o rodzaj wykorzystywanej szyny, która definiuje liczbę złotych pinów widocznych na karcie. Dostępne są karty z oznaczeniem UHS-I (UHS-1), UHS-II (UHS-2), UHS-III (UHS-3). Karty z klasyczną szyną UHS-I osiągają maksymalną prędkość odczytu wynoszącą do 104 MB/s. A do działania z maksymalnymi wartościami wymaga czytnika ze wsparciem standardu SD 3.01. Karty z szyną UHS-II osiągają maksymalną prędkość odczytu wynoszącą do 312 MB/s. Aby osiągać ten transfer na szynie UHS-2, niezbędny jest czytnik wspierający przesył danych z podwójnym rzędem pinów, wspierający standard SD 4.0. Z kolei karty z szyną UHS-III osiągają maksymalną prędkość 624 MB/s ze wsparciem standardu SD 6.00.

Symbol V (ang. Video Speed) - to klasa wydajności dla treści wizualnych która występuje w wariantach V6, V10, V30, V60 oraz V90. Liczby przy literce V oznaczają minimalną stałą prędkość zapisu wyrażoną w megabajtach na sekundę. Np. dla V6 minimalny stały zapis wynosi 6 MB/s, a dla V90 wynosi 90 MB/s. Kategoria ta odnosi się do najniższej stałej, zagwarantowanej prędkości zapisu na kartę. Z tego powodu symbol V coraz częściej pojawia się na nowoczesnych kartach SD będąc jedną z najważniejszych wytycznych podczas zakupu karty. Zwłaszcza, jeśli ta ma trafić do wideo rejestratora albo kamery.

Symbol A (ang. Applications) – to klasa wydajności, dla aplikacji, która tworzy oddzielną kategorię wydajności dla takich urządzeń jak smartfony, tablety czy przenośne konsole do gier. Dla tego typu urządzeń większą rolę mają inne parametry niż karty pamięci dedykowane dla kamer czy wideo rejestratorów. Zamiast zapisu ciągłego, smartfony i tablety zapisują i odczytują mniejsze porcje danych, które są zagospodarowywane na karcie w bardziej nieregularnych odcinkach czasu. W kartach tego typu najważniejszą cechą nie jest minimalny zapis ciągły (jak np. w kategorii C lub U), ale minimalny zapis losowy. Szybka karta do zapisu ciągłego wcale nie musi być szybka podczas zapisu losowego i odwrotnie. Dlatego wskaźnikiem klasyfikacji A nie są megabajty na sekundę (MB/s), ale operacje wejście-wyjście na sekundę (ang. IOPS). Klasa ta dedykowana jest ocenianiu wydajności dyskowej, pokazująca, jak wiele jednoczesnych operacji może wykonać urządzenie w ciągu jednej sekundy. Obecnie dostępne są karty A1 dla której minimalny odczyt losowy wynosi 1500 IOPS a minimalny zapis losowy 500 IOPS oraz karty A2 dla której minimalny odczyt losowy wynosi 4000 IOPS a minimalny zapis losowy 2000 IOPS.

3. Architektura i implementacja

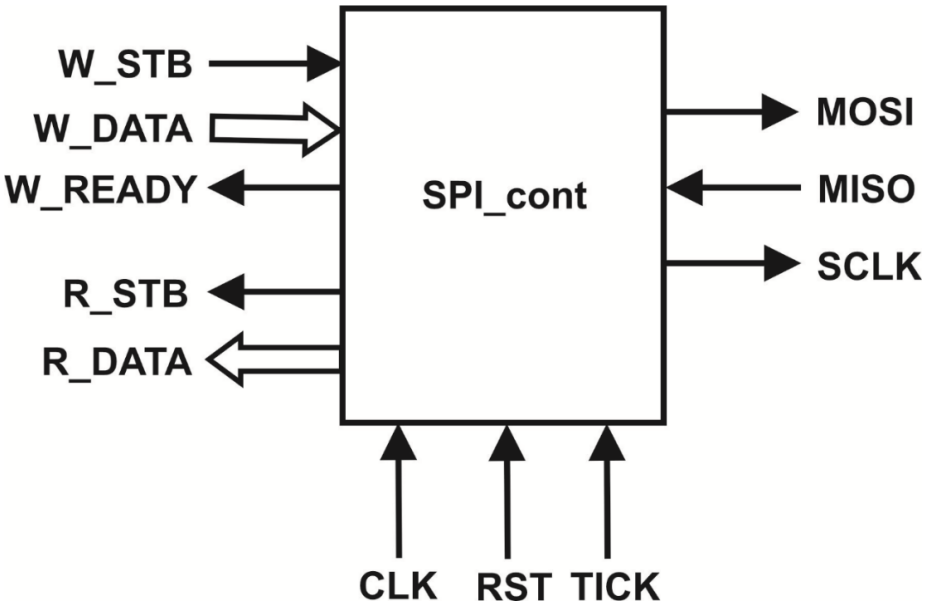


Rysunek 17 Ogólny schemat blokowy sterownika SD

Rysunek 17 przedstawia schemat funkcjonalny projektowanego sterownika. Składa się on z sterownika SPI który odpowiada za transmisje danych zgodnie z interfejsem SPI z lub do karty pamięci. Maszyna stanów stanowi automat, który steruje sygnałami z interfejsu oraz wysyła odpowiednie komendy do karty pamięci. Generator sygnału taktującego generuje sygnał z odpowiednią częstotliwością podczas inicjalizacji pracy karty oraz podczas transmisji danych.

3.1. Moduł sterownika SPI

Projektowany sterownik SPI stanowi najmniejszy moduł projektowanego sterownika i odpowiada za transmisję wykorzystującą protokół SPI. Transmisja ta odbywa się w trybie 0, gdy przy zboczu narastającym pobierana jest próbka (ang. Sample) a ustawienie wartości (ang. Setup) przy zboczu opadającym. Rysunek 18 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 wejścia wyzwalającego który służy do wyznaczania częstotliwości `SCLK`.



Rysunek 18 Schemat funkcjonalny modułu sterownik SPI

Tabela 3 Opis wyprowadzeń sygnałów kontrolera SPI

Nazwa sygnału:	**Opis:**
CLK	Wejście zegarowe
RST	Wejście resetujące układ (Reset)
TICK	Wejście wyzwalające działanie modułu
W_STB	Wejście strobujące zapis danych
W_DATA	8-bitowa magistrala danych zapisywanych
W_READY	Wyjście potwierdzające przesłanie danych
R_STB	Wyjście strobujące odczyt
R_DATA	8-bitowa magistrala danych odczytywanych
MOSI	Wyjście danych 1-bitowych interfejsu SPI
MISO	Wejście danych 1-bitowych interfejsu SPI
SCLK	Wyjście zegarowe interfejsu SPI

Implementacja powyższego modułu odwzorowuje interfejs przedstawiony na rysunku 18. Kod zawiera dwa procesy które opisują działanie projektowanego kontrolera. Pierwszy z nich służy do generowania sygnału SCLK i ma następującą postać:

```
always @(posedge CLK or posedge RST)
     INT_SCLK <= (RST) ? 0 : (W_STB) ? 1 : INT_SCLK ^ TICK;
```

Do rejestru `INT_SCLK` zapisywany jest wynik operacji sumy modulo 2 wartości z `INT_SCLK` oraz `TICK` który dostarczany jest z modułu nadrzędnego. Sygnał `TICK=1` powoduje zanegowanie `INT_SCLK`. Natomiast jeśli `TICK=0`, to `INT_SCLK` nie ulega zmianie. W przypadku gdy aktywny jest sygnał resetu `RST` do zmiennej zapisywana jest wartość logicznego zera, natomiast gdy aktywny jest sygnał strobujący `W_STB` wartość logicznej jedynki.

```
assign SCLK = INT_SCLK & receiving;
```

Wyjściowa wartość `SCLK` jest wynikiem iloczynu logicznego zmiennej `INT_SCLK` i rejestru `receiving`.

Drugi proces odpowiada za transmisję na narastającym i opadającym zboczu zegara. Składa się z kilku instrukcji warunkowych if:

```
always@(posedge CLK or posedge RST)
     if(RST)
     begin
       MOSI <= 1;
       receiving <= 0;
       sending <= 0;
       WR_DATA <= 0;
       RD_DATA <= 0;
       R_STB <= 0;
       R_DATA <= 0;
       period <= 0;
```

Pierwsza część odpowiada za inicjalizację wszystkich rejestrów i sygnałów po resecie modułu.

```
end else
  begin
  R_STB <= 0;
  R_DATA <= 0;
  if (W_STB)
    begin
    WR_DATA <= W_DATA;
    sending <= 1;
    period <= 7;
```

W następnym bloku przypisywane są wartości do rejestrów i sygnałów podczas, gdy aktywny jest sygnał strobujący `W_STB`. Dane z magistrali wejściowej `W_DATA` wczytywane są do wewnętrznego rejestru `WR_DATA`. Szyna wyjściowa `R_DATA` jest zerowana, rejestr `sending` jest zapisywany wartością 1 a rejestr `period` wartością 7.

W kolejnej części opisu projektowanego sterownika zawarte jest zachowanie na zboczu pozytywnym lub negatywnym.

```
      else if (sending && TICK && (INT_SCLK==1))
            begin
                  WR_DATA <= WR_DATA << 1;
                  period <= period - 1;
                  if(period[3])
                        begin
                              receiving <= 0;
                              sending <= 0;
                              MOSI <= 1;
                              R_DATA <= RD_DATA;
                              R_STB  <= 1;
                        end
                  else
                        begin
                              receiving <= 1;
                              MOSI <= WR_DATA[7];
                        end
            end
```

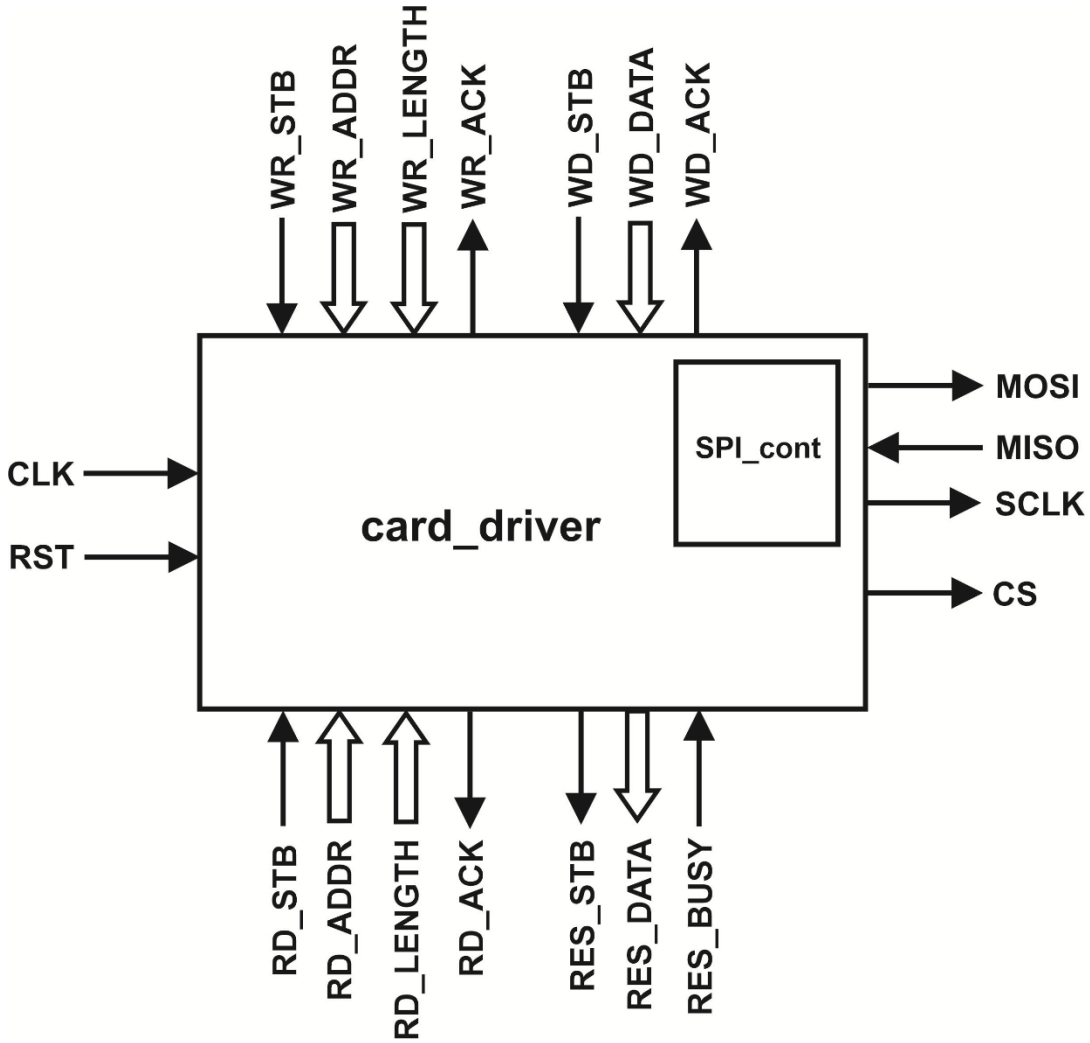
W tej części opisywane jest działanie na zboczu opadającym w którym przesuwany jest rejestr `WR_DATA` o jedną pozycję a następnie wysyłany na sygnał wyjściowy `MOSI`. Jeśli wartość rejestru `period` osiągnie wartość minus jeden i pojawi się wartość jeden na najstarszej pozycji rejestru `period` wówczas wykonywany jest odczyt rejestru `RD_DATA` na magistrale wyjściową `R_DATA` i ustawienie sygnału strobującego odczyt `R_STB`.

```
end else if (receiving && TICK && (INT_SCLK==0))

begin

      RD_DATA <= { RD_DATA, MISO};

      end

end
```

W ostatniej części opisywane jest zachowanie na zboczu narastającym, gdzie zapisywane są dane do rejestru `RD_DATA` z wejściowego sygnału `MISO` i zerowanie sygnału strobującego odczyt `R_STB`.

