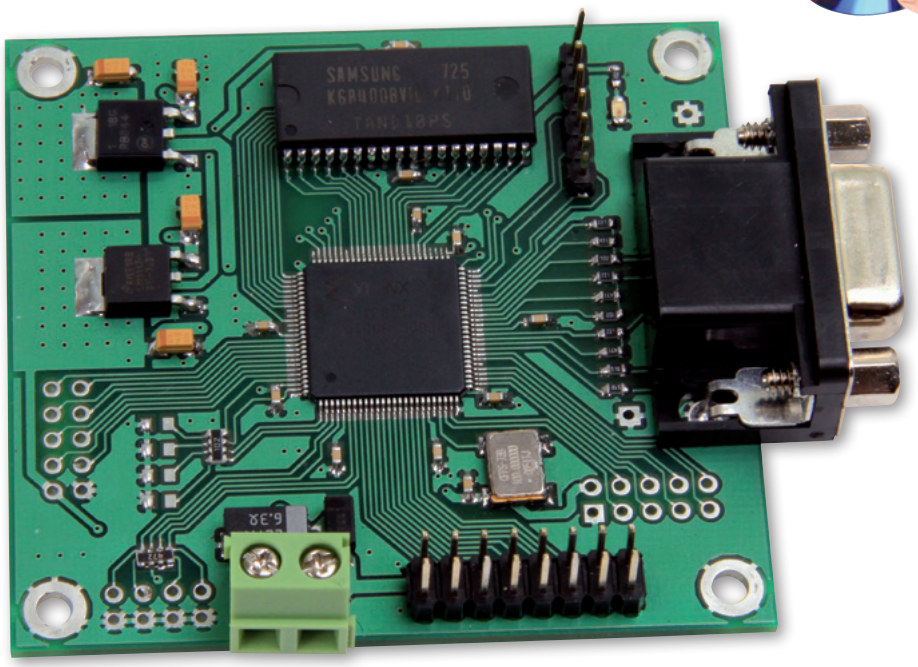
Mini moduł graficzny VGA

AVT 5264

Zaprezentowano opis modułu przekształcającego klasyczny monitor VGA w duży, czytelny wyświetlacz przeznaczony głównie do współpracy z urządzeniami wyposażonymi w proste mikrokontrolery. Moduł charakteryzuje się przede wszystkim prostym interfejsem, funkcjonującym bez trudnych do spełnienia wymagań czasowych i ma własną pamięć graficzną. Jakość obrazu co prawda nie pozwala na zastosowanie jej do wyświetlania zdjęć czy filmów, lecz z powodzeniem wystarcza do celów informacyjnych, np. do wyświetlania napisów lub wykresów. Oprócz tego pełna dostępność kodu źródłowego umożliwia Czytelnikom wzbogacenie modułu o wiele nowych funkcji lub wbudowanie go w większy system cyfrowy.

Rekomendacje: *moduł przyda się osobom zajmującym się reklamą wizualną oraz konstruktorom używającym mikrokontrolerów*



Zastanówmy się, w jaki sposób można dołączyć duży wyświetlacz o rozdzielczości 640 na 480 pikseli do urządzenia opartego na prostym i tanim mikrokontrolerze? Zakup takiego wyświetlacza jako części elektronicznej szczególnie dla hobbysty jest raczej mało opłacalny z kilku powodów. Po pierwsze, wyświetlacze o wyższych rozdzielczościach typowo mają sterowniki bez własnej pamięci obrazu. Oznacza to konieczność ciągłego i bardzo szybkiego wpisywania do nich danych graficznych, aby bez przerwy odświeżać zawartość wyświetlacza. Po drugie, za podobną cenę (kilkaset zł) można kupić fabrycznie nowy monitor LCD do komputera PC: z obudową, kompletem kabli i własnym zasilaczem. Monitory LCD z interfejsem VGA można równie dobrze pozyskać jako złom pozostały po modernizacji sprzętu komputerowego w domach lub firmach (nie wspominając nawet o starych lampowych monitorach CRT, których chyba każdy się chętnie pozbywa.

Decyzje projektowe czyli jak się do tego zabrać?

Na początek naszkicuję cechy sygnału VGA, by łatwiej uzasadnić wybór najważniejszych podzespołów.

Wygenerowanie na monitorze VGA obrazu o rozdzielczości 640×480 punktów z podstawową częstotliwością odświeżania 60 Hz wiąże się z wysyłaniem do niego kolejnych pikseli z częstotliwością 25,175 MHz (w prostych implementacjach stosuje się częstotliwości 25,0 MHz i nie sprawia to żadnych problemów). Jeden piksel w jednym momencie stanowią 3 sygnały analogowe: składowa czerwona (R), zielona (G) i niebieska (B). Dodatkowo trzeba wygenerować dwa cyfrowe sygnały synchronizacji: pionowej i poziomej.

Z racji tego, że potrzebujemy sygnałów analogowych, a generujemy je cyfrowo, potrzebne będą 3 przetworniki cyfrowo-analogowe (DAC), po jednym dla każdego koloru. Częstotliwość zmian sygnału może być relatywnie duża, więc należy zastosować dość szybkie układy. Od ich rozdzielczości (liczby bitów) będzie zależeć liczba możliwych do uzyskania kolorów. W najprostszych układach generujących sygnał VGA (np. płytki testowe układów programowalnych) często stosuje się przetworniki DAC utworzone z kilku odpowiednio połączonych rezystorów. Zaletą takiego przetwornika jest prawie zerowy koszt i spora szybkość działania, a główną wadą trudności w uzyskaniu rozdzielczości wyższych, niż 4...5 bitów, ponieważ jest trudno dobrać odpowiednio dokładne rezystory. W tanim urządzeniu VGA, wyświetlającym jedynie informacje i niewymagającym dużej liczby kolorów, rozwiązanie z rezystorowym przetwornikiem DAC wydaje się być optymalne, dlatego zastosowałem je w omawianym projekcie.

