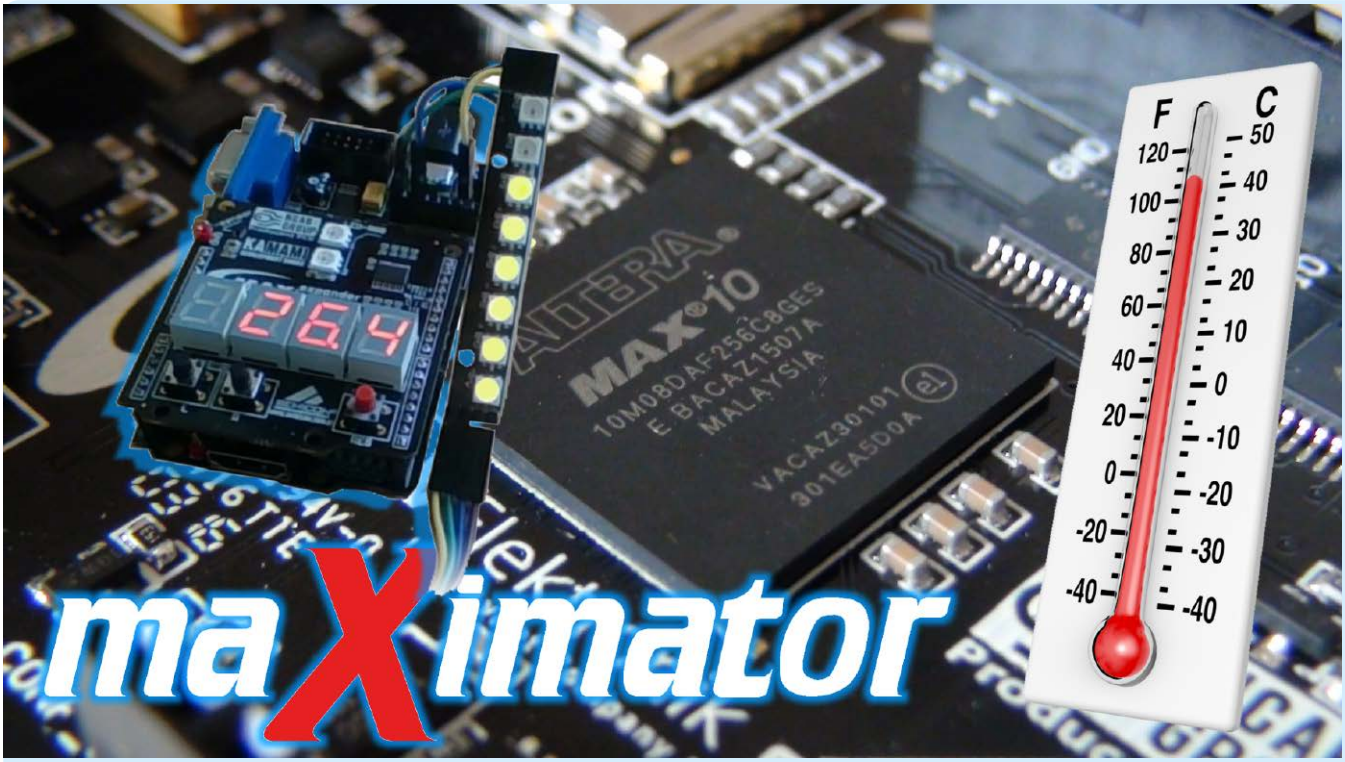
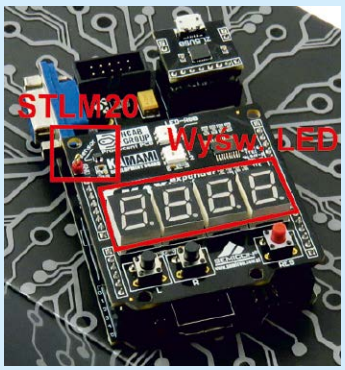


Pierwsze kroki z FPGA (8)

Wskaźnik temperatury z STLM20 na diodach RGB-LED WS2812B

Miesiąc temu przedstawiliśmy projekt cyfrowego miernika temperatury, który wyświetlał temperaturę zmierzoną za pomocą analogowego sensora temperatury na 3-pozycyjnym, 7-segmentowym wyświetlaczu LED. Teraz pokażemy nieco bogatszy funkcjonalnie termometr, który cyfrowe wskazania prezentowane na wyświetlaczu wzbogaca efektem świetlnym wyświetlanym na LED-RGB WS2812B.



