POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Elektronika i Telekomunikacja
SPECJALNOŚĆ: Aparatura Elektroniczna

PRACA DYPLOMOWA MAGISTERSKA

Układ eliminacji przeplotu w obrazie wideo na układzie programowalnym FPGA

Video scan doubler based on FPGA

AUTOR:
Balthasar Szczepański

PROWADZĄCY PRACĘ:
Dr inż. Grzegorz Głomb
Katedra Metrologii Elektronicznej i Fotonicznej

OCENA PRACY:

WROCŁAW 2015

1. Spis treści

1. Spis treści....2
2. Wprowadzenie....4
2.1. Cel pracy....4
2.2. Sygnał wideo....4
2.2.1. Struktura czasowa sygnału wideo....4
2.2.2. Standardy wideo....8
2.2.3. Problemy ze zgodnością standardów....10
2.3. Istniejące rozwiązania....11
2.3.1. Urządzenia typu scandoubler....11
2.3.2. FPGA jako układ przetwarzający sygnał....12
3. Platforma sprzętowa i programistyczna....13
3.1. Platforma sprzętowa....13
3.1.1. Komputer Commodore Amiga 500....13
3.1.2. Moduł FPGA....15
3.1.3. Pamięć DDR2....16
3.1.4. Układ FPGA....16
3.1.5. Pamięć BlockRam....17
3.1.6. Porty IO....17
3.1.7. Odbiornik sygnału wideo....18
3.2. Platforma programistyczna....18
4. Budowa i działanie układu....19
4.1. Część sprzętowa....19
4.1.1. Pozyskanie sygnału wideo....19
4.1.2. Interfejs wejściowy....21
4.1.3. Interfejs wyjściowy....23
4.1.4. Wyjście testowe....25
4.1.5. Zasilanie....25
4.2. Część programowa....26
4.2.1. Zasada działania....26
4.2.2. Rozdzielczość sygnału wyjściowego....27
4.2.3. Generacja sygnałów zegarowych....27
4.2.4. Generacja sygnału wyjściowego....28
4.2.5. Generacja sygnału testowego....30
4.2.6. Rozpoznawanie sygnału wejściowego i odtwarzanie pozycji....34
4.2.7. Buforowanie obrazu....37
4.2.8. Kontroler pamięci DDR2....38
4.3. Uruchomienie urządzenia....44
5. Testowanie....45
5.1. Symulacja....45
5.2. Praca krokowa....45

5.3. Port VGA....46
5.4. Obserwacja pracy urządzenia....46
6. Podsumowanie....49
6.1. Przebieg pracy....49
6.2. Ocena....49
6.3. Dalszy rozwój....49
7. Bibliografia....50

2. Wprowadzenie

2.1. Cel pracy

Celem niniejszej pracy było opracowanie oraz wykonanie prototypu urządzenia opartego o układ FPGA, którego działanie polega na konwersji sygnału wideo RGB o częstotliwości odświeżania poziomego ok. 15kHz do sygnału zgodnego ze standardem VESA, o częstotliwości odświeżania poziomego ok. 31kHz, który może być odebrany przez monitor VGA. Urządzenie powinno obsługiwać różne rozdzielczości sygnału wejściowego, a także eliminować tzw. przeplot w przypadku sygnału z przeplotem.

Wiele komputerów starszej generacji generuje sygnał wideo, który nie jest zgodny ze współczesnym standardem VESA, a zatem niemożliwe jest korzystanie z tych komputerów przy użyciu współczesnych monitorów. Stąd potrzeba stworzenia urządzenia do konwersji sygnału. Większość dostępnych na rynku urządzeń do konwersji sygnału wideo generuje na wyjściu sygnał, który nie jest zgodny ze standardem VESA i w związku z tym nie jest obsługiwany przez wszystkie monitory VGA. Celem tej pracy jest stworzenie urządzenia, które dokonuje konwersji na sygnał zgodny ze standardem VESA..

Zastosowanie układu FPGA w połączeniu z szybką pamięcią DDR2 pozwala na przetwarzanie sygnału wideo w czasie rzeczywistym.

Niniejsza praca stanowi kontynuację pracy inżynierskiej z roku 2013 o temacie "Zastosowanie układu FPGA do przetwarzania sygnału wideo", w ramach której stworzono pierwszy prototyp urządzenia. Kontynuacja polega na eliminacji ograniczeń poprzedniej wersji, dodaniu obsługi wyższej rozdzielczości oraz sygnału z przeplotem.

Urządzenie opracowano z w celu zastosowania go w komputerze Amiga 500. Jednak niewielkie zmiany w projekcie mogą pozwolić na przystosowanie go do współpracy z innym komputerem pod warunkiem, że wewnątrz tego komputera dostępny jest sygnał wideo w postaci cyfrowej.

2.2. Sygnał wideo

2.2.1. Struktura czasowa sygnału wideo

Istnieją różne standardy sygnału wideo. Różnią się sposobem kodowania, sposobem przesyłania, itp. Jednak niezależnie od standardu struktura czasowa sygnału jest jednakowa [1]. To znaczy, w sygnale można wyróżnić pewne odcinki czasowe, które występują w każdym standardzie.

W sygnale wideo można wyróżnić następujące składowe: sygnał koloru oraz sygnały synchronizacji: pionowej i poziomej. W zależności o standardu, te sygnały mogą być przesyłane osobno lub razem.

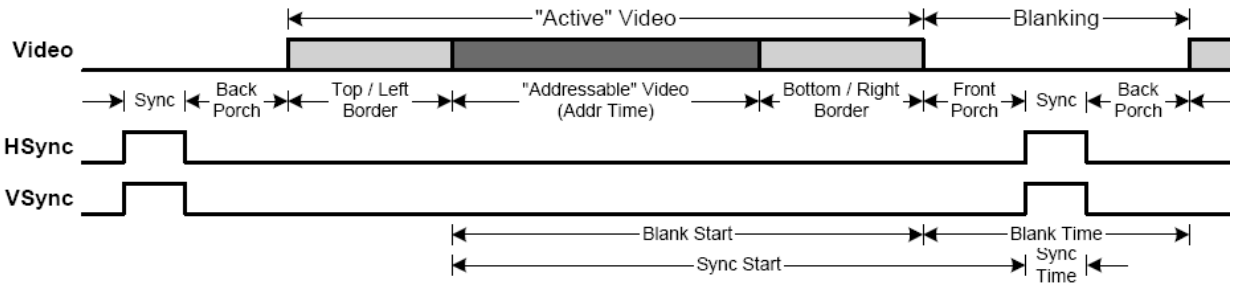
Sygnał dzieli się na tzw. ramki, (*frames*) następujące jedna po drugiej. W czasie jednej ramki przesyłana jest informacja o jednej klatce obrazu. Częstotliwość, z jaką powtarzają się ramki nazywana jest częstością odświeżania (*refresh rate*) lub częstotliwością pionową (*vertical frequency*).

Ramka obrazu z kolei dzieli się na tzw. linie (*lines*). Jedna linia to czas przeznaczony na przesłanie informacji o jednym wierszu pikseli obrazu. Liczba linii zależy od rozdzielczości obrazu. Linie zawsze wysyłane w kolejności od góry do dołu. Częstotliwość z jaką powtarzają się linie nazywana jest poziomą częstotliwością (*horizontal frequency*).

Istnieją dwie możliwe konfiguracje linii. Kolejne linie mogą reprezentować kolejne wiersze obrazu licząc od góry, wówczas mówimy o sygnale progresywnym (*progressive video*). W drugiej konfiguracji klatka obrazu jest podzielona na dwie ramki. W pierwszej ramce linie reprezentują kolejne nieparzyste wiersze obrazu, W drugiej ramce linie reprezentują kolejne parzyste wiersze obrazu licząc od góry. Wówczas mówimy o sygnale z przeplotem (*interlaced video*).

Tylko część linii zawiera informacje o wierszach obrazu. Ramka dzieli się na następujące obszary (rys. 2.2.1): obszar aktywny (*active video*) oraz obszar nieaktywny (*blanking*). Obszar aktywny składa się z tych linii, które zawierają informacje o obrazie. W obszarze aktywnym można wyróżnić obszar adresowalny (*addressable video*) oraz ramki: górną i dolną (*top/bottom border*).

Obszar adresowalny to ta część widzialnego obrazu, która zawiera rzeczywistą treść obrazu. Ramki górna i dolna nie zawierają treści, stanowią tło. W zależności od standardu w sygnale ramka może występować bądź nie występować w sygnale.



Rys. 2.2.1. Struktura sygnału wideo [2]

Linie należące do obszaru nieaktywnego nie zawierają informacji o obrazie. W czasie trwania obszaru nieaktywnego odbiornik przechodzi na początek kolejnej klatki.

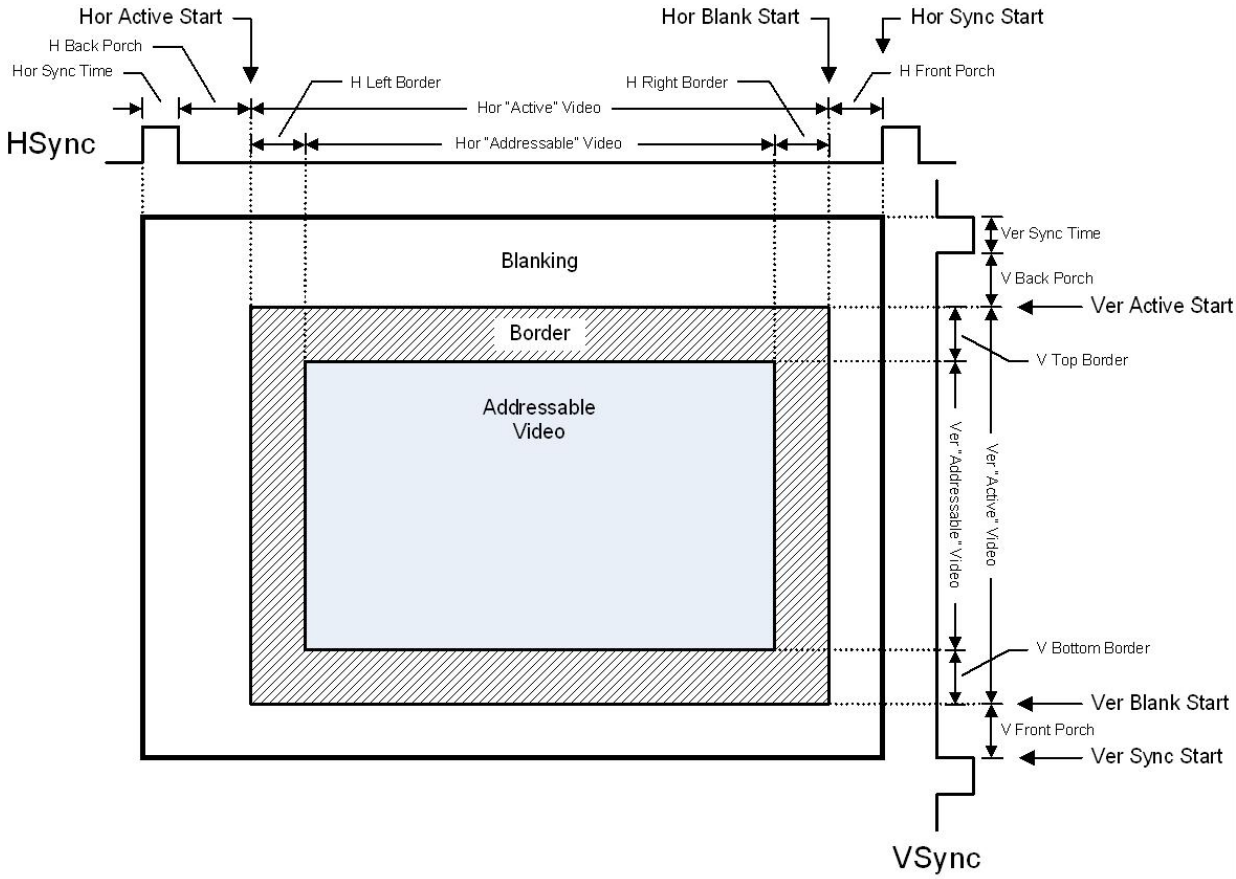
W obszarze nieaktywnym można wyróżnić obszar synchronizacji (*sync time*). W obszarze synchronizacji sygnał synchronizacji pionowej jest aktywny. Odbiornik przechodzi na początek kolejnej klatki po wykryciu sygnału synchronizacji. W zależności od standardu sygnał synchronizacji może być aktywny stanem wysokim (mówimy wtedy o polaryzacji dodatniej - *positive polarity*) lub stanem niskim (ujemna polaryzacja - *negative polarity*).

Obszar poprzedzający sygnał synchronizacyjny nazywa się *front porch*, obszar następujący po sygnale synchronizacyjnym nazywa się *back porch*.

Linia sygnału wideo dzieli się na piksele. Jeden piksel sygnału zawiera informacje o jednym pikselu obrazu. Piksele następują kolejno, licząc od lewej strony. Częstotliwość z jaką przesyłane są piksele nazywana jest *pixelclock frequency*. Nie cała linia zawiera informacje o obrazie. Linie dzielą się na obszary analogicznie do omówionego powyżej podziału ramki w pionie (rys. 2.2.1).

Podobnie jak w pionie, można wyróżnić obszar aktywny i nieaktywny. W obszarze aktywnym stan sygnału koloru w danej chwili odpowiada kolorowi aktualnie aktywnego piksela. Obszar aktywny dzieli się na obszar adresowalny i ramkę, interpretacja jest taka sama, jak powyżej.

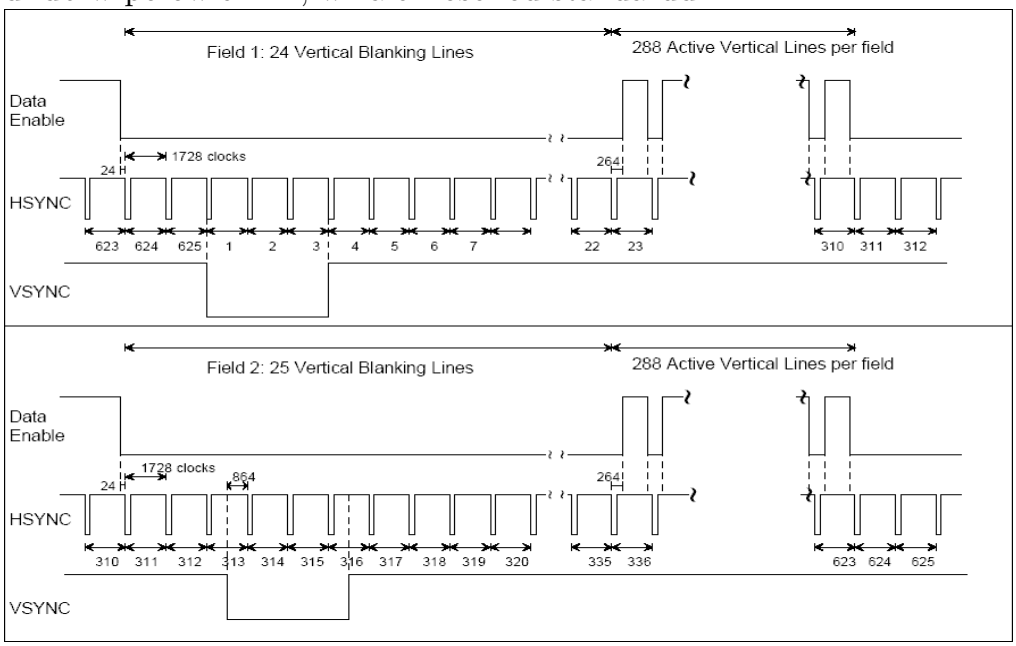
Obszar nieaktywny dzieli się na *front porch*, *sync time* i *back porch*. W obszarze synchronizacji aktywny jest sygnał synchronizacji poziomej. Podobnie jak sygnał synchronizacji poziomej może być aktywny stanem wysokim lub niskim. Po wykryciu sygnału synchronizacji poziomej odbiornik przechodzi na początek kolejnej linii. Czasową strukturę sygnału w pionie i w poziomie przedstawia rys. 2.2.2.



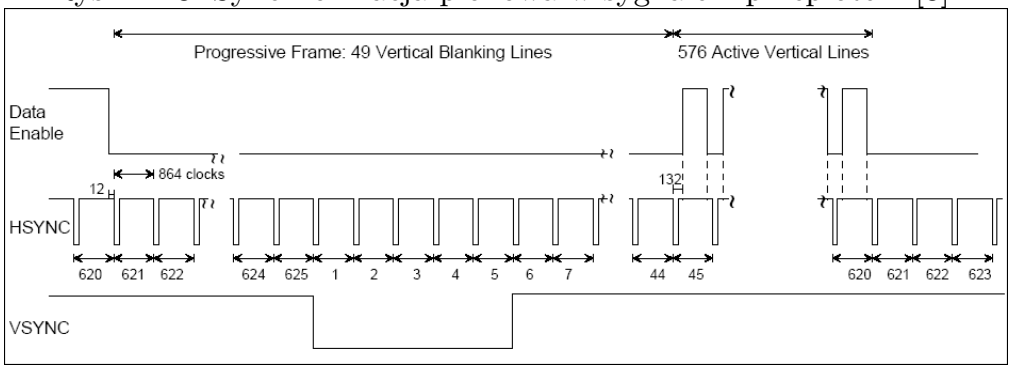
Rys. 2.2.2. Struktura sygnału wideo [2]

Sygnał progresywny musi być odróżnialny od sygnału z przeplotem. W sygnale z przeplotem ramka parzysta i nieparzysta też muszą być odróżnialne. Osiągnięto to za pomocą wzajemnego położenia sygnałów synchronizacji pionowej i poziomej.

W sygnale progresywnym sygnał synchronizacji pionowej zaczyna się na początku pierwszej linii (rys. 2.2.4). W sygnale z przeplotem sygnał synchronizacji pionowej zaczyna się na początku pierwszej linii ramki nieparzystej i w połowie pierwszej linii ramki parzystej (rys. 2.2.3). Koniec sygnału synchronizacyjnego znajduje się na początku lub w połowie linii, w zależności od standardu.