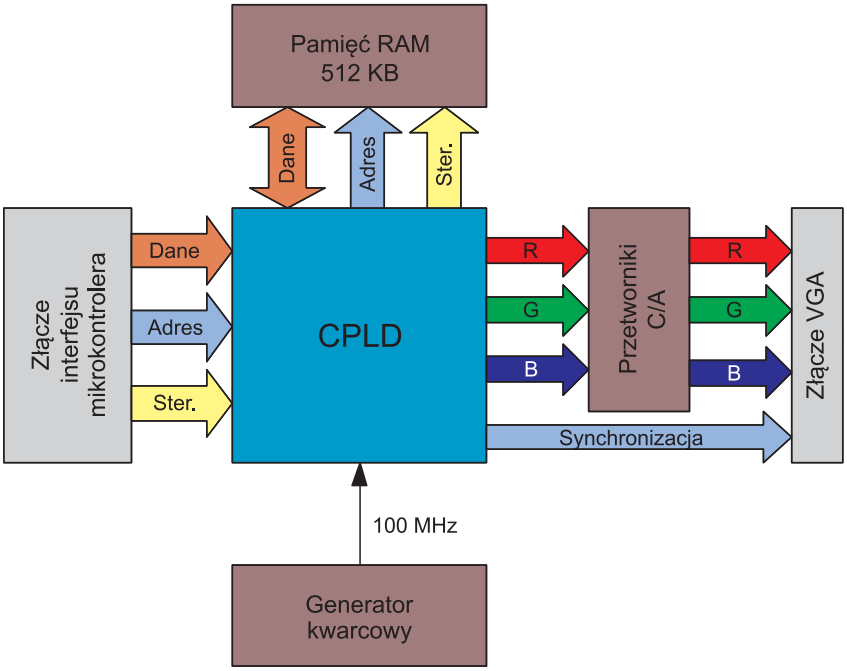
AVT-5264 w ofercie AVT:
AVT-5264A – płytka drukowana

Podstawowe informacje:
- Napięcie zasilania: 5 VDC
- Pobór prądu: ok. 100 mA
- Rozdzielczość obrazu: 640×480 pikseli, 256 kolorów
- Interfejs równoległy, 8-bitowy, zgodny z CMOS 3,3 V (nie akceptuje napięć o poziomach TTL i wyższych)
- Płytka dwustronna o wymiarach 66×62 mm

Dodatkowe materiały na CD i FTP:
ftp://ep.com.pl, user: 12089, pass: 776m3t3q
- wzory płytek PCB
- karty katalogowe i noty aplikacyjne elementów oznaczonych w wykazie elementów kolorem czerwonym

Projekty pokrewne na CD i FTP:
(wymienione artykuły są w całości dostępne na CD)

AVT-959	VGA – Tester (EP 12/2006)
AVT-949	Konwerter VGA – LCD (EP 9/2006)
AVT-942	Prosty tester VGA (EP 8/2006)
AVT-539	Sterownik VGA (EP 11/2004)
AVT-1355	Konwerter VGA ↔ TV (EP 8/2002)



Rysunek 1. Schemat blokowy modułu mini karty VGA

Zastanówmy się teraz, jakim układem można w relatywnie prosty sposób wygenerować dane graficzne przekazywane do przetworników DAC. Musi to być układ odpowiednio szybki, zdolny do odczytu danych z pamięci i wysyłania ich do przetworników z częstotliwością ok. 25 MHz. Większość tanich mikrokontrolerów jednoukładowych nie może sprostać temu zadaniu, ponieważ ich interfejsy I/O, mimo wysokich częstotliwości taktowania rdzenia, nie pracują zbyt szybko. A nawet jeśli mikrokontroler mógłby wygenerować sygnał VGA, to pojawiłby się kłopot z dostępnością pamięci przechowującej kopię obrazu. Przy jakości obrazu generowanego przez omawiane tutaj urządzenie, informacja o kolorze każdego piksela zajmuje 8 bitów, a rozdzielczość wynosi 640×480 pikseli. Dlatego do przechowania całego obrazu bez kompresji potrzeba 300 kB.

Przy powyższych wymaganiach optymalnym rozwiązaniem wydaje się być zastosowanie układu logiki programowalnej z dołączoną zewnętrzną pamięcią. Obecnie najpopularniejszymi odmianami układów programowalnych są CPLD oraz FPGA. W projekcie modułu graficznego VGA zdecydowałem się na zastosowanie CPLD z kilku powodów. Przede wszystkim zastosowanie prostszych układów CPLD (o niewielkich zasobach) jest tańsze, niż użycie FPGA. Dodatkowym czynnikiem obniżającym koszty jest brak konieczności stosowania układu pamięci nieulotnej przechowującego konfigurację, ponieważ CPLD mają wewnętrzną, nieulotną pamięć konfiguracji. Ponadto, w tak prostym projekcie, jak omawiany tutaj moduł, nie wykorzystalibyśmy wszystkich zasobów oferowanych przez FPGA, jak np. układy mnożące czy wewnętrzną pamięć RAM (w większości FPGA dostępnych cenowo dla przeciętnego klienta, wewnętrzna pamięć RAM nie wystarczyłaby do przechowania całego obrazu VGA). Ostateczny wybór padł więc na układ CPLD o oznaczeniu XC2C128 z rodziny CoolRunner-II firmy Xilinx. Zastosowany układ w wersji z sufiksem **-VQG100** ma wyprowadzenia kompatybilne z układem XC2C256-VQG100 o dwukrotnie większej liczbie makrocel, co otwiera drogę do wzbogacenia możliwości funkcjonalnych urządzenia (w podstawowej wersji omawianej w artykule zajęto bowiem prawie wszystkie zasoby układu XC2C128).

W opracowanym przeze mnie mini module graficznym we wcześniejszej wersji użyłem układu CPLD typu EPM3128 firmy Altera o podobnej liczbie zasobów logicznych, jednak w ostatecznie układ ten przegrał ze wspomnianym wcześniej produktem Xilinx'a ze względu na wyższą cenę i trudności w zakupie prawdopodobnie wynikające z jego mniejszej popularności.