3.2. Moduł sterownika karty pamięci SD

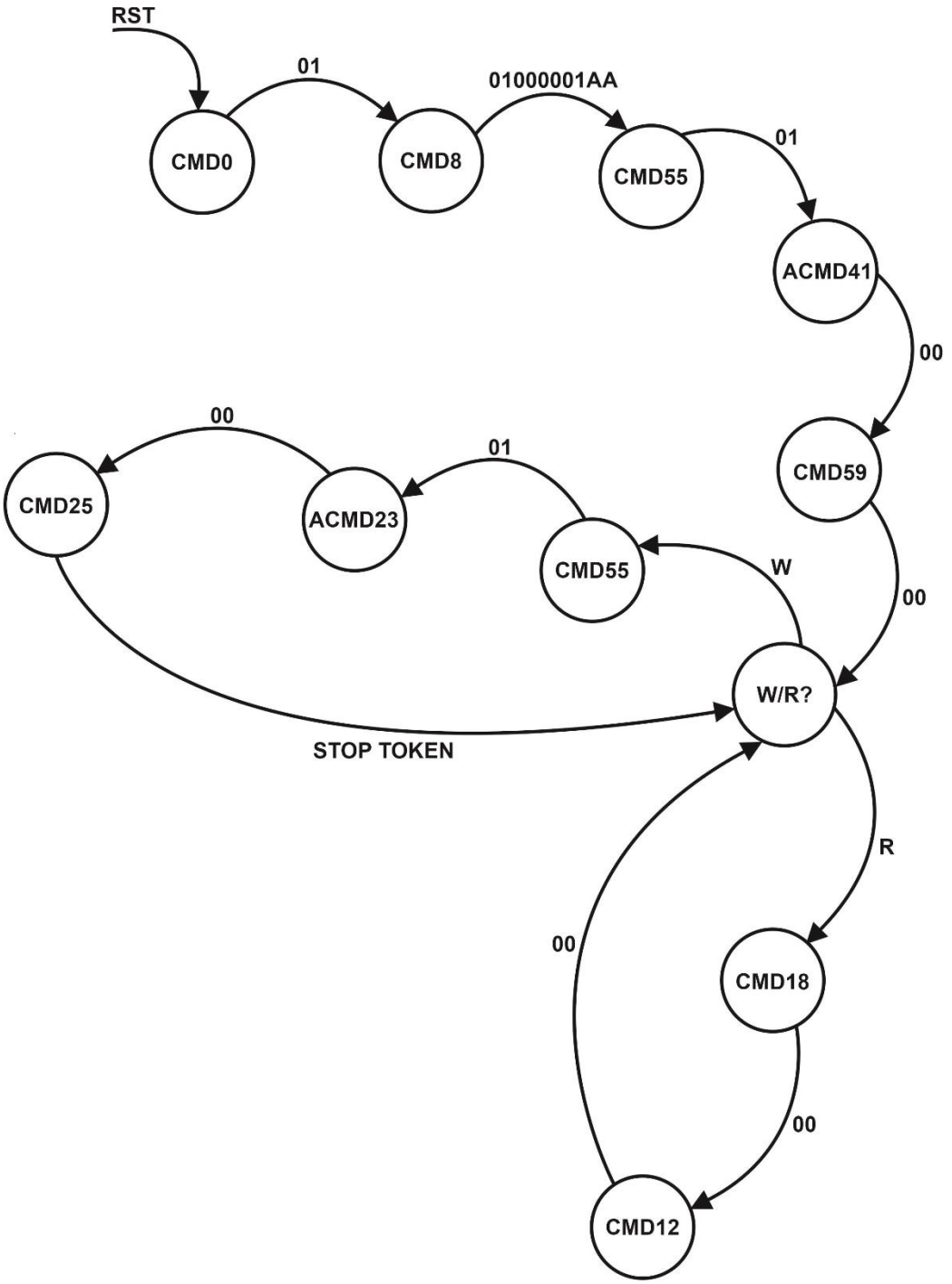
Sterownik kart pamięci stanowi automat wysyłający komendy inicjalizacji oraz obsługi zapisu i odczytu danych na kartę pamięci. Moduł ten posiada wejście zegarowe CLK, resetujące RST oraz CS (ang. Chip Select) służące do wyboru układu, na którym wykonywane są operacje. Sterownik card_driver posiada wejście strobujące zapis adresu WR_STB wraz z 32-bitowym wejściem do zapisu adresu WR_ADDR oraz 23-bitowe wejście do podania ilości zapisywanych bloków a także wyjście 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 oraz wartość na wejście RD_LENGHT określające liczbę odczytywanych bloków. 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 19 Schemat funkcjonalny projektowanego sterownika karty pamięci SD

Tabela 4 Opis wyprowadzeń projektowanego sterownika kart 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_LENGHT	23-bitowe wejście adresowe określające liczbę bloków do zapisu
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_LENGHT	23-bitowe wejście adresowe określające liczbę bloków do odczytu
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 20 Uproszczony (koncepcyjny) graf automatu sterownika kart pamięci SD.
Domyślnie automat pozostaje w aktualnym (dotychczasowym) stanie

Na rysunku 20 przedstawiono automat zgodnie z którym działa zaprojektowany sterownik. Pozostaje on domyślnie w danym stanie, dopóki nie otrzyma odpowiedzi od karty pozwalającej mu przejść do następnego stanu. 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 (zapisana heksadecymalnie) 0x01. Następnie wysyłana jest komenda CMD8:

```
01001000 00000000 00000000 00000001 10101010 10000111
```

dla której odpowiedź (zapisana heksadecymalnie) wynosi 0x01000001AA co oznacza, że karta SD jest w wersji co najmniej drugiej 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 wartości heksadecymalnej 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 wartością heksadecymalną 0x00 oznacza to, że proces inicjalizacji przebiegł pomyślnie. Następnie wysyłana jest komenda CMD59:

```
01111011 00000000 00000000 00000000 00000000 00000000
```

która wyłącza obsługę sum kontrolnych CRC. Jeśli karta odpowie wartością heksadecymalną 0x00 automat przechodzi do stanu oczekiwania na strobujący sygnał zapisu lub odczytu. Jeśli sygnał strobujący zapis jest aktywny wówczas wysyłana jest komenda CMD 55:

```
01110111 00000000 00000000 00000000 00000000 11111111
```

po której wysyłana jest komenda ACMD 23:

```
01010111 00000000 00000000 WR_LENGTH[23:0]
```

gdzie `WR_LENGTH[23:0]` oznacza liczbę bloków która zostanie zapisana na karcie pamięci podczas operacji zapisu. Jeżeli karta odpowie wartością heksadecymalną 0x00 wówczas wysyłana jest komenda zapisu wielu bloków danych CMD25 o postaci:

```
01011001 ADDR[31:0] 11111111
```

gdzie `ADDR[31:0]` oznacza adres, pod którym jest dokonywany zapis. Koniec tej operacji następuje po wysłaniu żetonu (ang. token) `11111101` co informuje kartę o końcu procesu zapisu. Po czym automat wraca do stanu oczekiwania na zapis lub odczyt. Jeżeli natomiast sygnał strobujący odczyt jest aktywny wówczas 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 11111111
```

a następnie automat wraca do stanu oczekiwania na sygnał strobujący zapis lub odczyt.

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

Implementacja sterownika karty zawiera interfejs znajdujący się na rysunku 19. Posiada ona proces odpowiadający za generowanie sygnału zegarowego.

```
always @(posedge CLK or posedge RST)
      tickcounter <= (RST) ? START_DIVIDER-1
            : (TICK) ? divider_m1 : tickcounter-1;
```

Do rejestru `ticketcounter` zapisywana jest podczas aktywnego sygnału resetu `RST` wartość początkowa `START_DIVIDER – 1`. Następnie, jeśli aktywny jest sygnał `TICK` do rejestru `tickcounter` zapisywana jest wartość `divider_m1`. Natomiast w pozostałych przypadkach jest to wartość `tickcounter – 1`.

Drugi proces opisuje automat przedstawiony na rysunku 20.

```
always@(posedge CLK or posedge RST)
if (RST) begin
      state <= 0;
      CS <= 1;
      W_STB <= 0;
      W_DATA <= 0;
      RES_STB <= 0;
      RES_DEBUG <= 1;
      RES_DATA <= 0;
      statecnt <= 0;
      statecnt_wr <= 0;
      statecnt_rd <= 0;
      shreg <= 0;
      WR_ACK <= 0;
      WD_ACK <= 0;
      RD_ACK <= 0;
      len_counter <= 0;
      divider_m1 <= START_DIVIDER-1;
      RES_DATA_IS_00 <= 0;
      RES_DATA_IS_01 <= 0;
      RES_DATA_IS_AA <= 0;
      RES_DATA_IS_FF <= 0;
```

W pierwszej instrukcji warunkowej ustawiane są wartości początkowe (zwykle zera) do wszystkich wejść, wyjść i rejestrów, z których korzysta automat.

```
end else begin
      CS <= 0;
      W_STB <= 0;
      W_DATA <= 0;
      RES_STB <= 0;
      RES_DATA <= 0;
      WR_ACK <= 0;
      WD_ACK <= 0;
      RD_ACK <= 0;
      W_DATA <= 8'b11111111;
      RES_DEBUG <= 1;
      RES_DATA <= R_DATA;
      RES_DATA_IS_00 <= R_DATA==8'h00;
      RES_DATA_IS_01 <= R_DATA==8'h01;
      RES_DATA_IS_AA <= R_DATA==8'hAA;
      RES_DATA_IS_FF <= R_DATA==8'hFF;
```

W instrukcji else z pierwszej instrukcji warunkowej ustawiane są domyślne wartości początkowe dla rejestrów, wejść oraz wyjść.

```
case(state)
0: begin CS<=1; if (RES_BUSY==0) state <= 1; end
1:  begin  statecnt  <=  15;  RES_STB  <=  1;  RES_DATA  <=  "S";
   CS<=1;state <= 2; end
```

Następnie opisany jest automat za pomocą instrukcji case, w której dzięki wartości zapisanej w rejestrze `state` możliwe jest przechodzenie i wykonywanie kolejnych stanów. W pierwszym stanie sprawdzane jest wyjście `RES_BUSY` i jeśli jest wolne automat przechodzi do drugiego stanu w którym wysyłany jest znaku „S”. Znak ten można zobaczyć na terminalu podczas działania modułu w celu weryfikacji jego działania.

```
2: begin W_STB <= W_READY&&!R_STB; CS<=1; if(R_STB) state<=3; end
```

Przez kolejne 10 stanów (stan od 2 do 11) przesyłany jest stan wysoki do karty przy ustawionym sygnale Chip Select.

