

Organy

Jakub Kałużka, 170995@student.pwr.wroc.pl

Piotr Kłys, 171168@student.pwr.wroc.pl

Politechnika Wrocławska

Dokumentacja jest częścią projektu, realizowanego w ramach przedmiotu *Układy Cyfrowe i Systemy Wbudowane 2* pod kierunkiem dra inż. Jarosława Sugiera.

Copyright © 2011 Jakub Kałużka, Piotr Kłys

Udziela się zezwolenia do kopiowania, rozpowszechniania i/lub modyfikacji tego dokumentu zgodnie z zasadami Licencji GNU GPL.

Autorzy dołożyli wszelkich starań, aby zawarte w tej publikacji informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych i autorskich. Autorzy nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w tej publikacji.

SPIS TREŚCI

1. Przedstawienie tematu	3
1.1. Cel i zakres projektu	3
1.2. Teoria problemu	3
1.3. Opis sprzętu	3
2. Opis projektu	6
2.1. Hierarchia plików źródłowych	6
2.2. Opis funkcjonalny modułu „dekoder”	9
2.3. Opis funkcjonalny modułu „wyswietlacz”	15
3. Implementacja	22
4. Podsumowanie	29
5. Bibliografia	30

1. Przedstawienie tematu

1.1. Cel i zakres projektu

Niniejszy projekt został stworzony przez Jakuba Kałużkę (170995) i Piotra Kłysa (171168) na potrzeby zaliczenia kursu „*Układy Cyfrowe i Systemy Wbudowane 2 - projekt*”, prowadzonego przez dra inż. Jarosława Sugiera na VI semestrze studiów informatycznych na Wydziale Elektroniki Politechniki Wrocławskiej w semestrze letnim 2010/2011. Tematem projektu było zrealizowanie w języku VHDL na platformę sprzętową *Spartan-3E Starter* firmy *Xilinx* aplikacji symulującej w swoim działaniu organy. Napisana aplikacja miała łączyć ze sobą poprzez płytę drukowaną wraz z głośnikiem klawiaturę umożliwiającą granie poszczególnych dźwięków oktawy oraz wyświetlacz VGA wyświetlający klawiaturę organów i wskazujący aktualnie naciśnięty klawisz.

1.2. Teoria problemu

Napisana aplikacja miała łączyć ze sobą poprzez płytę drukowaną wraz z głośnikiem, klawiaturę umożliwiającą granie poszczególnych dźwięków oktawy oraz wyświetlacz VGA wyświetlający klawiaturę organów i wskazujący aktualnie naciśnięty klawisz. Istotą problemu było jednoczesne odgrywanie odpowiedniego dźwięku z przyjętej oktawy (przyjęto oktawę dwukreślną) z jednoczesną wizualizacją naciśniętego klawisza. Należało zadbać ponadto o to aby uzyskiwany za pomocą przetwornika *DAC* dźwięk odpowiadał częstotliwościowo danemu dźwiękowi z oktawy i przypisać go w odpowiedni sposób do podłączonej do płyty klawiatury. Dźwięki na klawiaturze należało porozmieszczać tak aby zmapowana na ekranie klawiatura opowiadała klawiaturze prawdziwych organów. W implementacji należało wykorzystać gotowe moduły zewnętrzne służące do odczytu kodu klawisza z klawiatury (*PS2_Kbd*) i obsługi wysyłania danych do przetwornika cyfrowo-analogowego *LTC2624*.

1.3. Opis sprzętu

Platformą sprzętową na której realizowany był niniejszy projekt jest produkt firmy *Xilinx* – układ *FPGA Spartan-3E XC3S500E*. Produkt ten należy do rodziny programowalnych cyfrowych układów logicznych o dużym stopniu scalenia w których istnieje możliwość przeprogramowania 'w locie' poprzez zastosowanie mechanizmu częściowej rekonfiguracji. Dzięki temu układ może dostosować własną strukturę tak aby lepiej sprostać zadaniom, realizowanym w danym momencie. Aby zdefiniować zachowanie układu *FPGA* używa się języka opisu sprzętu takiego jak *Verilog* czy *VHDL*. Następnie przy pomocy narzędzi syntezy generuje się listę połączeń, która potem w procesie implementacji jest odwzorowywana w konkretnym układzie. Należy zwrócić uwagę, że proces syntezy dopuszcza tworzenie układów logicznych dowolnych rozmiarów, podczas gdy proces implementacji jest próbą wpisania go do konkretnego układu *FPGA*, gdzie może zabraknąć

Organy

zasobów do realizacji zadanej logiki. Do zaprogramowania układu *FPGA* służy plik binarny, który zawiera informacje o konfiguracji układu.

Układ należy do rodziny *Virtex-II/Spartan-3* i oznaczony jest symbolem „E” co określa, iż jest ukierunkowany przez producenta na zwiększoną liczbę zasobów logicznych, takich jak liczba komórek logicznych, bloków CLB czy plastrów. Nazwa układu - *XC3S500E* określa ponadto, iż składa się on z 500 tys. bramek logicznych, co potwierdza tabela z wartościami poszczególnych logicznych zasobów udostępniona przez producenta w dokumentacji produktu:



XILINX®

Rys1. Fizyczny wygląd Spartan-3E

Device	System Gates	Equivalent Logic Cells	CLB Array (One CLB = Four Slices)				Distributed RAM Bits	Block RAM Bits	Dedicated Multipliers	DCMs	Maximum User I/O	Maximum Differential I/O Pairs
			Rows	Columns	Total CLBs	Total Slices						
XC3S500E	500K	10,476	46	34	1,164	4,656	73K	360K	20	4	232	92

Rys2. Specyfikacja techniczna XC3S500E

Zastosowana platforma sprzętowa posiada układ *XC2C64A CoolRunner CPLD* pozwalający na automatyczną konfigurację tuż po załączeniu zasilania. Odbywa się to po przez pamięć *XCF04S* przechowującą plik z konfiguracją *FPGA*.

Platforma posiada także programator *USB* pozwalający w sposób bezpośredni skonfigurować układ *FPGA*. Jest to bardzo istotna kwestia, gdy brak jest złącza *LPT* w komputerze. Cały programator jest oparty o układ firmy *Cypress CY7C68013A* i pracuje w standardzie *USB 2.0* w trybie *High Speed*.

Spartan-3E Starter Kit został wyposażony w złącze *RJ-45*. Jest to złącze pozwalające na wykorzystanie technologii *Ethernet*, zawierającej standardy wykorzystywane w budowie głównie lokalnych sieci komputerowych. Obejmuje ona specyfikację kabli oraz przesyłanych nimi sygnałów. Technologia *Ethernet* opisuje również format ramek i protokoły z dwóch najniższych warstw *Modelu OSI*. Całość jest obsługiwana przez układ *LAN83C185* firmy *SMSC*. Pełna zgodność ze standardem *IEEE 802.3/802.3u*.

Na płycie znajdują się dwa złącza od komunikacji szeregowej *RS232*. Jedno jest żeńskie a drugie męskie. Interfejs ten jest bardzo przydatny w komunikacji z innym urządzeniem, mimo swojego archaicznego mechanizmu jest bardzo prosty do realizacji upraszczając wiele spraw w porównaniu np. z *USB*. Aby była możliwa komunikacja

Organy

szeregowa użyto konwertera napięć *ST3232* na napięcia akceptowalne przez układ *FPGA* w tym przypadku to 3,3 V.

Prócz wyżej wymienionego interfejsu szeregowego *RS232* znajdującego się na płycie, jest jeszcze jedno złącze pracujące w standardzie szeregowym – złącze o standardzie *PS/2*. Dzięki niemu istnieje możliwość podłączenia do płyty klawiatury bądź myszy od komputera PC.

Następny interfejs to *VGA*. Jest to bardzo prosty interfejs składający się z 3-bitowego przetwornika *DAC*. Przetwornik ten operuje na 4 kanałach i za jego sprawą możliwe jest uzyskanie na wyświetlaczu *VGA* ośmiu kolorów włącznie z czarnym oraz białym.

Przetworniki analogowo-cyfrowe i cyfrowo-analogowe - układy sproduktowane przez firmę Linear Technology, *LTC2624* to przetwornik *CAD* (cyfrowo-analogowy) 12-bitowy a *LTC1407* jest przetwornikiem *ADC* (analogowo-cyfrowy) 12-bitowy. Oba przetworniki wykorzystują interfejs szeregowy *SIP*.

Prócz wyżej wymienionych interfejsów płyta ma trzy złącza gold-pinowe oraz złącza rozszerzeń *Hirose FX2*. Złącza te zostały pozostawione użytkownikowi do wykorzystania w dowolny sposób.

Dodatkowe elementy interfejsu użytkownika znajdujące się na płycie to:

- Wyświetlacz LCD 2x16 z wbudowanym odpowiednikiem sterownika graficznego firmy Hitachi 44780. Jest on bardzo popularnym interfejsem graficznym używanym w systemach wbudowanych ze względu na prostotę obsługi oraz na dostępność do materiałów związanych z współpracą systemów z wyświetlaczem.
- 8 diód LED.
- 4 mikro-switchy.
- 4 przełączniki pozycyjne.
- Obrotowy przełącznik

Bardzo ważnymi elementami zastosowanej platformy sprzętowej są pamięci: *DDR SDRAM 64 MB × 16*, *NOR Flash 16 MB*, *SPI Flash 16 MB*. Można je wykorzystać do zwiększenia wydajności projektów oraz do konfiguracji *FPGA*.