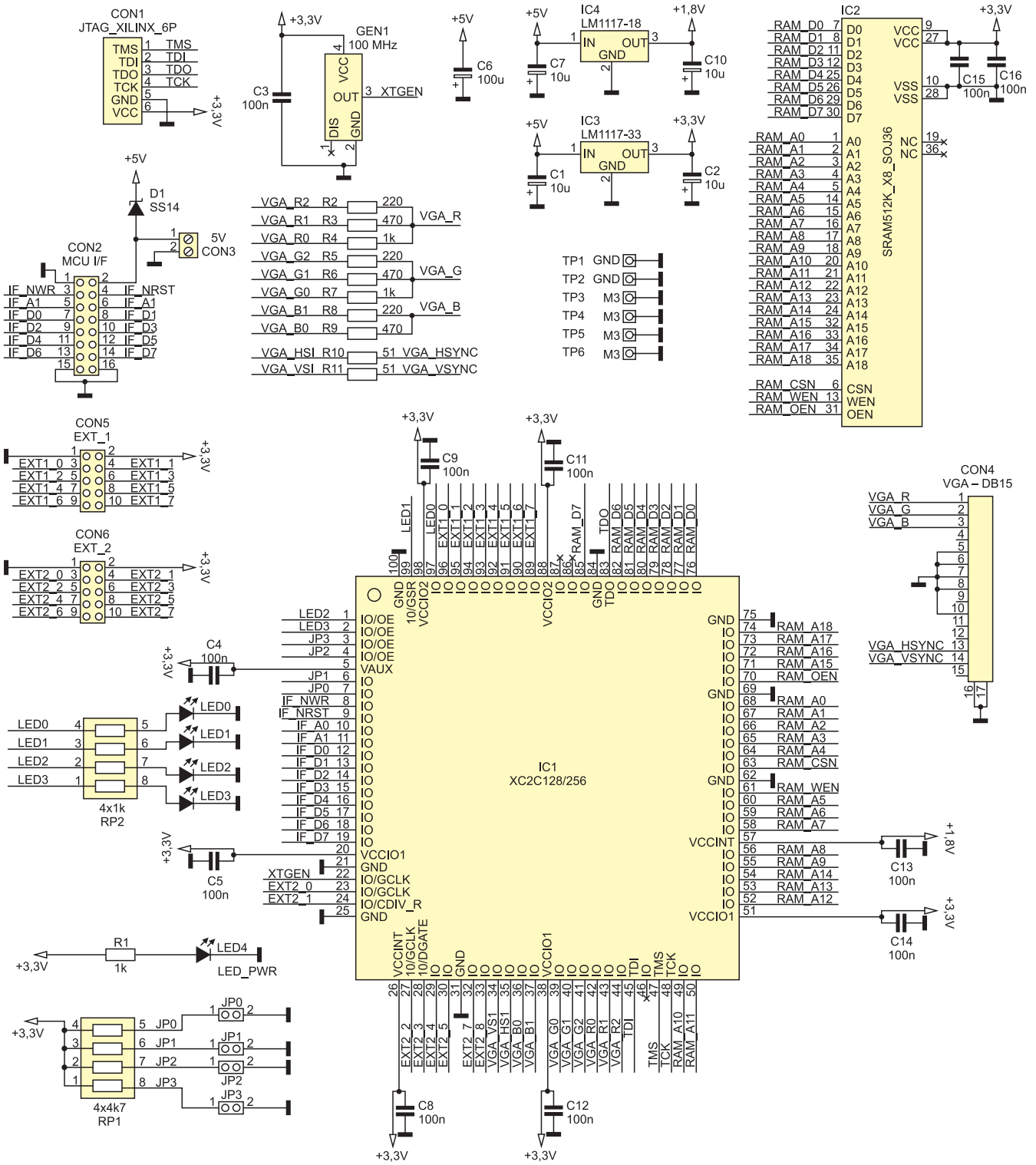
Jako pamięć obrazu dołączoną do układu CPLD zastosowano szybki układ statycznej pamięci SRAM typu K6R4008V1D-KI10 o pojemności 512 kB i organizacji 8-bitowej. Zastosowanie pamięci SRAM podyktowane było prostotą i dużą popularnością jej interfejsu. Dzięki temu, nawet jeśli wspomniane układy pamięci nie będą dostępne, to bez trudu będzie można znaleźć dla nich w pełni funkcjonalny zamiennik. W najgorszym przypadku trzeba będzie przeprojektować płytkę drukowaną dopasowując ją do innej obudowy lub zmienić konfigurację układu CPLD dla pamięci o innej organizacji. Na przykład układy pamięci SRAM o organizacji 16-bitowej wymagają dodatkowych dwóch sygnałów selekcji bajtu, które można wygenerować dopisując dosłownie kilka linijek do kodu źródłowego w języku opisu sprzętu.

Na koniec pozostaje wybór źródła sygnału zegarowego dla modułu graficznego. Popularnym sposobem jego uzyskania dla układów programowalnych jest zastosowanie scalonego generatora kwarcowego. Generator tego rodzaju wystarczy podłączyć do napięcia zasilania i na jego wyjściu uzyskujemy sygnał zegarowy o częstotliwości deklarowanej przez producenta, który możemy podać na wejście taktujące układu cyfrowego. Częstotliwość zastosowanego w projekcie generatora jest dość wysoka, bo wynosi aż 100 MHz, choć wystarczy podawać kolejne piksele z częstotliwością nieco ponad 25 MHz. Szybszy zegar umożliwia jednak łatwą (tzn. zajmującą mało zasobów CPLD) implementację jakby „dwuportowego" dostępu do pamięci RAM: można przez to bez trudu i szybko zapisywać oraz prawie jednocześnie odczytywać z niej dane.

Budowa modułu

Schemat blokowy modułu zamieszczono na **rysunku 1**, natomiast dokładne połączenia elektryczne możemy przeanalizować na **rysunku 2**. Najważniejszym komponentem urządzenia jest układ CPLD. Na nim spoczywa odpowiedzialność za odbiór i buforowanie danych z mikrokontrolera, odczyt i zapis danych do pamięci oraz generowanie sygnałów cyfrowych, z których jest tworzony sygnał VGA.