Fotografia 1. Skład zestawu testowego

Prezentowany w artykule termometr jest przeznaczony do mierzenia temperatur w warunkach domowych. Na wyświetlaczu może wyświetlić temperaturę o wartości od 0 do +51,1 stopni Celsjusza z dokładnością do 0,1 stopnia. Na pasku diod LED-RGB są wyświetlane kombinacje kolorów i liczby świecących elementów, odpowiadające temperaturom z zakresu +12…+36,6 stopni Celsjusza. Na diodach uzyskujemy rozdzielczość pomiaru 0,5 stopnia. Wartość można odczytać na podstawie koloru i ilości zapalonych diod zgodnie z **tabelą 1**.

W odróżnieniu od wcześniej prezentowanych projektów, projekt prezentowany w artykule wymaga dołączenia nie tylko shielda maXimator Expander (na którym umieszczono wyświetlacze LED oraz sensor temperatury STLM20 – **fotografia 1**), ale także linijki z 8 diodami LED-RGB WS2812B. W przykładzie użyto modułu KAmodWS2812-8, który dołączono do układu FPGA za pomocą pinów UART-a (**fotografia 2**):

- WS8212B → maXimator,
- GND → GND,
- +VDD → +5V,
- Din → TxD.

Wszystkie oznaczenia są umieszczone na płytkach, jak widać na fotografii. Układ zacznie działać od razu po włączeniu zasilania.

Opis projektu

Projekt składa się z czterech plików napisanych w języku VHDL. Plik główny *rgb_termometr* (najwyższy poziom w hierarchii) zawiera wszystkie niezbędne elementy do odczytania wartości napięcia z przetwornika analogowo-cyfrowego oraz przekazuje te wartości do bloków obsługujących wyświetlacz i diody RGB.

Blok opisany w pliku *wysw* (**rysunek 3**) odpowiada za obsługę wyświetlacza siedmiosegmentowego oraz algorytmy do uśredniania i obliczania temperatury w stopniach Celsjusza. Blok opisany w pliku *led_gen* pobiera wartość temperatury z bloku *wysw* i na tej podstawie ustala ile i jakich kolorów powinno być wyświetlonych na diodach RGB. Blok opisany w pliku *led_sterownik* zamienia kod koloru podawany przez *led_gen* na sygnał

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: **ftp://ep.com.pl**, user: **97325**, pass: **6yfwxr8q**

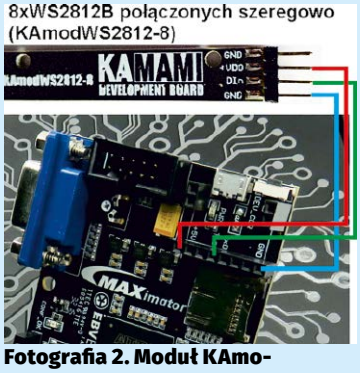
Tabela 1. Wartość temperatury wskazywanej za pomocą LED-RGB

Kolor\ Liczba świecących diod	1	2	3	4	5	6	7	8
Niebieski	12,0	12,5	13,0	13,5	14,0	14,5	15,0	15,5
Jasnoniebieski	16,0	16,5	17,0	17,5	18,0	18,5	19,0	19,5
Zielony	20,0	20,5	21,0	21,5	22,0	22,5	23,0	23,5
Żółty	24,0	24,5	25,0	25,5	26,0	26,5	27,0	27,5
Czerwony	28,0	28,5	29,0	29,5	30,0	30,5	31,0	31,5
Różowy	32,0	32,5	33,0	33,5	34,0	34,5	35,0	35,5

szeregowy, który jest wysyłany na linijkę diod.

Moduły funkcjonalne

led_sterownik (**tabela 2**). Blok *led_sterownik* zrealizowano jako maszynę stanów. Przechodzi ona przez 5 stanów: wysoka jedynka (*h1*), niska jedynka (*l1*), wysokie zero (*h0*), niskie zero (*l0*) oraz stan reset (*resett*).