Rys. 2.2.3. Synchronizacja pionowa w sygnale z przeplotem [3]



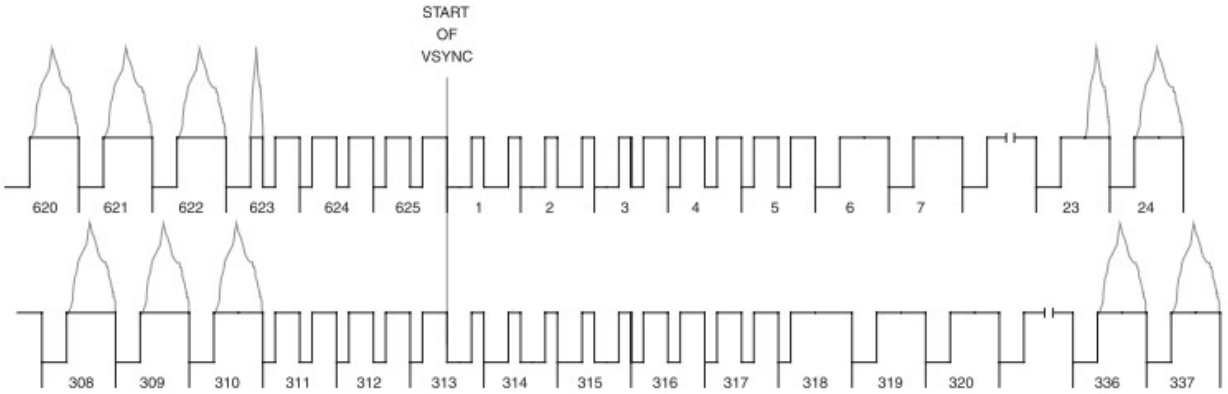
Rys. 2.2.4. Synchronizacja pionowa w sygnale progresywnym [3]

W cyfrowym sygnale wideo może jeszcze występować sygnał *data enable*, który znajduje się w stanie wysokim gdy sygnał wideo znajduje się w obszarze aktywnym.

Sygnały synchronizacji pionowej i poziomej mogą być przesyłane jedną linią. Wówczas mówimy o tzw. synchronizacji kompozytowej (*composite sync.*)

Dla większości linii obrazu sygnał synchronizacji kompozytowej jest identyczny z sygnałem synchronizacji poziomej Wyjątek stanowią linie dla których sygnał synchronizacji pionowej jest aktywny oraz kilka linii przed i za nimi (rys. 2.2.5). Liczba tych linii zależy od standardu.

Dla tych linii przed i po synchronizacji pionowej impuls synchronizacyjny jest dwa razy krótszy i występuje dwa razy: na początku i w połowie linii. Dla linii w czasie trwania synchronizacji pionowej impuls synchronizacyjny również występuje dwa razy, ale jest dużo dłuższy.



Rys. 2.2.5. Synchronizacja kompozytowa[3]

2.2.2. Standardy wideo

Istniejące standardy wideo różnią się między sobą długością odcinków czasowych omówionych w powyższym rozdziale. Można wyróżnić trzy grupy standardów.

Pierwszą grupę stanowią sygnały wideo w standardzie tzw. telewizyjnym: 576i oraz 480i [1], często niepoprawnie nazywane standardem PAL lub NTSC (standardy PAL i NTSC określają także sposób kodowania koloru i dotyczą tylko sygnału kompozytowego). Są to standardy używane w telewizji analogowej.

Tabela 1.2.1 przedstawia parametry sygnałów w standardach 576i i 480i. W tabeli użyto angielskich nazw.

	576i	480i
video mode	interlaced	interlaced
vertical front porch	2,5 (2) lines	3,5 (3) lines
vertical sync time	2,5 lines	3 lines
vertical back porch	19,5 (20) lines	16 (16,5) lines
vertical active region	288 lines	240 lines
vertical total time	312,5 lines	262,5
refresh rate	50 Hz	60 Hz
horizontal front porch	~2 µs	~4,7 µs
horizontal sync time	~4 µs	~4,7 µs
horizontal back porch	~5 µs	~1,5 µs
horizontal active region	~53 µs	~52,6 µs
horizontal total time	64 µs	~63,5 µs
horizontal frequency	15,626 kHz	15,750kHz
pixelclock	undefined	undefined

Tabela 2.2.1. Parametry sygnałów 576i i 480i [1]

Tam, gdzie wartość jest różna dla ramki parzystej i nieparzystej podano w nawiasie wartość dla ramki parzystej, a bez nawiasu dla nieparzystej.

Pozioma rozdzielczość nie jest zdefiniowana i zależy od urządzenia nadającego sygnał. Standardy definiują wyłącznie sygnał wideo z przeplotem. Urządzenia nieobsługujące przeplotu generują obraz w którym treść ramki parzystej i nieparzystej jest jednakowa. Skutkuje to zmniejszoną o połowę rozdzielczością pionową. Niektóre urządzenia generują sygnał składający się wyłącznie z ramek nieparzystych. Nie jest to zgodne ze standardem.

Istnienie dwóch standardów o częstościach odświeżania 50 i 60Hz wynika z faktu, że dawniej urządzenia nadające sygnał wideo były synchronizowane częstotliwością sieci energetycznej, która w zależności od państwa wynosi 60 lub 50Hz.

Drugą grupę stanowią tryby wideo zdefiniowane w standardzie VESA [2]. VESA definiuje tryby wideo przeznaczone dla monitorów komputerowych. Obecnie produkowane komputery i monitory wspierają ten standard.

W tabeli 2.2.2 przedstawiono wybrane tryby wideo standardu VESA. W tabeli użyto angielskich nazw.

Pixel Format	Refresh Rate	Horizontal Frequency	Pixel Frequency
640 x 400	85 Hz	37.9 kHz	31.500 MHz
720 x 400	85 Hz	37.9 kHz	35.500 MHz
640 x 480	60 Hz	31.5 kHz	25.175 MHz
	72 Hz	37.9 kHz	31.500 MHz
	75 Hz	37.5 kHz	31.500 MHz
800 x 600	60 Hz	37.9 kHz	40.000 MHz
	72 Hz	48.1 kHz	50.000 MHz
	75 Hz	46.9 kHz	49.500 MHz
1024 x 768	60 Hz	48.4 kHz	65.000 MHz
	70 Hz	56.5 kHz	75.000 MHz
	75 Hz	60.0 kHz	78.750 MHz
1280 x 1024	60 Hz	64.0 kHz	108.000 MHz
	75 Hz	80.0 kHz	135.000 MHz
	85 Hz	91.1 kHz	157.500 MHz

Tabela 2.2.2. Wybrane tryby wideo wg. standardu VESA [2]

Kolejną grupę stanowią tryby wideo zdefiniowane w standardzie CEA-861 [3], CEA-861 definiuje tryby wideo przeznaczone dla telewizji cyfrowej i sprzętu RTV (telewizory, odtwarzacze wideo, itp.). Obecnie produkowany sprzęt RTV wspiera ten standard.

W tabeli 2.2.3 przedstawiono wybrane tryby wideo w standardzie CEA-861. W tabeli użyto angielskich nazw. Można zauważyć, że standardzie CEA zdefiniowano między innymi tryby zgodne ze standardami 576i i 480i ,przy czym określono rozdzielczość poziomą, która nie była w nich zdefiniowana

									(kHz)	(Hz)	(MHz)
	Code	H active	V active	I/P_	H total	H blank[5]	V total	V blank[5]	H Freq[5]	V Freq[4]	Pixel Freq[5]
50Hz	17,18	720	576	Prog	864	144	625	49	31.250	50.000	27.000
	19	1280	720	Prog	1980	700	750	30	37.500	50.000	74.250
	20	1920	1080	Int	2640	720	1125	22.5[1]	28.125	50.000	74.250
	21,22	1440[2]	576	Int	1728[2]	288	625	24.5[1]	15.625	50.000	27.000
	23,24	1440[2]	288	Prog	1728[2]	288	312	24	15.625	50.080	27.000
	23,24	1440[2]	288	Prog	1728[2]	288	313	25	15.625	49.920	27.000
	23,24	1440[2]	288	Prog	1728[2]	288	314	26	15.625	49.761	27.000
	29,30	1440[2]	576	Prog	1728[2]	288	625	49	31.250	50.000	54.000
	31	1920	1080	Prog	2640	720	1125	45	56.250	50.000	148.500
	39	1920	1080	Int	2304	384	1250	85	31.250	50.000	72.000
60Hz[3]	1	640	480	Prog	800	160	525	45	31.500	60.000	25.200
	2,3	720	480	Prog	858	138	525	45	31.500	60.000	27.027
	4	1280	720	Prog	1650	370	750	30	45.000	60.000	74.250
	5	1920	1080	Int	2200	280	1125	22.5[1]	33.750	60.000	74.250
	6,7	1440[2]	480	Int	1716[2]	276	525	22.5[1]	15.750	60.000	27.027
	8,9	1440[2]	240	Prog	1716[2]	276	262	22	15.750	60.115	27.027
	8,9	1440[2]	240	Prog	1716[2]	276	263	23	15.750	59.886	27.027
	14,15	1440[2]	480	Prog	1716[2]	276	525	45	31.500	60.000	54.054
	16	1920	1080	Prog	2200	280	1125	45	67.500	60.000	148.500
	35,36	2880	480	Prog	3432	552	525	45	31.500	60.000	108.108
100 Hz	40	1920	1080	Int	2640	720	1125	22.5[1]	56.250	100.00	148.500
	41	1280	720	Prog	1980	700	750	30	75.000	100.00	148.500
	42, 43	720	576	Prog	864	144	625	49	62.500	100.00	54.000
	44, 45	1440[2]	576	Int	1728[2]	288	625	24.5[1]	31.250	100.00	54.000

Tabela 2.2.3 - Wybrane tryby wideo wg. standardu CEA-861 [3]

2.2.3. Problemy ze zgodnością standardów

Zarówno współczesne komputery jak i monitory wspierają tryby wideo zgodne ze standardem VESA. Jednak komputery starszej generacji generują sygnał niezgodny z tym standardem. Wynika to z dwóch powodów. Pierwszym z nich jest fakt, że pierwsza wersja standardu VESA powstała w roku 1994 [2].

Drugą przyczyną jest fakt, że komputery projektowano tak, aby można było z nich korzystać po podłączeniu do zwykłego odbiornika telewizyjnego, co pozwalało na zakup samego komputera bez monitora.

Komputery te generują sygnał wideo z przeplotem w standardzie 576i lub 480i, albo w postaci zmodyfikowanej, bez przeplotu (288p, 240p). Jednym z takich komputerów jest użyty w niniejszej pracy komputer Commodore Amiga 500. Wybrane komputery pracujące w standardzie telewizyjnym to m.in.[4]:

- wszystkie modele Commodore Amiga,
- Commodore 64, 128,
- 8- i 16 bitowe modele Atari,
- komputery PC z kartą CGA,
- komputery Apple I, II.

Współczesne monitory zgodne ze standardem VESA nie są w stanie prawidłowo odebrać sygnału wideo w standardzie telewizyjnym. Częstotliwość pozioma w standardzie telewizyjnym wynosi około 15 kHz lub i jest około dwóch razy mniejsza od najmniejszej częstotliwości poziomej zdefiniowanej w standardzie VESA

(31,5 kHz), a zatem znajduje się poza zakresem wspieranych częstotliwości. Oznacza to, że monitor fizycznie nie jest w stanie dostroić się do tego sygnału.

Część monitorów wspiera również tryby zdefiniowane w standardzie CEA-861, jednak zwykle dotyczy to trybów o częstotliwości poziomej powyżej 31 kHz i bez przeplotu.

Wynika z tego, że nie można używać współczesnego monitora do odbioru sygnału pochodzącego z tych komputerów.

2.3. Istniejące rozwiązania

Problem niezgodności standardów można rozwiązać na kilka sposobów. Pierwszym, najbardziej oczywistym sposobem jest korzystanie z monitorów wspierających dany standard wideo. Nie jest to jednak idealne rozwiązanie.

W praktyce oznacza to korzystanie ze starszych modeli monitorów. Ponieważ nie są one już produkowane, są one coraz trudniej dostępne. Ponadto są to monitory kineskopowe, które zajmują więcej miejsca i są bardziej niekorzystne dla zdrowia w porównaniu z innymi typami monitorów. Dodatkowo, w przypadku posiadania komputerów w różnych standardach konieczne jest korzystanie z dwóch różnych monitorów, co oznacza więcej miejsca zajętego przez monitory.

Alternatywnym rozwiązaniem jest korzystanie z monitorów typu multisync. Są to monitory, które potrafią się dostroić zarówno do częstotliwości ~15 kHz jak i ~31 kHz. Obecnie, z powodu bardzo niskiego zapotrzebowania na takie monitory, są one trudno dostępne na rynku i kosztują więcej niż zwykłe monitory.

Niektóre współczesne telewizory wspierają sygnał wideo w obu standardach.

2.3.1. Urządzenia typu scandoubler

Istnieją specjalne urządzenia tzw. *scandoublery*, które konwertują sygnał wideo. [5] Scandoubler jest urządzeniem, które zamienia sygnał w standardzie 288p lub 240p (czyli 576i i 480i bez przeplotu) na sygnał o częstotliwości poziomej około 31 kHz . Urządzenie które potrafi również zamienić sygnał z przeplotem (576i, 480i) na sygnał o częstotliwości około 31 kHz bez przeplotu nazywane jest *flicker-fixer*.

Zwiększanie częstotliwości poziomej dokonuje się przez powielenie linii. Każda linia sygnału wejściowego jest kopiowana i podawana w sygnale wyjściowym dwukrotnie. Każda kopia zajmuje czas dwa razy krótszy niż oryginalna linia. Stąd pochodzi nazwa *scandoubler* (ang. "podwajacz linii obrazu").

Otrzymuje się w ten sposób sygnał wyjściowy o wysokości 576 linii i częstotliwości poziomej 31,25 kHz i pionowej 50 Hz lub sygnał o wysokości 480 linii i częstotliwości poziomej 31,5 kHz i pionowej 60 Hz. Taki sygnał jest zgodny ze standardem CEA, ale nie jest zgodny ze standardem VESA. Jest duże prawdopodobieństwo, że sygnał będzie wspierany przez monitor VGA, ponieważ częstotliwość pozioma mieści się we wspieranym zakresie częstotliwości, ale nie jest to pewne ponieważ monitor nie musi obsługiwać trybu wideo niezgodnego ze standardem.

Taki stan rzeczy wynika z faktu, że konwersja do formatu zgodnego ze standardem VESA wymaga niesynchroniczności sygnału wejściowego i wyjściowego. To prowadzi do zwiększenia złożoności urządzenia, a co za tym idzie, zwiększenia jego kosztu. Urządzenie generujące sygnał niesynchroniczny z sygnałem wejściowym nie jest scandoublerem w ścisłym znaczeniu tego słowa, ponieważ jego funkcjonalność nie polega wyłącznie na podwajaniu linii obrazu.

Scandoubler może pobierać sygnał wideo na jeden z dwóch sposobów. Pierwszym z nich jest podłączenie go do analogowego portu wideo źródła sygnału. W takim rozwiązaniu konieczna jest konwersja A/C, próbkowanie sygnału oraz odtwarzanie zegara *pixelclock*. Przykładem urządzeń działających w ten sposób są [5]:

- Adavanced Micro Interfaces amiVGA,
- Eyetech EZ-VGA,
- Unitech Electronics Video Crisper,
- Elbox Scandoubler FF.

Ponieważ komputery, których dotyczy problem niezgodności standardów charakteryzują się mniejszą od współczesnych skalą integracji, sygnał nieskonwertowany do postaci analogowej może być dostępny na płycie głównej komputera. Można zatem podłączyć wejście urządzenia bezpośrednio do sygnału cyfrowego wewnątrz komputera. Przykładem urządzeń działających w ten sposób są:

- Individual Computers Indivision ECS,
- DCE Scan Magic,
- Micronix Scandy1200,
- ICD Flicker Free Video2.

2.3.2. FPGA jako układ przetwarzający sygnał

Układy FFGA są wysoce konfigurowalnymi układami logicznymi. można w nich zaimplementować funkcjonalność dowolnego układu cyfrowego, która zmieści się w zasobach, jakimi dysponuje dany układ FPGA. Z tego powodu nadają się do zastosowania w przetwarzaniu sygnału wideo.

Przetwarzanie sygnału w czasie rzeczywistym wymaga szybkiego kopiowania dużej ilości danych. Niesie to z sobą specyficzne wymagania, co do struktury układu. Układ FPGA może zostać zaprogramowany tak, aby był jak najlepiej dostosowany do wymagań konkretnej aplikacji.

Niektóre ze scandoublerów dostępnych na rynku zbudowane są w oparciu właśnie o układ FPGA. Na przykład w ICD Flicker Free Video został wykorzystany układ Xilinx XC320-100, a w Individual Computers Indivision ECS wykorzystano układ Altera EP1C3T100C8N [5]. W niniejszej pracy również wykorzystano układ FPGA do realizacji projektu.

3. Platforma sprzętowa i programistyczna

3.1. Platforma sprzętowa

3.1.1. Komputer Commodore Amiga 500.

Jako źródło sygnału wybrano komputer Commodore Amiga 500 z płytą główną w wersji 8A.1. Jest to wersja płyty głównej stosowana standardowo w komputerach Amiga 500+. Wybrany egzemplarz jest wyposażony w chipset ECS, również stosowany w komputerach Amiga 500+.

Rys. 3.1.1. Komputer Amiga 500

Amiga 500 jest komputerem opartym o 16/32 bitowy procesor Motorola 68000 pracujący z częstotliwością 7,09375MHz [7]. Komputer jest wyposażony w specjalizowane układy scalone tworzące tzw. chipset. W zależności od wersji komputer jest wyposażony w chipset w wersji OCS (*original chipset*) lub ECS (*enhanced chipset*). Układy stanowiące chipset są odpowiedzialne za generację sygnału audio, wideo i obsługę peryferiów. Układy te pracują niezależnie od procesora.

W zależności od wersji chipsetu komputer obsługuje do 2MB pamięci *chip ram* dostępnej dla procesora i chipsetu oraz do 8MB pamięci *fast ram* dostępnej tylko dla procesora. Komputer jest w stanie obsłużyć do czterech stacji dyskietek 3,5". W przypadku zastosowania zewnętrznego kontrolera może również obsłużyć dysk twardy o wielkości do 4GB. Komputer posiada jeden port równoległy, jeden port szeregowy, dwa porty dla urządzeń wejściowych (mysz, joystick), jeden port dla zewnętrznych stacji dyskietek, porty wideo: kompozytowy oraz RGB, dwukanałowy port audio, oraz złącze rozszerzeń służąca do podłączania dodatkowych urządzeń. Standardowo komputer jest wyposażony w jedną stację dyskietek 3,5" i klawiaturę.

Komputer generuje 12 bitowy sygnał wideo o rozdzielczości pionowej wynoszącej w zależności od standardu 200 lub 256 linii bez przeplotu albo 400 lub 512 linii z przeplotem. Są trzy rozdzielczości poziome, *LORES* - 320 pikseli, *HIRES* - 640

pikseli oraz *SUPER HIRES* (*SHRES*) - 1280 pikseli. Rozdzielczość SHRES jest dostępna tylko dla komputerów z chipsetem ECS.

Tabela 3.1.1 pokazuje parametry generowanego sygnału w różnych trybach wideo.

Standard napięć:	TTL 5V					
polaryzacja sygnałów synchronizujących:	ujemna					
Liczba bitów:	12					
Częstotliwość odświeżania pionowego [Hz]:	50			60		
Częstotliwość odświeżania poziomego [kHz]	15,625			15,646		
Częstotliwość pixelclock [MHz]	7,09375	14,1875	28,375	7,159	14,318	28,636
Szerokość obrazu adresowalnego [piksele]	320	640	1280	320	640	1280
Szerokość obrazu widzialnego [piksele]	376	752	1504	376	752	1504
Całkowita szerokość obrazu [piksele]	454	908	1816	454	908	1816
Wysokość obrazu adresowalnego [linie]	256 (512)			200 (400)		
Wysokość obrazu widzialnego [linie]	286 (572)			240 (480)		
Całkowita wysokość obrazu [linie]	312 (625)			262 (525)		
Czas trwania jednej linii [us]	64			63,416		
Czas trwania jednego piksela [ns]	140,97	70,485	35,242	139,68	69,841	34,920

Tabela 3.1.1. Parametry sygnału wideo komputera Amiga 500 [6, 7]

Część danych przedstawionych w tabeli pochodzi z własnych pomiarów. W nawiasach przedstawiono dane dla sygnału z przeplotem.