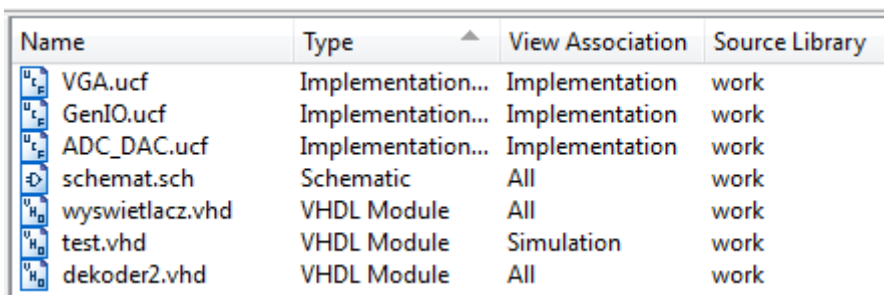
Płyta posiada oscylator kwarcowy 50 MHz dla *FPGA*. Można go zmienić mnożąc lub dzieląc częstotliwość przy użyciu bloków *DCM*. Jeżeli nie jest to wystarczające dla użytkownika to można skorzystać z dodatkowych źródeł generujących, a mianowicie złącze dip 8 znajdującego się na płycie dla oscylatora kwarcowego lub wykorzystać generator zewnętrzny który będzie generował sygnał zegarowy dostarczany przez złącze *SMA* klawiatury oraz przetwornika *Spartan 3E-Starter (Xilinx)*.

Do płyty poprzez przetwornik *DAC* istnieje także możliwość podłączenia głośnika i odtwarzania na nim dźwięku.

Organy

2. Opis projektu

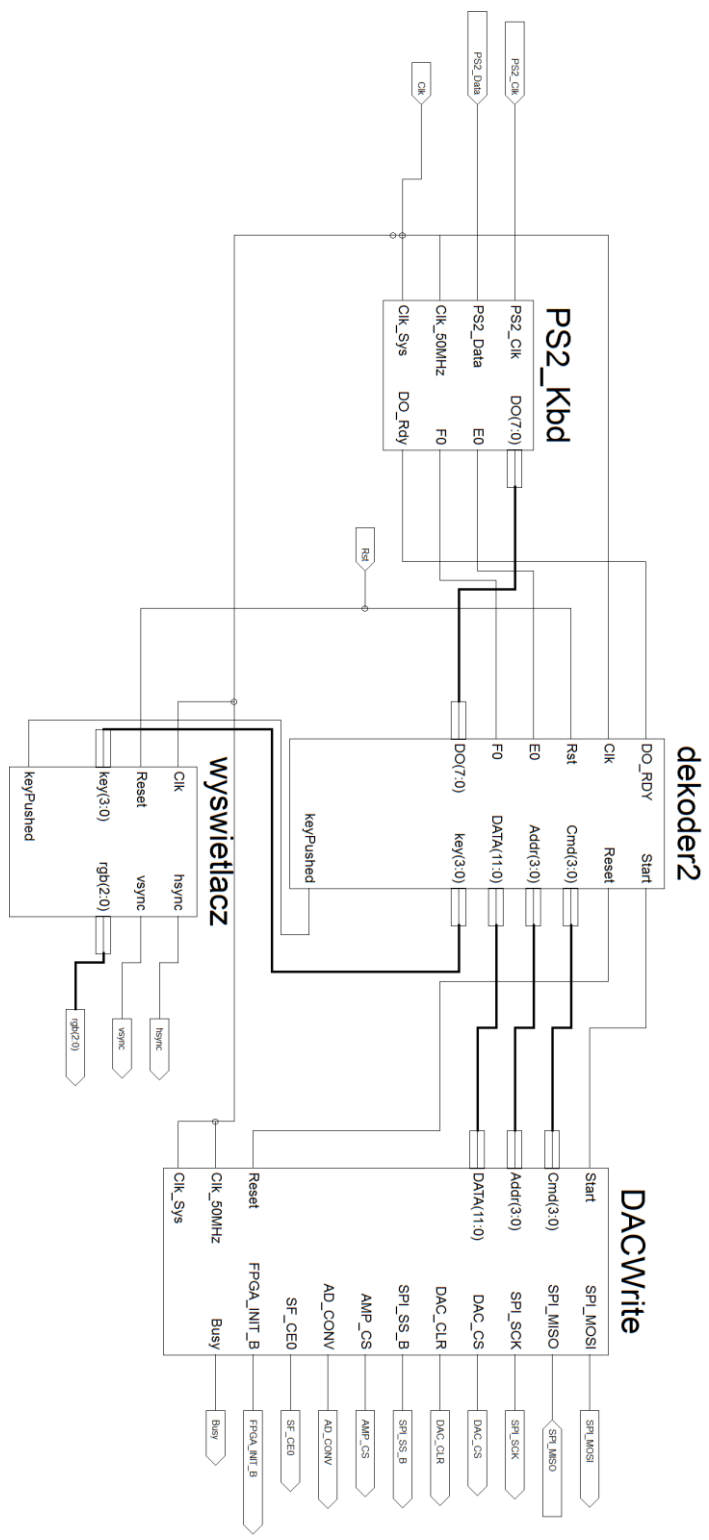
2.1. Hierarchia plików źródłowych



Name	Type	View Association	Source Library
VGA.ucf	Implementation...	Implementation	work
GenIO.ucf	Implementation...	Implementation	work
ADC_DAC.ucf	Implementation...	Implementation	work
schemat.sch	Schematic	All	work
wyswietlacz.vhd	VHDL Module	All	work
test.vhd	VHDL Module	Simulation	work
dekoder2.vhd	VHDL Module	All	work

Rys3. Hierarchia plików w aplikacji Organy

Pliki źródłowe projektu zostały przedstawione na powyższym rysunku. Pierwsze 3 pliki liczone od góry służą do translacji odpowiednich portów, tak aby program załadowany na platformę Xilinx Spartan-3E mógł przysyłać informacje na odpowiednie porty innych modułów znajdujących się na niej. Pliki te zostały zaimportowane ze strony internetowej *Zakładu Sieci Komputerowych*. **Schemat.sch** przedstawia logiczne rozmieszczenie modułów *.vhd tak, aby zrealizowanie postawionego zadania było możliwe.

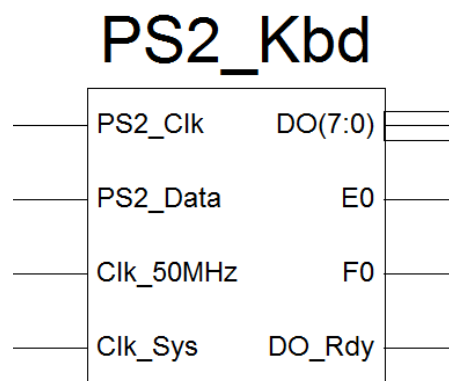


Rys4. Schemat ogólny aplikacji Organy

Organy

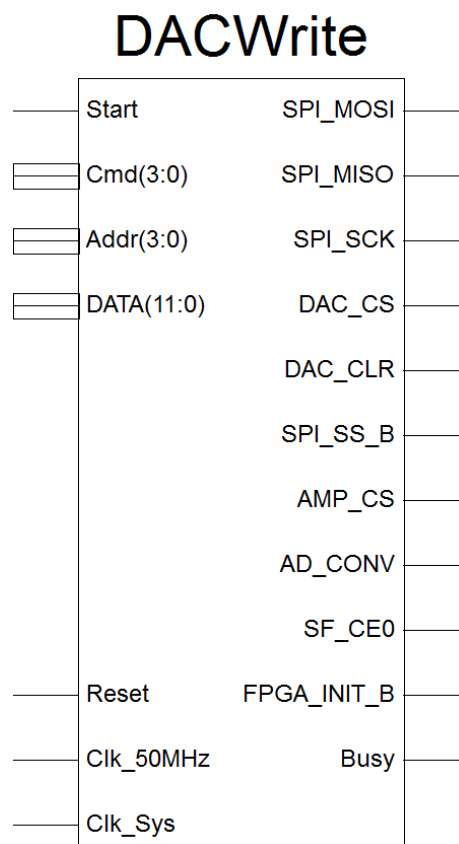
Moduły kodu VHDL **dekoder2.vhd** oraz **wyswietlacz.vhd** są autorskimi skrzynkami, realizującymi założone zadanie. Ponadto ze schematu możemy wyczytać, że użyte zostały jeszcze 2 moduły, mianowicie **DACWrite** oraz **PS2_Kbd**. Są to moduły pobrane ze strony prowadzącego zajęcia, które po krótkce zostały poniżej opisane. Każdy z tych modułów został przedstawiony graficznie poniżej.

PS2_Kbd - Moduł ten realizuje operację wejścia w projektowanym systemie. Obsługuje on klawiaturę podłączoną przez port *PS2*. Jego obecność jest na tyle ważna, iż dekoduje on kod naciśniętego klawisza i przesyła tę sekwencję do modułu autorskiego **dekoder**.



Rys5. Schemat ogólny modułu PS2_Kbd

DACWrite - Moduł obsługuje wysyłanie danych do przetwornika cyfrowo-analogowego LTC2624 połączonego z głośnikiem, a jego moduł wejściowy stanowi **dekoder**. Jako dane wejściowe **DACWrite** otrzymuje adres kanału na który ma wysłać przekonwertowaną do postaci analogowej fale piłokształtną oraz komendę odpowiedzialną za rodzaj wykonywanej operacji. Procesem przesyłu danych steruje impuls Start, który przyjmując wartość 1 zatrzymuje dane na wejściach Cmd, Addr i DATA oraz rozpoczyna ich szeregowe wysłanie do układu. Czas trwania operacji sygnalizuje wyjście Busy. 10 wyjść modułu przeznaczonych jest do dołączenia do wyprowadzeń zewnętrznych FPGA. Oprócz sygnałów używanych do właściwej komunikacji z przetwornikiem zawierają one szereg innych sygnałów trwale blokujących pozostałe urządzenia na płycie korzystające z magistrali SPI: przetwornik ADC, jego przedwzmacniacz oraz trzy rodzaje pamięci flash.



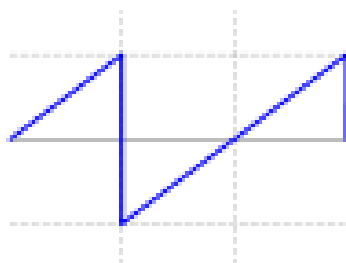
Rys6. Schemat ogólny modułu DACWrite

Organy

Oprócz modułów zewnętrznych do projektu zostały dołączone także pliki o rozszerzeniu ucf – GenIO i ADC_DAC, które oprócz przypisania sygnału do wyprowadzenia często podają szereg parametrów trybu pracy WE/WY (wydajności prądowe, rezystory pull-up / pull-down etc.). Plik GenIO odpowiada za sterowanie Clk_50MHz, BTN_*, SW(3:0), ROT_*, LED(7:0), PS/2, VGA, RS-232 natomiast zadaniem ADC_DAC jest sterowanie przetwornikami ADC / DAC.