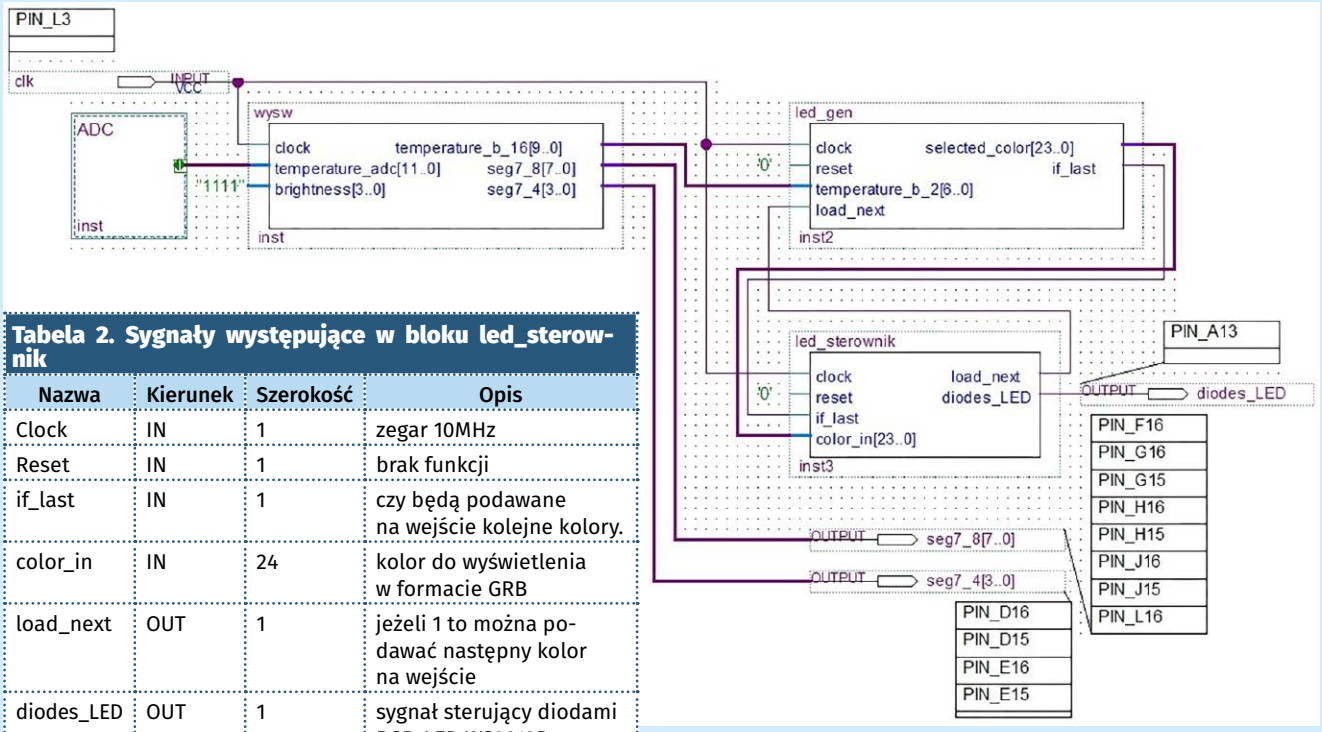
Fotografia 2. Moduł KAmodWS2812-8

Automat zaczyna cykl pracy od zapamiętania wejścia *color_in* do sygnału *color_24*. Licznik *color_counter_high* liczy, który bit koloru jest w danym momencie czytany – dane wysyła od najważniejszych bitów. W zależności od wartości tego bitu wchodzi w stan *h1* albo *h0*. Każdy stan ma licznik, który wskazuje przez jaki czas maszyna ma pozostać w tym stanie. Po upływie tego czasu przechodzi do następnego stanu, którym jest *l1* po stanie *h1* lub *l0* po stanie *h0*. Znowu czeka w tym stanie na odliczenie licznika do zadanej wartości. O tym, do którego stanu przejdzie dalej decyduje kolejny przeczytany bit. Jeżeli licznik *color_counter_high* jest większy od zera to przechodzi do odpowiedniego dla kolejnego bitu stanu *h1* albo *h0*. Kiedy wynosi *0* to w zależności od wartości sygnału wejścia *if_last* zaczyna czytać bity od początku (23 bit) lub – jeżeli *if_last* = *1* - to przechodzi do stanu *resett*, który trwa określony czas. Po tym czasie przechodzi do stanu *h1* lub *h0*, jeżeli sygnał *if_last* będzie mieć wartość 1. W przeciwnym przypadku ponownie wchodzi w stan *resett*.