W komputerze są dostępne dwa wyjścia wideo - analogowe kompozytowe i analogowe RGB. Sygnał cyfrowy RGB jest dostępny na płycie głównej. Tabela 3.1.2 przedstawia linie wchodzące w skład sygnału wideo (nazewnictwo wg. [8]). Sygnały taktujące 14,18758 i 28,375MHz nie są dostępne na płycie głównej i muszą być odtworzone.

Zdecydowano się na wybór tego komputera ze względu na dostęp do dwóch egzemplarzy tego komputera, dostęp do dokumentacji i schematów, stosunkowo łatwo dostępne sygnały na płycie głównej oraz wcześniejsze doświadczenie w pracy z tym komputerem, zarówno w ramach pracy inżynierskiej jak i przedtem.

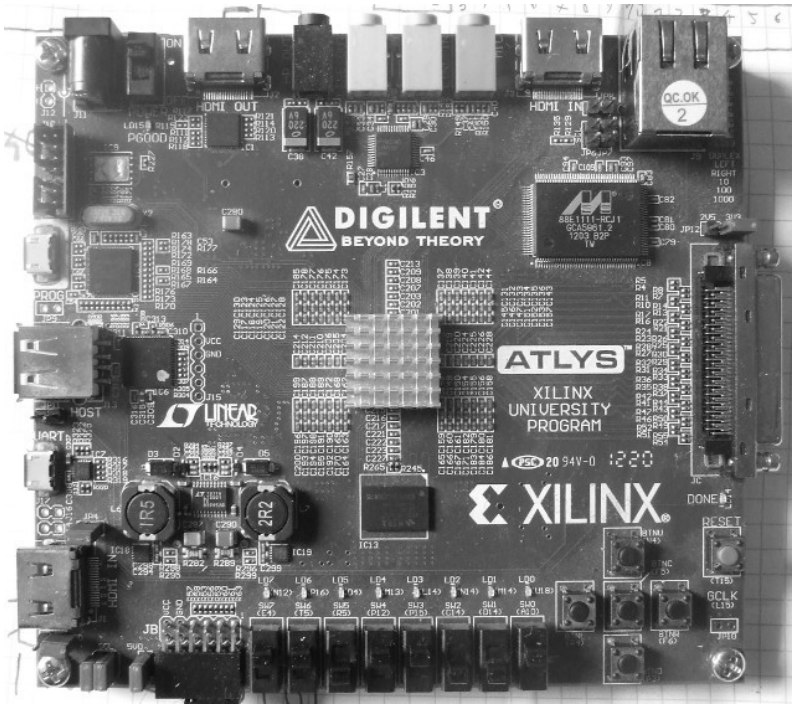
nazwa sygnału	opis
7M	sygnał zegarowy 7,09379 MHz
CDAC	sygnał 7M opóźniony o 1/4 okresu
HSYNC	sygnał synchronizacji poziomej
VSYNC	sygnał synchronizacji pionowej
CSYNC	sygnał synchronizacji kompozytowej
R[3:0], G[3:0], B[3:0]	bity koloru

Tabela 3.1.2. Linie sygnału wideo [8]

3.1.2. Moduł FPGA

Projekt zrealizowano na module Atlys firmy Digilent. Moduł Atlys jest wyposażony w następujące elementy [9]:

- Xilinx Spartan-6 LX45 FPGA w obudowie 324pin BGA,
- pamięć DDR2 - 128 MB,
- pamięć Flash 16MB x4 SPI,
- porty HDMI - 2 wejściowe i dwa wyjściowe,
- 8 diód LED, 6 przycisków, 8 przełączników,
- port USB do programowania i przesyłu danych,
- USB-UART i port USB-HID (dla myszy/klawiatury),
- oscylator CMOS 100 MHz,
- port ethernet 10/100/1000 kbit/s,
- kodek AC-97z wejściami i wyjściami audio,
- 48 linii I/O,



Rys. 3.1.1. Moduł Atlys

3.1.3. Pamięć DDR2

Moduł Nexys3 jest wyposażony w pamięć MIRA P3R1GE3EGF G8E [11]. Jest to pamięć typu DDR2 zgodna ze standardem JEDEC [10]. Tabela 3.1.3 przedstawia najważniejsze parametry pamięci. W tabeli 3.1.4 opisano sygnały pamięci.

pojemność	128 MB
szerokość słowa	16 bitów
liczba banków	8
rozmiar strony	2kB
długość *burst*'a	4, 8
maksymalna częstotliowość taktowania	400 MHz
standard napięć	SSTL_18

Tabela 3.1.3. Najważniejsze parametry pamięci [11]

nazwa	**kierunek**	**opis**
A[13:0]	wejście	linie adresu
BA[2:0]		wybór banku
CK, /CK		zegar różnicowy
DQS, /DQS	dwukierunkowe	sygnały wyzwalania
LDQS, /LSQS		
UDQS, /UDQS		
RDQS, /RDQS		
/WE	wejście	sygnały sterujące
/CAS		
/RAS		
CKE		
DM, UDM, LDM		maska danych
DQ[15:0]	dwukierunkowe	linie danych

Tabela 3.1.4. Sygnały pamięci [11]

3.1.4. Układ FPGA

Układem FPGA zastosowanym w module jest Xilinx Spartan-6 LX45 w obudowie CSG324C. Tabela 3.1.5 zawiera opis najważniejszych parametrów układu. W tabeli użyto angielskich nazw.

Logic Cells	43 661
Slices	6 822
Flip-Flops	54 576
Max Distributed RAM (kb)	401
DSP48A1 Slices	58
Block RAM (kB)	2088
CMTs	4
Memory Controller Blocks	2
Total I/O Banks	4
Max User I/O	358
Max Clock frequency	400MHz

Tabela 3.1.5. Najważniejsze parametry układu Spartan-6 LX45 [12]

3.1.5. Pamięć BlockRam

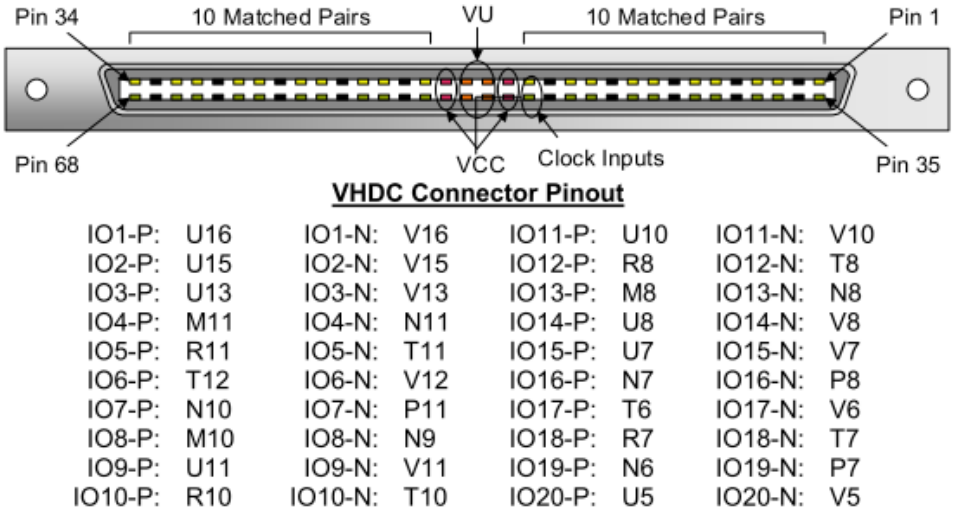
Układ Spartan-6 LX16 jest wyposażony w 116 bloków pamięci o pojemności 18 kB co daje całkowitą pojemność 2088 kB. Każdy blok składa się z dwóch części o pojemności 9kB, które mogą być skonfigurowane do pracy jako dwie niezależne pamięci [13]. Szerokość szyny danych jest konfigurowalna i może wynosić 1, 2, 4, 8, 9, 16, 18, 32 lub 36 bitów. Dla szerokości 9, 18 i 36 bitów pojemność bloku wynosi 9kB, dla pozostałych - 8kB.

Pamięć może pracować trybie jednoportowym lub jednym z dwóch trybów dwuportowych. W trybie dwuportowym możliwy jest jednoczesny dostęp do pamięci na obu portach, przy czym porty mogą być taktowane różnymi sygnałami zegarowymi i mieć różne szerokości szyny danych. Są dwa tryby dwuportowe: Simple Dual-port oraz True Dual-port. W trybie Simple Dual-Port jeden port służy do zapisu, drugi do odczytu. W trybie True Dual-port oba porty mają możliwość odczytu i zapisu.

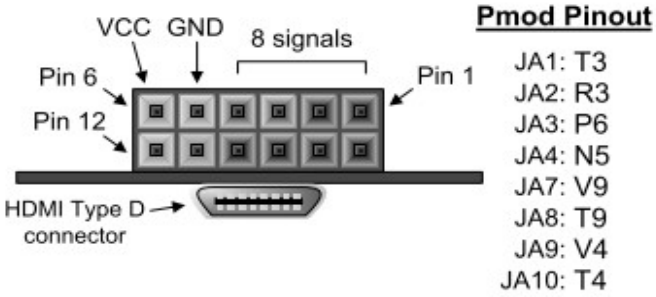
3.1.6. Porty IO

Na module Atlys jest dostępnych 48 linii IO: 40 na złączu VHDC, 8 na złączu Pmod [9]. W niniejszej pracy korzystano ze złącza VHDC z dołączonym adapterem VmodMIB, na którym jest dostępnych 5 portów Vmod, identycznych jak port Pmod na module Atlys. W projekcie wykorzystano 4 porty Vmod i 1 port Pmod.

Port Pmod jest złączem o 12 pinach. 8 z nich to linie IO, 2 to napięcie zasilania 3,3V, 2 to masa. Linie IO pracują w stan-dardzie napięć 3,3V. Rys. 3.1.2 przedstawia rozkład pinów na złączu VHDC oraz ich przyporządkowanie pinom układu FPGA. Rys. 3.1.3 przedstawia rozkład pinów na złączu Pmod oraz ich przyporządkowanie pinom układu FPGA.



Rys. 3.1.2. Rozkład pinów na złączu VHDC [9]



Rys. 3.1.3. Rozkład pinów na złączach Pmod [9]

3.1.7. Odbiornik sygnału wideo

Generowany sygnał wideo był testowany na dwóch standardowym monitorze VGA Fujitsu Siemens B17-2. Sygnał analogowy z komputera Amiga 500 był podłączony do monitora Phillips CM8833.

3.2. Platforma programistyczna

Środowisko programistyczne, wykorzystane w projekcie to Xilinx ISE Design Suite w wersji 14.2 na darmowej licencji WebPack. Zdecydowano się na wybór tego środowiska, ponieważ producent oprogramowania jest jednocześnie producentem układu FPGA, co zapewnia pełne wsparcie tego układu przez środowisko. Program był pisany w języku VHDL.

Do wgrywania plików wynikowych do modułu Atlys korzystano z programu Digilent Adept w wersji 2.4.2. Jest to program stworzony przez producenta modułu Atlys i jest przeznaczony do współpracy z modułami tego producenta.

4. Budowa i działanie układu

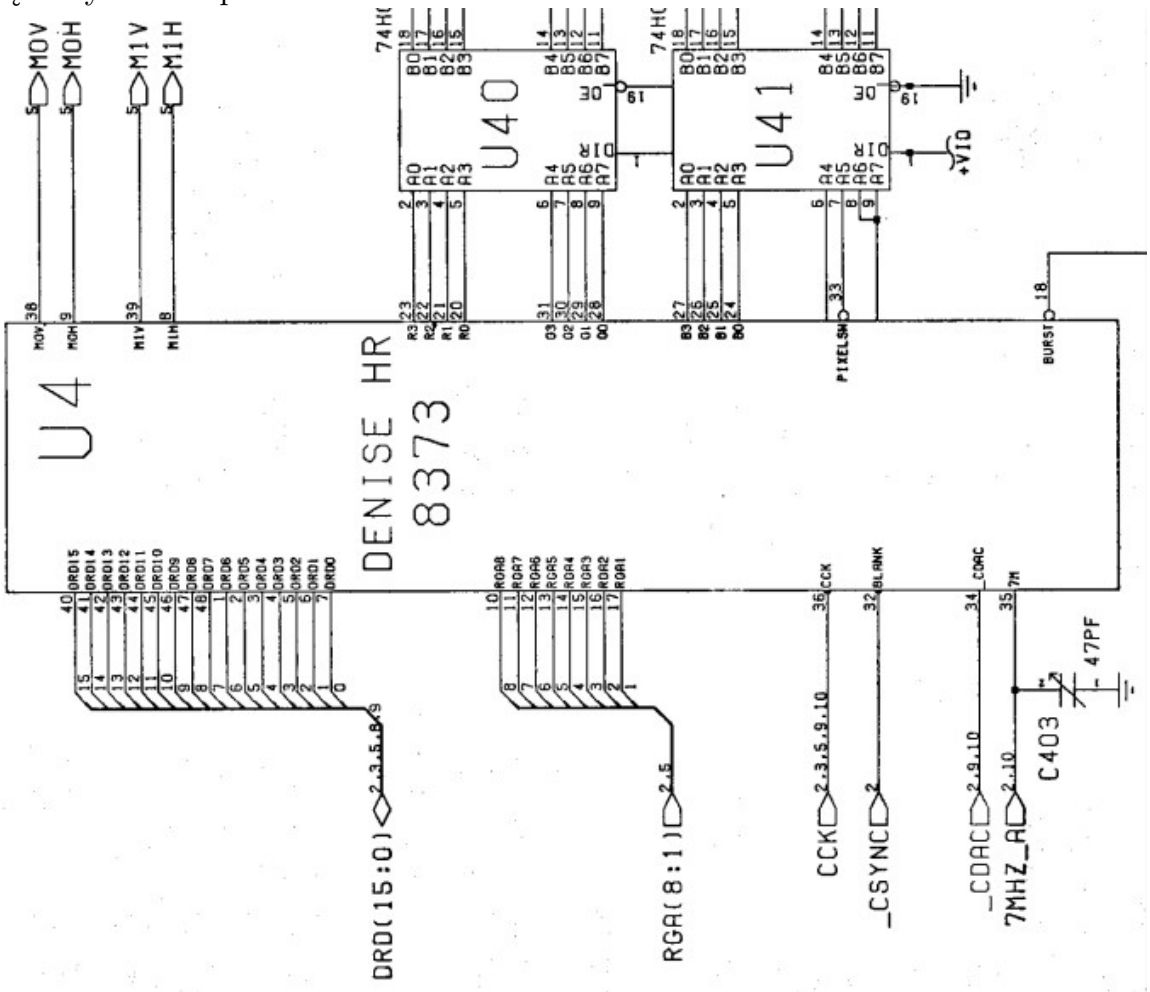
4.1. Część sprzętowa

4.1.1. Pozyskanie sygnału wideo

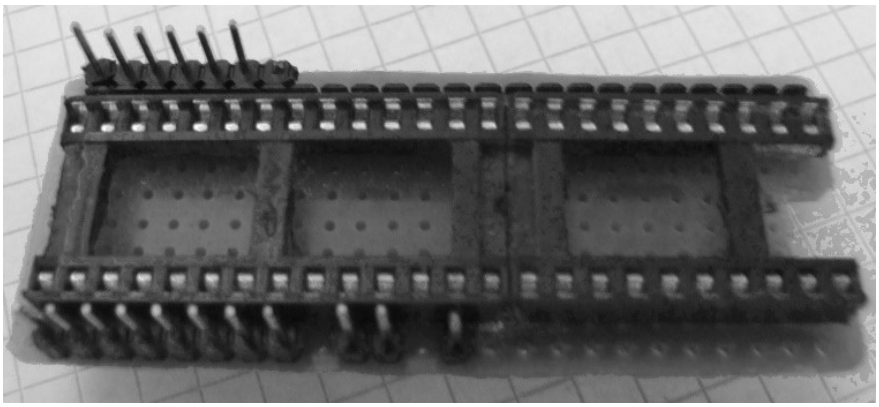
Sygnał wideo w postaci cyfrowej składa się z linii wymienionych w tabeli 3.1.2. Jest on generowany przez układ U4 (8373 Denise). [7] Sygnały potrzebne w niniejszej pracy są dostępne na płycie głównej, w pobliżu tego układu.

W przeciwieństwie do poprzedniej wersji urządzenia realizowanej w ramach pracy inżynierskiej, gdzie sygnał wideo pobrano poprzez przylutowanie przewodów w odpowiednie miejsca na płycie głównej, w tej wersji przyjęto założenie, że urządzenie nie powinno ingerować w płytę główną komputera. Zamiast tego przyjęto rozwiązanie polegające na pobraniu potrzebnych sygnałów przez wykonanie odpowiedniego adaptera i umieszczenie go w podstawce układu U4.

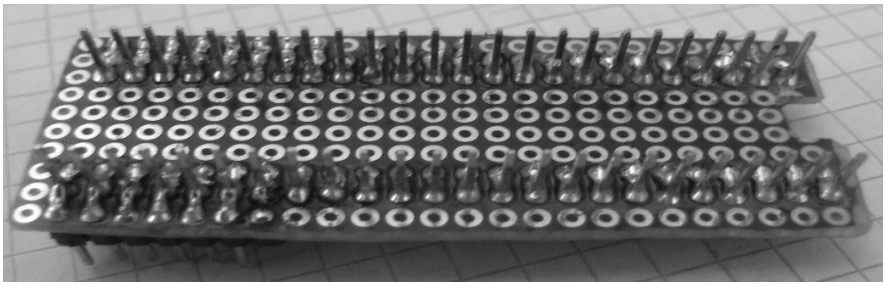
Na rys. 4.1.1 przedstawiono fragment schematu płyty głównej przedstawiający rozmieszczenie sygnałów na pinach układu U4. Rys. 4.1.2 i 4.1.3 przedstawiają widok od góry i dołu wykonanego adaptera. Rys. 4.1.4 przedstawia adapter podłączony do komputera.