Przetwornik DAC zbudowano z kilku rezystorów. Tory sygnałów kolorów czerwonego i zielonego wyposażono w przetworniki 3-bitowe (rezystory R2...R4 dla koloru czerwonego i R5...R7 dla zielonego), natomiast do wygenerowania sygnału koloru niebieskiego użyto przetwornika 2-bitowego (rezystory R8 i R9). Przetworniki te stają się kompletne w momencie dołączenia do złącza CON4 monitora VGA. W monitorze, pomiędzy końcem przewodu każdego koloru a masą, jest włączony rezystor o rezystancji 75 Ω. Według zaleceń, na wyjściach poszczególnych kolorów na złączu VGA powinno występować napięcie 0...0,7 V. Im napięcie z tego zakresu jest wyższe, tym jaśniej będzie wyświetlana składowa danego koloru.



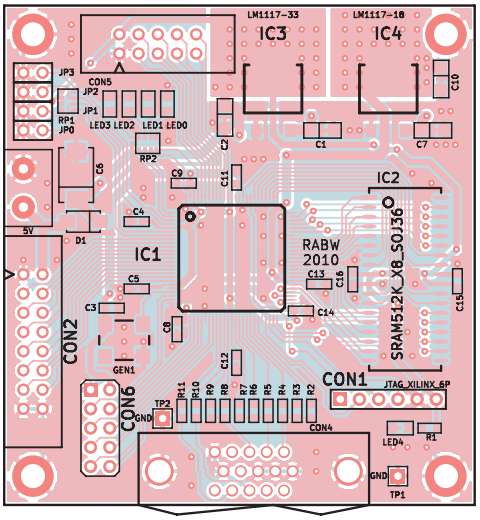
Rysunek 2. Schemat ideowy modułu mini karty VGA

Pamięć SRAM jest dołączona do układu CPLD bez dodatkowych elementów pośredniczących. Do komunikacji z nią nie są także używane specjalizowane wyprowadzenia CPLD.

Układy modułu graficznego wymagają dwóch napięć zasilania. Napięcie 1,8 V potrzebne jest dla rdzenia układu CPLD, natomiast linie wejścia/wyjścia CPLD oraz wszystkie inne układy scalone na płytce (pamięć SRAM i generator kwarcowy) zasilane są napięciem 3,3 V. Aby uniezależnić się od napięcia zewnętrznego, wszystkie układy zasilane są ze stabilizatorów IC3 i IC4. Są to popularne układy LM1117 z ustalonymi fabrycznie napięciami wyjściowymi. Zalecanym napięciem zasilania modułu jest 5 V, jednak nie ma przeszkód, aby zasilać go wyższym napięciem. Trzeba jednak wtedy pamiętać o:

1) zastosowaniu kondensatorów C1, C6 i C7 przeznaczonych do pracy przy odpowiednio wyższym napięciu,
2) wziąć pod uwagę moc wydzielaną w stabilizatorach IC3 i IC4. Napięcie zasilania możemy podać na złącze śrubowe CON3 lub na wyprowadzenia 2 (+5 V), 1, 15, 16 (masa) złącza CON2, które zostało przewidziane jako interfejs pomiędzy modułem, a układem z mikrokontrolerem.

Płytka drukowana modułu graficznego jest wyposażona także w kilka dodatkowych elementów umożliwiających Czytelnikom własne eksperymentowanie. Są to: diody LED0... LED3, zworki JP0...JP3 oraz dwa złącza rozszerzeń CON5 i CON6. Mogą one być przydatne podczas rozbudowy modułu, ale w wersji podstawowej nie trzeba ich montować.



Rysunek 3. Schemat montażowy modułu mini karty VGA

Montaż i uruchomienie

Wzór płytki drukowanej zamieszczono na płycie CD oraz na serwerze FTP, natomiast schemat montażowy pokazano na **rysunku 3**. Warto wykonywać go w omówionej niżej kolejności.

Najpierw wlutowujemy rezystory, drabinki rezystorowe, kondensatory, diodę D1 i ewentualnie diody LED. Następnie montujemy stabilizatory IC3 oraz IC4 i wstępnie sprawdzamy poprawność montażu. Podłączamy napięcie zasilające +5 V i sprawdzamy prawidłowość napięć 1,8 V i 3,3 V. Jeśli mamy oscyloskop lub miernikiem częstotliwości, to możemy zamontować i sprawdzić działanie generatora kwarcowego GEN1 – na jego wyjściu powinien być przebieg o częstotliwości 100 MHz. Jeśli napięcia są prawidłowe i generator działa, to montujemy pozostałe elementy: układ CPLD, pamięć SRAM i złącza.

Po zmontowaniu całości i podaniu napięcia zasilającego 5 V (np. na złącze CON3) stabilizatory IC3 i IC4 nie powinny się nagrzewać, a cały moduł powinien pobierać prąd niewiele większy od 100 mA.

Teraz możemy przystąpić do wgrania pliku programującego do układu CPLD. Do tego celu potrzebny jest programator z interfejsem JTAG. Podłączamy go do złącza CON1. W fazie testów układ CPLD był programowany za pomocą wtyczki *JTAG-SPI Full Speed* produkcji Digilent, współpracującej z darmowym programem *Digilent Adept*.

W materiałach uzupełniających do artykułu dostarczone są gotowe, przykładowe pliki programujące dla układów XC2C128 i XC2C256, aby nie było konieczności kompilowania całego projektu.

Jeśli konfiguracja zostanie prawidłowo zapisana w układzie CPLD, to po włączeniu zasilania modułu i podłączeniu do niego monitora VGA powinniśmy ujrzeć nieruchomy obraz składający się z pikseli o kolorach wyglądających na losowe – to zawartość niezainicjalizowanych komórek układu pamięci SRAM.

Obsługa modułu

Interfejs modułu jest prosty w obsłudze i nie narzuca żadnych trudnych do spełnienia zależności czasowych. Dane i sygnały sterujące dla modułu VGA doprowadzamy do złącze CON2 (wg schematu z rysunku 2). Do tego złącza możemy także dołączyć napięcie zasilania. Wówczas wystarczy tylko jeden przewód łączący nasz moduł z mikrokontrolerem. W **tabeli 1** zostały opisane szczegółowo funkcje poszczególnych wyprowadzeń złącza CON2.