Tabela 3. Sygnały występujące w bloku led_gen

Nazwa	Kierunek	Szerokość	Opis
clock	IN	1	zegar 10MHz
reset	IN	1	brak funkcji
temperature_b_2	IN	7	temperatura w stopniach Celsjusza podzielona przez 2
load_next	IN	1	informacja otrzymywana z bloku led_sterownik z informacją, czy należy podać następny kolor na wyjście
selected_color	OUT	24	kolor podawany do bloku led_sterownik w formacie GRB
if_last	OUT	1	informacja wysyłana do bloku led_sterownik informująca czy dany kolor jest ostatnim wysyłanym

W module występuje sygnał wyjściowy *load_next*. Informuje on, że można w tym czasie zmienić kolor na wejściu. Zmiana koloru raz w czasie całego stanu wysokiego. Jest on zapamiętywany, kiedy sygnał *load_next* jest równy 0. Sygnał reset nie ma żadnej funkcjonalności. Sygnał zegarowy musi mieć wartość 10 MHz (domyślna w maXimatorze), aby sygnał do diod



Rysunek 3. Projekt termometru RGB

Tabela 2. Sygnały występujące w bloku led_sterownik

Nazwa	Kierunek	Szerokość	Opis
Clock	IN	1	zegar 10MHz
Reset	IN	1	brak funkcji
if_last	IN	1	czy będą podawane na wejście kolejne kolory.
color_in	IN	24	kolor do wyświetlenia w formacie GRB
load_next	OUT	1	jeżeli 1 to można podawać następny kolor na wejście
diodes_LED	OUT	1	sygnał sterujący diodami RGB-LED WS2812B

miał odpowiedni „kształt” w funkcji czasu. Sygnał wyjściowy sterujący diodami LED przyjmuje wartość *1* dla stanów *h1* i *h0* oraz *0* dla pozostałych stanów.

led_gen (**tabela 3**). Blok *led_gen* odpowiada za wygenerowanie odpowiednich kolorów dla 8 diod RGB-LED WS2812B. Na wejściu przyjmuje 7-bitową wartość temperatury w stopniach Celsjusza, gdzie najmłodszy bit jest równy 0,5 stopnia. Zakres temperatur, które może przyjąć *led_gen* wynosi od 0 do 63,5 stopnia.

Moduł wybiera nowy kolor, kiedy sygnał *load_next* – pochodzący z modułu *led_sterownik* – jest równy 1. Na podstawie 4 najwyższych bitów wartości temperatury wejściowej *temperature_b_2* określa, w jakim kolorze będzie wyświetlana temperatura. Progi kolorów wyglądają następująco: 12…15,5– ciemnoniebieski, 16…19,5– jasnoniebieski, 20…23,5 – zielony, 24…27,5 – żółty, 28…31,5 – czerwony, 32,0…35,5 – różowy. Kod koloru jest zapisywany do sygnału *temp_color*. Na podstawie wartości 3 najmłodszych bitów ustalane jest, ile z ośmiu diod modułu KAmodWS2812-8 powinno być włączonych. Dla wartości *0* jest to *1* dioda. Kiedy na *3* najmniej znaczących bitach w sygnale *temperature_b_2* są same jedynki, to włączone są wszystkie diody.

Kody kolorów są wysyłane jeden po drugim. Licznik *count_diodes* odlicza, ile kolorów diod powinno być jeszcze wysłanych i ustala na wyjściu *selected_color* wartość *temp_color*, jeżeli dioda ma się świecić lub same zera, jeżeli dioda nie powinna się świecić. Po ośmiu kolorach licznik *count_diodes* przyjmuje wartość 8 i ustawia bit *if_last*, który sygnalizuje, że więcej kolorów nie będzie wysyłanych.

wysw (**tabela 4**). Moduł *wysw* jest odpowiedzialny za konwertowanie temperatury z przetwornika A/C do zadanych formatów oraz wyświetlaniu jej na wyświetlaczu siedmiosegmentowym. W module występuje sygnał *slow_clk*, który jest inkrementowany przy każdym zboczu narastającym zegara. Służy on jako wejście *‚clock enable’* do wyzwalania procesów, które należy włączać z dużo mniejszą częstotliwością niż częstotliwość głównego sygnału zegarowego.

Proces odczytywania temperatury jest uruchamiany co *4096* taktów zegara. Za każdym razem aktualna wartość z przetwornika ADC z pinu podłączonego do analogowego termometru STLM20 jest dodawana do sygnału *temperature_temp1*. Dodawane jest *128* pomiarów. Kiedy dodane zostanie taka liczba pomiarów, temperaturą jest *12* najwyższych bitów sumy (sygnału *temperature_temp1*). Odjęta jest od danej wartości (*0xBE7*) i pomnożona przez odpowiedni współczynnik, zgodnie ze wzorem:

temperature_temp2 = 3047 – temperature_temp1;
temperature_b_10 = t_przejściowa * 8 + temperature_temp2/4 + temperature_temp2/32 + temperature_temp2/64;

W ten sposób otrzymana jest temperatura w stopniach Celsjusza, gdzie bity *temperature_b_10(12 downto 4)* oznaczają temperaturę pomnożoną przez *10* (np. *0x018* to *2,4* stopnia).