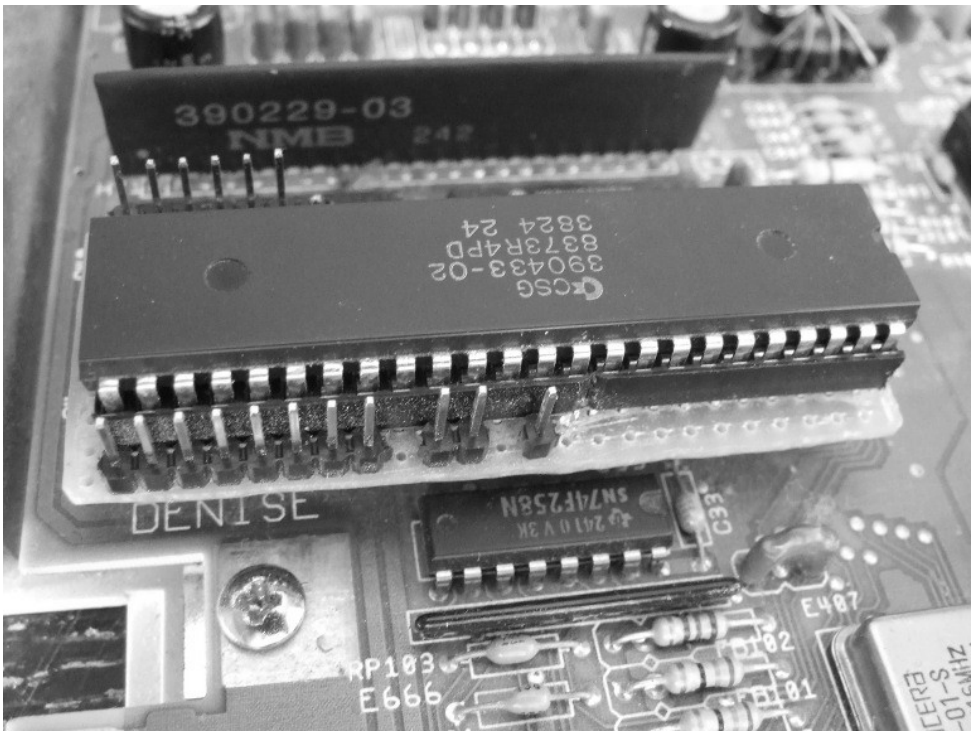
Rys. 4.1.1. Rozmieszczenie sygnałów na pinach układu U4 [7]



Rys. 4.1.2. Adapter, widok od góry



Rys. 4.1.3. Adapter, widok od dołu

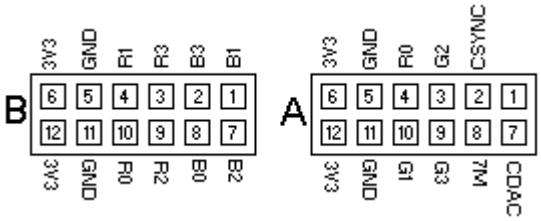


Rys. 4.1.4. Adapter podłączony do komputera

Na pinach układu U4 nie są dostępne sygnały HSYNC i VSYNC, co oznacza, że do przetwarzania sygnału wideo trzeba korzystać z sygnału CSYNC, co jest trudniejsze, niż przy użyciu sygnałów HSYNC i CSYNC.

4.1.2. Interfejs wejściowy

Na interfejs wejściowy wykorzystano 2 porty Vmod adaptera VmodMIB: A i B. Rozmieszczenie sygnałów wejściowych na portach Vmod przedstawia rys. 4.1.6. Sygnał zegarowy musi być umieszczony na pinie do tego przystosowanym, pozostałe sygnały można było rozmieścić dowolnie. Sygnały pobrane z adaptera należy podłączyć do odpowiednich wejść.



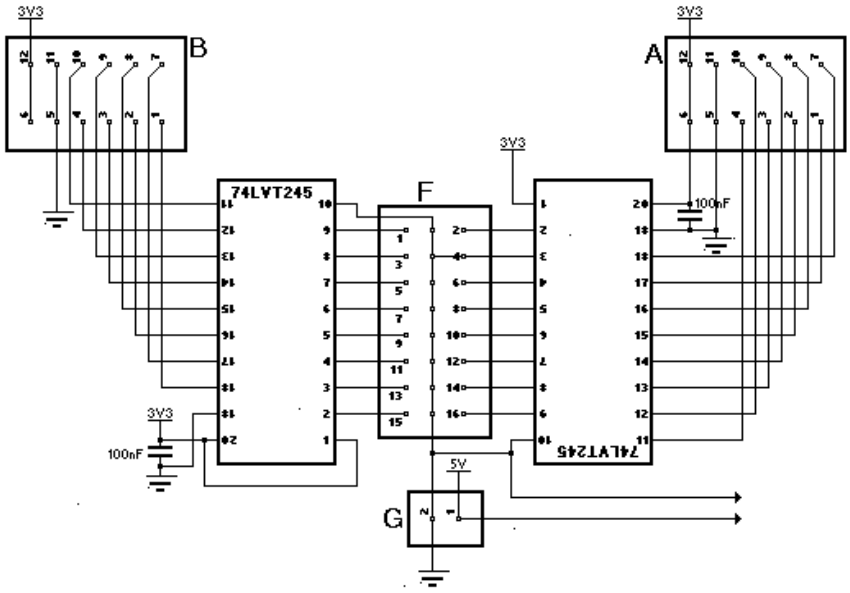
Rys. 4.1.6. Rozmieszczenie sygnałów wejściowych

Ponieważ sygnał wideo jest w standardzie napięć 5V, a porty Pmod akceptują sygnały w standardzie 3,3V konieczna jest konwersja poziomów napięć. Konwersji można dokonać an dwa sposoby: za pomocą rezystorowego dzielnika napięcia lub za pomocą układu scalonego.

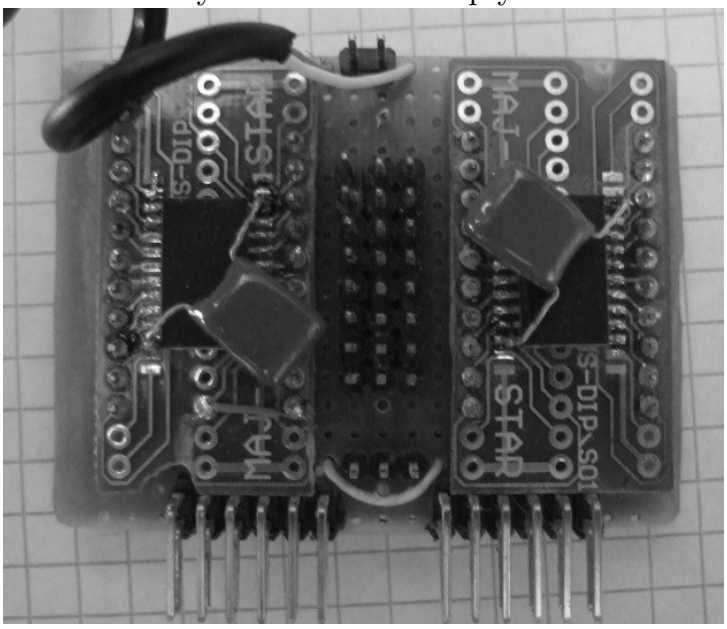
W czasie realizacji poprzedniej wersji urządzenia okazało się, że rezystorowy dzielnik napięć nie jest dobrym rozwiązaniem, ponieważ sygnał wyjściowy dzielnika narastał i opadał zbyt wolno, co powodowało gubienie cykli zegarowych, co z kolei prowadziło do zniekształceń obrazu. Wobec tego w tej wersji urządzenia zastosowano bramki 74LVT245. Układ 74LVT245 pracuje w standardzie 3,3V, i akceptuje poziomy TTL 5V na wejściu.

Poprzednia wersja urządzenia umożliwiała odtwarzanie sygnału zegarowego ~14 MHz przez wykonanie operacji XOR na sygnałach 7M i CDAC. za pomocą bramki logicznej 74LS86. W tej zdecydowano się na odtwarzanie sygnału zegarowego wewnątrz FPGA, między innymi dlatego, że projekt zakładał również odtworzenie sygnału ~128 MHz, czego nie da się zrobić przy pomocy zewnętrznych bramek logicznych.

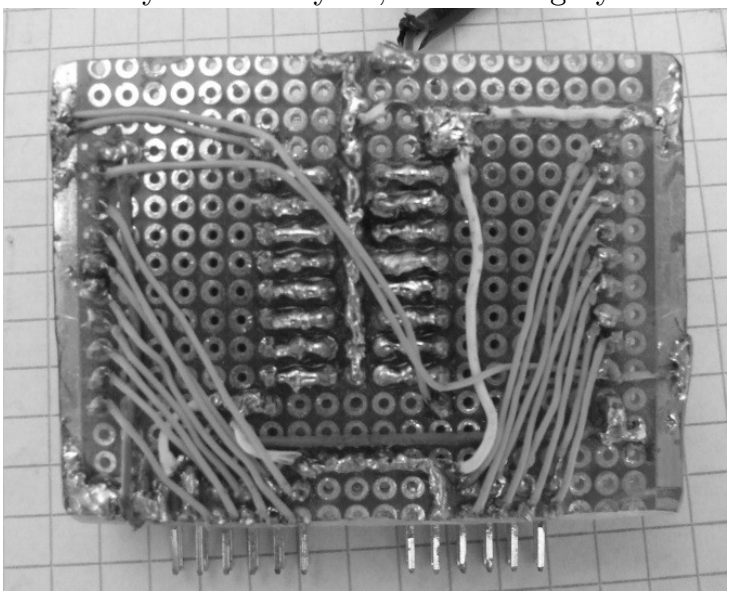
Interfejs wejściowy zrealizowano na uniwersalnej płytce drukowanej. Schemat przedstawiono na rys. 4.1.7. Rys. 4.1.8 i 4.1.9 przedstawiają widok płytki od góry i od dołu. Rys. 4.1.10 przedstawia płytkę podłączoną do sygnału wideo z układu U4.



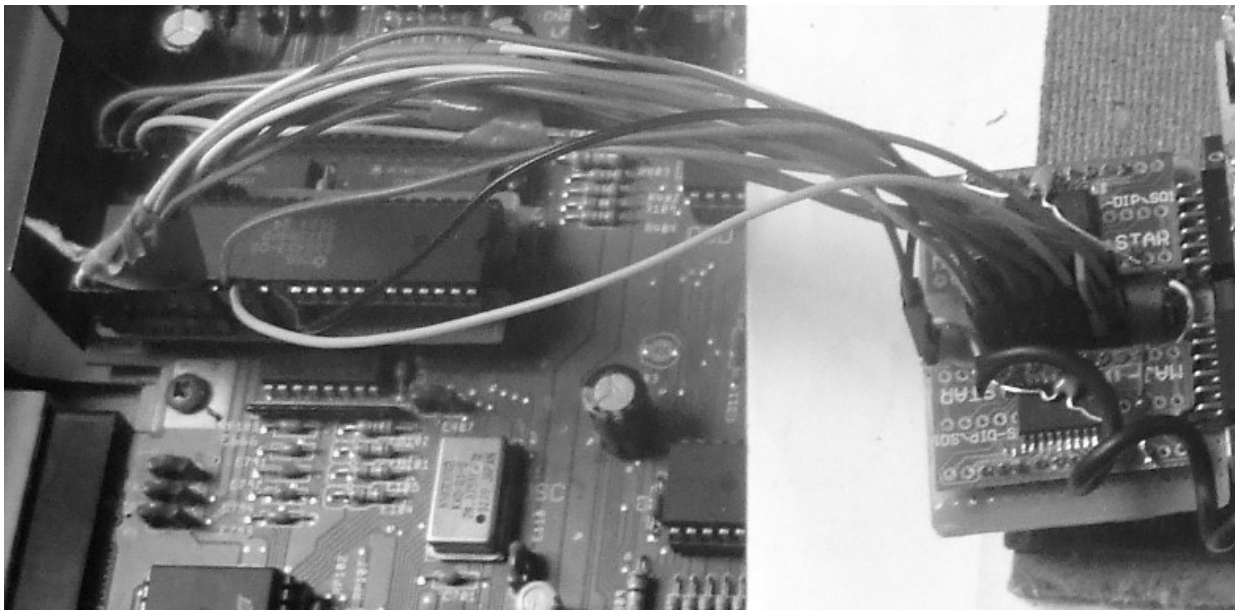
Rys. 4.1.7. Schemat płytki



Rys. 4.1.8. Płytka, widok od góry



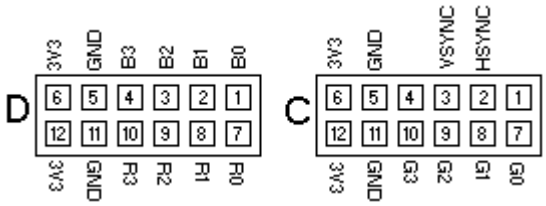
Rys. 4.1.9. Płytka, widok od dołu



Rys. 4.1.10. Płytka podłączona do sygnału wideo

4.1.3. Interfejs wyjściowy

Na interfejs wyjściowy przeznaczono 2 porty Vmod adaptera VmodMIB: C i D. Rozmieszczenie sygnałów wyjściowych na tych portach przedstawia rys. 4.1.11. Sygnał wyjściowy należy doprowadzić na złącze VGA. Rys. 4.1.12 i tabela 4.1.1 przedstawiają rozmieszczenie pinów na złączu VGA.



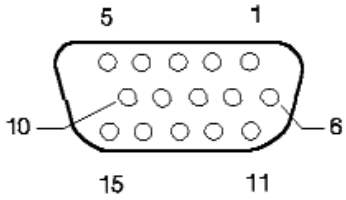
Rys. 4.1.11. Rozmieszczenie sygnałów wejściowych

Sygnał koloru należy przekonwertować do postaci analogowej. Do konwersji zastosowano rezystancyjny przetwornik C/A (rys. 4.1.13). Analogowy sygnał VGA przyjmuje wartości od 0 do ok. 0,7V, a rezystancja wejściowa monitora VGA wynosi 75Ω [16].

Napięcie analogowe w takim przetworniku można wyznaczyć ze wzoru:

$$V_{VGA} = \frac{\sum_{i=0}^{3} \frac{D_i}{R_i}}{\frac{1}{R_{VGA}} + \sum_{i=0}^{3} \frac{1}{R_i}}$$

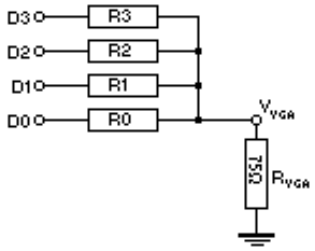
Przyjęto wartości R_3=510Ω, R_2=1kΩ, R_1=2kΩ, R_0=3,9Ω. Wówczas maksymalne napięcie analogowe wynosi 0,719V.



Rys. 4.1.12. Złącze VGA [16]

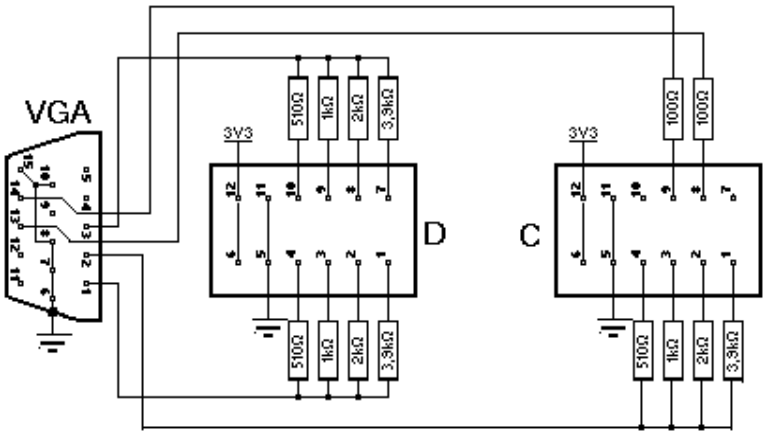
Pin	Signal Description
1	Red
2	Green
3	Blue
4	Monitor ID 2
5	Ground
6	Red Ground
7	Green Ground
8	Blue Ground
9	Plug
10	Ground
11	Monitor ID 0
12	Monitor ID 1
13	Horizontal Synchronization
14	Vertical Synchronization
15	Monitor ID 3

Tabela 4.1.1. Opis sygnałów na złączu VGA [16]

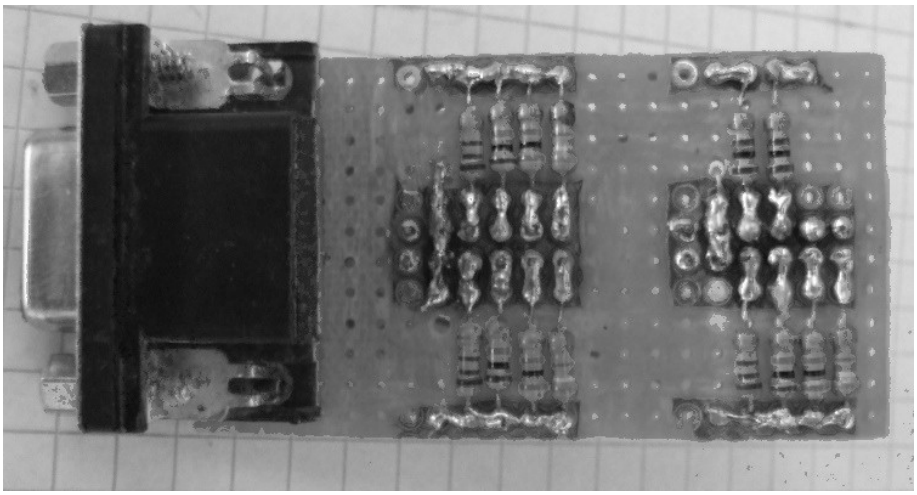


Rys. 4.1.13. Przetwornik C/A

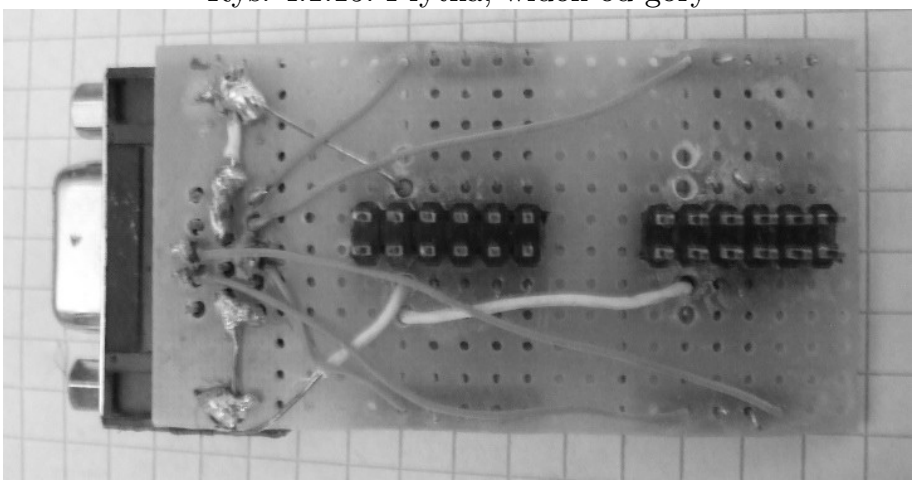
Interfejs wyjściowy zrealizowano na uniwersalnej płytce drukowanej. Schemat przedstawiono na rys. 4.1.14.