2.2. Opis funkcjonalny modułu „dekoder”

Jest to odrębny moduł napisany przez autorów niniejszego dokumentu. Moduł ten korzysta z zewnętrznych bibliotek obsługujących wyrażenia logiczne, arytmetyczne oraz inne standardowe, potrzebne do prawidłowego funkcjonowania modułu. Posiada on 6 sygnałów wejściowych (*DO_RDY*, *Clk*, *Rst*, *E0*, *F0*, *DO*) oraz 7 sygnałów wyjściowych (*Start*, *Cmd*, *Addr*, *DATA*, *Reset*, *key*, *keyPushed*). Sygnały te wykorzystywane są w komunikacji między modułami **PS2_Kbd**, **DACWrite** oraz **wyświetlacz**. Sygnał *DO* dostarcza informacji o kodzie naciśniętego klawisza na klawiaturze podłączonej do portu **PS2**. Innymi sygnałami z tego modułu są *F0* i *E0*, którym zostaje nadana wartość gdy naciśnięto klawisz specjalny, albo puszczono dowolnie jaki klawisz. Sygnały wyjściowe przesyłają informacje dla pozostałych dwóch modułów, w celu odgrywania dźwięku oraz wyświetlania animacji na monitorze. Ponadto zdefiniowane zostały stany w których znajdować się będzie automat MOORE’a zaimplementowany w tym module. Zdefiniowane zostały też sygnały tymczasowe jak na przykład 8 bitowy licznik, który wykorzystywany jest przy przesyłaniu sygnału *DATA* na wejście przetwornika DAC. 8 bitowy iterator użyty przy generacji fali piłokształtnej czy 9 bitowa wartość częstotliwości dźwięku unikalna dla każdego z nich w oktawie. Zadeklarowano także licznik zliczający takty zegara, co wykorzystane jest do odpowiedniej modulacji dźwięku. Za pomocą odpowiednich sygnałów dźwięk zostaje stopniowo ścisany, co ma imitować realistyczne wybrzmiewanie dźwięku odgrywanego na prawdziwych organach.



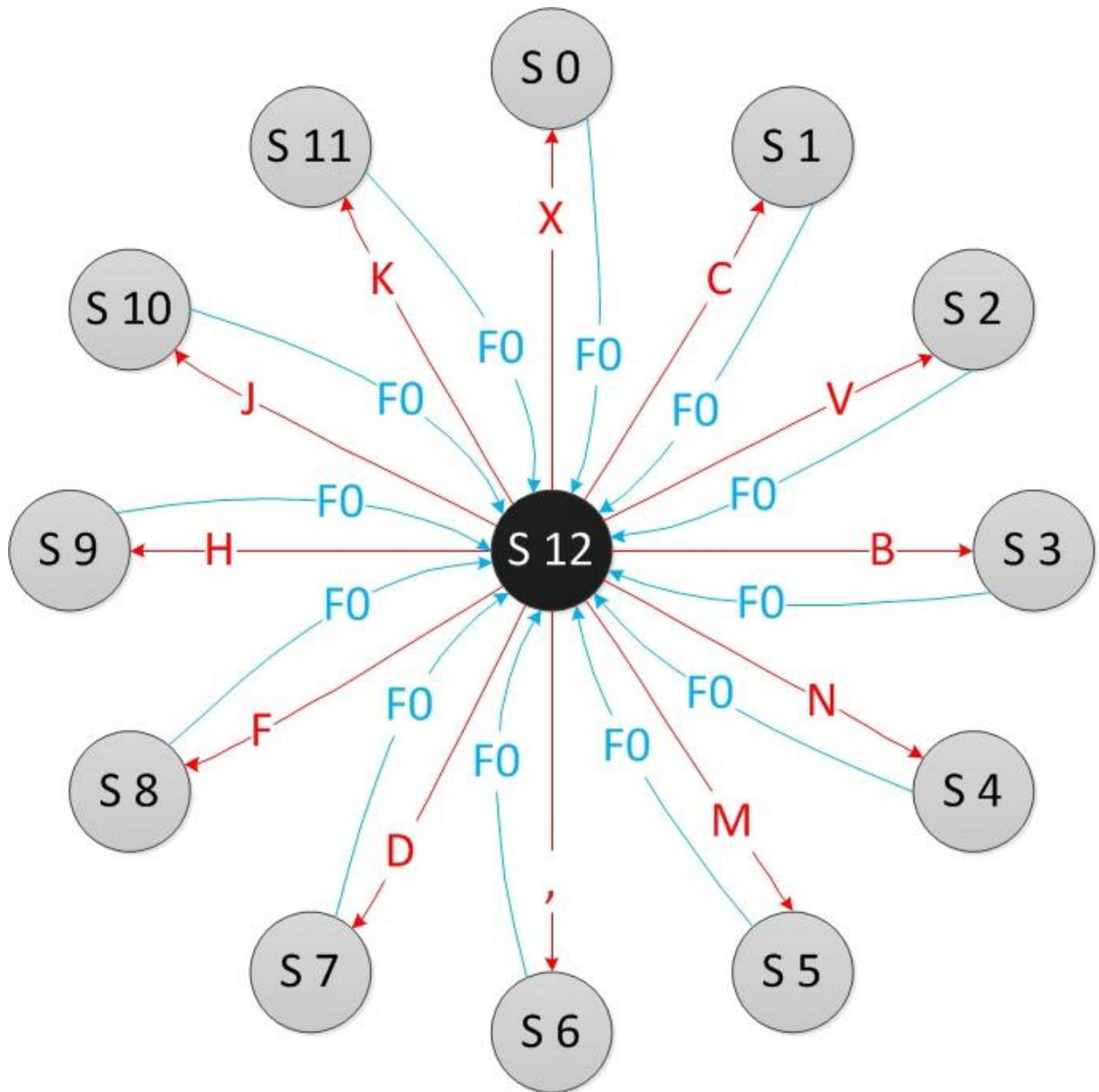
Rys7. Idea fali piłokształtnej

Organy

Process1 przedstawia przechodzenie przez kolejne stany automatu w zależności od narastającego zbocza zegara oraz wartości specjalnych $F0$ i $E0$ dostarczonych przez moduł **PS2_Kbd**.

W procesie następnym na podstawie stanu w którym się znajdujemy, oraz odpowiedniej wartości sygnału DO dostarczonej przez moduł klawiatury akcja odgrywa się zgodnie z zamieszczoną poniżej maszyną stanów.

Organy



Rys8. Maszyna stanów automatu zastosowanego w projekcie

Maszyna startuje z jałowego stanu s_{12} . W stanie tym oczekuje się na naciśnięcie jednego z określonych klawiszy, przykładowo klawisz „x” o kodzie binarnym: **00100010**, powoduje przejście maszyny do stanu s_0 , a puszczenie klawisza, ustawia maszynę na stan jałowy. Zastosowana została tutaj oktawa dwukreślna, aby częstotliwość odgrywanych dźwięków była przyjazna dla ucha.

		klawisz	hex	bin	f	f in program	f in program
--	--	---------	-----	-----	---	--------------	--------------

Organy

						- dec	- bin
1	c^2	x	22	00100010	523,25	373	101110101
2	cis^2	c	21	00100001	554,4	352	101100000
3	d^2	v	2a	00101010	587,3	333	101001100
4	dis^2	b	32	00110010	622,3	314	100111001
5	e^2	n	31	00110001	659,3	296	100101000
6	f^2	m	3a	00111010	698,5	280	100010111
7	fis^2	,	41	01000001	740	264	100000111
8	g^2	d	23	00100011	784	249	011111001
9	gis^2	f	2b	00101011	830,6	235	011101011
10	a^2	h	33	00110011	880	222	011011101
11	b^2	j	3b	00111011	932,3	209	011010001
12	h^2	k	42	01000010	987,8	198	011000101

Tab1. Wartości poszczególnych zmiennych zależnych od klawiszy i przydzielonych im częstotliwości z oktawy dwukreślonej

Powyższa tabela przedstawia wszystkie wartości które pojawiają się w module **dekoder2**. Obliczenia te zostały wykonane na podstawie prostych wzorów fizycznych i zależności. Poniższe obliczenia przedstawiają proces obliczenia wartości wykorzystanej w programie (*f in program*) dla dźwięku c^2 .

$$f = 523,25 \text{ Hz}$$

$$T = \frac{1}{f} = 0,0019111323 \dots s = 1911132,3 \dots ns$$

$$T = n * 20ns$$

$$n = \frac{T}{20} = \frac{1911132,3}{20} = 95556,615$$

$$f(\text{in program}) = \frac{95556,615}{256} = 373$$

W ostatniej linijce obliczeń wartość n została podzielona przez 256, ponieważ wysyłanych jest 2^8 sample'i tego dźwięku. Format wyjściowy *DATA* jest 12 bitowy, dlatego dopełniać należy ten sygnał czterema zerami (0000).