```
12: begin W_DATA <= 8'b01000000; W_STB <= W_READY && !R_STB;
if(R_STB)   state <= 13; end
13: begin W_DATA <= 8'b00000000; W_STB <= W_READY && !R_STB;
if(R_STB)   state <= 14; end
14: begin W_DATA <= 8'b00000000; W_STB <= W_READY && !R_STB;
if(R_STB)   state <= 15; end
15: begin W_DATA <= 8'b00000000; W_STB <= W_READY && !R_STB;
if(R_STB)   state <= 16; end
16: begin W_DATA <= 8'b00000000; W_STB <= W_READY && !R_STB;
if(R_STB)   state <= 17; end
17: begin W_DATA <= 8'b10010101; W_STB <= W_READY && !R_STB;
if(R_STB)   state <= 19; end
```

W następnych stanach wysyłana jest 48-bitowa komenda CMD0. Wykonywane jest to w stanach od 12 do 17.

```
18: begin W_STB <= W_READY && !R_STB;  RES_STB <= R_STB;
RES_DATA <= R_DATA; if (R_STB)   state <= 19; end

19: begin statecnt <= statecnt-1; if (RES_DATA_IS_01)
state <= 20; else if(statecnt[SC_SIZE]) state <= 0;
else state<=18; end
```

W następnym stanie 18 odczytywane są dane które wysyła karta po otrzymaniu komendy. Jeśli otrzymana jest wartość heksadecymalna 0x01 która zapisana jest w rejestrze `RES_DATA_IS_01` wówczas automat przechodzi do kolejnego stanu w przeciwnym wypadku wraca do stanu 0 i wysyła komendę ponownie. Sprawdzenie to odbywa się w stanie 19. Warto tu zauważyć, że porównywanie oczekiwanej odpowiedzi przechodzi przez dodatkowy rejestr. Wszystkie kolejne komendy obsługiwane są w podobny sposób.

```
71: begin CS<=1; divider_m1 <= TRANSMISSION_DIVIDER-1; if (TICK)
state <= 72; end
72: begin CS<=1; if (WR_STB) state <= 80; else if (RD_STB)
state <= 158; end
```

Po wysłaniu początkowych komend i poprawnej inicjalizacji karty automat wczytuje nową wartość do rejestru `divider_m1` zmieniając przy tym częstotliwość (stan 71). A następnie

automat czeka na sygnał strobujący a wykonuje się to w stanie 72. W stanie tym również ustawiane są wartości zmiennych potrzebnych przy wykonywaniu odczytu lub zapisu danych.

```
132: begin shreg[31:24] <= WD_DATA; WD_ACK <= WD_STB; if (WD_STB)
state <= 133; end
133: begin W_DATA <= shreg[31:24];  W_STB <= W_READY && !R_STB;
if (R_STB) state <= 134; end
134: begin statecnt_wr <=statecnt_wr-1; if (statecnt_wr[SC_SIZE])
state <= 135; else state <= 132; end
```

W stanach od 132 do 134 wykonywany jest zapis bloku na kartę pamięci pod adres wskazany wcześniej w komendzie CMD25. Dane wczytywane są do rejestru `ADDR[31:24]` a następnie do `W_DATA` gdzie wysyłane są do karty. Proces ten jest powtarzany dla 512 bajtów.

```
139: begin if (len_counter[LEN_WIDTH-1]) state <= 140; else
state <= 125; end
```

W stanie 139 sprawdzana jest liczba powtórzeń zapisu dla określonej wartości bloków przeznaczonych do zapisu.

```
183: begin W_STB <= W_READY && !R_STB;   RES_STB <= R_STB;
RES_DATA <= R_DATA;  RES_DEBUG <= 0; if (R_STB)   state <= 184;
end
184:  begin  statecnt_rd  <=statecnt_rd-1;  RES_DEBUG  <=  0;
if (statecnt_rd[SC_SIZE]) state <= 185 ; else state <= 183; end
```

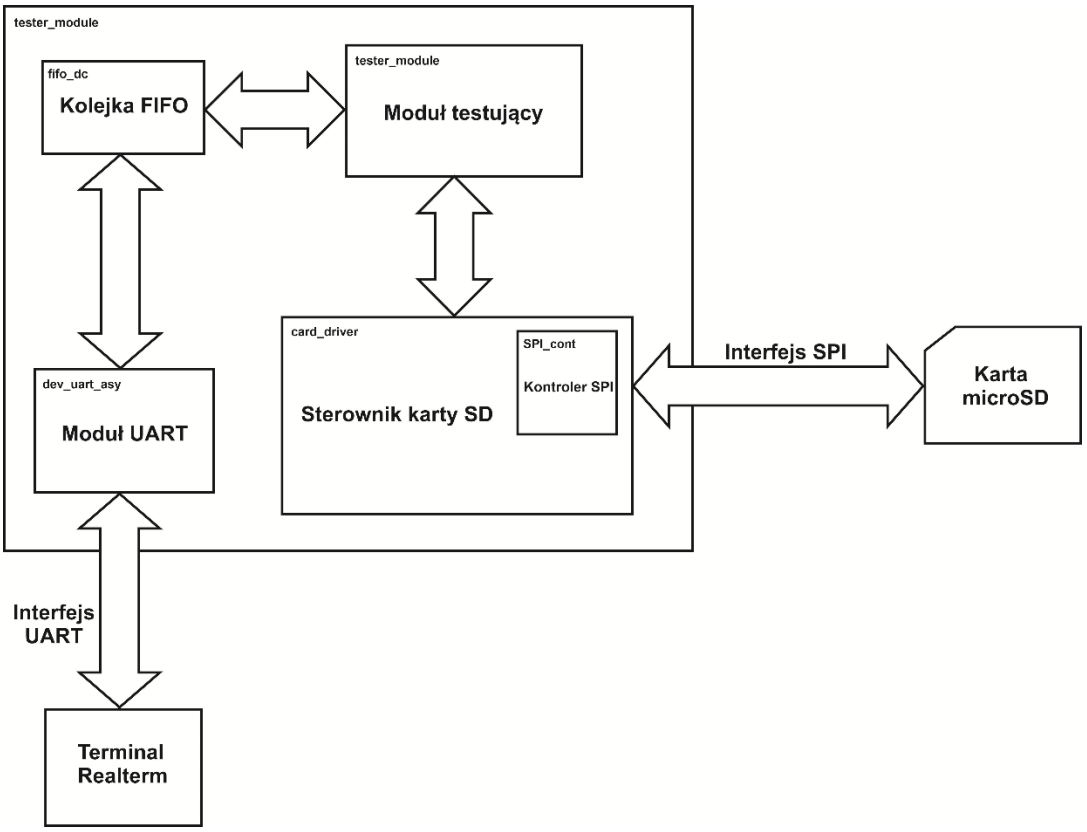
Proces odczytu bloku jest podobny do zapisu i wykonywany jest w stanie 183 i 184.

```
204: begin CS<=1; RD_ACK <= 1; RES_STB <= 1; RES_DATA <= "O";
state <= 205; end
```

Po zakończeniu procesu odczytu wystawiany jest sygnał `RD_ACK` który potwierdza proces odczytu i informuje o tym, że się zakończył.

3.3. Moduł testujący

Dla poprawnego przetestowania projektowanego układu wykorzystano 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 21 przedstawia schemat połączeń pomiędzy modułami od karty pamięci microSD do terminala, na którym obserwowane były wyniki podczas testowania modułu w sprzęcie. Wszystkie moduły zostały dodane i uruchomione w projekcie programu Lattice Diamond 3.10 a samo uruchomienie na układzie MACHXO2 4000HC firmy Lattice z wykorzystaniem płytki laboratoryjnej przedstawionej na rysunku 22.



Rysunek 21 Schemat blokowy testowanego modułu sterownika

Rysunek 22 Zdjęcie uruchomieniowej płytki laboratoryjnej

Implementacja testera posiada kilka procesów obsługi modułu UART, kolejki FIFO oraz testowanego modułu.

```
always @(posedge CLOCK50 or posedge RESET)
if (RESET) begin
      RD_EN <= 1;
      state <= 0;
end else case(state)
      0:  if (!RD_EMPTY && TX_RDY) begin RD_EN <= 1; state <= 1;
end else begin RD_EN <= 0; state <= 0; end
      1:  begin RD_EN <= 0; state <= 2; end
      2:  begin TX_STB <= 1; TX_DAT <= RD_Q; state <= 3; end
      3:  begin TX_STB <= 0; state <= 0; end
endcase
```

Pierwszy z nich stanowi automat stanów który odpowiada za obsługę modułu UART, ustawiając wartości na wyjściu `TxD`.

```
always @(posedge CLOCK50 or posedge RESET)
      RxD_r <= (RESET) ? 8'b11111111 : {RxD_r, RxD};
```

Następny proces ustawia rejestr `RxD_r` który replikowany jest sygnałem wejściowym `RxD`. Najstarszy bit z rejestru `RxD_r` wysyłany jest do modułu UART na wejście `RxD`.

```
always @(posedge CLOCK50 or posedge RESET)
if (RESET) begin
  WR_STB <= 0;
  WR_ADDR <= 0;
  WR_LENGTH <= 0;
  RD_STB <= 0;
  RD_ADDR <= 0;
  RD_LENGTH <= 0;
end else begin
  RD_STB <= 0;
  WR_STB <= 0;
  if (WR_ACK) WR_ADDR <= WR_ADDR + BURST_SIZE;
  if (RD_ACK) RD_ADDR <= RD_ADDR+1;
  if (RX_STB && (RX_DAT=="c")) begin
     WR_ADDR <= 0;
     RD_ADDR <= 0;
  end else if (RX_STB && (RX_DAT=="z")) begin
     WR_STB <= 1;
     WR_LENGTH <= BURST_SIZE;
  end else if (RX_STB && (RX_DAT=="o")) begin
     RD_STB <= 1;
     RD_LENGTH <= 1;
  end
end
```

Kolejny proces odpowiada za odbiór komend z UART i wysyłania danych do karty SD. Zapis rozpoczyna się, gdy odebrany znak `RX_DAT` jest równy wartości odpowiadającej kodowi ASCII (ang. American Standard Code for Information Interchange) znaku „z”, zaś odczyt, gdy odebrany zostaje znak „o”. Jeśli natomiast odebrany znak `RX_DAT` jest równy wartości odpowiadającej kodowi znaku „c” wówczas wartości adresowe są zerowane. W procesie tym można ustawić adres, od którego rozpocznie się zapis lub odczyt oraz liczbę bloków, dla których zostanie wykonana operacja.

```
always @(posedge CLOCK50 or posedge RESET)
if (RESET) begin
  WD_STB <= 1;
  WD_DATA <= "A";
end else if (WD_ACK) WD_DATA <= WD_DATA+1;
```

Powyższy proces służy do zdefiniowania danych które zostaną zapisane na karcie. W celu testowania blok karty pamięci został zapisany znakami ASCII zaczynając od litery A (0x41).

W celu przeprowadzenia pomiaru prędkości należało zmodyfikować procesy w module testującym.