Rys. 4.1.14. Schemat płytki



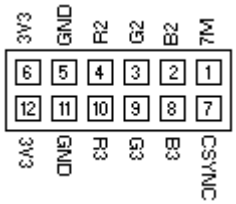
Rys. 4.1.15. Płytka, widok od góry



Rys. 4.1.15. Płytka, widok od dołu

4.1.4. Wyjście testowe

Poza interfejsem wejściowym i wyjściowym urządzenie wyposażono także w wyjście testowe, na które podawany jest wygenerowany sygnał wideo o parametrach identycznych, jak sygnał pochodzący z komputera. Umożliwia to testowanie urządzenia bez konieczności podłączania go do komputera, który zajmuje miejsce na biurku. Na wyjście testowe wykorzystano port Pmod. Rys. 4.1.16 przedstawia rozmieszczenie sygnałów na porcie Pmod.



Rys. 4.1.16. Rozmieszczenie sygnałów na wyjściu testowym

4.1.5. Zasilanie

Moduł Atlys może być zasilany przez port przez złącze J11 [9]. Źródłem zasilania jest komputer będący źródłem sygnału wideo. Jak widać na rys. 4.1.7, poza sygnałem wideo z komputera wyprowadzone jest zasilanie 5V oraz masa. Bramki 74LVT245 są zasilane z modułu Atlys napięciem 3,3V.

4.2. Część programowa

4.2.1. Zasada działania

Przed przystąpieniem do realizacji urządzenia należało wypracować zasadę jego działania. Ogólna koncepcja nie zmieniła się zasadniczo od czasu poprzedniej wersji. Urządzenie ma przyjmować sygnał wideo na wejściu i podawać przetworzony sygnał na wyjście.

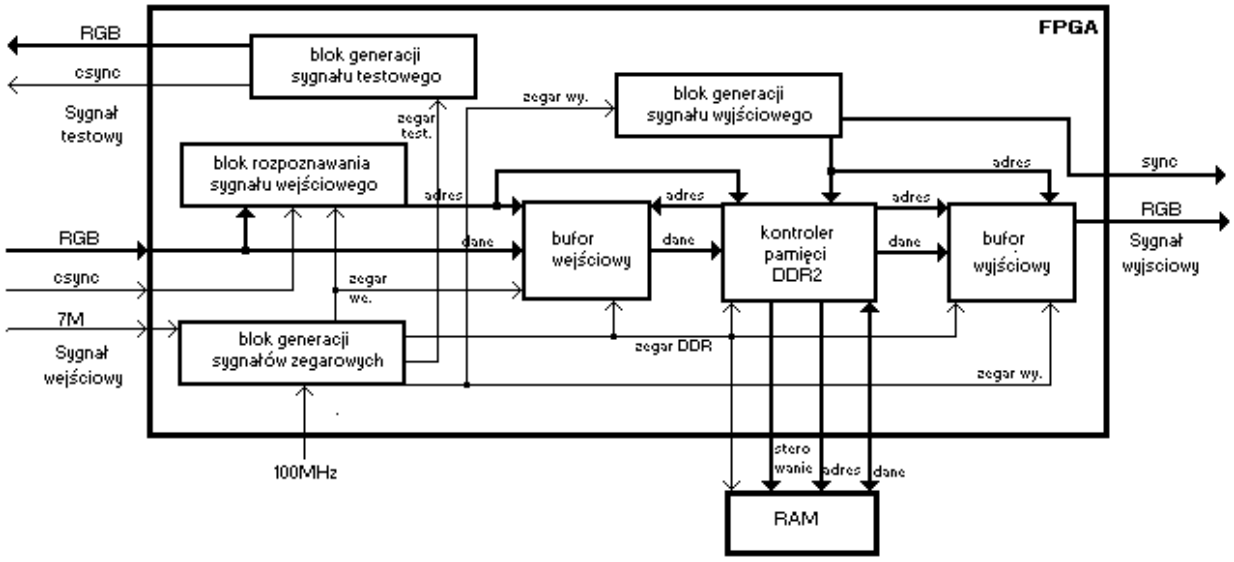
Ponieważ sygnał wyjściowy nie jest synchroniczny z sygnałem wejściowym, całą klatkę obrazu trzeba przechowywać w pamięci RAM. Do tego celu wykorzystano pamięć DDR2. Pamięć nie może być jednocześnie zapisywana i odczytywana. Ponadto odczyt i zapis nie mogą odbywać się w trybie ciągłym. To oznacza konieczność buforowania danych na wejściu i wyjściu. Przepływ danych w urządzeniu ukazuje rys. 4.2.1.

Rys. 4.2.1. Przepływ danych

Funkcjonalność urządzenia podzielono na części składowe. Można wyróżnić następujące części:

- generacja sygnałów zegarowych
- buforowanie danych wejściowych i wyjściowych
- rozpoznawanie sygnału wejściowego
- generacje sygnału wyjściowego
- generacja sygnału testowego
- sterowanie pamięcią DDR2
- wybór trybu pracy

Na rys. 4.2.2 przedstawiono schemat blokowy funkcjonalności urządzenia. Realizację poszczególnych części opisano w kolejnych podrozdziałach.



Rys. 4.2.2. Schemat blokowy urządzenia

4.2.2. Rozdzielczość sygnału wyjściowego

Należało wybrać taką rozdzielczość sygnału wyjściowego, żeby wymagania co do pojemności i szybkości działania były możliwie małe. Oznacza to, że najlepszym rozwiązaniem będzie wybór najniższej rozdzielczości, która pozwoli na wyświetlenie całej klatki obrazu wejściowego.

Maksymalna rozdzielczość sygnału wejściowego w trybie HIRES to 640 na 256 (512).Uwzględniając obszar nieadresowalny wynosi ona 752 na 286 (572). Najmniejsza rozdzielczość zawarta w standardzie VESA, w której można zmieścić obraz wejściowy to 800 na 600. Podobnie, jak w pierwszej wersji zdecydowano się zastosować w urządzeniu rozdzielczość 800 na 600 o częstotliwości pionowej 60 Hz.

Rozdzielczość 800 na 600 nie wystarcza do obsłużenia sygnału wejściowego w trybie SHRES o rozdzielczości poziomej. Najmniejsza rozdzielczość zawarta w standardzie VESA zdolna zmieścić klatkę obrazu wejściowego to 1280 na 1024. Ponieważ całkowity obszar aktywny ma szerokość 1504 pikseli, obsłużenie trybu SHRES wymaga wykrywania początku i końca obszaru adresowalnego.

Ostatecznie zdecydowano się na generację obu rozdzielczości. Za pomocą przełącznika można wybrać tryb pracy. Tabela 4.2.1 przedstawia parametry sygnału wideo w obu trybach pracy.

pixelclock:	40 MHz	108 MHz
vertical frequency	60,317 Hz	60,020 Hz
horisontal frequency	37,879 kHz	63,981 kHz
horizontal total time:	1056 px	1688 px
horizontal addresable time:	800 px	1280 px
horizontal front porch:	40 px	48 px
horisontal back porch:	88 px	248 px
horisontal sync time:	128 px	112 px
vertical total time:	628 lines	1066 lines
vertical addressable time:	600 lines	1024 lines
vertical front porch:	1 line	1 line
vertical back porch:	23 lines	38 lines
vertical sync time:	4 lines	3 lines

Tabela 4.2.1. Parametry sygnału wideo [2]

4.2.3. Generacja sygnałów zegarowych

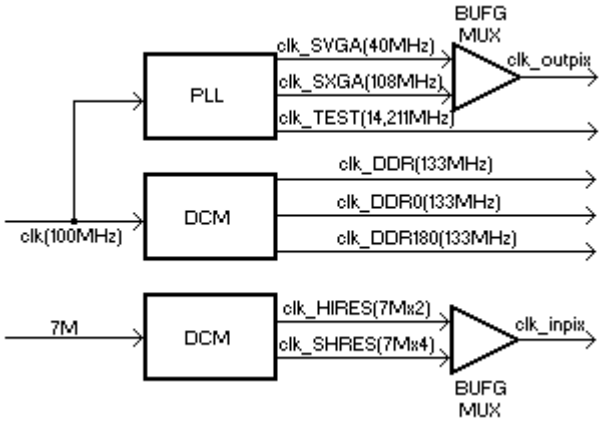
Urządzenie potrzebuje do działania kilku sygnałów zegarowych: Trzy zegary *pixelclock* dla sygnału wideo wejściowego, wyjściowego i testowego, oraz zegar dla pamięci DDR. W zależności od trybu pracy pixelclock wejściowy powinien mieć dwukrotną lub czterokrotną częstotliwość sygnału 7M, a pixelclock wyjściowy powinien mieć częstotliwość 40 lub 108 MHz. Pamięć DDR pracuje z częstotliwością 133MHz i potrzebuje zegara różnicowego.

Generację sygnałów zegarowych zrealizowano w pliku `clock_manager.vhd`. Do wygenerowania sygnałów `CLK_HIRES` o dwukrotnie wyższej częstotliwości i `CLK_SHRES` o czterokrotnie wyższej częstotliwości Użyto zasobu DCM [14]. Odpowiedni plik wygenerowano za pomocą narzędzia *Clocking wizard*. Oba sygnały są multipleksowane do sygnału `clk_inpix` za pomocą multipleksera zegarowego BUFGMUX. Nie ma osobnego zegara dla trybu LORES. Tryb LORES jest traktowany tak samo jak HIRES, przy czym każdy piksel jest próbkowany dwukrotnie.

Do generacji sygnałów `clk_SVGA` o częstotliwości 40MHz, `clk_SXGA` o częstotliwości 108MHz i `clk_test` o częstotliwości 14,211MHz użyto zasobu PLL. Odpowiedni plik wygenerowano za pomocą narzędzia *Clocking wizard*. Sygnały `clk_SVGA` i `clk_SXGA` są multipleksowane do sygnału `clk_outpix` za pomocą multipleksera zegarowego BUFGMUX.

Uzyto zasobu DCM do generacji trzech sygnałów o częstotliwości 133MHz: `clk_DDR`, `clk_DDR0`, i `clk_DDR180`. Do wygenerowania pliku również użyto narzędzia *Clocking wizard*. Sygnał `clk_DDR180` jest przesunięty w fazie o 180 stopni względem pozostałych sygnałów. Sygnał `clk_DDR` służy do taktowania kontrolera pamięci, pozostałe dwa do taktowania samej pamięci.

Generację sygnałów zegarowych zilustrowano na rys. 4.2.3.



Rys. 4.2.3. Generacja sygnałów zegarowych

4.2.4. Generacja sygnału wyjściowego

Generację sygnału wyjściowego zrealizowano w pliku `output_timing.vhd`. Ta część kodu odpowiada za generację sygnałów synchronizacyjnych (`out_hsync`, `out_vsync`), sygnałów wskazujących na obszar aktywny (`out_DE`, `out_hDE`) oraz chwilowych współrzędnych.

Generacja sygnału wideo oparta jest o dwa liczniki: (`hcount` i `vcount`) Z każdym taktem zegara inkrementowany jest licznik `hcount`. Jeżeli licznik osiągnie o jeden mniejszą od wartości sygnału `htotal_len`, jest zerowany. Przy każdym zerowaniu licznika `hcount` inkrementowana jest wartość `vcount`. Jezeli licznik osiągnie wartość o 1 mniejszą od wartości sygnału `htotal_len`, wtedy przy zerowaniu licznika `hcount` zerowany jest również licznik `vcount`. W ten sposób uzyskuje się odpowiednią długość linii oraz całego obrazu.

Wartość hcount równa zero oznacza początek sygnału synchronizacyjnego. Dlatego zerowaniu licznika hcount towarzyszy zerowanie sygnału hsync. W momencie doliczenia do wartości o 1 mniejszej od wartości sygnału hsync_len sygnał hsync przyjmuje ponownie wartość jedynki logicznej. W ten sposób generowany jest sygnał synchronizacji poziomej.

Alalogicznie generuje się sygnał synchronizacji pionowej. Wartość vcount równa zero oznacza początek sygnału synchronizacyjnego. Zerowaniu licznika vcount towarzyszy zerowanie sygnału vsync. W momencie doliczenia do wartości o 1 mniejszej od wartości sygnału vsync_len sygnał vsync przyjmuje ponownie wartość jedynki logicznej.

Obszar adresowalny zaczyna pewną liczbę pikseli na prawo od początku synchronizacji poziomej i pewną liczbę linii w dół od początku synchronizacji pionowej. Zatem wartości współrzędnych pionowej i poziomej są równe wartościom liczników vcount i hcount przesuniętym o pewną stałą wartość.

Każdemu zerowaniu licznika hcount towarzyszy przypisanie sygnałowi hpos wartości 0 - hact_start, analogicznie każdemu zerowaniu licznika vcount towarzyszy przypisanie sygnałowi vpos wartości 0 - vact_start. Kazdej inkrementacji licznika towarzyszy inkrementacja sygnału współrzędnej.

W momencie osiągnięcia przez licznik hpos wartości szesnastkowej FFF sygnał hDE przyjmuje stan jedynki logicznej. Jeżeli dodatkowo licznik vpos ma wartość mniejszą od wartości sygnału vact_len stan jedynki logicznej przyjmuje też sygnał DE. Gdy licznik osiągnie wartość o 1 mniejszą od wartości sygnału hact_len oba sygnały są zerowane. W ten sposób Sygnał DE wskazuje na obszar aktywny, natomiast hDE wskazuje na obszar aktywny rozciągnięty w pionie na wszystkie linie obrazu. Sygnał hDE służy do synchronizacji kontrolera pamięci DDR. Wartości sygnałów, do których porównywane są liczniki zależy od wybranego trybu pracy.

Poniżej przedstawiono kod procesu realizującego opisaną powyżej funkcjonalność:

```
process(out_pix_clk,reset) begin
  if rising_edge(out_pix_clk) then
    if  reset = '1' then
      hcount <= x"fff";
      vcount <= x"fff";
      hpos <= x"fff";
      vpos <= x"fff";
      hsync <= '1';
      vsync <= '1';
      DE <= '0';
    else
      hcount <= hcount + x"001";
      hpos <= hpos + x"001";
      if hcount >= (htotal_len - x"001") then
        hcount <= x"000";
        hpos <= x"000" - hact_start;
        vcount <= vcount + x"001";
        vpos <= vpos + x"001";
```

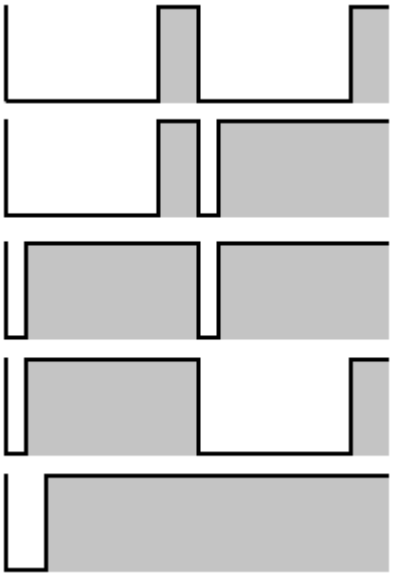
```
        hsync <='0';
        if vcount >= (vtotal_len - x"001") then
          vcount <= x"000";
          vpos <= x"000" - vact_start;
          vsync <= '0';
        elsif vcount = (vsync_len - x"001") then
          vsync <= '1';
        end if;
      elsif hcount = (hsync_len - x"001") then
        hsync <='1';
      end if;
      if hpos = x"fff" then
        hDE <= '1';
        if vpos < vact_len then
          DE <= '1';
        end if;
      elsif  hpos = (hact_len-x"001") then
        DE <= '0';
        hDE <= '0';
      end if;
    end if;
  end if;
end process;
```

4.2.5. Generacja sygnału testowego

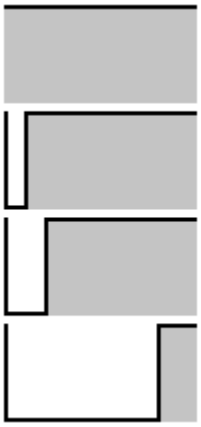
Generację sygnału testowego zrealizowano w pliku `test_signal_generator.vhd`. Ta część kodu odpowiada za generację sygnałów synchronizacyjnych (`test_hsync`, `test_vsync`, `test_csync`), oraz sygnału koloru (`test_R`, `test_G`, `test_B`). W zależności od wybranego trybu pracy generowany jest sygnał o 525 lub 625 liniach z przeplotem, albo o 312 lub 262 liniach bez przeplotu.

Zadanie generacji sygnałów podzielono na kilka procesów. W pierwszym procesie, analogicznie jak przy generacji sygnału wyjściowego, inkrementowane są liczniki `hcount` i `vcount` i zerowane gdy osiągną wartości odpowiednio `hmax` i `vmax`. Przy czym nie są zliczane całe linie, ale połówki linii. To ułatwia generację sygnału `test_csync`. Wartości `hmax` i `vmax` zależą od wybranego trybu pracy.

Drugi proces odpowiada za generację sygnałów synchronizacyjnych. W zależności od numeru linii sygnał `test_csync` przyjmuje jeden z przebiegów przedstawionych na rys. 4.2.4. (patrz podrozdział 2.2.1) Gdy rozpatrzy się połowy linii przebiegi wyglądają tak jak na rys. 4.2.5.



Rys. 4.2.4. Możliwe przebiegi sygnału synchronizacji kompozytowej dla pełnej linii.



Rys. 4.2.4. Możliwe przebiegi sygnału synchronizacji kompozytowej dla połowy linii.

Długości impulsów synchronizacyjnych przypisano stałym `sync_normal`, `sync_short` i `sync_inverted`. W momencie do osiągnięcia przez licznik odpowiednich wartości sygnałowi `slope` przypisywana jest jedna z tych stałych. Dodatkowo ustawiana jest wartość bitu `short`. Stan zerowy oznacza brak impulsu synchronizującego w liniach o parzystym numerze.

Gdy wartość licznika `hcount` jest mniejsza od stałej `sync_normal`, a numer linii jest parzysty (najmniej znaczący bit `vcount` równy 0) sygnał `test_hsync` przyjmuje wartość zera logicznego, w przeciwnym razie przyjmuje wartość jedynki, W ten sposób generowany jest sygnał synchronizacji poziomej.

Gdy wartość licznika `vcount` jest mniejsza od wartości `short0` lub gdy przyjmuje wartość pomiędzy `inverted1`, a `short2` (w przypadku sygnału z przeplotem) sygnał `test_vsync` przyjmuje wartość zera logicznego, w przeciwnym razie przyjmuje wartość jedynki, W ten sposób generowany jest sygnał synchronizacji pionowej.