Uwaga! Należy pamiętać, że na wejścia zastosowanego układu CPLD nie można podawać napięć wyższych od 3,3 V. Dlatego, chcąc zastosować moduł w urządzeniach z mikrokontrolerami, w których na liniach I/O przy wysokim poziomie występuje napięcie 5 V, należy zastosować konwersję poziomów napięć. Może to być np. 74LVC245 zasilany z napięciem 3,3 V. Czasami wystarczą nawet same rezystory włączone szeregowo z każdym wyprowadzeniem układu zasilanego niższym napięciem, lecz jest to rozwiązanie prowizoryczne i nie gwarantuje najlepszych parametrów czasowych czy odporności na zakłócenia.

W interfejsie modułu możemy wyróżnić dwie linie sterujące: sygnał zerowania IF_NRST oraz strobujący zapis IF_NWR. Dane podajemy na osiem linii IF_D0...IF_D7, natomiast wejścia IF_A0 i IF_A1 służą do wyboru rodzaju danych wpisywanych do IF_D.

Stanem nieaktywnym linii sterujących IF_NWR i IF_NRST jest „1”. Pojawienie się impulsu ujemnego (chwilowy stan „0”) na wyprowadzeniu IF_NRST powoduje wyzerowanie modułu i przejście do stanu, jak po włączaniu napięcia zasilania, ale zawartość zewnętrznej pamięci SRAM pozostaje bez zmian. Z kolei sygnał IF_NWR umożliwia wpisywanie danych do modułu: jego opadające zbocze powoduje zatrzaśnięcie stanu linii danych (IF_D[7:0]) i adresowych (IF_A[7:0]). Przykładowe przebiegi podczas zapisu dwóch wartości pokazano na **rysunku 4**.

Za pomocą magistrali IF_D możemy wpisywać 4 rodzaje danych (**tabela 2**) do wewnętrznych układów modułu VGA.

Po podaniu na wejścia IF_A[1:0] poziomów „00”, każdy bajt doprowadzony na wejścia IF_D[7:0] i zatrzaśnięty opadającym zboczem sygnału IF_NWR pojawi się na ekranie monitora jako pojedynczy piksel o kolorze zależnym od liczby podanej na IF_D. W tabeli 2 podano, które bity danych odpowiadają poszczególnym kolorom. Każdy kolejny wpis ustawia sąsiedni piksel. Po wyzerowaniu zaczniemy wpisywać dane począwszy od pik-

Tabela 1. Opis funkcji poszczególnych wyprowadzeń złącza CON2

Nr wypr.	Nazwa sygnału	Opis
1	GND	Masa
2	+5V	Zasilanie +5V (dostępne także na złączu CON3)
3	IF_NWR	Opadające zbocze zatrzaskuje dane podane na IF_D[7:0] i IF_A[1:0]
4	IF_NRST	Ujemny impuls zeruje logikę modułu VGA
5	IF_A0	Wejścia adresowe. Umożliwiają wybór rejestru, do którego chcemy wpisać dane z wyprowadzeń IF_D[7:0]. Najbardziej znaczący bit to IF_A1.
6	IF_A1	
7	IF_D0	Wejścia danych. W zależności od kombinacji A[1:0] można tutaj wpisywać dane graficzne lub wybrane bity adresu kursora. Najbardziej znaczący bit to IF_D7.
8	IF_D1	
9	IF_D2	
10	IF_D3	
11	IF_D4	
12	IF_D5	
13	IF_D6	
14	IF_D7	
15	GND	Masa
16	GND	Masa

Wykaz elementów

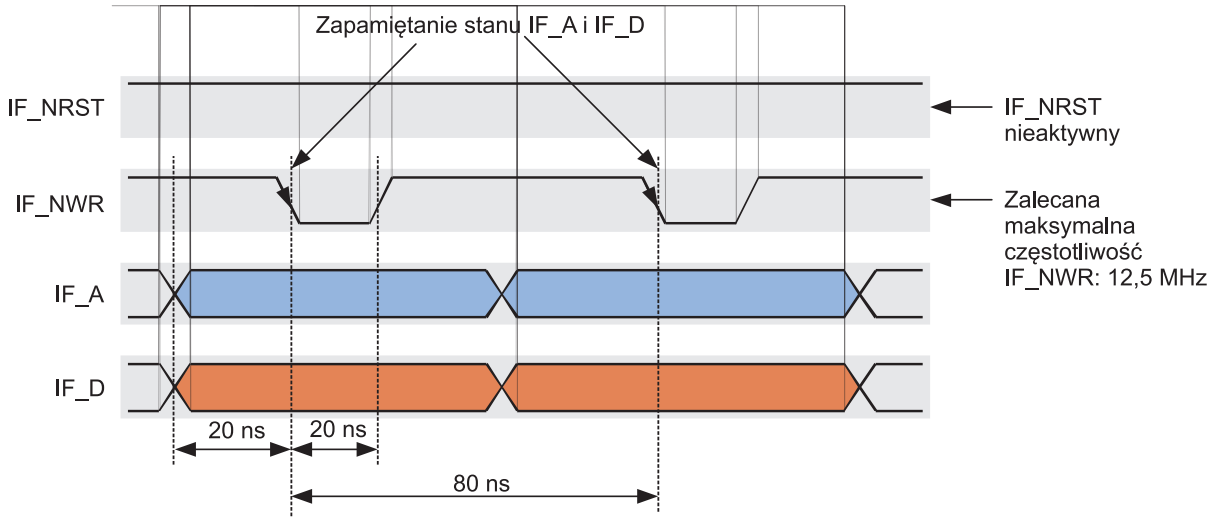
Rezystory:
R10, R11: 51 Ω (0603)
R2, R5, R8: 220 Ω (0603)
R3, R6, R9: 470 Ω (0603)
R1, R4, R7: 1 kΩ (0603)
RP1: drabinka rezystorowa 4×1 kΩ (1206)
RP2: drabinka rezystorowa 4×4,7 kΩ (1206)