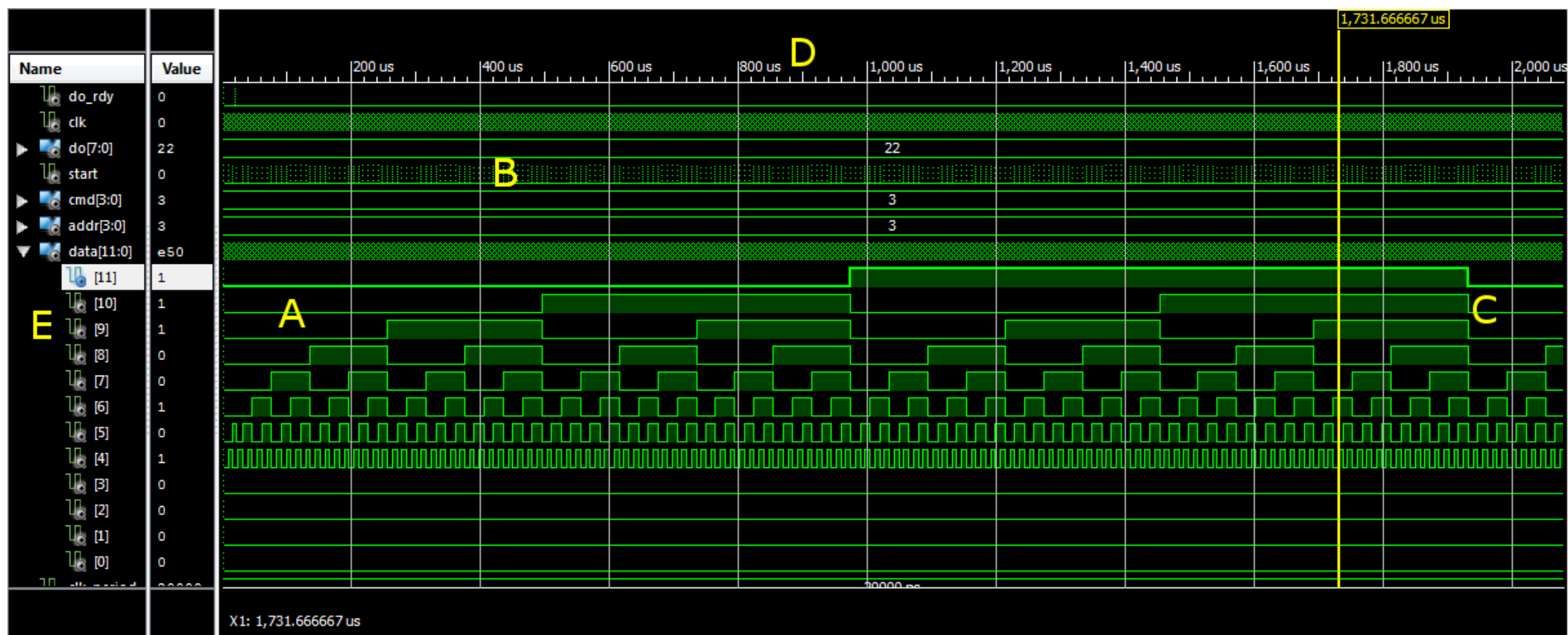
Process 3 – ta część modułu odpowiedzialna jest za wysyłanie fali piłokształtnej na przetwornik **DAC**, a w momencie puszczenia klawisza uruchamiana zostaje procedura ściszenia dźwięku. Przez pierwsze 0.3 sekundy odgrywany zostaje ten sam sygnał, następnie w okresie czasu 0.3 – 0.5 dzięki zastosowaniu konkatencji czterech zer, z sygnałem *tmp* przechowującym wartość aktualnie wysyłaną na przetwornik cyfrowo-analogowy, uzyskano obniżenie tonowe dźwięku. Kolejnymi przedziałami czasowymi są: 0.5 – 0.7, 0.7 – 0.9, 0.9 –

Organy

1.1. W przypadku odmierzenia okresu czasu powyżej 1.1 sekundy, odgrywanie dźwięku zostaje zatrzymane.

```
DATA <=tmp&"0000";           -- 0.0 – 0.3
DATA <="0"&tmp&"000"; -- 0.3 – 0.5
DATA <="00"&tmp&"00"; -- 0.5 – 0.7
DATA <="000"&tmp&"0"; -- 0.7 – 0.9
DATA <="0000"&tmp;           -- 0.9 – 1.1
DATA <="0000000000000000"; -- > 1.1
```

Architektura **dekoder2** kończy się 3 poleceniami, które definiują odpowiednio kanał **D** przetwornika **DAC**, co zrealizowane zostało przez wysłanie sygnału `Addr` o wartości `0011` a do modułu graficznego wysyłana zostaje informacja o tym, czy rozpocząć animację na monitorze.



Rys9. Przebieg czasowy test-bencha dla modułu *dekoder2*.

Przedstawiony powyżej przebieg czasowy charakteryzuje symulowane działanie modułu *dekoder2* na przedziale 2 ms. Zastosowana sekwencja testowa zakładała zmianę sygnału DO_RDY co 1 mikrosekundę i inkrementowana była wartość co kolejne 20 mikrosekund.

A – przedstawia inkrementację sygnału DATA, który będzie w pełnej implementacji wysyłany na wejście przetwornika DAC. Przedstawiona tutaj narastająca fala symbolizuje falę piłokształtną, którą należało uzyskać aby poprawnie odgrywany był dźwięk.

B – widzimy tutaj sygnały Start ustawiane za każdym razem gdy wysyłana ma być informacja na przetwornik DAC. Bez takiej charakterystyki, niemożliwe byłoby odgrywanie dźwięku. Otrzymując sygnał Start a następnie sprawdzając wartość sygnału DATA, odgrywany jest przez ułamek sekundy dźwięk. Odpowiednio częste ustawianie i zmienianie tych wartości skutkuje płynnym odgrywaniem dźwięku oraz, odpowiednią wysokością tonów.

C – punkt ten przedstawia sytuację, gdy licznik osiągnął zadeklarowane maksimum dla tego dźwięku, tym samym zostaje on zerowany, a fala piłokształtna od początku zaczyna być formowana,

D – skala czasu narastająca od strony lewej do prawej, pozwala na śledzenie stanu maszyny i układu w odpowiednich momentach czasu,

E – legenda sygnałów ustawianych, rozwinięty w tym przypadku został sygnał data w celu zobrazowania tworzenia się fali piłokształtnej.

2.3. *Opis funkcjonalny modułu „wyswietlacz”*

Moduł „wyswietlacz” odpowiedzialny jest za komunikację pomiędzy modulem „dekoder” a portem wyświetlacza VGA do którego podłączony jest monitor. Monitor otrzymujący poprzez port VGA odpowiednie dane wyświetla zmapowaną w postaci klawiatury organów/fortepianu klawiaturę dołączoną do płyty oraz jej aktualnie naciśnięty klawisz. Moduł „dekoder” współpracujący bezpośrednio z klawiaturą podłączoną do płyty dostarcza modułowi „wyświetlacz” informacji o tym czy klawisz jest naciśnięty i który to klawisz. Informacja ta dostarczana jest do modułu „wyświetlacz” w postaci sygnałów wejściowych: 4-bitowego wektora `key` typu `STD_LOGIC_VECTOR` przekazującego informację o kodzie naciśniętego klawisza oraz sygnału `keyPushed` typu `STD_LOGIC` określającego czy dany klawisz jest naciśnięty – sygnał przyjmuje wartość 1 w przypadku naciśnięcia klawisza lub wartość 2 gdy klawisz nie jest naciśnięty. Kody do naciskanych na klawiaturze klawiszy przekazywanych modułowi przydzielono w następujący sposób:



W skład sygnałów wyjściowych modułu wchodzi sygnały odpowiedzialne za współpracę z wyświetlaczem VGA: dwa sygnały typu `STD_LOGIC` służące do synchronizacji poziomej (`hsync`) oraz pionowej (`vsync`) wyświetlacza oraz 3-bitowy wektor `rgb` typu `STD_LOGIC_VECTOR` określający w jakim kolorze ma być wyświetlany dany piksel na ekranie. Modelem przestrzeni barw jest model RGB, a dany kolor tworzony jest poprzez zestawienie trzech barw – czerwonej (R), zielonej (G) i niebieskiej (B). Barwy te reprezentowane są w wektorze `rgb` z kolejnością zgodną z nazwą modelu, a więc najstarszy bit wektora odpowiada za barwę czerwoną, środkowy za barwę zieloną, a najmłodszy za barwę niebieską. Dzięki odpowiedniemu zestawianiu barw, a więc przesłaniu odpowiednich wartości w wektorze `rgb` możliwe jest uzyskanie na wyświetlaczu VGA następujących kolorów:

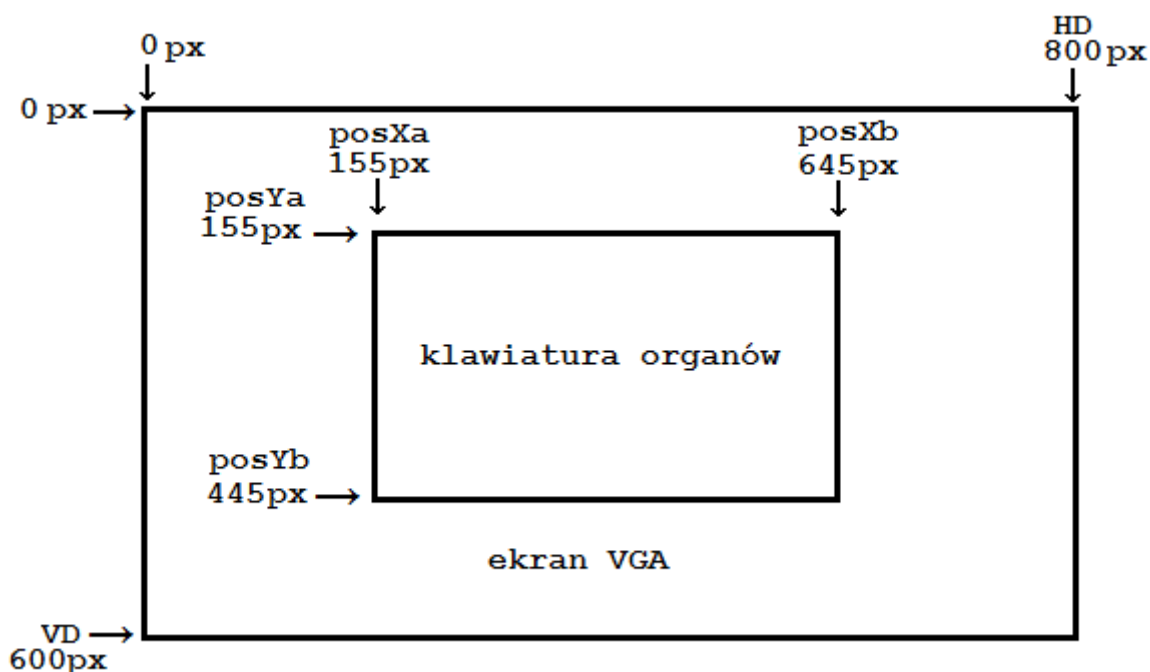
Tab2. Kolory możliwe do uzyskania w module VGA

Organy

Moduł „wyświetlacz” oprócz sygnałów wejściowych i wyjściowych posiada szereg stałych oraz sygnałów wewnętrznych niezbędnych do poprawnej obsługi wyświetlacza VGA. Zmienne zadeklarowane w module – HD (horizontal display) oraz VD (vertical display) to wektory 10-bitowe typu STD_LOGIC_VECTOR określające obszar wyświetlania obrazu, a więc rozdzielczość ekranu. Rozdzielczość ustalona jest na 800 pikseli poziomo i 600 pikseli pionowo. Jest to rozdzielczość zgodna ze standardem SVGA, o częstotliwości odświeżania 72Hz i czasowi dostępności jednego piksela równemu cyklowi zegara (50MHz).