Sygnał `test_csync` przyjmuje wartość zera logicznego gdy wartość licznika `hcount` jest mniejsza od wartości `slope`, za wyjątkiem sytuacji, gdy a numer linii jest nieparzysty, a sygnał short ma wartość 0. w przeciwnym razie przyjmuje

wartośćzera logicznego. Poniżej przedstawiono kod procesu realizującego opisaną wyżej funkcjonalnośc:

```
process(test_pix_clk, reset) begin
  if reset='1' then
    test_csync<='1';
    test_vsync<='1';
    test_hsync<='1';
    short<='1';
  elsif rising_edge(test_pix_clk) then
    if (hcount < slope) and ((not vcount(0) or short)='1') then
      test_csync<='0';
    else
      test_csync<='1';
    end if;
    if (hcount < sync_normal) and (vcount(0)='0') then
      test_hsync <='0';
    else
      test_hsync <='1';
    end if;

    if hcount = x"000" then
      if (vcount < short0) or ((vcount < short2)and(vcount >= inverted1)) then
        test_vsync <= '0';
      else
        test_vsync <= '1';
      end if;

      if (vcount = inverted0) or (vcount = inverted1) then
        slope <= sync_inverted;
        short <= '1';
      elsif (vcount = short0) or (vcount = short1) or (vcount = short2) or (vcount =
short3) then
        slope <= sync_short;
        short <= '1';
      elsif (vcount = normal0) or (vcount = normal1) then
        slope <= sync_normal;
        short <= '0';
      else
        slope <= slope;
        short <= short;
      end if;
    end if;
  end if;
end process;
```

Następny proces przelicza wartości liczników na wartości współrzędnych. Dla pierwszej połowy linii wartość współrzędnej poziomej `hcount2` jest równy wartości licznika `hcount`, dla drugiej połowy linii wartość współrzędnej jest równa sumie wartości licznika i długości połowy linii `hlen`.

Dla linii, w których wartość licznika `vcount` nie jest większa od wartości `inverted1` wartość współrzędnej pionowej `vcount` jest równa wartości licznika z

najmłodszym bitem ustawionym na zero. Dla pozostałych linii wartość vcount2 jest równa różnicy `vcount` z najmłodszym bitem ustawionym na zero i sygnału `inverted1`. Dzięki takiemu przeliczeniu w sygnale z przeplotem otrzymuje się wyłącznie współrzędne parzyste w jednej klatce obrazu i współrzędne nieparzyste w drugiej. Wyliczone w ten sposób współrzędne służą do generacji sygnału koloru w kolejnym procesie.

W owym procesie porównuje się współrzędne z odpowiednimi wartościami, żeby ustalić, czy dany piksel znajduje się w obszarze aktywnym i czy znajduje się w obszarze adresowalnym. Poza obszarem aktywnym bity koloru są zerowane. Wewnątrz obszaru adresowalnego wartość koloru zależy od wartości liczników, poza nim przyjmują stałą wartość. Opisaną powyżej funkcjonalność realizuje następujący kod:

```
--pixel coordinates
process(hcount, vcount, inverted1) begin
  if vcount(0)='1' then
    hcount2 <= hcount+hlen;
  else
    hcount2 <= hcount;
  end if;
  if vcount > inverted1 then
    vcount2 <= (vcount(11 downto 1)&"0")-inverted1;
  else
    vcount2 <= vcount(11 downto 1)&"0";
  end if;
end process;
--color values
process(test_pix_clk, reset) begin
  if reset='1' then
    test_R<=x"0";
    test_G<=x"0";
    test_B<=x"0";
  elsif rising_edge(test_pix_clk) then
    if (hcount2 >= act_start_h) and (hcount2 < act_stop_h) and (vcount2 >= act_start_v)
and (vcount2 < act_stop_v) then
      if (hcount2 >= (act_start_h+hoffset)) and (hcount2 <
(act_start_h+hoffset+addr_len_h)) and (vcount2 >= (act_start_v+voffset)) and (vcount2 <
(act_start_v+voffset+addr_len_v)) then
        test_R <= std_logic_vector(hcount2(3 downto 0));
        test_G <= std_logic_vector(vcount2(3 downto 0));
        test_B <= std_logic_vector((vcount2(0) xor hcount(4)) & hcount (7 downto 5));
      else
        test_R<=x"b";
        test_G<=x"6";
        test_B<=x"2";
      end if;
    else
      test_R<=x"0";
      test_G<=x"0";
      test_B<=x"0";
```

```
    end if;
  end if;
end process;
```

4.2.6. Rozpoznawanie sygnału wejściowego i odtwarzanie pozycji

Rozpoznawanie sygnału wejściowego zrealizowano w pliku `input_timing.vhd`. W tej części kodu rozpoznawane są parametry sygnału (przeplot, położenie obszaru adresowalnego) i odtwarzane są chwilowe współrzędne pikseli.

Rozpoznawanie sygnału wejściowego opiera na wykrywaniu impulsów sygnału synchronizacji kompozytowej `in_csync`. Po wykryciu zbocza opadającego resetowany jest licznik `hcount1`, do wartości równej 1, który jest inkrementowany w każdym cyklu zegara. Ponieważ impuls synchronizacyjny może wystąpić także w połowie linii, konieczne było wprowadzenie drugiego licznika `hcount2`, który również jest resetowany do wartości 1, ale tylko wtedy, gdy odstęp od poprzedniego impulsu jest wystarczająco długi (wartość `hcount1` większa od wartości minimalnej `min_h_len`) dodatkowo, do sygnału `len_h` przypisuje się zmierzoną w ten sposób długość linii. Gdy licznik `hcount2` osiągnie wartość o 1 mniejszą niż `len_h,` jest zerowany. Dzięki temu licznik zawiera prawidłową wartość we wszystkich liniach obrazu.

Po wykryciu zbocza narastającego wartość licznika `hcount1` jest porównywana z wartością graniczną `min_v_len`. Jeżeli nie jest mniejsza, oznacza to występowanie impulsu synchronizacji pionowej. Wówczas ustawiana jest flaga `vsflag1`, w przeciwnym razie jest kasowana. Jeżeli przedtem flaga nie była ustawiona, oznacza to początek impulsu. W takim wypadku porównuje się licznik `hcount2` z wartością `len_h_2`, żeby sprawdzić, czy jest to początek pierwszego czy drugiego półobrazu. Trzybitowy sygnał `fields` przechowuje aktualnie ustalony numer półobrazu oraz dwa poprzednie. Jeżeli środkowy bit ma wartość przeciwną niż pozostałe dwa, oznacza to sygnał wideo z przeplotem.

Jeżeli został wykryty początek pierwszego półobrazu lub przeplot nie występuje, wtedy zerowane są liczniki `vcount1` i `vcount2`. a do sygnału `len_v` wpisywana jest całkowita liczba linii (czyli wartość licznika `vcount1`). Jeżeli jest to początek drugiego półobrazu, wtedy w sygnale `len_f` zapamiętywana jest długość pierwszego.

Gdy resetuje się licznik `hcount2` inkrementowane są liczniki `vcount1` i `vcount2`. Dodatkowo, współrzędna pionowa `pos_v` zwiększana jest o 2 dla sygnału z przeplotem lub o 1 dla sygnału bez przeplotu. Gdy wartość licznika `vcount2` jest o 2 mniejsza od początku obszaru aktywnego (`voffset - 2`), wówczas sygnał `pos_v` przyjmuje wartość -2 dla sygnału z przeplotem lub -1 dla sygnału bez przeplotu. Gdy wartość licznika `vcount2` jest o 2 mniejsza od początku obszaru aktywnego w drugim półobrazie (`len_f + voffset - 1`), wtedy sygnał `pos_v` przyjmuje wartość -1. Gdy licznik `vcount2` osiągnie wartość o 1 mniejszą od wartości `len_v` jest on zerowany. Licznik współrzędnej poziomej `pos_h` inkrementuje się w każdym cyklu zegara i jest mu przypisywana wartość -2, gdy licznik `hcount2` doliczy do wartości o 3 mniejszej od początku obszaru aktywnego. (`hoffset`).

Sygnały hDE oraz DE, generowane są na podstawie współrzędnych `pos_h` i `pos_v` identycznie, jak w przypadku sygnału wyjściowego. Ponieważ zegar *pixelclock* może mieć dwu- lub czterokrotną częstotliwość sygnału `7M`, zmiana trybu pracy powoduje przesunięcie bitowe sygnałów porównawczych.

Poniżej przedstawiono kod procesu realizującego omówioną wyżej funkcjonalność:

```
process(in_pix_clk,reset) begin
  if reset='1' then
    prev_in_csync <= '1';
    hcount1 <= x"000";
    hcount2 <= x"000";
    vcount1 <= x"000";
    vcount2 <= x"000";
    len_h <= x"000";
    len_v <= x"000";
    len_f <= x"000";
    fields <= "000";
    s_interlace <='0';
    vsflag1 <= '0';
    voffset <= x"000";
    hoffset <= x"000";
    pos_v <= x"000";
    pos_h <= x"000";
    DE <= '0';
    hDE <= '0';
  elsif rising_edge(in_pix_clk) then
    voffset <= def_v_off;
    hoffset <= def_h_off;
    prev_in_csync <= in_csync;
    hcount1<=hcount1+x"001";
    hcount2<=hcount2+x"001";
    pos_h<=pos_h+x"001";
    s_interlace <= (fields(0) xor fields(1)) and (fields(1) xor fields(2));
    if hcount2 = (hoffset - x"003") then
      pos_h <= x"ffe";
    end if;
    if prev_in_csync = '1' and in_csync = '0' then
      hcount1 <= x"001";
      if hcount1 >= min_h_len then
        hcount2 <= x"001";
        len_h <= hcount1;
        -- len_hs <= hscount1;
      end if;
    end if;
    if prev_in_csync = '0' and in_csync = '1' then
      if hcount1 >= min_v_len then
        vsflag1 <= '1';
        if vsflag1 = '0' then
          if hcount2 < len_h_2 then
            fields(0)<='0';
            len_v <= vcount1;
            vcount1 <= x"000";
```

```
          vcount2 <= x"000";
        else
          fields(0)<='1';
          if s_interlace = '1' then
            len_f <= vcount1;
          else
            len_v <= vcount1;
            vcount1 <= x"000";
            vcount2 <= x"000";
          end if;
        end if;
        fields(2) <= fields(1);
        fields(1) <= fields(0);
      end if;
    else
      vsflag1 <= '0';
    end if;
  end if;
  if hcount2 >= (len_h - x"001") then
    if vcount2 = (voffset - x"002") then
      if s_interlace = '1' then
        pos_v <= x"ffe";
      else
        pos_v <= x"fff";
      end if;
    elsif (vcount2 = (len_f + voffset - x"001")) and (s_interlace = '1') then
      pos_v <= x"fff";
    else
      if s_interlace = '1' then
        pos_v <= pos_v + x"002";
      else
        pos_v <= pos_v + x"001";
      end if;
    end if;
    hcount2 <= x"000";
    vcount1 <= vcount1 + x"001";
    if vcount2 >= (len_v - x"001") then
      vcount2 <= x"000";
    else
      vcount2 <= vcount2 + x"001";
    end if;
  end if;
  if pos_h = x"fff" then
    if pos_v < act_v_len then
      DE <= '1';
    end if;
    hDE <= '1';
  elsif pos_h = (act_h_len - x"001") then
    DE <='0';
    hDE <='0';
  end if;
end if;
```

```
end process;
```

Gdy urządzenie pracuje w trybie SHRES, wtedy całkowity obszar aktywny obrazu wejściowego nie mieści się w obrazie wyjściowym. Żeby zapobiec utracie danych trzeba wykryć położenie obszaru aktywnego, które nie jest zdefiniowane i może być zmieniane z poziomu systemu operacyjnego. Można tego dokonać tylko przez analizę sygnału koloru.

Wewnątrz obszaru aktywnego ale poza obszarem aktywnym kolor ma stałą wartość. Zatem zmiana wartości koloru świadczy o obszarze adresowalnym. Wartość koloru jest porównywana z wartością w poprzednim cyklu. Jeżeli zostanie wykryta zmiana aktualizuje się odpowiednie rejestry. `border_l` oraz `border_r` zapamiętują położenie najbardziej wysuniętych na lewo i prawo pikseli w wartości różnej od sąsiadujących z nimi w poziomie pikseli. `border_t` i `border_b` przechowują położenie najbardziej wysuniętych w górę i w dół linii w których choć jeden piksel różni się od pozostałych.

Na początku kolejnej klatki sprawdza się, czy wszystkie skrajne wartości mieszczą się w wyznaczonym wcześniej obszarze adresowalnym. Jeżeli nie, obszar adresowalny jest przesuwany, poprzez aktualizację sygnałów `det_v_off` i `det_h_off` tak, żeby żadna z wykrytych skrajnych wartości nie znalazła się na zewnątrz.

Gdy urządzenie pracuje w trybie HIRES, położenie obszaru adresowalnego jest stałe.

4.2.7. Buforowanie obrazu

Dane wejściowe muszą być na bieżąco odczytywane w celu późniejszego zapisania ich w pamięci DDR2. Podobnie dane wyjściowe muszą być na bieżąco wysyłane i muszą być przygotowane wcześniej. Oznacza to konieczność buforowania danych wejściowych i wyjściowych.

W projekcie zastosowano dwa bufory - jeden wejściowy i jeden wyjściowy. Projekt zakłada, że w buforze mieszczą się dwie pełne linie obrazu. Wówczas, gdy jedna linia jest odczytywana druga jest zapisywana. Po przejściu obrazu do kolejnej linii następuje zmiana.

Bufory zrealizowano za pomocą zasobów Block RAM [13]. Odpowiednie pliki wygenerowano przy użyciu narzędzia *Block Memory Generator*. Bufor wejściowy jest pamięcią dwuportową w trybie *Simple Dual-port*, czyli ma jeden port wejściowy i jeden port wyjściowy taktowane niezależnymi sygnałami zegarowymi. Szerokość słowa na porcie wejściowym to 12 bitów, na wyjściowym 24 bity. Port wejściowy ma 12-bitowe wejście adresowe a port wyjściowy 11-bitowe. Bufor mieści 4096 słów 12-bitowych i wykorzystuje 3 zasoby Block RAM 18k. Bufor wyjściowy ma takie same parametry jak bufor wejściowy z tą różnicą, że port wejściowy ma szerokość słowa 24 bity, a wyjściowy 12.

Port wejściowy bufora wejściowego jest adresowany licznikami współrzędnych wejściowych. najstarszy bit adresu jest podłączony do najmłodszego bitu pionowej współrzędnej. Pozostałe bity są podłączone do najmłodszych bitów współrzędnej

poziomej. Wejście danych jest podłączone do wejściowego sygnału koloru, a wejście write enable do sygnału `in_DE`. Port wejściowy jest taktowany zegarem `clk_inpix`. Przy takim podłączeniu zapamiętywane są tylko piksele należące do obszaru adresowalnego, a każdy kolejny piksel trafia pod kolejny adres. Połową bufora zajmuje linia parzysta, a drugą nieparzysta. Port wyjściowy jest sterowany przez kontroler pamięci DDR2 i taktowany zegarem `clk_DDR`.

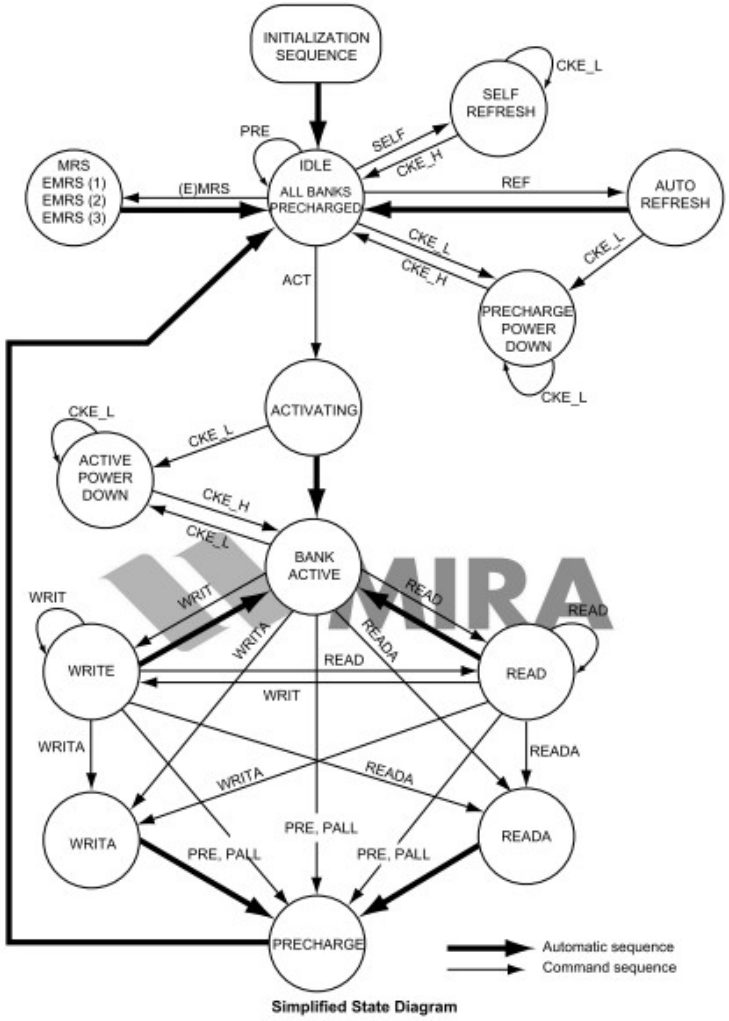
W buforze wyjściowym port wejściowy również jest sterowany przez kontroler pamięci DDR2 i taktowany zegarem `clk_DDR`. Port wyjąciowy jest adresowany licznikami współrzędnych wyjściowych. Przy rozdzielczości 800x600 każda linia wyjściowa jest wyświetlana podwójnie dla sygnału bez przeplotu, a pojedynczo dla sygnału z przeplotem. Przy rozdzielczości 1280x1024 każda linia wyjściowa jest wyświetlana poczwórnie dla sygnału bez przeplotu, a podwójnie dla sygnału z przeplotem. Dlatego, w zależności od trybu pracy najstarszy bit adresu jest podłączony do zerowego, pierwszego lub drugiego bitu współrzędnej pionowej. Pozostałe bity są podłączone do najmłodszych bitów współrzędnej poziomej. Wyjście danych jest podawane na wyjściowy sygnał koloru, gdy sygnał `out_DE` ma wartość 1. Ponieważ wyjście danych jest opóźnione o 1 cykl zegara względem wejścia adresowanego, sygnały synchronizacji muszą też być opóźnione o 1 cykl. Port wyjściowy jest taktowany zegarem clk_outpix.

4.2.8. Kontroler pamięci DDR2

Kontroler RAM jest centralnym elementem urządzenia. Jego zadaniem jest kopiowanie do pamięci DDR2 danych z bufora wejściowego i kopiowanie danych z pamięci DDR2 do bufora wyjściowego. Kontroler musi śledzić aktualną pozycję sygnału wyjściowego i wejściowego i w odpowiednim momencie wykonać operację odczytu lub zapisu pod odpowiednim adresem. Sygnał wejściowy i wyjściowy nie są synchroniczne, dlatego często się zdarza, że w czasie gdy może rozpocząć się operacja zapisu odczyt jeszcze się nie skończył, lub odwrotnie, w czasie gdy można rozpocząć operację odczytu wciąż trwa zapis. To znaczy, że kontroler musi kolejkować operacje odczytu i zapisu.

W tym projekcie pamięć pracuje z częstotliwością 133MHz co oznacza transfer 266M słów na sekundę. Jest to częstotliwość w zupełności wystarczająca na potrzeby tego projektu. Dużym problemem w poprzedniej wersji urządzenia była pamięć, której maksymalna częstotliwość (80MHz) była zbyt niska, przez co urządzenie nie mogło poprawnie działać.