Kondensatory:
C3...C6, C8, C9, C11...C16: 100 nF (0603)
C1, C2, C7, C10: 10 µF/6,3 V (tantalowy A)
C6: 100...470 µF/6,3 V (tantalowy D)

Półprzewodniki:
IC1: XC2C128 lub XC2C256, sufiks -VQ100, lub -VQG100
IC2: K6R4008V1D-KI10 (SOJ-36)
IC3: LM1117-3,3 (TO-252)
IC4: LM1117-1,8 (TO-252)
D1: SS14
LED0...LED4: diody LED SMD, obudowa 0805

Inne:
CON1: goldpin 6p
CON2: goldpin 2×8p lub „wannowe” 16p
CON3: śrubowe terminal block 2p, raster 5 mm
CON4: DB15 gęste, żeńskie, do druku
CON5, CON6: goldpin 2×5

Na CD: karty katalogowe i noty aplikacyjne elementów oznaczonych w wykazie elementów kolorem czerwonym



Rysunek 4. Przebiegi czasowe podczas zapisu danych

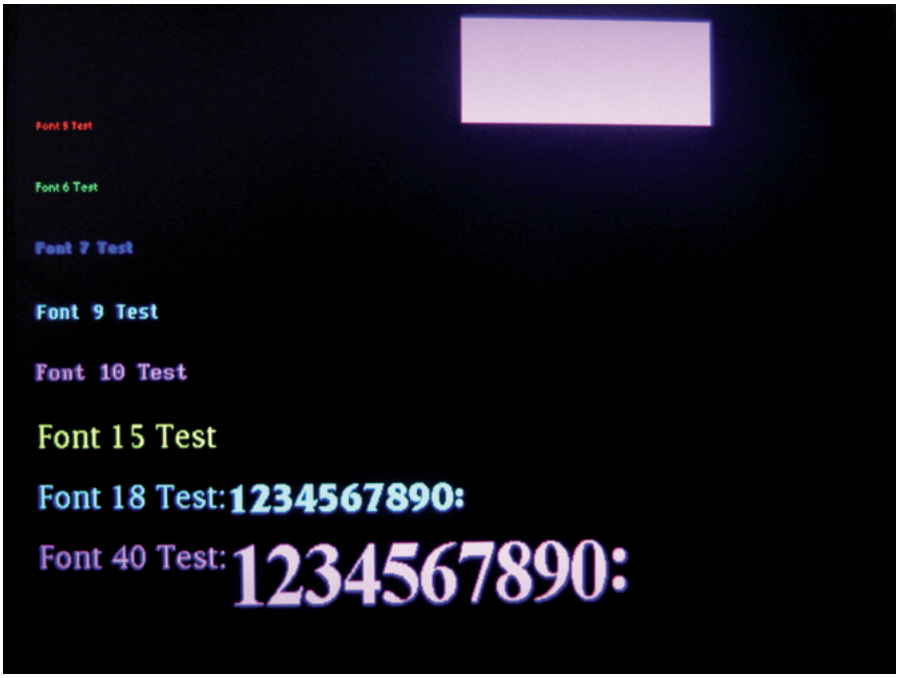
sela znajdującego się w lewym górnym rogu ekranu – powiemy wtedy, że *kursor* znajduje się pod adresem 0. Kolejne wpisy będą przesuwały aktualne położenie kursora w prawo aż do końca linii (zapełnimy danymi całą linię), a potem na początek następnej linii i tak do końca obrazu. Ostatni (skrajny prawy) piksel pierwszej (górnej) linii obrazu ma adres 639. Pierwszy piksel drugiej linii ma adres 640 itd. Ostatni piksel (prawy dolny) będzie miał adres 640*480-1.

Oczywiście nie musimy wpisywać danych graficznych zawsze rozpoczynając od lewego górnego rogu ekranu (od adresu 0). Adres kursora możemy zmienić w dowolnym momencie, tak aby zmieniać jedynie wybraną część wyświetlanego obrazu. Z racji tego, że adres kursora opisuje 19-bitowa liczba, a interfejs wejściowy ma tylko 8 bitów (oszczędność wyprowadzeń mikrokontrolera), wpisywanie adresu kursora odbywa się porcjami. Najmniej znaczące bity adresu kursora podajemy jak poprzednio na magistralę IF_D i zapisujemy opadającym zboczem IF_NWR, lecz tym razem przy poziomach na wyprowadzeniach IF_A[1:0] równych „01”. Bardziej znaczące bity (15:8) adresu kursora wpiszemy podając na wejścia IF_A[1:0] poziomy „10”, a trzy najbardziej znaczące bity (18:16) przy IF_A[1:0] – „11”. Oczywiście, wpisując dane przy IF_A[1:0] innym niż „00”, obraz wyświetlany na monitorze nie zmieni się.

Tabela 2. Rodzaje danych wpisywanych do mini modułu VGA

Wejścia adresowe		Znaczenie danych zatrzaśniętych na wejściach IF_D[7:0]	Znaczenie poszczególnych bitów na wejściach danych							
IF_A1	IF_A0		IF_D7	IF_D6	IF_D5	IF_D4	IF_D3	IF_D2	IF_D1	IF_D0
0	0	Dane graficzne (1 piksel)	czerw., bit 2	czerw., bit 1	czerw., bit 0	ziel., bit 2	ziel., bit 1	ziel., bit 0	nieb., bit 1	nieb., bit 0
0	1	Najmniej znaczące bity adresu kursora	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1	0	Średnie bity adresu kursora	bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8
1	1	Najbardziej znaczące bity adresu kursora	wartość nieistotna	–	–	–	–	bit 18	bit 17	bit 16

Interfejs modułu VGA jest dość szybki. Zgodnie z przeprowadzonymi symulacjami, bezpieczną częstotliwością wprowadzania danych do modułu VGA jest 12,5 MHz (okres 80 ns), czyli 1/8 częstotliwości sygnału tak-