```
`ifdef TEST_RW
.Data(RD_ACK ? "r" : "w"),
.WrEn(RD_ACK || WR_ACK),
.AlmostFull(BUSY),
`endif
```

Na kolejkę FIFO należało wysłać znak „r” lub „w” a następnie przeprowadzić testy dla zapisu lub odczytu dla określonej liczby bloków. Cała instrukcja została zdefiniowana dla instrukcji sterującej `TEST_RW`.

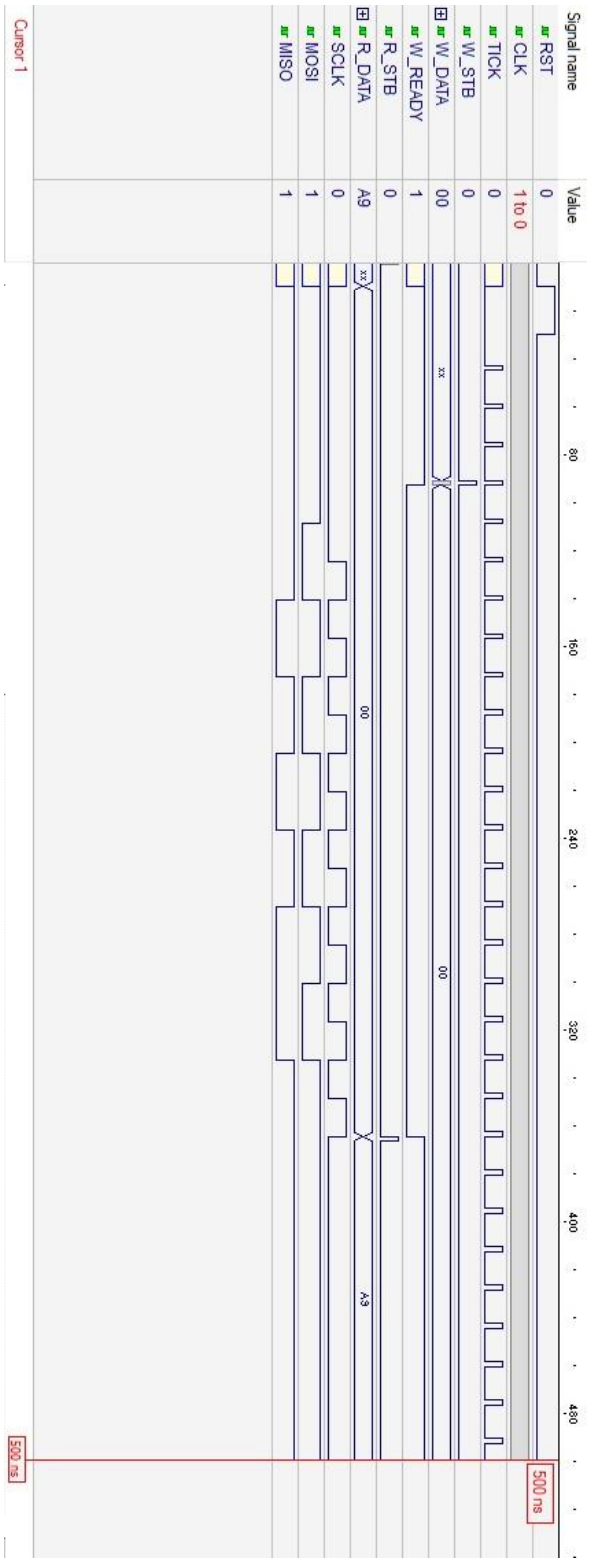
```
ifdef TEST_RW

wire read_nwrite = SWITCH[3];

wire [31:0] burst_len =
  SWITCH[2:0]==0 ? 1 :
  SWITCH[2:0]==1 ? 2 :
  SWITCH[2:0]==2 ? 5 :
  SWITCH[2:0]==3 ? 10 :
  SWITCH[2:0]==4 ? 100 :
  SWITCH[2:0]==5 ? 1000 :
  SWITCH[2:0]==6 ? 10000 : 100000;

reg [7:0] rwstate;

always @(posedge CLOCK50 or posedge RESET)
if (RESET) begin
  WR_STB <= 0;
  WR_ADDR <= 0;
  WR_LENGTH <= 0;
  RD_STB <= 0;
  RD_ADDR <= 0;
  RD_LENGTH <= 0;
  rwstate <=0;
end else case (rwstate)
  0: begin
      RD_STB <= read_nwrite;
      WR_STB <= !read_nwrite;
      RD_LENGTH <= burst_len;
      WR_LENGTH <= burst_len;
      if (RD_ACK) rwstate <= 1;
  end
  1: begin
    RD_STB <= 0;
     WR_STB <= 0;
    if (!BUSY) rwstate <= 2;
  end
  2: begin
    RD_ADDR <= RD_ADDR + burst_len;
     WR_ADDR <= WR_ADDR + burst_len;
    rwstate <= 0;
  end
endcase
`endif
```

Liczba bloków w trybie wieloblokowym została zdefiniowana jako rejestr `burst_len` który ustawiany jest w zależności od wartości trzech przełączników sterujących. Natomiast w zależności od ustawienia czwartego przełącznika ustawiany jest zapis lub odczyt który jest testowany.

4. Symulacja

4.1. Symulacja kontrolera SPI

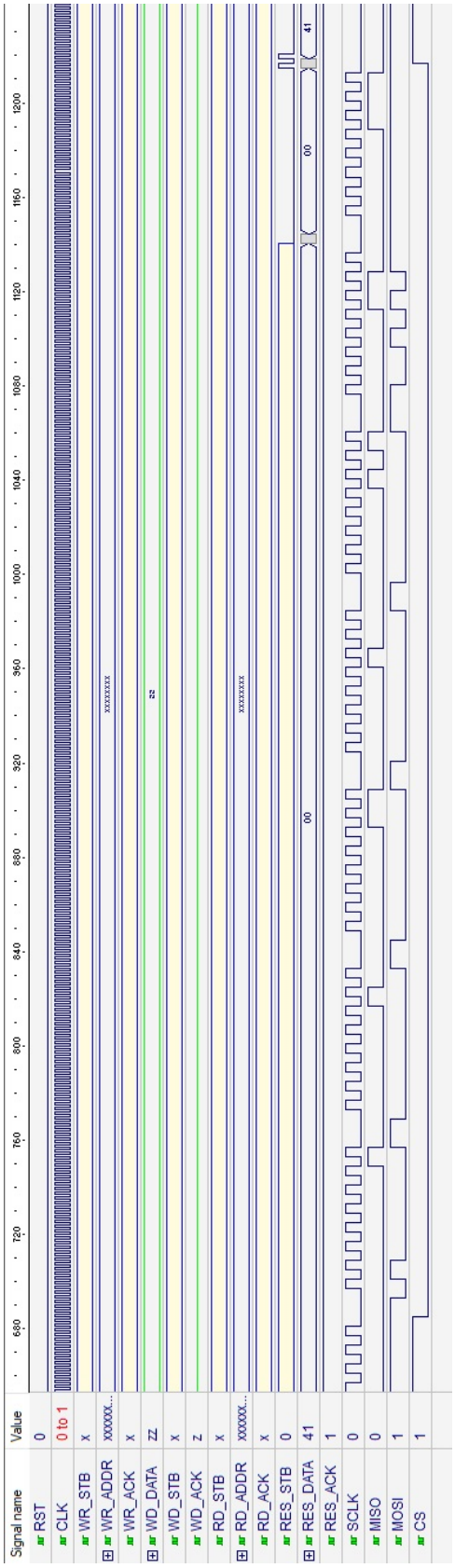
Rysunek 23 przedstawia wyniki zapisu i odczytu jednego bajtu danych poprzez protokół SPI, do zapisu (wysłania) użyto wartości 01010101_B (85_{10}) natomiast do odczytu (odebrania) wartości 10101001_B (169_{10}).



Rysunek 23 Wyniki symulacji kontrolera SPI

4.2. Symulacja sterownika karty pamięci

Rysunek 24 przedstawia przesyłanie komendy CMD0 do karty pamięci przez wyjście MOSI oraz przykładowe dane odbioru danych na wejście MISO.



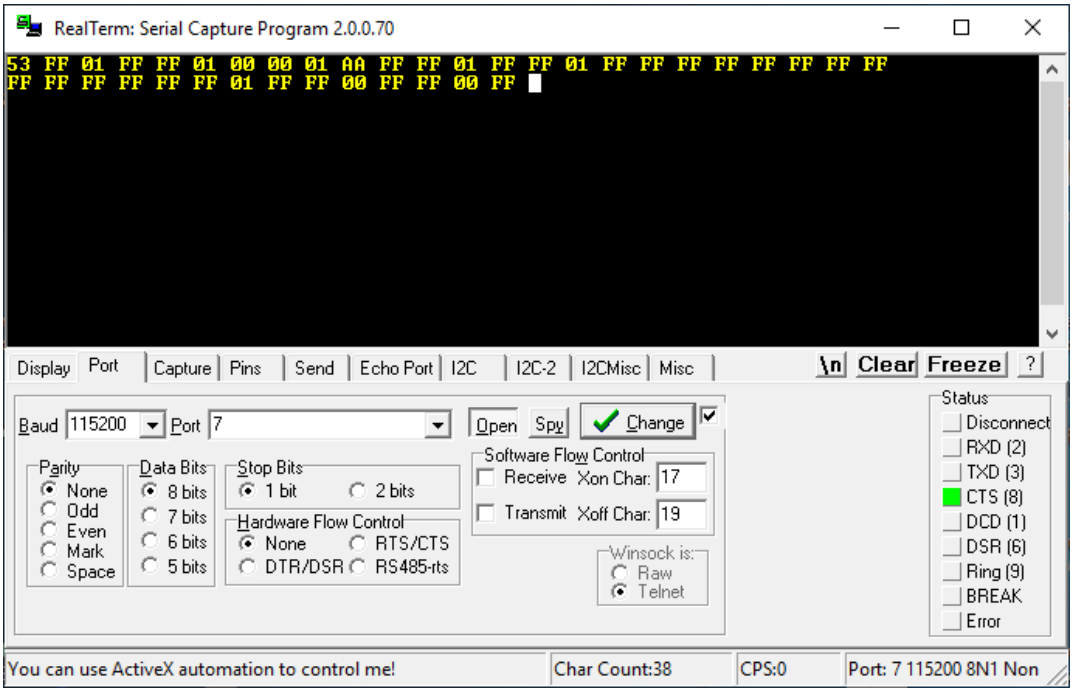
Rysunek 24 Wyniki symulacji przesłania komendy CMD0 i odbioru przykładowych danych

5. Testowanie w układzie FPGA

5.1. Otrzymane wyniki działania modułu sterownika

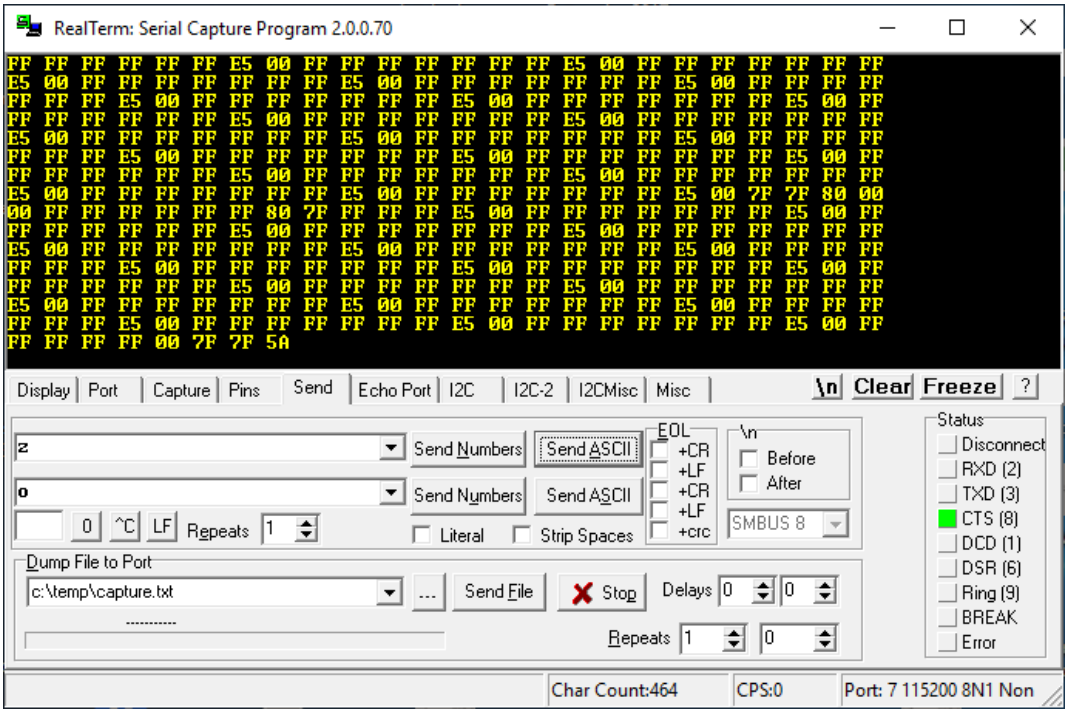
Do testów zastosowano moduł testujący a wyniki były obserwowane na terminalu Realterm. Otrzymane rezultaty z testów to:

a) Wyniki procesu inicjalizacji karty pamięci



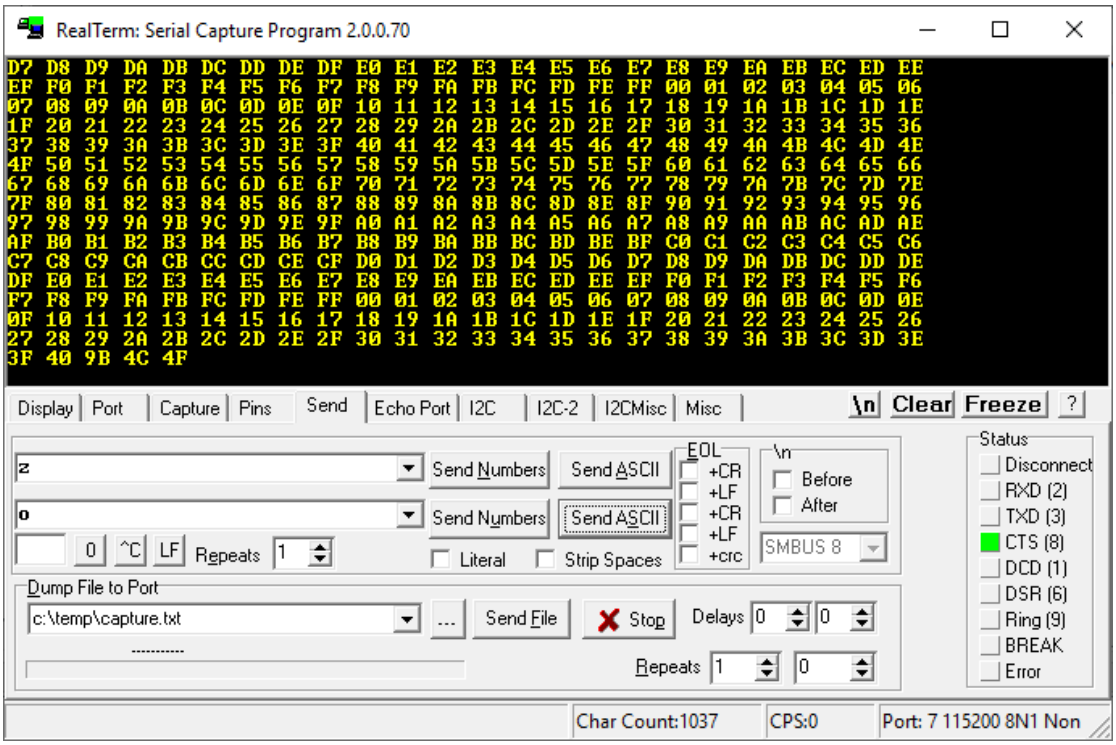
Rysunek 25 Wartości otrzymane po inicjalizacji karty pamięci

b) Wyniki zapisu danych na karcie

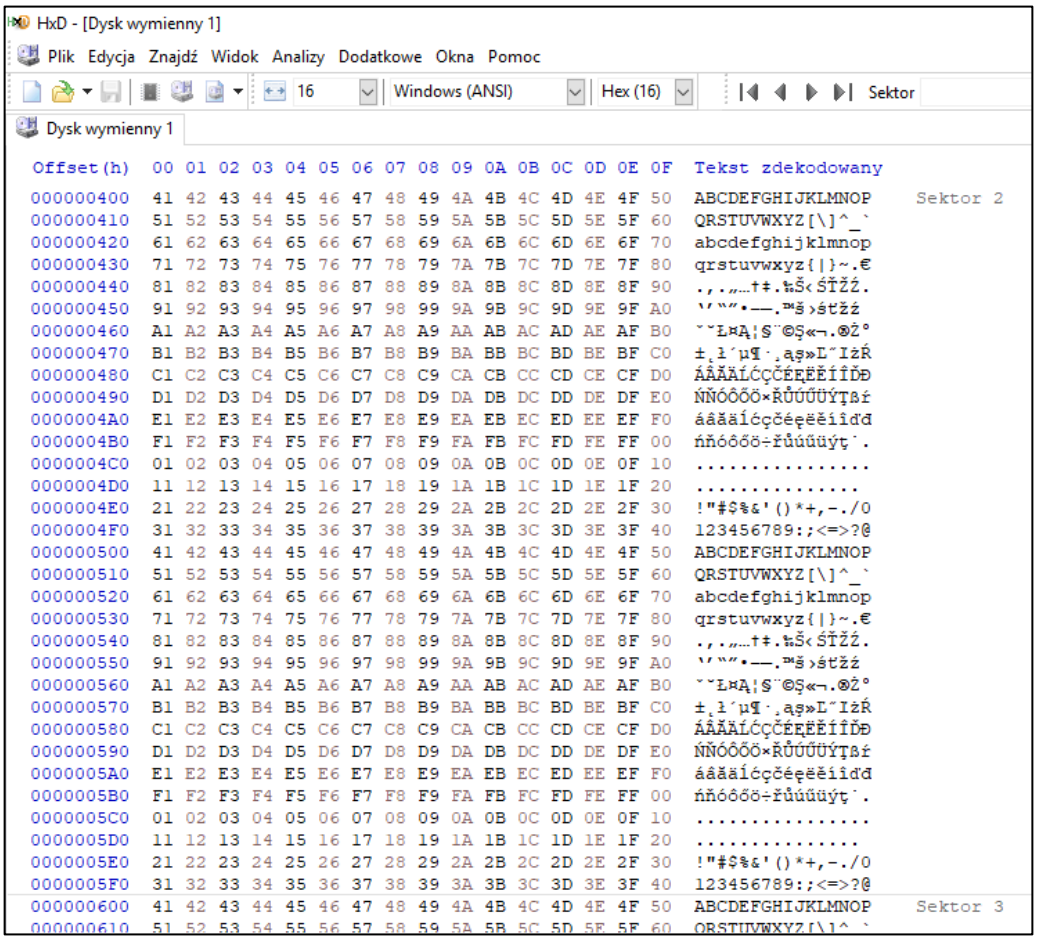


Rysunek 26 Wartości otrzymane po zapisie wielu bloków na karcie pamięci

c) Wyniki odczytu danych z karty



Rysunek 27 Wartości otrzymane po odczycie wielu bloków z karty pamięci



Rysunek 28 Podgląd zapisanych danych w programie HxD

5.2. Sposób pomiaru prędkości oraz otrzymane wyniki

Pomiaru prędkości dokonano z wykorzystaniem następujących ilości bloków w trybie wieloblokowym: 1, 2, 5, 10, 100, 1000, 10000. W szczególności, prędkość zapisu i odczytu zmierzono poprzez pomiar liczby wyemitowanych znaków sterujących „w" (w zapisie heksadecymalnym 0x77) dla operacji zapisu lub „r" (w zapisie heksadecymalnym 0x72) dla operacji odczytu, wysyłanych po ukończeniu każdej operacji wieloblokowej, przez określony czas. Wynikową prędkość transferu danych wyznaczono wg wzoru:

$$V = \frac{liczba\ odczytanego\ znaku * multiburst * \ długość\ bloku\ (512)}{10}$$

Do odczytu zapisanego znaku „w" lub „r" została użyta funkcja Capture z terminala Realterm, która umożliwia przechwytywanie wyników do pliku.

Pierwszego z pomiarów dokonano dla ustawionych parametrów:

```
parameter START_DIVIDER = 255;
parameter TRANSMISSION_DIVIDER = 4;
```

które konfigurowały docelowy dzielnik zegara jako TRANSMISSION_DIVIDER + 1 = 5, dlatego maksymalna prędkość wynosiła:

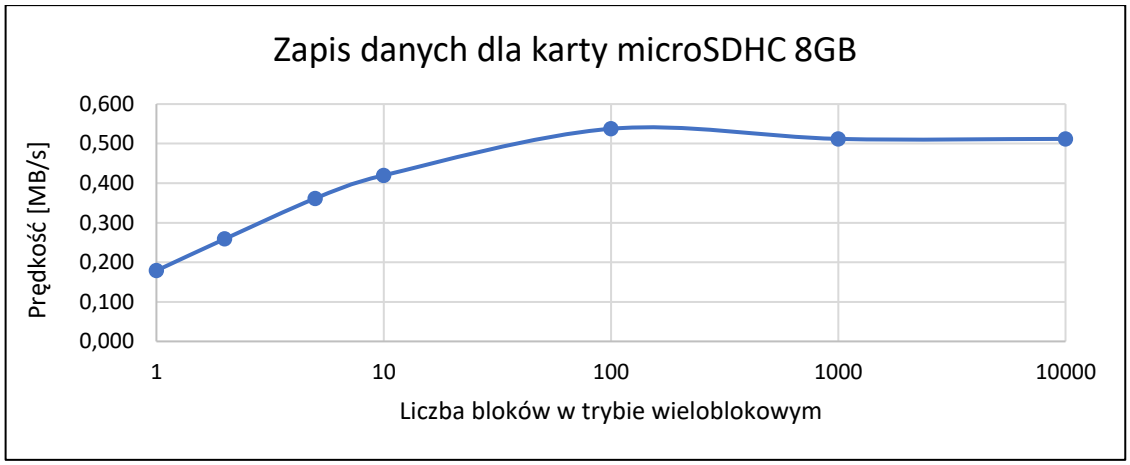
$$V_{max} = 50\ 000\ 000\ [\text{Hz}]\ /5[\text{dzielnik}]/8\ [\text{takt/bajt}]/2\ [1/\text{takt}]\ = 0{,}625\ [\text{MB/s}]$$

Pomiarów odczytu i zapisu dokonano dla karty pamięci SANDISK microSDHC 8GB.

a) Wyniki pomiarów dla karty pamięci SANDISK microSDHC 8GB dla Vmax = 0,625 MB/s

Tabela 5 Wyniki dla zapisu danych na karcie pamięci microSDHC 8GB dla Vmax = 0,625 MB/s

Czas [s]	Liczba bloków w trybie wieloblokowym	Liczba wyemitowanych znaków "w" (0x77)	Długość bloku	Otrzymana prędkość [MB/s]
10	1	3495	512	0,179
10	2	2530	512	0,259
10	5	1412	512	0,361
10	10	820	512	0,420
10	100	105	512	0,538
10	1000	10	512	0,512
10	10000	1	512	0,512

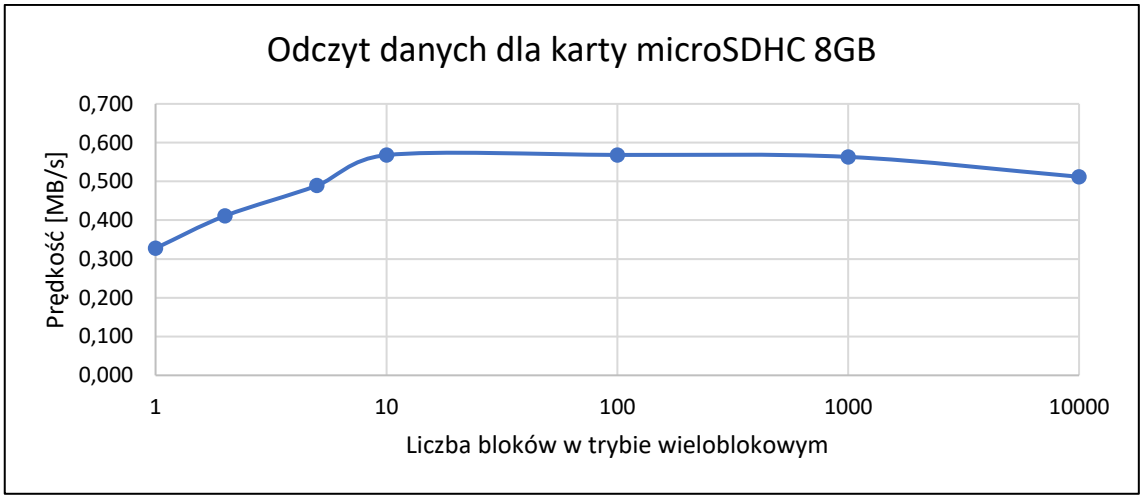


Rysunek 29 Zapis danych dla karty microSDHC 8GB dla Vmax = 0,625 MB/s

Tabela 5 i rysunek 29 przedstawia prędkość zapisu, która wraz ze wzrostem liczby bloków w trybie wieloblokowym rośnie do wartości 0,538 MB/s a następnie spada do wartości 0,512 MB/s.

Tabela 6 Wyniki dla odczytu danych na karcie pamięci microSDHC 8GB dla Vmax = 0,625 MB/s

Czas [s]	Liczba bloków w trybie wieloblokowym	Liczba wyemitowanych znaków "r" (0x72)	Długość bloku	Otrzymana prędkość [MB/s]
10	1	6400	512	0,328
10	2	4015	512	0,411
10	5	1911	512	0,489
10	10	1109	512	0,568
10	100	111	512	0,568
10	1000	11	512	0,563
10	10000	1	512	0,512



Rysunek 30 Odczyt danych dla karty microSDHC 8GB dla Vmax = 0,625 MB/s

Tabela 6 i rysunek 30 przedstawia prędkość odczytu, która wraz ze wzrostem liczby bloków w trybie wieloblokowym narastała do wartości 0,568 MB/s a następnie delikatnie spadła do 0,512 MB/s.

Następnego pomiaru dokonano dla ustawionych parametrów:

```
parameter START_DIVIDER = 255;
parameter TRANSMISSION_DIVIDER = 1;
```

które definiowały docelowy dzielnik zegara jako TRANSMISSION_DIVIDER + 1 = 2, zaś maksymalna prędkość wynosiła:

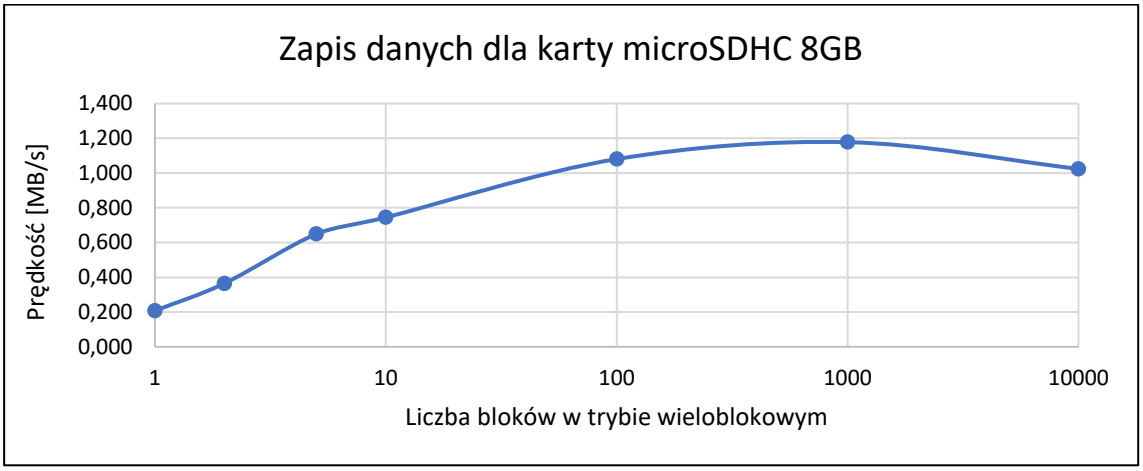
$$V_{max} = 50\,000\,000\ [\text{Hz}]\ /2[\text{dzielnik}]/8\ [\text{takt/bajt}]/2\ [1/\text{takt}] = 1{,}562\ [\text{MB/s}]$$

Pomiarów odczytu i zapisu dokonano dla karty pamięci SANDISK microSDHC 8GB.

b) Wyniki pomiarów dla karty pamięci SANDISK microSDHC 8GB dla Vmax = 1,562 MB/s

Tabela 7 Wyniki dla zapisu danych na karcie pamięci microSDHC 8GB dla Vmax = 1,562 MB/s

Czas [s]	Liczba bloków w trybie wieloblokowym	Liczba wyemitowanych znaków "w" (0x77)	Długość bloku	Otrzymana prędkość [MB/s]
10	1	4056	512	0,208
10	2	3565	512	0,365
10	5	2536	512	0,649
10	10	1455	512	0,745
10	100	211	512	1,080
10	1000	23	512	1,178
10	10000	2	512	1,024

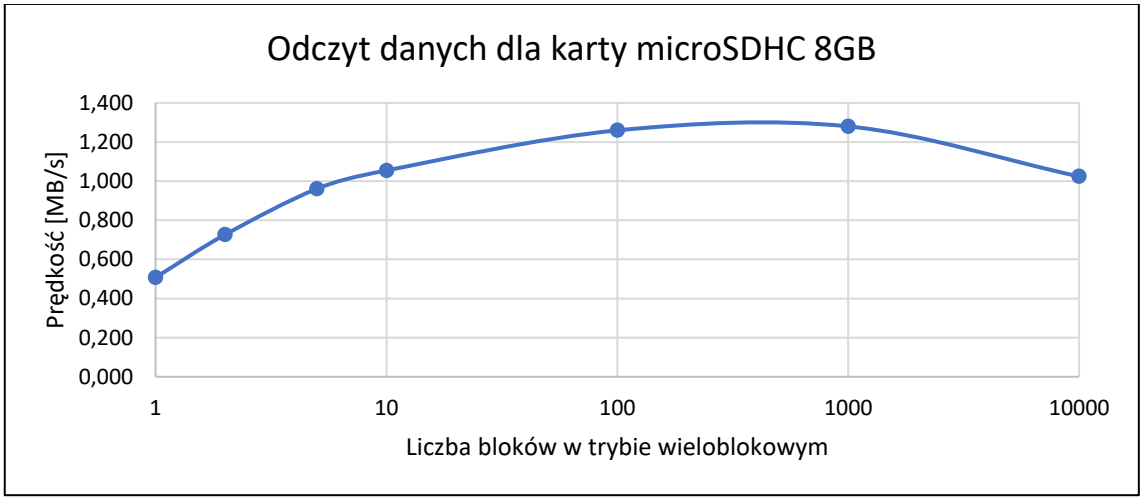


Rysunek 31 Zapis danych dla karty microSDHC 8GB dla Vmax = 1,562 MB/s

Tabela 7 i rysunek 31 przedstawia prędkość zapisu, która wraz ze wzrostem liczby bloków w trybie wieloblokowym narastała do wartości 1,17 MB/s a następnie delikatnie spadła do 1,024 MB/s.

Tabela 8 Wyniki dla odczytu danych na karcie pamięci microSDHC 8GB dla Vmax = 1,562 MB/s

Czas [s]	Liczba bloków w trybie wieloblokowym	Liczba wyemitowanych znaków "r" (0x72)	Długość bloku	Otrzymana prędkość [MB/s]
10	1	9914	512	0,508
10	2	7094	512	0,726
10	5	3754	512	0,961
10	10	2060	512	1,055
10	100	246	512	1,260
10	1000	25	512	1,280
10	10000	2	512	1,024



Rysunek 32 Odczyt danych dla karty microSDHC 8GB dla Vmax = 1,562 MB/s

Tabela 8 i rysunek 32 przedstawia prędkość odczytu, która wraz ze wzrostem liczby bloków w trybie wieloblokowym wzrastała do 1,28 MB/s a następnie zmalała do 1,024 MB/s.

Kolejnych pomiarów dokonano przy jednoczesnym ustawieniu parametrów:

```
parameter START_DIVIDER = 255;
parameter TRANSMISSION_DIVIDER = 0;
```

które definiowały docelowy dzielnik zegara jako TRANSMISSION_DIVIDER + 1 = 1, dlatego maksymalna prędkość wynosiła:

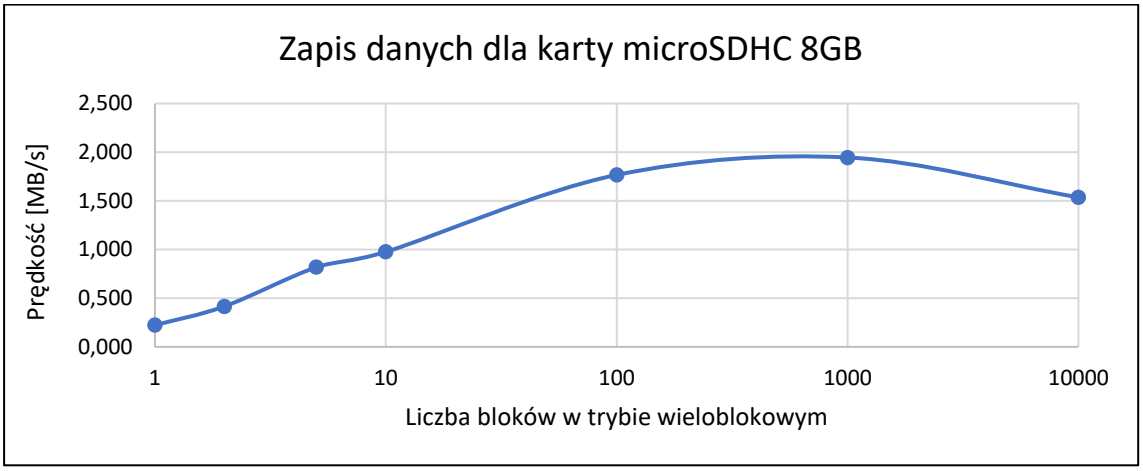
$$V_{max} = 50\,000\,000\ [\mathrm{Hz}]\ /1[\mathrm{dzielnik}]/8\ [\mathrm{takt/bajt}]/2\ [1/\mathrm{takt}] = 3{,}125\ [\mathrm{MB/s}]$$

Pomiarów odczytu i zapisu dokonano dla karty pamięci SANDISK microSDHC 8GB, SANDISK ULTRA microSDHC 32GB oraz dla karty SANDISK ULTRA ANDROID microSDXC 64GB a wyniki przedstawiono poniżej.

c) Wyniki pomiarów dla karty pamięci SANDISK microSDHC 8GB dla Vmax = 3,125 MB/s

Tabela 9 Wyniki dla zapisu danych na karcie pamięci microSDHC 8GB dla Vmax = 3,125 MB/s

Czas [s]	Liczba bloków w trybie wieloblokowym	Liczba wyemitowanych znaków "w" (0x77)	Długość bloku	Otrzymana prędkość [MB/s]
10	1	4352	512	0,223
10	2	4061	512	0,416
10	5	3194	512	0,818
10	10	1906	512	0,976
10	100	345	512	1,766
10	1000	38	512	1,946
10	10000	3	512	1,536

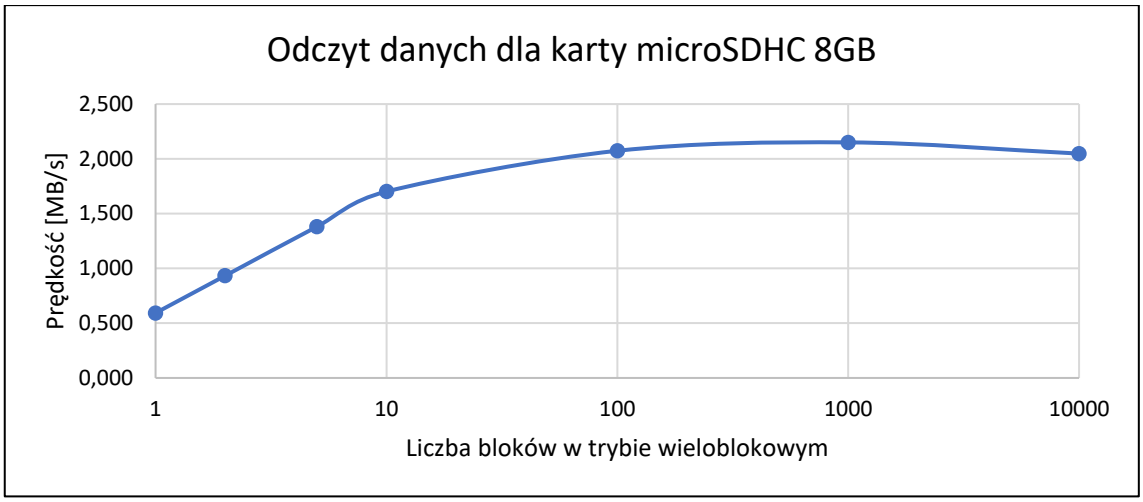


Rysunek 33 Zapis danych dla karty microSDHC 8GB dla Vmax = 3,125 MB/s

Tabela 9 i rysunek 33 przedstawia prędkość zapisu, która wraz ze wzrostem liczby bloków w trybie wieloblokowym narastała do wartości 1,946 MB/s a następnie delikatnie spadła do 1,536 MB/s.

Tabela 10 Wyniki dla odczytu danych na karcie pamięci microSDHC 8GB dla Vmax = 3,125 MB/s

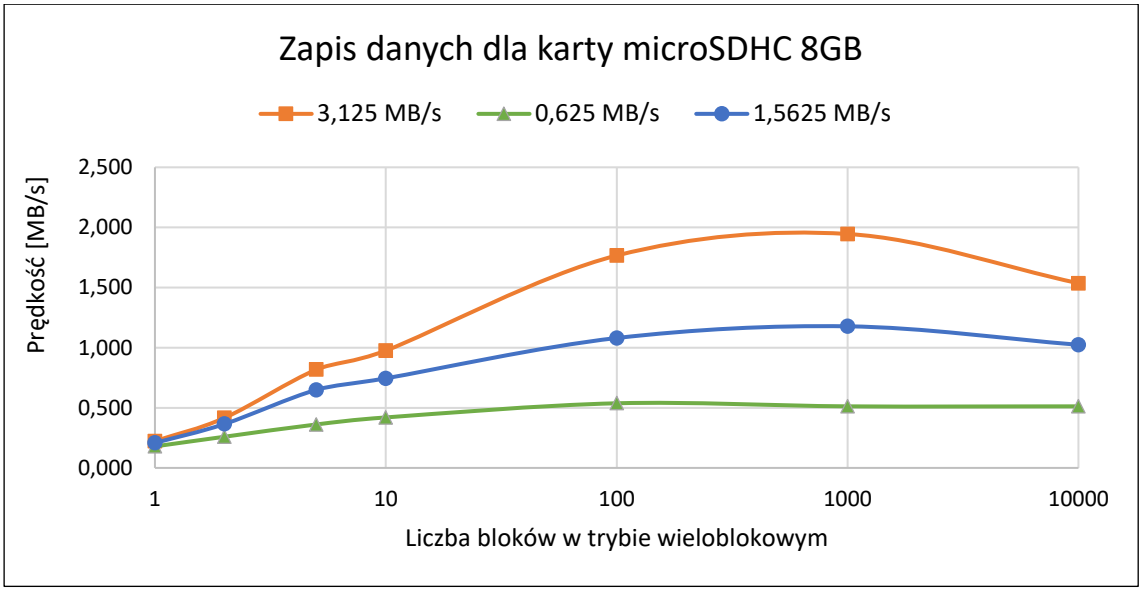
Czas [s]	Liczba bloków w trybie wieloblokowym	Liczba wyemitowanych znaków "r" (0x72)	Długość bloku	Otrzymana prędkość [MB/s]
10	1	11579	512	0,593
10	2	9105	512	0,932
10	5	5394	512	1,381
10	10	3323	512	1,701
10	100	405	512	2,074
10	1000	42	512	2,150
10	10000	4	512	2,048



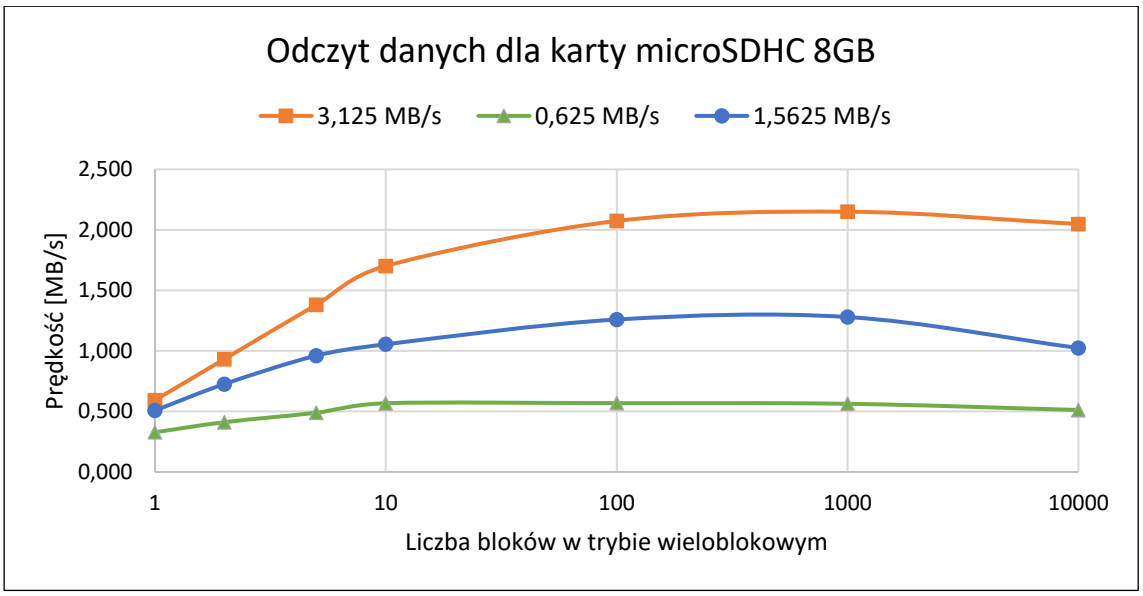
Rysunek 34 Odczyt danych dla karty microSDHC 8GB dla Vmax = 3,125 MB/s

Tabela 10 i rysunek 34 przedstawia prędkość odczytu, która wraz ze wzrostem liczby bloków w trybie wieloblokowym rośnie do wartości 2,15 MB/s a następnie zaczyna spadać do wartości 2,048 MB/s.

d) Porównanie wyników pomiarów dla karty pamięci microSDHC 8GB



Rysunek 35 Porównanie wyników dla zapisu danych na karcie pamięci microSDHC 8GB dla różnych Vmax



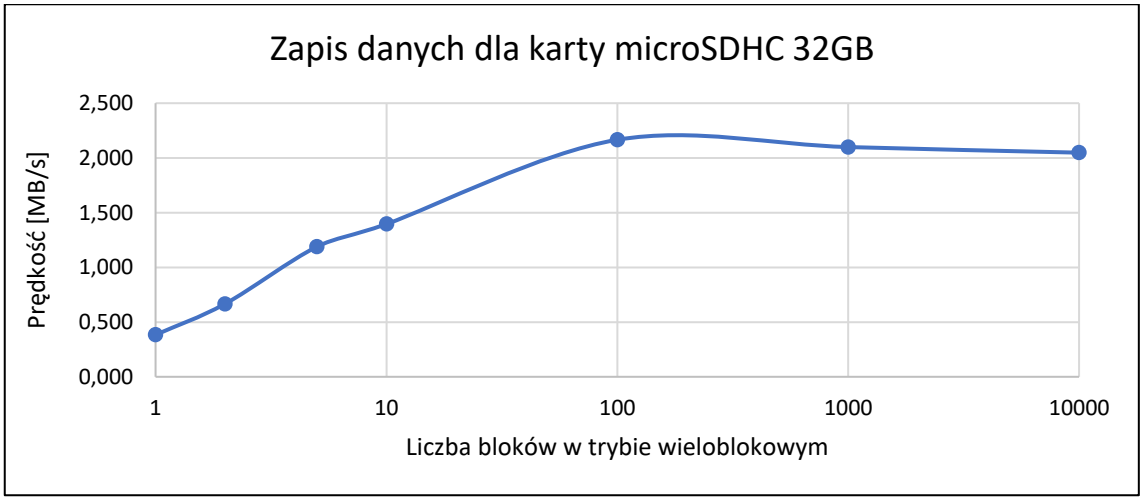
Rysunek 36 Porównanie wyników dla odczytu danych na karcie pamięci microSDHC 8GB dla różnych Vmax

Rysunek 35 oraz rysunek 36 przedstawiają porównanie otrzymanych wyników dla zapisu i odczytu danych dla karty pamięci microSDHC 8GB przy różnych wartościach prędkości maksymalnej Vmax. Dla wszystkich pomiarów prędkość rośnie wraz ze wzrostem liczby bloków w trybie wieloblokowym a następnie delikatnie spada.

e) Wyniki pomiarów dla karty pamięci SANDISK ULTRA microSDHC 32GB dla Vmax = 3,125 MB/s

Tabela 11 Wyniki dla zapisu danych na karcie pamięci microSDHC 32GB dla Vmax = 3,125 MB/s

Czas [s]	Liczba bloków w trybie wieloblokowym	Liczba wyemitowanych znaków "w" (0x77)	Długość bloku	Otrzymana prędkość [MB/s]
10	1	7488	512	0,383
10	2	6509	512	0,667
10	5	4644	512	1,189
10	10	2726	512	1,396
10	100	423	512	2,166
10	1000	41	512	2,099
10	10000	4	512	2,048

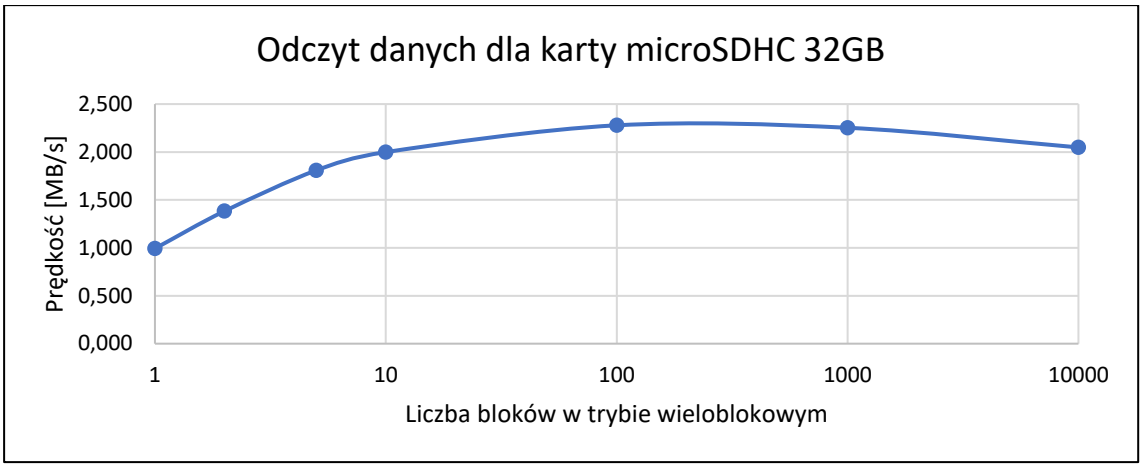


Rysunek 37 Zapis danych dla karty microSDHC 32GB dla Vmax = 3,125 MB/s

Prędkość zapisu rośnie od wartości 0,3 MB/s a następnie ustatkowuje się na prędkości ok 2,0 MB/s co przedstawiono na rysunku 37 i tabeli 11.

Tabela 12 Wyniki dla odczytu danych z karty pamięci microSDHC 32GB dla Vmax = 3,125 MB/s

Czas [s]	Liczba bloków w trybie wieloblokowym	Liczba wyemitowanych znaków "r" (0x72)	Długość bloku	Otrzymana prędkość [MB/s]
10	1	19390	512	0,993
10	2	13502	512	1,383
10	5	7064	512	1,808
10	10	3904	512	1,999
10	100	445	512	2,278
10	1000	44	512	2,253
10	10000	4	512	2,048



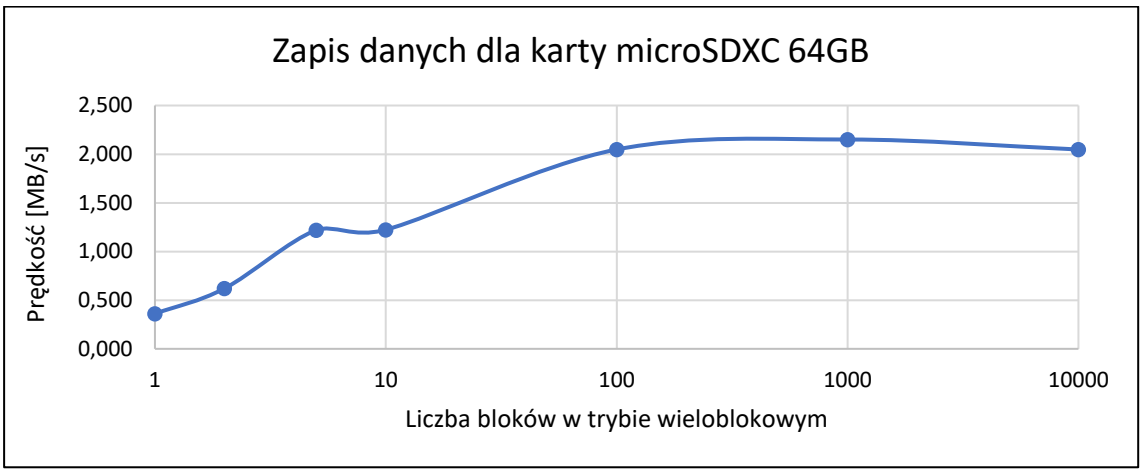
Rysunek 38 Odczyt danych dla karty microSDHC 32GB dla Vmax = 3,125 MB/s

Tabela 12 i rysunek 38 przedstawia prędkość odczytu, która wraz ze wzrostem liczby bloków w trybie wieloblokowym rośnie do wartości 2,2 MB/s a następnie zaczyna spadać do wartości 2,0 MB/s.

f) Wyniki pomiarów dla karty pamięci SANDISK ULTRA ANDROID microSDXC 64GB dla Vmax = 3,125 MB/s

Tabela 13 Wyniki dla zapisu danych na karcie pamięci microSDXC 64GB dla Vmax = 3,125 MB/s

Czas [s]	Liczba bloków w trybie wieloblokowym	Liczba wyemitowanych znaków "w" (0x77)	Długość bloku	Otrzymana prędkość [MB/s]
10	1	7046	512	0,361
10	2	6052	512	0,620
10	5	4762	512	1,219
10	10	2390	512	1,224
10	100	400	512	2,048
10	1000	42	512	2,150
10	10000	4	512	2,048

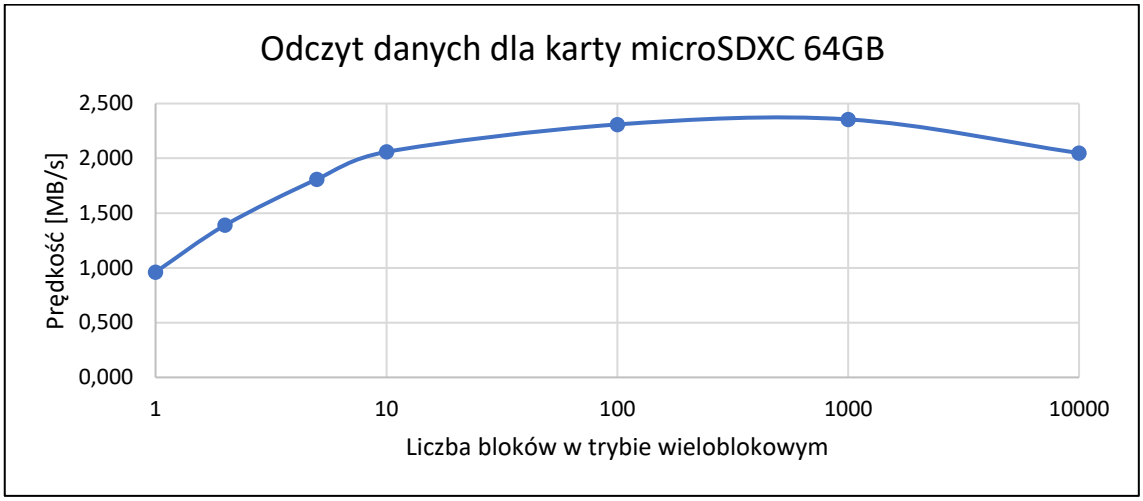


Rysunek 39 Zapis danych dla karty microSDXC 64GB dla Vmax = 3,125 MB/s

Tabela 13 i rysunek 39 przedstawia prędkość zapisu, która wraz ze wzrostem liczby bloków w trybie wieloblokowym narastała do wartości 2,1 MB/s a następnie spadła do 2,0 MB/s.