Sygnały wewnętrzne zadeklarowane w module to: 10-bitowe sygnały v_count_reg, v_count_next typu STD_LOGIC_VECTOR oraz 11-bitowe sygnały h_count_reg, h_count_next typu STD_LOGIC_VECTOR pełniące funkcję liczników synchronizujących modułu, sygnały v_sync_reg, h_sync_reg, v_sync_next, h_sync_next typu STD_LOGIC będące buforami wyjściowymi, sygnały h_end, v_end typu STD_LOGIC określające status sygnału, sygnał video_on określający czy ekran jest włączony czy wyłączony, a także wektory 10-bitowe pixel_x, pixel_y typu STD_LOGIC_VECTOR określające położenie aktualnego piksela w poziomie (pixel_x) i pionie (pixel_y) oraz posXa, posXb, posYa, posYb określające położenie na ekranie wyświetlanej klawiatury od punktu (0,0) będącego lewym górnym rogiem ekranu, gdzie posXa i posYa – pozycje początkowe (w poziomie – 155 pikseli i pionie – 155 pikseli) od których rozpoczyna się rysowanie klawiatury, posXb i posYb pozycje końcowe (w poziomie – 645 pikseli i pionie – 445 pikseli) na których rysowanie klawiatury się kończy. Podział ten przedstawia poniższy rysunek:

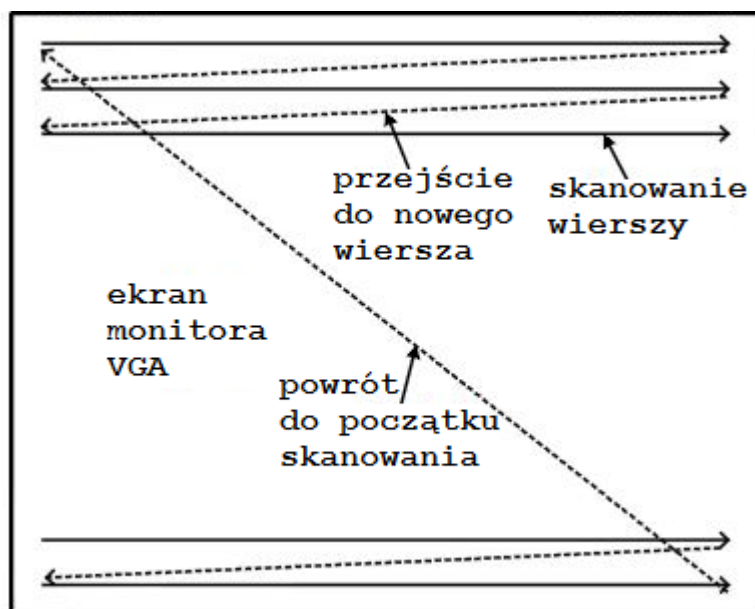
Organy



Rys11. Graficzne zobrazowanie problemu rozmieszczenia klawiatury

Idea działania modułu „wyświetlacz” opiera się na generowaniu przez kontroler VGA sygnałów synchronizujących w poziomie oraz w pionie, które koordynują dostarczanie danych wideo (informacja czy włączyć piksel i jaki kolor mu nadać) do każdego piksela ekranu. Kontroler w trakcie gdy wiązka elektronów przechodzi przez wyświetlacz indeksuje dane do bufora, a następnie odzyskuje je i przekazuje do wyświetlacza dokładnie w czasie przemieszczania wiązki w miejscu danego piksela. Informacja przekazywana jest jedynie wtedy, gdy wiązka porusza się od lewej do prawej i z góry na dół, co wiąże się z sekwencyjnością rysownia pikseli począwszy od lewego górnego rogu ekranu. W pozostałych przypadkach, a więc gdy wiązka wraca do lewej strony ekranu bądź kieruje się ku górze po osiągnięciu dolnej krawędzi informacja przekazywana nie jest. Pokazuje to schemat na poniższym rysunku.

Organy

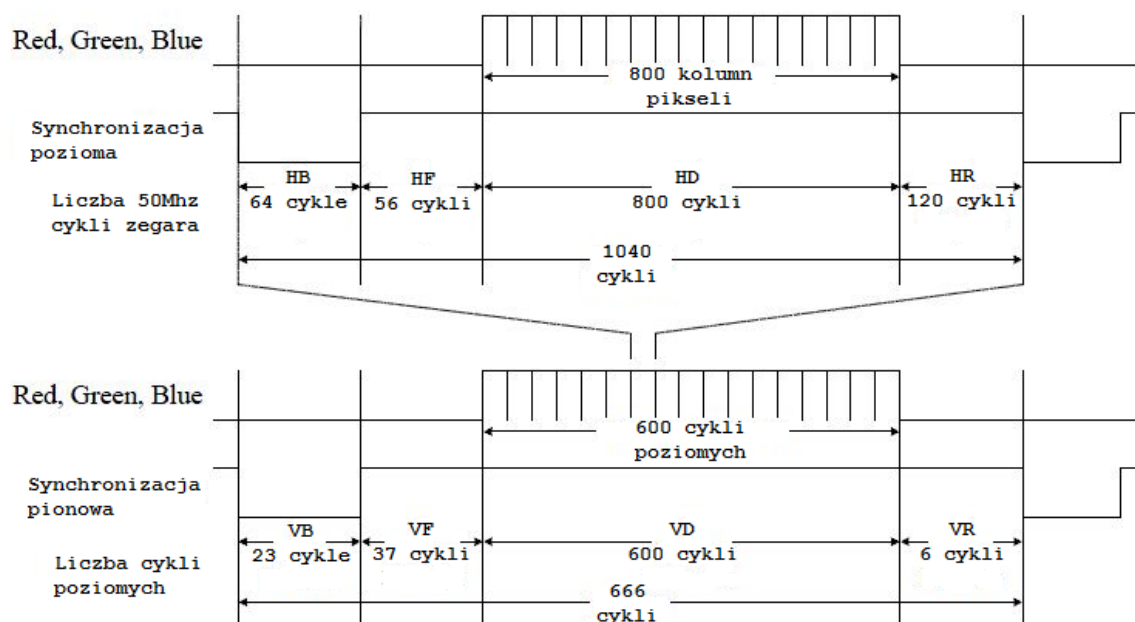


Rys12. Graficzne przedstawienie synchronizacji wertykalnej i horyzontalnej

Dzięki sygnałom synchronizującym możliwe jest zdeterminowanie czasu potrzebnego zarówno na zeskanowanie jednego wiersza (synchronizacja pozioma) jak i całego ekranu (synchronizacja pionowa) co określa tym samym rozdzielczość ekranu. Przyjmując iż wiązka elektronów przechodzi z jednego piksela na drugi w czasie równym cyklowi zegara (50MHz), wystarczy jedynie zliczać ilości cykli w rzędach oraz kolumnach uzyskując współrzędne (x,y) wyświetlanego w danym momencie czasu punktu, co umożliwia w łatwy i przystępny sposób jego manipulację.

Ważnym aspektem w procesie obsługi wyświetlacza VGA jest fakt iż, oprócz aktywnego ekranu na którym wyświetlane są piksele występuje także obszar gdzie piksele wyświetlane nie są (ekran nieaktywny). Istnienie tego obszaru jest o tyle istotne iż jest on wymagany do poprawnej synchronizacji, a więc powrotów wiązki elektronów do nowego wiersza wyświetlacza, bądź do początku wyświetlanego obszaru. Ekran nieaktywny ustalany jest za pomocą wyliczonych zgodnie z przyjętą rozdzielczością, częstotliwością odświeżania oraz czasem dostępności jednego piksela zmiennych, po których przekroczeniu sumy dla synchronizacji poziomej bądź pionowej dane wideo nie są przekazywane (ekran jest wyłączany). Wartości tych zmiennych oraz zakres ekranu aktywnego i nieaktywnego prezentuje poniższy rysunek.

Organy



Rys13. Idea synchronizacji, wyświetlania obrazu w zależności od cykli zegara

W przypadku synchronizacji poziomej wiązka elektronów przechodząca przez poszczególne wiersze ekranu znajduje się początkowo poza ekranem przechodząc ze stanu niskiego (HB – horizontal black porch) w stan wysoki (HF – horizontal front porch). Następnie przez czas 800 cykli zegara (HD – horizontal display area) do poszczególnych pikseli zostają przekazywane dane wideo – ekran aktywny. Po przejściu przez ostatnią kolumnę pikseli wiązka trafia ponownie do ekranu nieaktywnego gdzie następuje przejście do nowego rzędu pikseli (HR – horizontal retrace) i ponowne wejście w stan niski. Analogiczna sytuacja występuje w przypadku synchronizacji pionowej z tym, zliczaniu nie podlegają cykle zegara, a cykle przejść wiązki elektronów przez pojedynczy wiersz.