Fotografia 5. Obraz generowany przez program testujący

Tabela 3. Połączenia kabla łączącego płytkę testową z modułem VGA

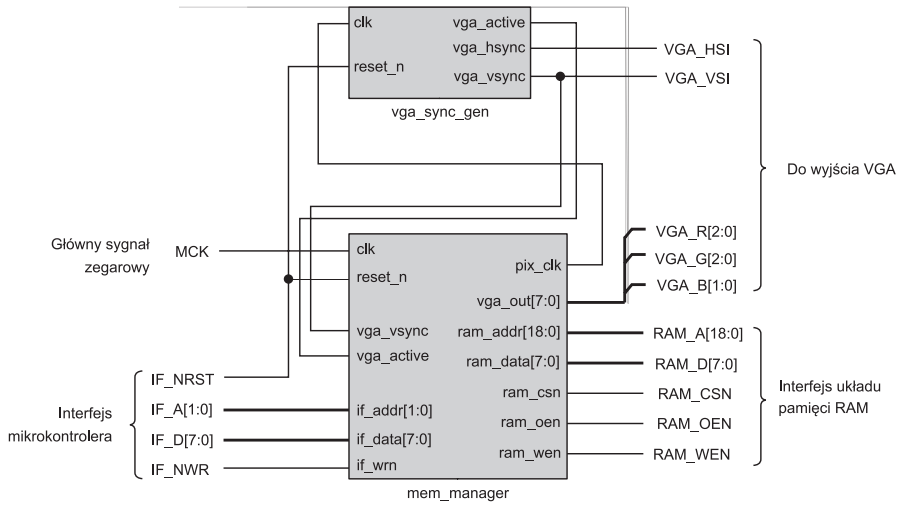
Złącze interfejsu karty VGA		Złącze rozszerzeń płytki ZL28ARM	
Sygnał	Nr wypr.	Nr wypr.	Sygnał
GND	1	39	GND
+5V	2	40	+5V
IF_NWR	3	3	PB0
IF_NRST	4	4	PB1
IF_A0	5	5	PB2
IF_A1	6	6	PB3
IF_D0	7	7	PB4
IF_D1	8	8	PB5
IF_D2	9	9	PB6
IF_D3	10	10	PB7
IF_D4	11	11	PB8
IF_D5	12	12	PB9
IF_D6	13	13	PB10
IF_D7	14	14	PB11
GND	15	29	GND
GND	16	29	GND

Listing 1. Funkcja *vgaBusWrite* z programu testowego wpisującą dane (*data*) na magistralę IF_D modułu VGA

```
static inline void vgaBusWrite(uint8_t data)
{
        vgaBusSet(data);
        VGA_NWR_0;
        VGA_NWR_1;
}
```

Listing 2. Funkcja służąca do ustawiania kursora na zadanych współrzędnych *x*, *y*

```
void vgaSetCursor(uint32_t x, uint32_t y)
{
        uint32_t address = y * VGA_X_SIZE + x;
        vgaSetRegAddress(REG_ADDR_L);
        vgaBusWrite(address);
        address = address >> 8;
        vgaSetRegAddress(REG_ADDR_M);
        vgaBusWrite(address);
        address = address >> 8;
        vgaSetRegAddress(REG_ADDR_H);
        vgaBusWrite(address);
        vgaSetRegAddress(REG_DATA);
}
```



Rysunek 6. Schemat blokowy funkcji realizowanych przez CPLD

tującego moduł. Jest to także maksymalna częstotliwość zmian sygnału IF_NWR, zapewniająca prawidłowe wpisanie danych. Można przyjąć, że minimalny czas trwania ujemnego impulsu na linii IF_NWR gwarantującego prawidłowy zapis wynosi 20 ns. Tyle należy więc przyjąć jako minimalny czas, jaki powinien upłynąć od ustawienia wejść danych i adresowych do podania opadającego zbocza na IF_NWR. Omawiane tutaj czasy zaznaczono na rysunku 4. Należy zaznaczyć, że są to czasy podane z zapasem, a więc na własną odpowiedzialność można spróbować przesyłać dane do modułu nieco szybciej. Nie wydaje się to jednak konieczne, ponieważ przy częstotliwości sygnału IF_NWR wynoszącej 12,5 MHz jesteśmy w stanie przesłać dane dla całego obrazu w ciągu ok. 25 ms.

Przykładowe oprogramowanie dla mikrokontrolera

W materiałach do artykułu jest przykładowy program testujący współpracę mini modułu VGA i płytki testowej ZL28ARM z mikrokontrolerem AT91SAM7XC256. W **tabeli 3** umieszczono listę połączeń kabla łączącego płytkę testową z modułem VGA – z takimi połączeniami działa dostarczony program testowy. Zasilanie (ok. 5 V) dla modułu VGA można pobierać bezpośrednio z płytki ewaluacyjnej.

Program testowy napisano w języku C i był kompilowany w pakiecie WinARM. Po poprawnym podłączeniu działającego modułu VGA do płytki ZL28ARM i zaprogramowaniu pamięci Flash mikrokontrolera plikiem wynikowym projektu, na ekranie powinniśmy ujrzeć obraz, jak na **fotografii 5**. W programie testowym, obsługę modułu VGA podłączono do biblioteki obsługi wyświetlaczy LCD. Nie jest ona zoptymalizowana dla omawianego modułu, stąd generowanie obrazu może wydawać się nieco powolne. Dzieje się tak dlatego, że rysowanie każdego piksela poprzedzone jest ustawianiem odpowiadającego mu adresu kursora, co znacznie spowalnia wpisywanie danych. Aby to zmienić, należałoby zoptymalizować działanie funkcji rysujących znaki i kształty, aby – gdzie to tylko możliwe – korzystały z automatycznej inkrementacji adresu kursora.

Na **listingu 1** zamieszczono funkcję *vgaBusWrite* z programu testowego wpisującą dane (*data*) na magistralę IF_D modułu pod wcześniej ustawiony adres. Wywołanie funk-

cji *vgaBusSet* oznacza zmianę poziomów wyprowadzeń odpowiadających sygnałom IF_D na poziomy określone stanami logicznymi podanymi w argumencie. Z kolei *VGA_NWR_0* i *VGA_NWR_1* są makrodefinicjami odpowiednio, zerującymi i ustawiającymi wyprowadzenia mikrokontrolera dołączone do wejścia IF_NWR modułu.