W tym samym miejscu jest wyliczana temperatura przestawiona jako binarna wartość w stopniach Celsjusza, gdzie najmniej znaczący bit to *0,5* stopnia (*temperature_b_2*). Przekazywana jest ona do modułu, gdzie jest wykorzystywana do wyświetlania temperatury na diodach RGB. W procesie tym przeliczana jest także temperatura z postaci binarnej do kodu BCD za pomocą funkcji *to_bcd* - najwyższe bity sygnału *temperature_b_10(12 downto 4)* przekształcane do sygnału BDC *temperature_decimal_10*. Po cztery kolejne bity odpowiadają za kolejne cyfry.

Wyświetlanie na wyświetlaczach 7-segmentowych jest realizowane poprzez przypisanie selektywne. W zależności od wartości *slow_clk(12 downto 11)* wybierane jest jedna anoda i odpowiednie 4 bity z sygnału *temperature_decimal_10*, które następnie są konwertowane na format 7-segmentowy, który jest wpisywany do *seg7_8_temp*.

Tabela 4. Sygnały występujące w bloku wysw

Nazwa	Kierunek	Szerokość	Opis
clock	IN	1	zegar 10MHz
temperature_adc	IN	12	temperatura z przetwornika ADC
brightness	IN	4	jasność wyświetlacza siedmiosegmentowego
temperature_b_16	OUT	10	temperatura w stopniach Celsjusza podzielona przez 16
seg7_8	OUT	8	sygnał na katody wyświetlacza siedmiosegmentowego
seg7_4	OUT	4	sygnał na anody wyświetlacza siedmiosegmentowego

Tabela 5. Sygnały występujące w bloku termometr_rgb

Nazwa	Kierunek	Szerokość	Opis
clk	IN	1	zegar 10MHz
seg7_8	OUT	8	sygnał na katody wyświetlacza siedmiosegmentowego
seg7_4	OUT	4	sygnał na anody wyświetlacza siedmiosegmentowego
diodes_LED	OUT	1	sygnał sterujący diodami RGB-LED WS2812B

Moduł daje możliwość wybrania jasności wyświetlacza siedmiosegmentowego poprzez wybranie na wejściu *brightness* odpowiedniej wartości od *0* do *15*. Zrealizowane jest poprzez sprawdzanie czy 4 bity rejestru *slow_clk(7 downto 4)* są mniejsze, czy większe od wartości *brightness*. Jeżeli warunek jest spełniony, to wyświetlacz jest gaszony. W projekcie termometru jasność na stałe jest ustawiona na wartość maksymalną.

termometr_rgb (**tabela 5**). Najwyżej położony w hierarchii moduł opisowy w VHDL, łączy wszystkie wyżej wymienione modułu oraz zawiera przetwornik ADC, który pobiera wartość z pinu wejścia analogowego numer *0*. Jest on podłączony do kanału *15*. Przetwornik zawiera moduły *adc*, *fingertemp_adc_sequencer* oraz pętlę *pll* generującą zegary na podstawie zegara wejściowego 10 MHz. Wartość z przetwornika jest zapisywana do sygnału *counter*, który jest wysyłany do bloku *wysw*. Blok ten zwraca sygnały, które są kierowane na piny do obsługi wyświetlacza siedmiosegmentowego oraz przekazuje sygnał z informacją o temperaturze do bloku *led_gen*. Blok ten przekazuje i dobiera sygnały z bloku *led_sterownik*, który na podstawie otrzymanej wartości koloru tworzy sygnał *diodes_LED*, który jest wyprowadzony na pin do sterowania diodami RGB.

Piotr Klasa, AGH

Dodatkowe informacje

Projekt powstał w Katedrze Elektroniki Wydziału Informatyki, Elektroniki i Telekomunikacji AGH, pod kierunkiem dr inż. Pawła Rajdy i dr inż. Jerzego Kasperka.

Poprzednie części kursu i dodatkowe materiały dostępne są na FTP: **ftp://ep.com.pl**, user: **97325**, pass: **6yfwxr8q**