Tabela 14 Wyniki dla odczytu danych z karty pamięci microSDXC 64GB dla Vmax = 3,125 MB/s

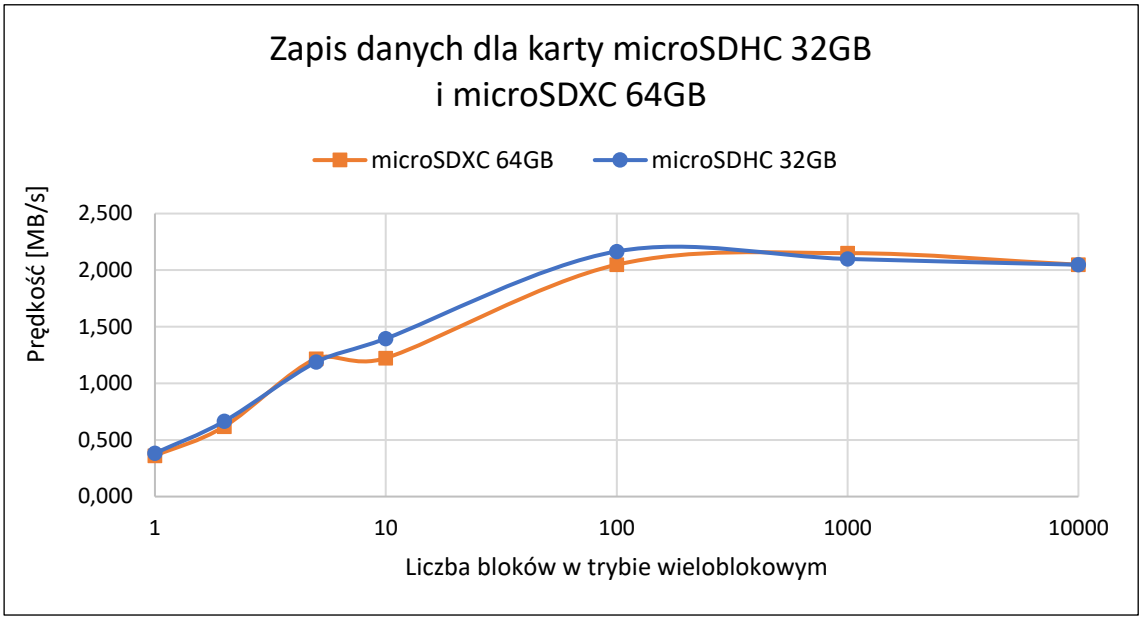
Czas [s]	Liczba bloków w trybie wieloblokowym	Liczba wyemitowanych znaków "r" (0x72)	Długość bloku	Otrzymana prędkość [MB/s]
10	1	18785	512	0,962
10	2	13568	512	1,389
10	5	7068	512	1,809
10	10	4021	512	2,059
10	100	451	512	2,309
10	1000	46	512	2,355
10	10000	4	512	2,048



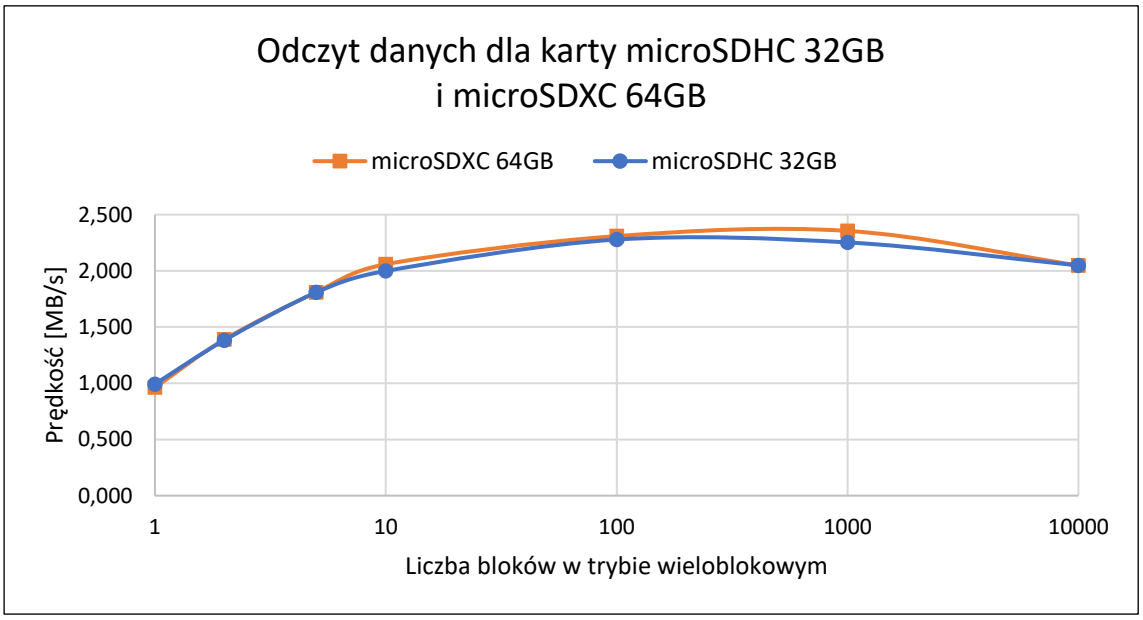
Rysunek 40 Odczyt danych dla karty microSDXC 64GB dla Vmax = 3,125 MB/s

Tabela 14 i rysunek 40 przedstawia prędkość odczytu, która wraz ze wzrostem liczby bloków w trybie wieloblokowym wzrastała do 2,3 MB/s wraz ze wzrostem odczytywanych bloków.

g) Porównanie wyników pomiarów na karcie pamięci microSDHC 32GB oraz microSDXC 64GB.



Rysunek 41 Zapis danych dla karty microSDHC 32GB i microSDXC 64GB dla Vmax = 3,125 MB/s



Rysunek 42 Odczyt danych dla karty microSDHC 32GB i microSDXC 64GB dla Vmax = 3,125 MB/s

Rysunek 41 oraz rysunek 42 przedstawia porównanie otrzymanych wyników dla zapisu i odczytu danych dla karty pamięci microSDHC 32GB oraz microSDXC 64GB przy tej samej wartości maksymalnej prędkości Vmax = 3,125 MB/s. Dla wszystkich pomiarów prędkość rośnie wraz ze wzrostem liczby bloków w trybie wieloblokowym a następnie delikatnie spada. Otrzymane wartości prędkości pomiędzy kartą microSDHC 32GB oraz micro SDXC 64GB są zbliżone, a drobne różnice wynikają z parametrów dwóch kart różnego typu.

6. Podsumowanie

Celem niniejszej pracy była implementacja sterownika karty pamięci SD. W tym celu zapoznano się z dokumentacją interfejsu SPI, działaniem kart pamięci MMC i SD oraz parametrami kart pamięci zgodnie z ich docelowym przeznaczeniem. Dokładne poznanie tych zagadnień wymagało znacznego poszerzenia wiedzy nabytej w trakcie studiów. Dla implementacji modułu sterownika SPI oraz sterownika karty pamięci w początkowej fazie pracy została wykonana symulacja. Następnie zostało stworzone środowisko testowe składające się z dodatkowych modułów, w którym zostały przeprowadzone testy z wykorzystaniem uruchomieniowej płytki laboratoryjnej.

Zapis i odczyt został zrealizowany przy użyciu komend obsługujących tryb wieloblokowy, co pozwala na zapis lub odczyt wielu bloków podczas jednej operacji. Sterownik SPI stanowi osobny moduł, dzięki czemu w razie potrzeby można dołączyć kolejne moduły dla obsługi wielu kart przy niewielkiej ingerencji w implementację sterownika karty.

Podczas pomiarów zaobserwowałem, że wraz ze wzrostem liczby bloków w trybie wieloblokowym rośnie prędkość zapisu lub odczytu. Przy czym na liczbę zapisywanych bloków bardziej wrażliwy jest zapis niż odczyt, dlatego otrzymane prędkości zapisu są mniejsze niż odczytu. Dla wszystkich wykonanych pomiarów odczyt jest szybszy niż zapis jednak dla dużej liczby bloków w trybie wieloblokowym tj. 10000 transmisja się nasyca i prędkość spada. Wykonane pomiary prędkości nie osiągnęły przypuszczalnej maksymalnej wartości, a prędkość kart pamięci SD bardzo dużo zależy od prędkości taktowania. Prędkość rośnie wraz ze wzrostem prędkości SCLK co oznacza to, że najprawdopodobniej część z mechanizmów zapisu i odczytu karty jest taktowanych również zegarem z linii SCLK.

Wszystkie cele postawione w projekcie udało się zrealizować, jednakże możliwe są dalsze kierunki prac w celu jego rozwoju - np. projekt może być rozwinięty o moduł obliczający sumy kontrolne CRC i dokonujący retransmisji bloków w trakcie wystąpienia błędów.

7. Bibliografia

[1] Zbigniew Hajduk, Wprowadzenie do języka Verilog, wydanie I, Wydawnictwo BTC, 2009

[2] J. Bhasker, Verilog HDL Synthesis, A Practical Primer, Star Galaxy Publishing, 1998

[3] Bogusz Jacek, Lokalne interfejsy szeregowe w systemach cyfrowych, wydanie I, Wydawnictwo BTC, 2004

[4] Jabłoński Tomasz, Karty SD/MMC w systemach mikroprocesorowych, wydanie I, Wydawnictwo BTC, 2009

[5] Martin Evans, Joshua Noble, Jordan Hochenbaum, Arduino w akcji, Wydawnictwo Helion, 2014

[6] How to Use MMC/SDC, http://elm-chan.org/docs/mmc/mmc_e.html

[7] Lecture 12: SPI and SD cards, http://www.dejazzer.com/ee379/lecture_notes/lec12_sd_card.pdf

[8] SD Specifications, Part 1, Physical Layer, Simplified Specification, Version 6.00, August 29, 2018, https://www.sdcard.org/downloads/pls/

[9] SD Standard Overview, https://www.sdcard.org/developers/overview/index.html

[10] Lexar, Memory Card, https://www.lexar.com/products/memory-cards/

[11] FPGA, CPLD and ASIC, http://www.latticesemi.com/

[12] The Design Verification Company - Aldec, www.aldec.com

[13] Program HxD, https://mh-nexus.de/en/hxd/

8. Spis tabel

9. Spis rysunków

Dodatek A. Wykaz plików na płycie CD

SPI_cont.rar

card_driver.rar

tester_module.rar

Praca magisterska.pdf

Praca magisterska.docx