Na **listingu 2** zawarto definicję funkcji służącej do ustawiania kursora na zadanych współrzędnych *x* oraz *y*, przy czym x=0 i y=0 oznaczają lewy górny róg ekranu. Na podstawie zadanych współrzędnych jest obliczany adres, który następnie zostaje wpisany do wewnętrznych rejestrów modułu VGA. Na końcu funkcji, wyjścia adresowe są zerowane (stała *REG_DATA*), przez co kolejne wywołania funkcji *vgaBusWrite* będą oznaczały wprowadzanie danych graficznych począwszy od wybranych współrzędnych.

Co się dzieje w CPLD?

Przeanalizujmy teraz bloki zawarte w układzie CPLD. Opis jego logiki został przedstawiony graficznie na schemacie blokowym na **rysunku 6**. Składa się on z dwóch części: układu zarządzającego dostępem do pamięci (moduł *mem_manager*) oraz układu generowania sygnałów synchronizujących (moduł *vga_sync_gen*).

Układ *vga_sync_gen* generuje przebiegi synchronizacji VGA_HSI i VGA_VSI oraz sygnałem *vga_active* sygnalizuje układowi zarządzania pamięcią, w którym momencie należy wysyłać do monitora dane graficzne. Jeśli aktualnie należy wysyłać sygnał video, to moduł *mem_manager* odczytuje kolejne komórki pamięci SRAM, a ich zawartość kieruje do wyjść VGA_R, VGA_G i VGA_B, a stąd wędrują prosto do przetworników DAC. Gdy wystąpi aktywny sygnał synchronizacji pionowej *vga_vsync* (rozdzielający poszczególne klatki obrazu), licznik adresu komórki do odczytu jest zerowany, by w kolejnej ramce zliczać adresy od początku.

Z wcześniejszego opisu wiemy, że nową treść obrazu można praktycznie w każdej chwili zapisywać do pamięci, np. za pomocą mikrokontrolera. Może się to odbywać dzięki szybkiej pamięci SRAM i sygnałowi zegarowemu MCK o dużej częstotliwości (100 MHz) dostarczanego do modułu *mem_manager* – wyświetlanie jednego piksela trwa bowiem aż 4 takty tego zegara. W tym czasie można z powodzeniem wykonać jeden cykl odczytu danych z pamięci SRAM, a także – jeśli przyjdą nowe dane – jeden cykl zapisu. To sprawia, że wpisując dane za pomocą mikrokontrolera, nie musimy czekać na zwolnienie pamięci (np. na przerwę w transmisji obrazu), lecz po zatrzaśnięciu danych, w kilkanaście do kilkudziesięciu nanosekund, wpisane dane graficzne mogą znaleźć się w pamięci SRAM.

Moduł *mem_manager* przechowuje w dwóch niezależnych rejestrach aktualne adresy odczytu (adres wyświetlanego piksela) oraz zapisu. Każdy takt sygnału *pix_clk*, przy aktywnym sygnale *vga_active*, powoduje zwiększenie wartości adresu odczytu. Natomiast każde wprowadzenie nowego słowa z magistrali IF_D[7:0] przy IF_A[1:0] = „00” zwiększa o 1 adres zapisu. W ciągu wspomnianych wcześniej 4 cykli głównego sygnału zegarowego MCK, specjalny rejestr modułu *mem_manager* multipleksuje adres aktualnie wystawiany na magistralę adresową pamięci SRAM: przez 2 cykle wystawiany jest adres odczytu, a przez pozostałe 2 adres zapisu. W ten sposób udało się uzyskać w module *mem_manager* jakby dwuportowy kontroler pamięci.

Czytelników zainteresowanych dokładną analizą działania logiki wewnątrz układu CPLD zachęcam, by zajrzeli do kodów źródłowych. Są one dostarczone w materiałach do artykułu wraz z plikiem UCF zawierającym przypisania wyprowadzeń modułu nadrzędnego do wyprowadzeń układu CPLD. Cały kod opisu sprzętu został napisany w języku Verilog i skompilowany w środowisku Xilinx ISE 11 z darmową licencją.

Uwagi końcowe

Mini moduł graficzny może być udoskonalany. Możliwości mamy dość duże. Przede wszystkim, chcąc uzyskać wyższą jakość obrazu należałoby zastosować co najmniej 16-bitowe kodowanie barw pikseli. Wiąże się to z zastosowaniem pamięci SRAM o większej pojemności i (najlepiej) 16-bitowej organizacji, a także dodaniem do układu prawdziwych przetworników DAC zamiast ich rezystorowej namiastki. Co prawda modyfikacja opisu układu w takim przypadku nie będzie zbyt trudna, ale nie unikniemy konieczności projektowania nowej płytki drukowanej. Teoretycznie, stosując 16-bitowe kodowanie pikseli można by uniknąć używania większej pamięci obrazu korzystając z palet kolorów, lecz może okazać się, że takie rozwiązanie pochłania więcej zasobów logicznych i jest znacznie bardziej skomplikowane w obsłudze niż kodowanie pikseli bez użycia palety.

Z racji naturalnie wymuszonej przenośności kodu pisanego w językach opisu sprzętu, nie powinno sprawić najmniejszych problemów zaimplementowanie modułu graficznego w układzie CPLD innej firmy – wystarczy tylko, aby miał odpowiednią ilość zasobów logicznych i mógł pracować z częstotliwościami rzędu 100 MHz, co obecnie nie jest dużym problemem. Nic także nie stoi na przeszkodzie, by wkomponować moduł graficzny w większy system cyfrowy implementowany w układzie FPGA. Będzie to o tyle rozsądne, że dużą częstotliwość głównego zegara modułu będzie można uzyskać z wewnętrznego modułu mnożnika częstotliwości, które wchodzą w skład typowego wyposażenia układów FPGA.

Sama płytka, na której zbudowano moduł graficzny może także służyć do innych celów. Umieszczenie na niej szybkiego układu CPLD i pamięci RAM oraz źródła sygnału zegarowego o dużej częstotliwości właściwie daje nam moduł, który może stać się elementem np. rejestratora stanów logicznych.

Robert Brzoza-Woch
rabw@poczta.fm