Powyższa ideologia została zrealizowana w module „wyswietlacz” w oparciu o cztery procesy oraz statyczne operacje przypisywania występujące poza nimi. Pierwszy z procesów jest licznikiem, który w przypadku pojawienia się ustawionego na wartość 1 sygnału resetującego (Reset) przypisuje licznikom zliczającym cykle zegara (`v_count_reg`, `h_count_reg`) oraz buforom (`v_sync_reg`, `h_sync_reg`) służącym lepszej synchronizacji wyświetlania i unikaniu błędów wartość 0. W przypadku napotkania zbocza narastającego zegara do liczników oraz buforów przypisywane są ich następne stany (`v_count_next`, `h_count_next`, `v_sync_next`, `h_sync_next`).

W procesie drugim następuje zliczanie licznika poziomego. Licznik poziomy zliczany jest aż do momentu osiągnięcia przez wiązkę elektronów końca wiersza, a więc do wartości 1040. Sytuacja ta zaznaczana jest w module poprzez przypisanie wartości 1 zmiennej `h_end`.

Organy

Dzięki temu sprawdzając przypisywaną poza procesami wartość h_end można ograniczyć zliczanie licznika do pożądanej wartości. W przypadku gdy h_end wynosi 1, a więc osiągnięto koniec wiersza licznik jest zerowany. W przeciwnym wypadku wartość jego zwiększana jest o 1.

Analogicznym procesem jest proces trzeci wyliczający licznik pionowy. Dla procesu tego ważne jest jaką wartość posiada h_end . Jeżeli h_end jest równe 0, a więc wiązka nie osiągnęła końca wiersza licznik poziomy się nie zmienia, w przeciwnym wypadku w zależności wyliczonej poza procesami wartości v_end - czy przekroczono zakres poziomego licznika, jest on zwiększany o 1 (v_end równe 0) bądź zerowany (v_end równe 1).

Ostatnim procesem jest proces w którym danym pikselom ekranu przekazywane są określone dane wideo, a więc proces określający co ma być wyrysowane na ekranie. W tym celu ustawiane są poza procesami współrzędne określające położenie klawiatury na ekranie (lewy górny róg i prawy dolny róg), a także określane jest czy mowa o ekranie aktywnym ($video_on$ równe 1 gdy wartość licznika poziomego mniejsza od HD i wartość licznika pionowego mniejsza od VD) oraz określenie odpowiedniej synchronizacji wyświetlania w poziomie (pomiędzy $HD+HF$ i $HD+HF+HR-1$) i w pionie (pomiędzy $VD+VF$ i $VD+VF+VR-1$). Początkowo sprawdzane jest czy ekran jest nieaktywny ($video_on$ równe 0). Jeśli tak to wypełniany jest on czarnym kolorem – wszystkie piksele są gaszone. W przypadku gdy ekran jest aktywny następuje rysowanie zielonego tła klawiatury. Następnie śledząc liczniki poziomy i pionowy wskazujące na aktualny piksel rysowane są warstwowo poszczególne części klawiatury. Na początku jest to czarne obramowanie, wypełniane następnie białym tłem. Tło zostaje podzielone sześcioma pionowymi, czarnymi kreskami w wyniku czego powstaje 7 białych klawiszy („x”, „c”, „v”, „b”, „n”, „m”, „”), do których jako ostatni element dorysowywanych jest 5 klawiszy czarnych („d”, „f”, „g”, „h”, „j”). W ten sposób uzyskana zostaje statyczna klawiatura organów mapująca klawiaturę komputera. Aby uzyskać obraz klawiatury zmieniającej się pod wpływem naciśnięcia klawisza, do klawiatury statycznej należy dodać kolejną z warstw rysowania. Wariant warstwy uruchamiane jest w zależności od tego czy klawisz jest wciśnięty ($keyPushed$ równe 1) i który to klawisz (4-bitowy wektor key przechowujący kod klawisza). Standardowa warstwa dynamicznej klawiatury polega na opuszczeniu klawisza poprzez zamalowanie na zielono (wypełnienie tła) części z której klawisz się przesunął do części którą klawisz zasłonił przy uwzględnieniu przesunięcia obramowania i domalowania klawiszy które mogły zostać zasłonięte (część klawisza). Po puszczeniu klawisza jako że statyczna klawiatura zawsze jest rysowana (warunkiem aktywny ekran) obraz dynamicznie się zmienia.

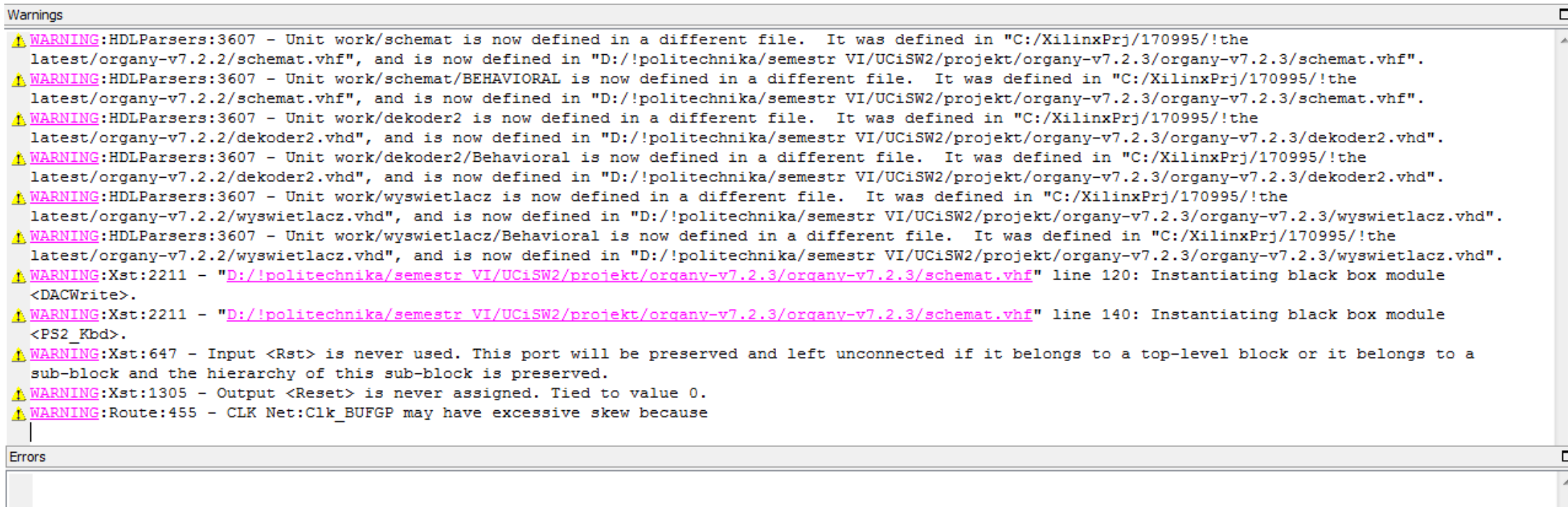
Symulacja modułu „wyświetlacz” z racji tego, iż kluczowe dla śledzenia aktualnego piksela, a więc rysowania na ekranie, liczniki były sygnałami wewnętrznymi (trudność przypisywania wartości w teście), a wyrysowywany obraz tworzony był na zasadzie warstwowej (nadpisywanie poszczególnych części warstw) symulacja modułu nie była przeprowadzana na zasadzie tworzenia plików testowych i ich symulacji w środowisku ISim, lecz w postaci bezpośredniego weryfikowania obrazu wyświetlanego na ekranie monitora podłączonego do wyświetlacza VGA. W przypadku błędów związanych z niedopasowaniem poszczególnych wyrysowywanych warstw następowała korekta w kodzie modułu i ponowne sprawdzanie uzyskiwanych rezultatów na ekranie monitora.

3. Implementacja

Aplikacja została napisana w języku VHDL w środowisku programistycznym Xilinx ISE Design Suite 12.4 pod systemem Windows 7 32-bit. Testowanie aplikacji odbywało się w sali laboratoryjnej Politechniki Wrocławskiej, w których znajdowało się IDE Xilinx ISE Design Suite 12.3. **Organy** były projektowane na platformę sprzętową Spartan-3E Starter firmy Xilinx, toteż testy nie mogłyby odbywać się bez jej udziału. W celu szybkiej wymiany danych, plików, oraz łatwej komunikacji między twórcami, wykorzystano program kontroli wersji *SVN Tortoise*. Za jego pomocą skutecznie archiwizowana była praca, dokonywana na poszczególnych etapach.

Miarą poprawnej implementacji danego projektu poza prawidłowością jego działania jest także jakość wygenerowanego raportu końcowego. Poniższy rysunek przedstawia zbiór wyników ostrzeżeń (*Warnings*) oraz błędów (*Errors*). Ilość ostrzeżeń na pierwszy rzut oka może nasuwać pytanie, dlaczego tak dużo? Jednakże pierwszych sześć ostrzeżeń dotyczy redefinicji modułów, które raz został zdefiniowane w sali laboratoryjnej, natomiast materiały do dokumentacji były uzyskiwane na platformach prywatnych, na których ścieżki dostępowe tych plików się różnią (co widać w opisach tych zdarzeń). Kolejne dwa ostrzeżenia dotyczą dołączania czarnych skrzynek z zewnątrz do aplikacji (DACWrite, PS2_Kbd). Otrzymano też informację zwrotną, że sygnały w założeniu mające resetować maszynę i moduł VGA (Rst, Reset) zostały zadeklarowane, ale nie użyte. Świadomie zostały one nie oznaczane, bo były niepotrzebne, jednakże w dalszej rozbudowie tej aplikacji, tudzież odrębnych modułów, zostawiono „otwartą furtkę” i możliwość oprogramowania tychże sygnałów. Najważniejszą jednak częścią jest druga część tego obrazka z nagłówkiem **Errors**. Mówi ona o tym, jakie krytyczne błędy występują w projekcie. Podstawową sprawą jest fakt, że jakkolwiek błąd, powoduje zatrzymanie syntezy kodu VHDL, co skutkuje brakiem generacji pliku programowalnego, czyli „układ nie działa”. Można też zauważyć że moduły zostały poprawnie napisane, ponieważ nie występują żadne zatraski. Oznacza to, że nie ma fizycznej możliwości, aby maszyna stanów znalazła się w stanie, przez projektantów nie określonym, albo nieprzewidzianym.