Pamięć, w którą jest wyposażony moduł Atlys jest pamięcią DDR2 zgodną ze standardem JEDEC [10, 11]. Pamięć jest wyposażona w sygnały wymienione w tabeli 3.1.4. Działanie pamięci opiera się o maszynę stanów. Rys. 4.2.4. przedstawia uproszczony diagram stanów. Sterowanie pamięcią polega na wysyłaniu sygnałami sterującymi odpowiednich komend. Tabela 4.2.2 ukazuje listę komend wspieranych przez pamięć.



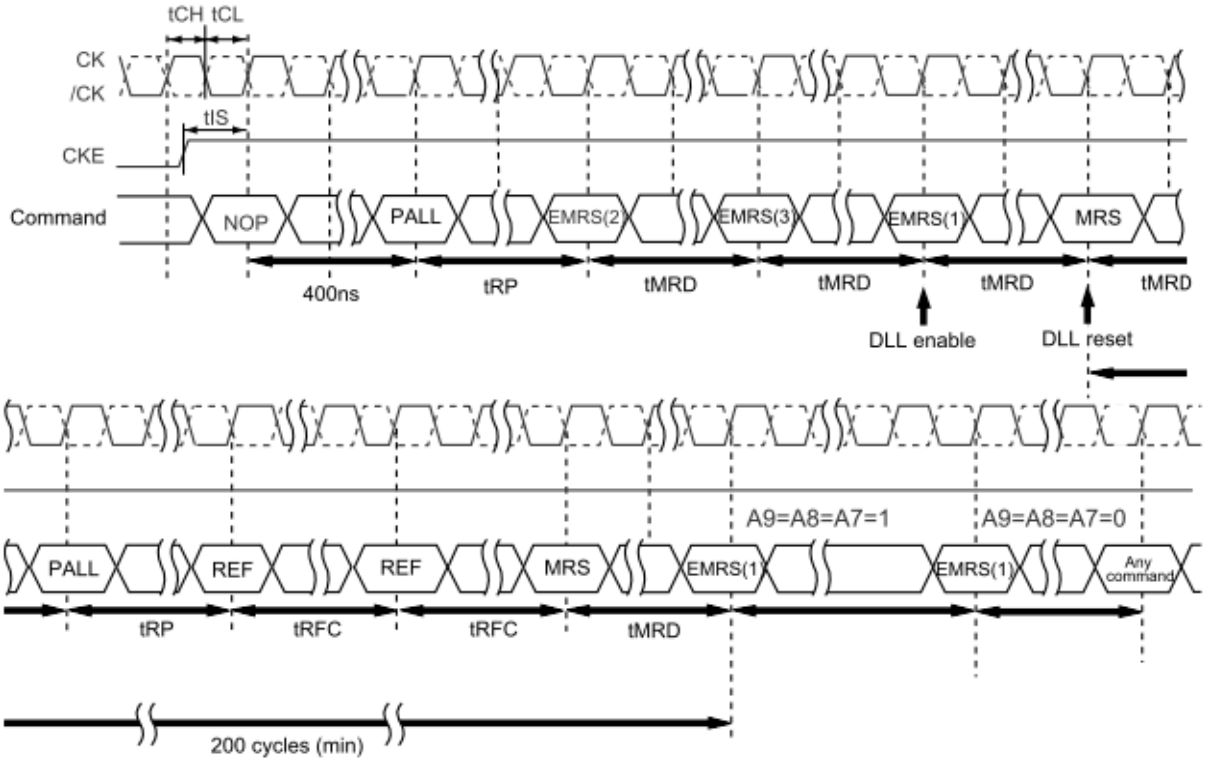
Rys. 4.2.4. Uproszczony diagram stanów [11]

Function	Symbol
Mode register set	MRS
Extended mode register set (1)	EMRS(1)
Extended mode register set (2)	EMRS(2)
Auto-refresh	REF
Self-refresh entry	SELF
Self-refresh exit	SELFX
Single bank precharge	PRE
Precharge all banks	PALL

Function	Symbol
Bank activate	ACT
Write	WRIT
Write with auto precharge	WRITA
Read	READ
Read with auto precharge	READA
No operation	NOP
Device deselect	DESL
Power-down mode entry	PDEN
Power-down mode exit	PDEX

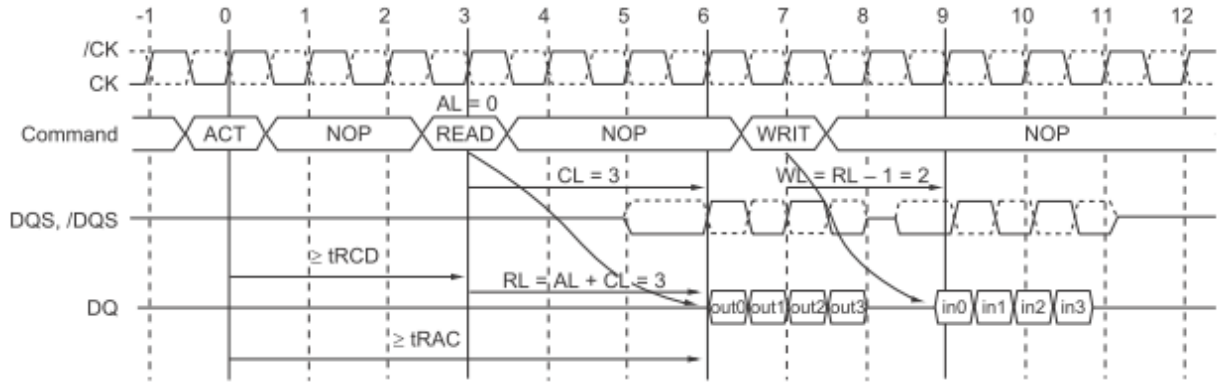
Tabela 4.2.1. Lista komend wspieranych przez pamięć [11]

Po uruchomieniu pamięć musi być zainicjowana. Procedura inicjalizacji została przedstawiona na rys. 4.2.5. W czasie inicjalizacji ustawiane są wartości rejestrów *Mode Regiser*, które odpowiadają za tryb pracy pamięci. Wartości wpisywane do rejestrów podaje się na linie adresowe. Po zakończeniu inicjalizacji kontroler może przejść do normalnego trybu pracy i wysyłać komendy np. odczytu czy zapisu.



Rys. 4.2.5. Procedura inicjalizacji pamięci [11]

Pamięć jest podzielona jest na banki, wiersze i kolumny. Przed wykonaniem odczytu lub zepisu należy najpierw aktywować wybrany wiersz w wybranym banku. W tym celu wywołuje się komendę ACT. Adres kolumny wybiera się za pomocą bitów adresowych Po upływie określonej liczby cykli zegarowych można wywołać komendę zapisu WRIT lub komendę odczytu READ. Podczas tej komendy bitami adresowymi ustawia się adres kolumny. Po określonej liczbie cykli zegarowych rozpoczyna się operacja zapisu lub odczytu pewnej liczy słów. Pamięć użyta w projekcie wspiera odczyt lub zapis 4 lub 8 słów. Dane są próbkowane na przecięciu się różnicowego sygnału DQS. W przypadku odczytu lub zapisu wewnątrz tego samego wiersza nie trzeba wysyłać kolejnej komendy aktywacji. Można wysyłać komendy w czasie trwania poprzedniej operacji. Przykładowy przebieg przedstawiono na rys. 4.2.6.



Rys. 4.2.6. Przykładowy przebieg odczytu i zapisu [11]

Początkowo projekt zakładał samodzielne napisanie obsługi pamięci DDR2 w języku VHDL. Jednak nie udało się tego dokonać pomimo wcześniejszego doświadczenia z pamięciami DDR. Żeby zmieścić się w założonym terminie zmieniono koncepcję.

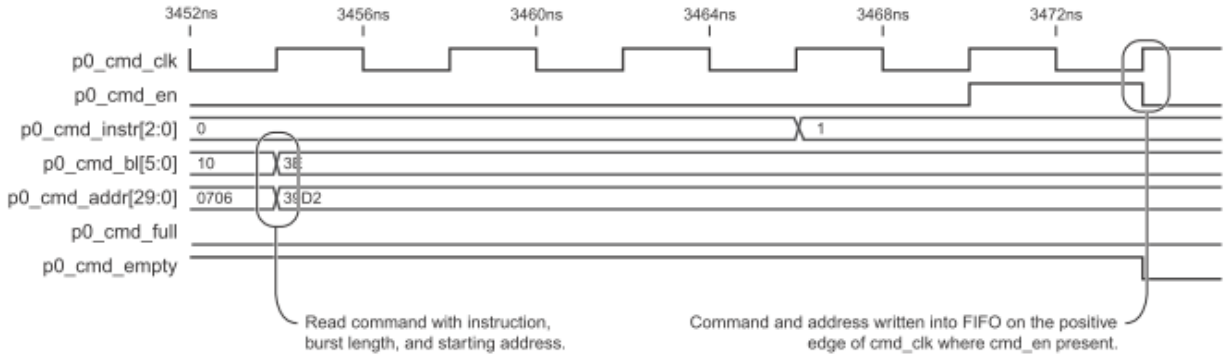
Zdecydowano się wykorzystać narzędzie *Memory Interface Generator* do wygenerowania interfejsu opartego o *Memory Control Block* [15]. Stworzony w ten sposób komponent steruje sygnałami pamięci i dostarcza postronie kodu użytkownika stosunkowo prosty w obsłudze interfejs. Generator pozwala na stworzenie interfejsu składającego się z maksymalnie dwóch portów dwukierunkowych i czterech portów jednokierunkowych. W projekcie użyto jednego portu dwukierunkowego.

Interfejs składa się z trzech niezależnych tzw. ścieżek danych: ścieżki zapisu odczytu i komend. Każda ścieżka jest może być taktowana niezależnym sygnałem zegarowym. Ścieżki danych są wyposażone w bufor FIFO o długości 64 słów, ścieżka komend jest wyposażona w bufor o długości 4 słów. Interfejs wspiera następujące komendy:

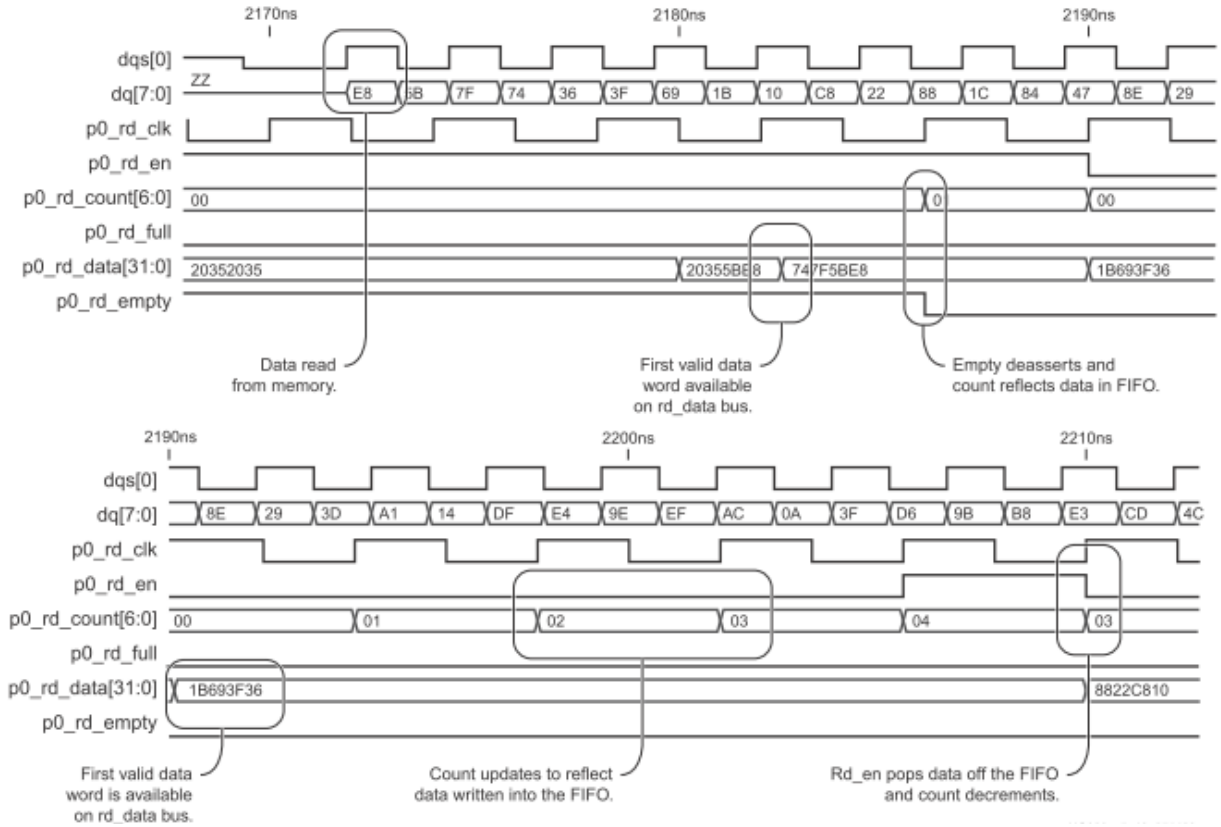
- Read
- Write
- Read with Auto Precharge
- Write with Auto Precharge
- Refresh

Komendy odświeżania Refresh można nie wysyłać, gdyż MCB sam dba o automatyczne odświeżanie.

W celu odczytania danych należy wysłać komendę Read (rys. 4.2.7). Wówczas ustala się adres początkowy oraz liczbę słów do odczytania, maksymalnie 64. Następnie można rozpocząć proces odczytu danych (rys. 4.2.8). Obecność danych w buforze sygnalizuje stan zerowy sygnału `rd_empty`. Ustawienie jedynki na `rd_en` powoduje wysunięcie jednego słowa z bufora. Można ustawiać jedynkę na `rd_en` nawet wtedy, gdy w buforze FIFO nie ma danych, wówczas sygnał jest ignorowany.

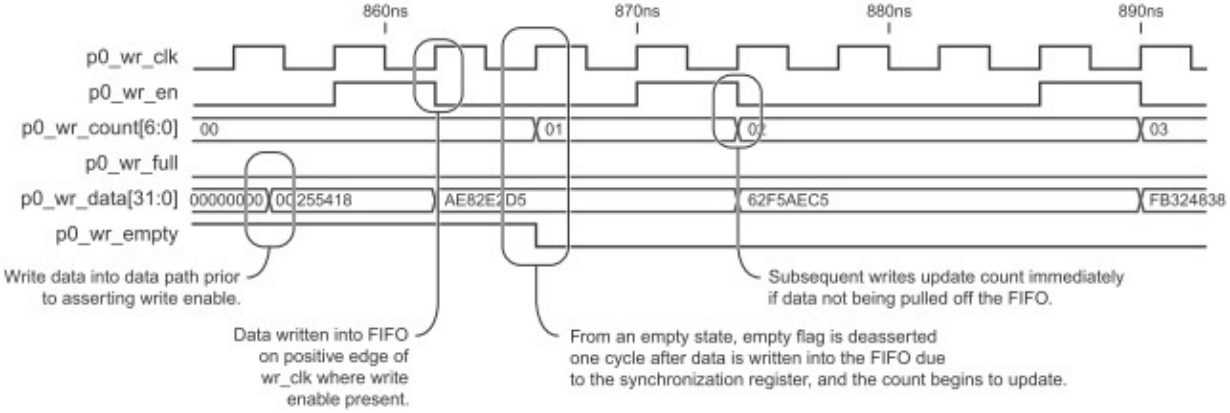


Rys. 4.2.7. Wysłanie komendy odczytu [15]

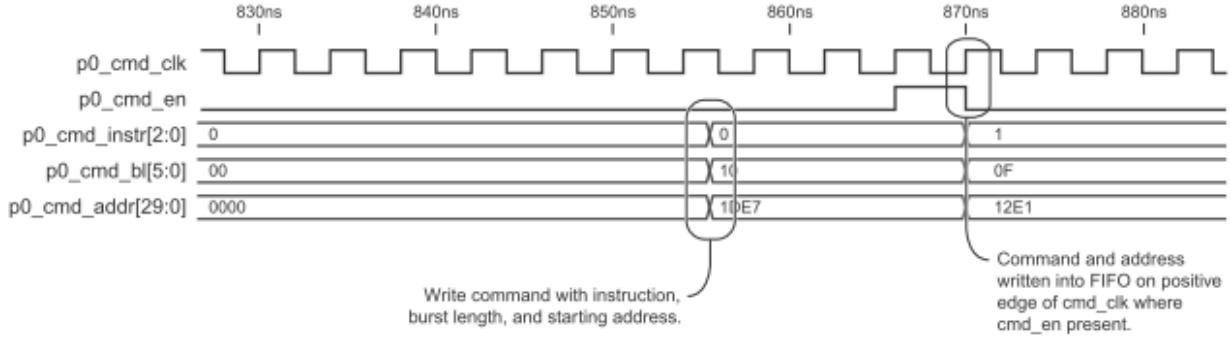


Rys. 4.2.8. Odczyt danych [15]

W przypadku zapisu danych należy najpierw wpisać dane do bufora FIFO rys. 4.2.9), a dopiero potem wysłać komendę zapisu rys. 4.2.10). ustawienie jedynki na sygnale wr_en powoduje wpisanie słowa do bufora FIFO. Podobnie, jak przy zapisie, przy wysyłaniu wybiera się adres początkowy i liczbę słów do zapisu.



Rys. 4.2.9. Zapis danych [15]



Rys. 4.2.10. Wysłanie komendy zapisu [15]

Ponieważ częstotliwość przesyłania danych w pamięci DDR jest dwukrotnie większa, niż częstotliwość danych, szerokość słowa w interfejsie MCB to 32 bity.

Proces obsługujący interfejs MCB oparto o maszynę stanów. Po uruchomieniu lub resecie maszyna znajduje się w stanie `s_reset`, w którym oczekuje na zakończenie inicjalizacji pamięci (pojawienie się jedynki na sygnale `calib_done`) Następnie maszyna przechodzi do stanu `s_wait`, w którym oczekuje na pojawienie się żądania odczytu lub zapisu pochodzące z osobnego procesu.

W przypadku wykrycia jedynki na fladze `rd_flag` maszyna przechodzi do stanu `s_setread`. Do odpowiednich rejestrów wpisywany jest adres bazowy i liczba 64-słowowych cykli odczytu. Flaga jest kasowana, a główny licznik `count` jest zerowany. W przypadku wykrycia jedynki na fladze `wr_flag` maszyna przechodzi do stanu `s_write` Do odpowiednich rejestrów wpisywany jest adres bazowy i liczba 64-słowowych cykli zapisu. Flaga jest kasowana, a główny licznik `count` jest zerowany.

W stanie `s_setread` zostaje wysłana komenda odczytu jeżeli bufor FIFO jest pusty, w przeciwnym razie maszyna czeka 1 cykl. Adres początkowy jest tworzony przez sumę adresu bazowego i wartości licznika `count`. Po wysłaniu komendy maszyna przechodzi do stanu `s_read`.