Organy



Warnings

- WARNING:**HDLParasers:3607 - Unit work/schemat is now defined in a different file. It was defined in "C:/XilinxPrj/170995/!the latest/organy-v7.2.2/schemat.vhf", and is now defined in "D:/!politechnika/semestr VI/UCiSW2/projekt/organy-v7.2.3/organy-v7.2.3/schemat.vhf".
- WARNING:**HDLParasers:3607 - Unit work/schemat/BEHAVIORAL is now defined in a different file. It was defined in "C:/XilinxPrj/170995/!the latest/organy-v7.2.2/schemat.vhf", and is now defined in "D:/!politechnika/semestr VI/UCiSW2/projekt/organy-v7.2.3/organy-v7.2.3/schemat.vhf".
- WARNING:**HDLParasers:3607 - Unit work/dekoder2 is now defined in a different file. It was defined in "C:/XilinxPrj/170995/!the latest/organy-v7.2.2/dekoder2.vhd", and is now defined in "D:/!politechnika/semestr VI/UCiSW2/projekt/organy-v7.2.3/organy-v7.2.3/dekoder2.vhd".
- WARNING:**HDLParasers:3607 - Unit work/dekoder2/Behavioral is now defined in a different file. It was defined in "C:/XilinxPrj/170995/!the latest/organy-v7.2.2/dekoder2.vhd", and is now defined in "D:/!politechnika/semestr VI/UCiSW2/projekt/organy-v7.2.3/organy-v7.2.3/dekoder2.vhd".
- WARNING:**HDLParasers:3607 - Unit work/wyswietlacz is now defined in a different file. It was defined in "C:/XilinxPrj/170995/!the latest/organy-v7.2.2/wyswietlacz.vhd", and is now defined in "D:/!politechnika/semestr VI/UCiSW2/projekt/organy-v7.2.3/organy-v7.2.3/wyswietlacz.vhd".
- WARNING:**HDLParasers:3607 - Unit work/wyswietlacz/Behavioral is now defined in a different file. It was defined in "C:/XilinxPrj/170995/!the latest/organy-v7.2.2/wyswietlacz.vhd", and is now defined in "D:/!politechnika/semestr VI/UCiSW2/projekt/organy-v7.2.3/organy-v7.2.3/wyswietlacz.vhd".
- WARNING:**Xst:2211 - "D:/!politechnika/semestr VI/UCiSW2/projekt/organy-v7.2.3/organy-v7.2.3/schemat.vhf" line 120: Instantiating black box module <DACWrite>.
- WARNING:**Xst:2211 - "D:/!politechnika/semestr VI/UCiSW2/projekt/organy-v7.2.3/organy-v7.2.3/schemat.vhf" line 140: Instantiating black box module <PS2_Kbd>.
- WARNING:**Xst:647 - Input <Rst> is never used. This port will be preserved and left unconnected if it belongs to a top-level block or it belongs to a sub-block and the hierarchy of this sub-block is preserved.
- WARNING:**Xst:1305 - Output <Reset> is never assigned. Tied to value 0.
- WARNING:**Route:455 - CLK Net:Clk_BUFPG may have excessive skew because

Errors

Rys14. Ostrzeżenia i błędy wygenerowane podczas syntezy projektowanego układu

Organy

Wygenerowany końcowy raport sumaryczny został przedstawiony obok. Na podstawie tego dokumentu można stwierdzić ile dokładnie zasobów zostało wykorzystanych w celu realizacji tego projektu. Tak więc 4 sumatory, 1 licznik, 10 rejestrów oraz 68 elementów porównujących dane zostało użyte w celu realizacji zadania wyżej opisanego.

Poniższa część raportu (*Device utilization summary*) przedstawia ile kłastrów (*Slices*), przerzutników (*Flip Flop*) oraz innych zasobów dostępnych na platformie Xilinx Spartan-3E zostało wykorzystanych. W ostatniej kolumnie przedstawione zostało procentowe użycie tychże elementów w stosunku do ich liczby globalnej.

```
HDL Synthesis Report

Macro Statistics
# Adders/Subtractors      : 4
  11-bit adder             : 1
  27-bit adder             : 1
  8-bit adder              : 1
  9-bit adder              : 1
# Counters                 : 1
  10-bit up counter        : 1
# Registers                : 10
  1-bit register           : 5
  11-bit register          : 1
  12-bit register          : 1
  27-bit register          : 1
  8-bit register           : 1
  9-bit register           : 1
# Comparators              : 68
  10-bit comparator greater : 19
  10-bit comparator less    : 20
  11-bit comparator greatequal : 2
  11-bit comparator greater : 3
  11-bit comparator less    : 8
  11-bit comparator lessequal : 2
  27-bit comparator greatequal : 5
  27-bit comparator less    : 5
  9-bit comparator equal     : 1
  9-bit comparator greatequal : 1
  9-bit comparator less     : 1
  9-bit comparator not equal : 1
```

Rys15. Raport syntezy kodu HDL

```
Device utilization summary:
-----
```

```
Selected Device : 3s500efg320-5
```

Number of Slices:	481	out of	4656	10%
Number of Slice Flip Flops:	182	out of	9312	1%
Number of 4 input LUTs:	854	out of	9312	9%
Number of IOs:	20			
Number of bonded IOBs:	20	out of	232	8%
Number of GCLKs:	1	out of	24	4%

Rys16. Podsumowanie wykorzystanych zasobów

Organy

Zamieszczony poniżej *Print Screen* przedstawia zebrane informacje w stosunku do przebiegów czasowych tego projektu.

```
Clock Information:
-----
```

Clock Signal	Clock buffer (FF name)	Load
Clk	BUFGP	182

```
Asynchronous Control Signals Information:
-----
```

Control Signal	Buffer (FF name)	Load
Rst	IBUF	23

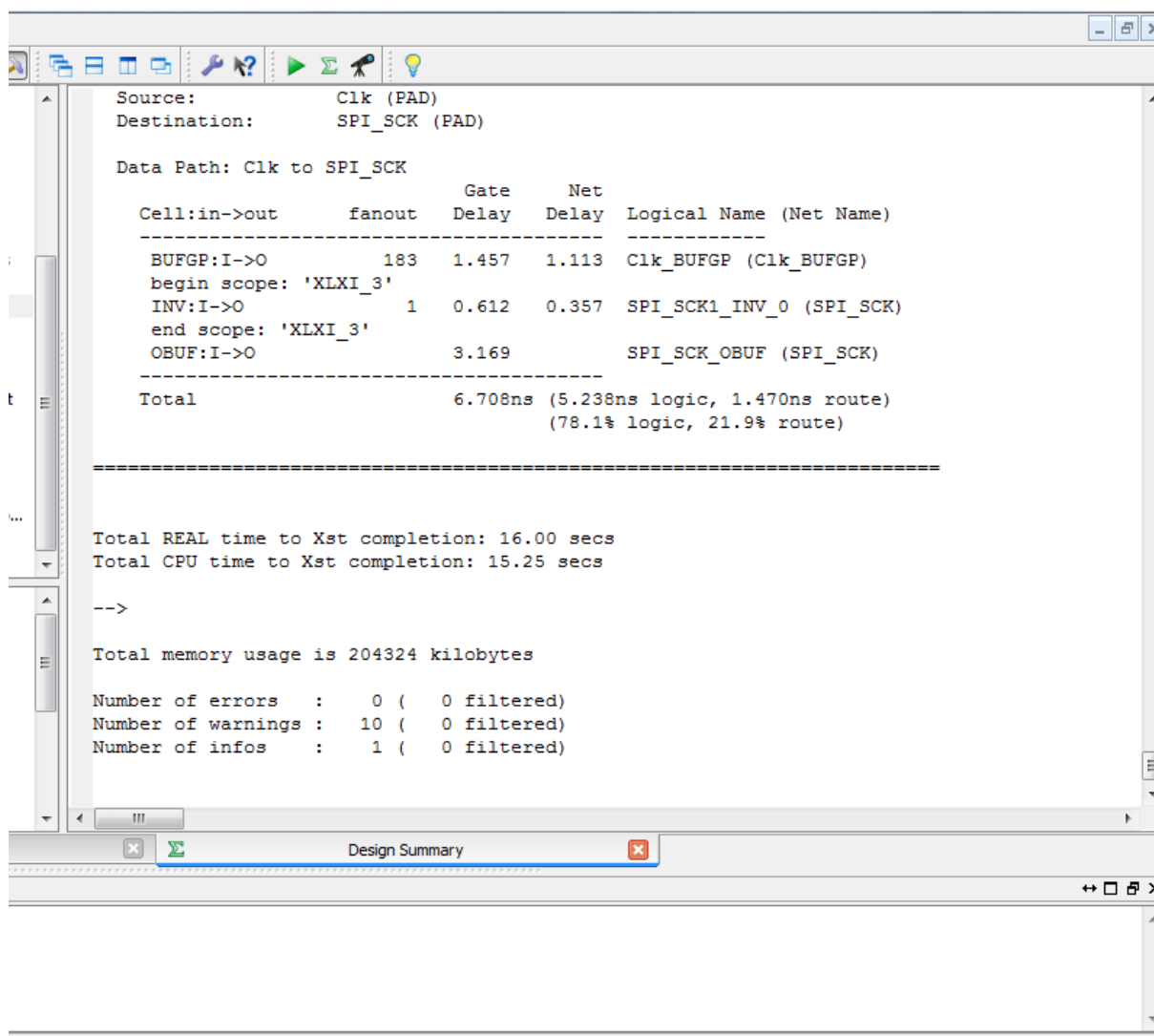
```
Timing Summary:
-----
Speed Grade: -5

Minimum period: 12.284ns (Maximum Frequency: 81.409MHz)
Minimum input arrival time before clock: 3.490ns
Maximum output required time after clock: 19.899ns
Maximum combinational path delay: 6.708ns
```

Rys17. Raport dotyczący danych zegarowych

Kolorem czerwonym zostały zaznaczone najważniejsze informacje. Mianowicie pierwsze od góry patrząc, to minimalne opóźnienie sygnału zegarowego. Im mniejsza jest ta wartość, tym większa staje się szacowana maksymalna częstotliwość pracy układu. W tym przypadku, wartość **81.940** MHz, jest zadowalająca. Wyjściową wartością jest ~50 MHz, ponieważ z taką częstotliwością pracuje zegar umiejscowiony na platformie Spartan. W przypadku pracy w tej częstotliwości, minimalne opóźnienie szacowane jest na **20 ns** stąd też wartość ta pojawiła się we wzorze wcześniejszym przy okazji przeliczania wartości programowych, różnych dźwięków oktawy.

Organy



Rys18. Podsumowanie błędów, ostrzeżeń i zajęcia pamięci przez układ

Dane dotyczące syntezy widoczne powyżej, informują odbiorcę o poprawności działania układu ze względu na ilość ostrzeżeń, błędów, informacji. Jak wyżej zostało to już opisane, w projekcie niniejszym zostało wygenerowanych 10 ostrzeżeń, z których większość można wyeliminować za pomocą przebudowania projektu, albo ponownemu dołączeniu tych modułów. Ostrzeżenia te nie były jakimiś strasznie rażącymi, dlatego nie skupiono się na dokładnej ich eliminacji. Łączna pamięć wykorzystana przez projekt wynosi **204324** kilobajtów.

Praca z układem jest bardzo prosta. Użytkownik ma przed sobą 4 główne elementy: *monitor CRT*, *klawiatura*, platforma *Xilinx Spartan-3E* oraz *głośniczki*. Modułem centralnym

Organy

jest płyta *Spartan*, łączy ona za pomocą odpowiednich portów pozostałe elementy. Pierwszym widokiem jaki widzi użytkownik, po uruchomieniu programu załadowanego do pamięci, jest czysta klawiatura wyświetlona na monitorze w taki oto sposób:



Rys19. Główny widok VGA

Przy naciśnięciu jednego z klawiszy **Organów** zostanie odgrywany dźwięk o specjalnej intonacji. Tabela znajdująca się z boku przedstawia, który klawisz klawiatury za jaki dźwięk jest odpowiedzialny.

Ponadto każdy z klawiszy zwizualizowanych na monitorze jednoznacznie identyfikuje klawisz z klawiatury, tak jak to pokazano poniżej:



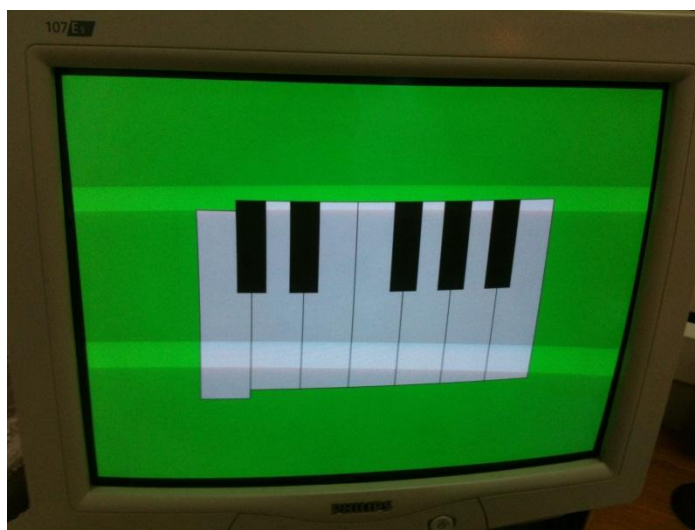
klawisz	Dźwięk
x	c2
c	cis2
v	d2
b	dis2
n	e2
m	f2
,	fis2
d	g2
f	gis2
h	a2
j	ais2
k	h2

Tab3. Każdy klawisz ma przypisany inny ton.

Rys20. Ogólne rozmieszczenie dźwięków na klawiaturze oktawy.

Organy

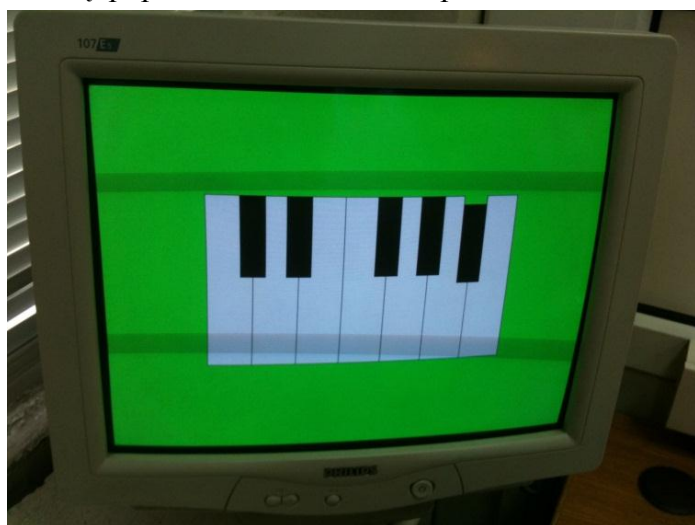
Po złączeniu tych informacji, można przedstawić w prosty sposób sytuację gdy naciśnięty zostaje klawisz „x” z klawiatury zewnętrznej podłączonej do Spartana. Skutkiem tego działania będzie odegranie dźwięku „C²” (ponieważ jest to oktawa dwukreślna, co zostało wyżej wytłumaczone), a na ekranie zobaczymy taki oto obraz:



Rys21. Naciśnięto klawisz „x” odegrany został dźwięk C

Animacja naciśniętego klawisza odbywa się poprzez obniżenie o 15 pikseli w dół prostokąta reprezentującego odpowiedni dźwięk, przy zachowaniu aktualnych pozycji pozostałych z nich. Najlepiej obrazuje to zdjęcie wykonane w laboratorium podczas testowania aplikacji.

Analogicznie wciśnięcie ostatniego klawisza czarnego (licząc od lewej strony) wiąże się z odegraniem dźwięku o częstotliwości b^2 co zwizualizowane zostaje następująco:



Rys21. Naciśnięto klawisz „k” odegrany został dźwięk b

Opisanie jakości odgrywanego dźwięku jest praktycznie nie możliwe, zwłaszcza że zależne to jest też od jakości buzzera dostarczonego przez wykładowcę. Na każdym głośniczku dźwięki te są w pewien sposób inaczej odgrywane. Jednakże zachowana jest skala oraz hierarchia oktawy dwukreślniej, co gwarantuje przyjemne dla ucha dźwięki. W celu lepszego zobrazowania faktycznych możliwości aplikacji do projektu został dołączony film promujący – instruktażowy (*spartan3-promo*), w którym wyjaśnione zostały praktyczne podstawy działania tego projektu, oraz zobrazowane zostało działanie wszystkich modułów razem połączonych.

4. Podsumowanie

Realizacja projektu organów w ramach zajęć z kursu „Układy Cyfrowe i Systemy Wbudowane 2 - projekt” powiodła się. Kluczem do poprawnej realizacji okazała się systematyczność i podział pracy na mniejsze części. Określenie pojedynczych podproblemów wraz z maksymalnym czasem potrzebnym na ich realizację pozwoliły na komfortową pracę i dotrzymywanie wyznaczonych terminów. Dzięki realizacji mniejszej części materiału łatwiej i szybciej można było rozwiązywać napotkane trudności i weryfikować ich rozwiązanie bezpośrednio w postaci działających modułów. Odpowiednie przygotowanie do zajęć projektowych, nie tylko pod względem merytorycznym ale również praktycznym, w postaci sporządzanego poza zajęciami projektowymi kodu VHDL, pozwoliło na maksymalne wykorzystanie dostępnego na zajęciach czasu na pracę na sprzęcie i weryfikację sporządzanego kodu w praktyce. Podział pracy pomiędzy członków grupy projektowej zapewniał równomierny wkład w zajęcia projektowe i efektywne wykorzystanie czasu poprzez uniknięcie redundancji rozwiązań tych samych problemów.

Mimo iż projekt został zrealizowany poprawnie jego realizację można było przeprowadzić lepiej. Mowa tu w szczególności o kodzie VHDL, który pod względem składniowym i objętościowym może być zrefaktoryzowany. Inne podejście do problemów jakie niosą ze sobą poszczególne moduły projektu, przejawiające się w stworzeniu bardziej optymalnych procesów występujących w tychże modułach może być drogą do ulepszenia stworzonego projektu. Oprogramowanie nowych klawiszy, co skutkowało by poszerzeniem „możliwości dźwiękowych” stworzonego projektu, a także bardziej realistyczna wizualizacja w formie klawiszy dwu- bądź trójwymiarowych to również godne uwagi ścieżki jakimi można rozwijać projekt. Warto nadmienić ponadto iż niektóre z pomysłów wykraczających poza zakres projektowy udało się zrealizować już podczas pracy nad organami. Wśród nich są m.in. operacja wyciszania klawisza po jego zwolnieniu przez użytkownika imitująca wybrzmiewanie dźwięku granego na prawdziwych organach, a także wizualizacja wciskanego klawisza poprzez obniżenie jego położenia względem pozostałych klawiszy co przy wyświetlaniu klawiatury jako rzut z góry daje złudzenie iż klawisz faktycznie został wciśnięty (podstawowa wersja wyświetlacza zgodna z początkowymi założeniami projektowymi - zaznaczanie wciskanego klawisza prostokątem również została zrealizowana).

5. Bibliografia

- **Spartan-3E FPGA, Starter Kit Board, User Guide, UG230 (v1.2) January 20, 2011,**
- **FPGA Prototyping by VHDL Examples. By Pong P. Chu, 2008 John Wiley & Sons, Inc.**
- **D/A Converter Control for Spartan-3E Starter Kit, Ken Chapman, Xilinx Ltd, 21stFebruary 2006**
- **Podstawy elektroniki cyfrowej, Józef Kalisz, Wydawnictwa Komunikacji i Łączności 2002.**

Urządzenia Cyfrowe i Systemy Wbudowane



XILINX®

ORGANY

Wrocław, 2011

Jakub Kałużka
Piotr Kłys