W stanie `s_read` odczytywane są dane z FIFO. Sygnał `rd_en` jest stale ustawiony na 1. Jeżeli sygnał `rd_empty` jest równy zero, oznacza to, że są dostępne dane. Wtedy licznik `count` jest inkrementowany, a sygnał `out_we` ustawiany na 1, żeby zapisać dane do bufora. W przeciwnym wypadku maszyna czeka 1 cykl zegarowy. Po odczytaniu 64 słów maszyna przechodzi do stanu `s_setwrite` lub, jeżeli to był ostani cykl odczytu w tej linii, do stanu `s_wait`.

W stanie `s_write` dane są zapisywane do FIFO. Sygnał `wr_en` jest stale ustawiony na 1, a licznik `count` jest inkrementowany. Po wysłaniu 64 słów maszyna przechodzi do stanu `s_setwrite`.

W stanie `s_setwrite` komenda zapisu jest wysłana jeżeli bufor FIFO jest pusty, w przeciwnym razie maszyna czeka 1 cykl. Adres początkowy jest tworzony przez sumę abresu bazoewgo i wartości licznika `count`. Po wysłaniu komendy maszyna przechodzi do stanu `s_write` lub, jeżeli to był ostatni cykl zapisu w tej linii, do stanu `s_wait`.

Najstarszy bit adresu buforów jest podłączony do dziesiątego bitu odpowiedniego adresu bazowego. Pozostałe bity adresów są podłączone do najmłodszych bitów licznika.

Drugi proces jest odpowiedzialny za ustawianie flag odczytu i zapisu. Ustawienie flagi powinno nastąpić w momencie przejścia sygnału wideo do kolejnej linii. Proces ustawia odpowiednie flagi po wykryciu zbocza narastającego na sygnale `in_hDE` lub `out_hDE`. Ponieważ te sygnały są taktowane innymi zegarami niż kontroler pamięci nie można wykluczyć występowania hazardów. Dlatego wykrywanie zbocza jest trochę bardziej złożone niż w przypadku sygnałów synchronicznych. Stan równy zero

dwa cykle pod rząd powoduje ustatienie tymczasowej flagi. Stan równy 1 dwa cykle pod rząd gdy flaga jest ustawiona oznacza wykrycie zbocza i skasowanie flagi. Po wykryciu zbocza, na podstawie wartości współrzędnej pionowej proces podejmuje decyzję czy i którą linięobrazu trzeba odczytać lub zapisać i ustawia odpowiednie sygnały.

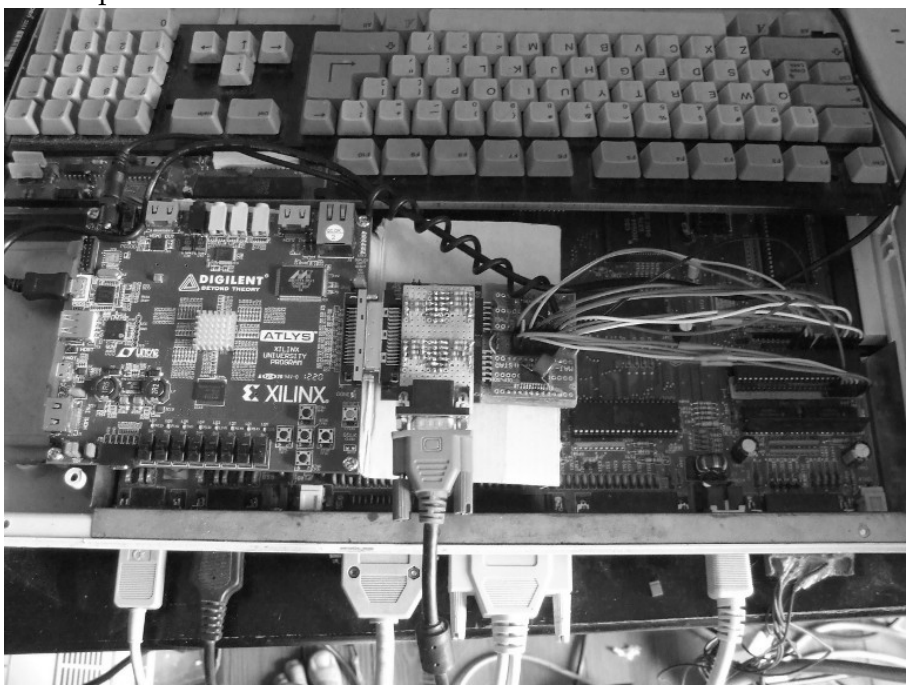
Narzędzie Memory Interfejs Generator zakłada, że MCB będzie taktowany zegarem zewnętrznym, a nie wygenerowanym wewnątrz FPGA. Z tego powodu w jednym z wygenerowanych plików umieszcza komponent IBUFG, który należy zamienić na BUFG, inaczej proces mapowania zakończy się niepowodzeniem.

4.3. Uruchomienie urządzenia

Aby uruchomić urządzenie należy wykonać następujące czynności: Górną część obudowy komputera należy zdjąć po odkręcaniu trzymających ją sześciu śrub. Pod obudową może znajdować się blaszany ekran. żeby zdjąć ekran trzeba odkręcić trzymające go śruby i odłączyć klawiaturę. Z lewej strony komputera znajduje się układ U4 (Denise). Układ należy wyjąć z podstawki, a w jego miejsce wstawić adapter. Układ włożyć w podstawkę na adapterze.

Moduł Atlys można położyć na stacji dyskietek. W celu zabezpieczenia przed zwarciem stacja powinna być przykryta papierem. Następnie do złącza można VHDC podłączyć adapter VmodMIB, do którego portów Vmod można podłączyć interfejs wejściowy i wyjściowy. Do interfejsu wejściowego należy podłączyć za pomocą przewodów sygnały z adaptera, a kabel zasilający włożyć do gniazdka J11.

Po włączeniu komputera włączy się również to urządzenie. Wtedy można je zaprogramować przez USB. Za pomocą przełącznika SW0 można wybrać tryb pracy. Środkowy przycisk służy do resetowania. Na rys. 4.3.1 przedstawiono na urządzenie podłączone do komputera.



Rys. 4.3.1. Urządzenie podłączone do komputera

5. Testowanie

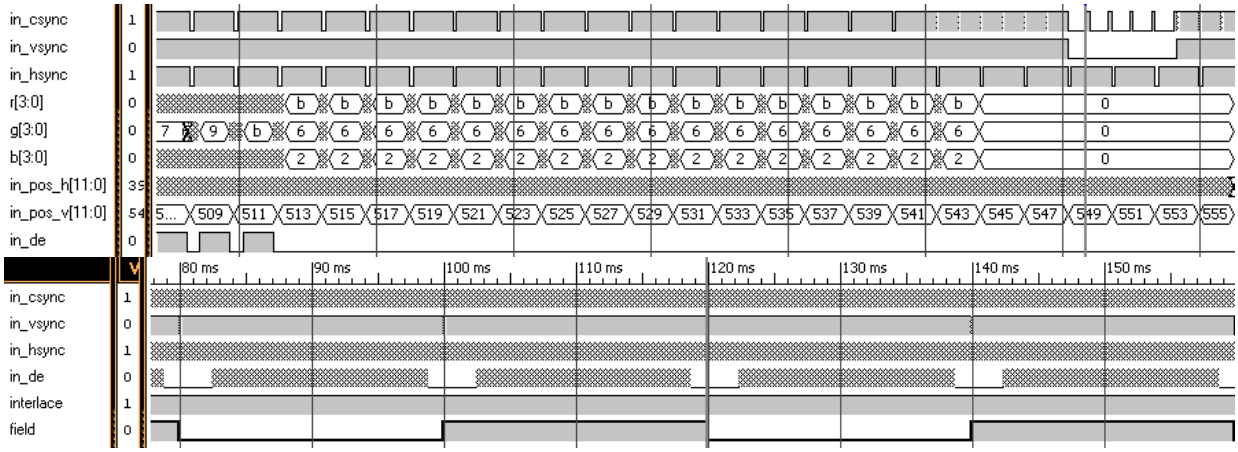
W czasie realizacji projektu, na różnych etapach pracy testowano poprawność działania urządzenia. Zastosowane środowisko programistyczne w wersji darmowej nie pozwala na śledzenie stanu sygnałów wewnątrz FPGA, co znacznie ogranicza możliwości debuggingu. W kolejnych podrozdziałach omówiono, w jaki sposób urządzenie było testowane i jakie były tego efekty

5.1. Symulacja

Przeprowadzono symulacje działania kodu odpowiedzialnego za generowanie sygnału testowego i za rozpoznawanie wejściowego sygnału wideo. Symulacja pozwoliła wykryć i poprawić błędy w kodzie. Do symulacji użyto programu ISim będącego częścią środowiska.

Kod odpowiedzialny za rozpoznawanie sygnału potrzebuje kilku klatek obrazu, żeby zsynchronizować się z sygnałem wideo. Dlatego czas symulacji ustawiono na 160ms. Wykonanie jednej symulacji takiego odcinka czasu trwało około 7 minut

Rys. 5.1.1 przedstawia przykładowe zrzuty ekranu symulatora.



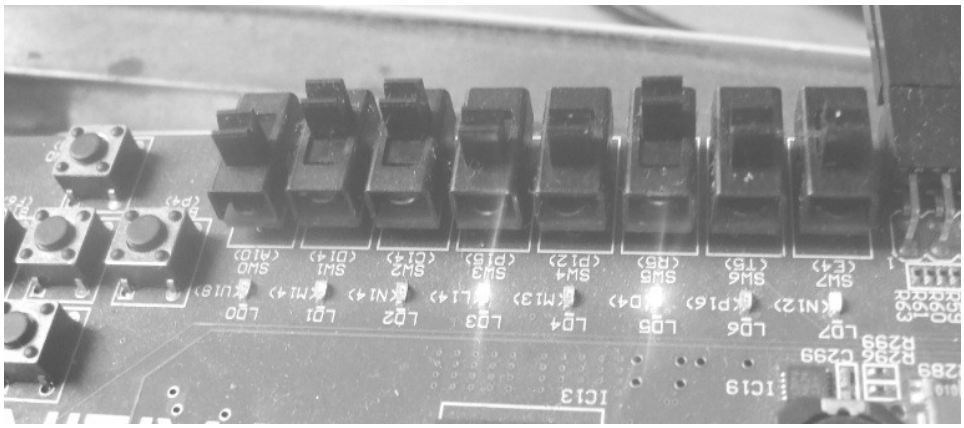
Rys. 5.1.1. Przykładowe symulacje

5.2. Praca krokowa.

O ile można było symulować generację i rozpoznawanie sygnału wideo, to kodu związanego z kontrolerem pamięci nie dało się już zasymulować. Wobec tego testowano maszynę stanową kontrolera pamięci za pomocą pracy krokowej.

Maszyna stanowa stanowa była taktowana sygnałem zegarowym generowanym za pomocą przycisków. Naciśnięcie jednego przycisku zmieniało stan zegara na wysoki, a nasiśnięcie drugiego na niski. Stan kontrolowanych sygnałów można było odczytać z diod LED. Za pomocą przełączników można było wybrać, który sygnał ma być wyświetlony.

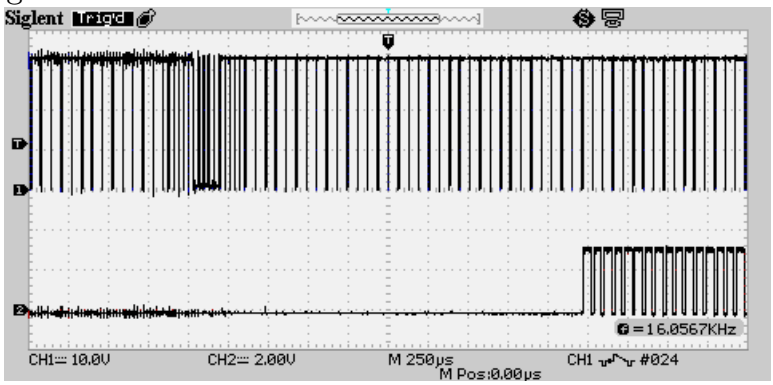
W ten sposób można było śledzić pracę maszyny stanowej. Dzięki temu wykryto i poprawiono błędy w kodzie.



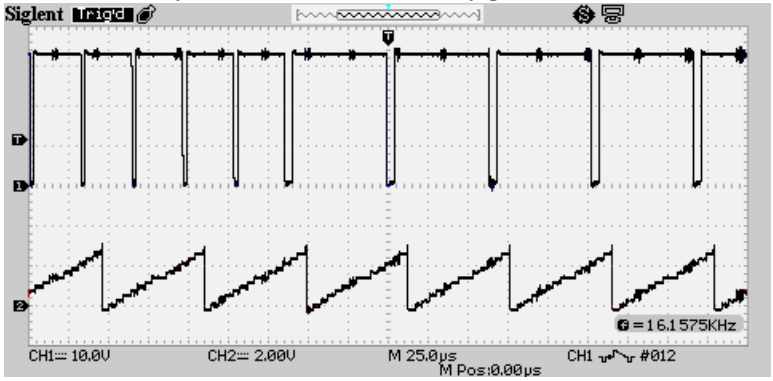
Rys. 5.2.1. Interfejs użyty do debuggingu

5.3. Port VGA

Niektóre sygnały śledzono, poprzez wyprowadzenie ich na port VGA i obserwację na oscyloskopie SDS 1152CML. Cztery najbardziej znaczące sygnały liczników podłączano do przetworników DAC, co umożliwiało śledzenie narastania liczników. W ten sposób można było potwierdzić sprawdzoną w symulacji poprawność rozpoznawania sygnału.



Rys. 5.3.1. Śledzenie sygnału DE



Rys. 5.3.2. Śledzenie współrzędnej poziomej

5.4. Obserwacja pracy urządzenia

Ostatnim etapem testowania było uruchomienie urządzenia i obserwacja jego pracy. Sygnał wyjściowy można było obserwować na monitorze VGA i porównywać z monitorem podłączonym bezpośrednio do komputera. W ten sposób można było zaobserwować, czy urządzenie spełnia swoją funkcję. Można było wykryć i poprawić

błędy. Na przykład rys. 5.4.1 pokazuje nieprawidłowy obraz spowodowany błędnym adresowaniem.

Rys. 5.4.1. Obraz zniekształcony przez błędne adresowanie.

Na rys. 5.4.2 widać pracę urządzenia w trybie rozdzielczości 800x600, a na rys. 5.4.3 pracę w rozdzielczości 1280x1024.

Rys. 5.4.2. Praca w rozdzielczości 800x600.

Rys. 5.4.3. Praca w rozdzielczości 1280x1024

Można zaobserwować, że obraz na wyjściu jest nieprawidłowy. Jest to spowodowane błędem, który nie został jeszcze wykryty. Biorąc pod uwagę charakter i regularność zniekształceń i można przypuszczać, że błąd dotyczy operacji kopiowania danych z bufora wejściowego do pamięci DDR..

6. Podsumowanie

6.1. Przebieg pracy

Praca polegała na opracowaniu i wykonaniu prototypu urządzenia przetwarzającego sygnał wideo korzystając z wiedzy dostępnej w literaturze oraz doświadczenia zdobytego w ramach realizowanej wcześniej pracy inżynierskiej. Cel ten został osiągnięty.

Opracowano koncepcję urządzenia opartą o układ FPGA przetwarzający sygnał wideo i zapamiętujący klatkę obrazu w pamięci DDR. Zidentyfikowano wymagania, co do interfejsów wejściowego i wyjściowego i wykonano te interfejsy.

Podzielono funkcjonalność urządzenia na podzadania:

- rozpoznawanie sygnału wejściowego,
- generacja sygnału wyjściowego,
- generacja sygnału testowego,
- buforowanie danych
- sterowanie pamięcią.

Następnie zrealizowano je tworząc kod w języku VHDL wykorzystując zasoby dostępne w układzie FPGA. W czasie pracy napotkano trudności w realizacji kontrolera pamięci, jednak udało się ukończyć to zadanie.

Użyto dostępnych środków do testowania działania urządzenia i poprawiania błędów. Ostatecznie, udało się stworzyć urządzenie spełniające przyjęte założenia.

6.2. Ocena

Udało się stworzyć urządzenie spełniające przyjęte założenia. Rozpoznawanie sygnału, buforowanie danych i generacja sygnału wyjściowego przebiegają prawidłowo. Również prawidłowo odbywa się sterowanie pamięcią DDR2, choć istnieje błąd powodujący przesunięcia przy kopiowaniu danych z bufora wejściowego. Błąd ten nie został usunięty tylko ze względu na ograniczenie czasowe.

Zdobyte doświadczenie pozwala stwierdzić, że układ FPGA jest narzędziem o bardzo dużych możliwościach i dobrze sprawdza się w przetwarzaniu sygnału wideo.

6.3. Dalszy rozwój

Wykonane w ramach niniejszej pracy urządzenie jest prototypem, co oznacza, że możliwy, a nawet wskazany jest jego dalszy rozwój.

Docelowe urządzenie powinno być zrealizowane na dedykowanej płytce drukowanej, co pozwoli ograniczyć jego rozmiary tak, żeby można było zmieścić urządzenie wewnątrz komputera.

Funkcjonalność urządzenia można rozszerzyć o niestandardowe możliwe do uzyskania na komputerach Amiga. Zapotrzebowanie na szybkość pamięci można zmniejszyć o 20% przez odpowiednie rozmieszczenie danych w pamięci. Obecnie 12-bitowy piksel zajmuje 16 bitów w pamięci DDR.

7. Bibliografia

1. *Video demysfified - a handbook for the digital enginee*r - Keith Jack, 2007
2. *VESA and Industry Standards and Guidelinesfor Computer Display Monitor Timing (DMT)* - Video Electronics Standards Association, 2007
3. *A DTV Profile for Uncompressed High Speed Digital Interfaces* - Consumer Electronics Association, 2006
4. *The obsolete technology website* (http://oldcomputers.net 26.06.2015)
5. *Big Book of Amiga Hardware - Flickerfixers & Scandoublers* (http://www.bigbookofamigahardware.com/bboah/CategoryList.aspx?id=13 28.06.2015)
6. *linux/drivers/video/amifb.c -- Amiga builtin chipset frame buffer device* - Geert Uytterhoeven with work by Roman Zippel, 2003 (http://stuff.mit.edu/afs/sipb/contrib/linux/drivers/video/ 08.07.2013)
7. *A500 plus Service Manual* - Commodore International, 1991
8. *Amiga Hardware Reference Manual* - Commodore International
9. *Atlys™ Board Refference Manual* - Digilent Inc., 2013
10. *DDR2 SDRAM Specification* - JEDEC Solid State Technology Association, 2008
11. *Data Sheet 1G bits DDR2 SDRAM P3R1GE3JGF(128M words × 8 bits) P3R1GE4JGF(64M words × 16 bits)* - MIRA
12. *Spartan-6 Family Overview* - Xilinx Inc., 2011
13. *Spartan-6 FPGA Block RAM Resources User Guide* - Xilinx Inc., 2011
14. *Spartan-6 FPGA Clocking Resources User Guide* - Xilinx Inc., 2013
15. *Spartan-6 FPGA Memory controller User Guide* - Xilinx Inc., 2010
16. *VGA / XGA / XGA-2 Technical Reference Manual* - International Business Machines Corporation, 1992