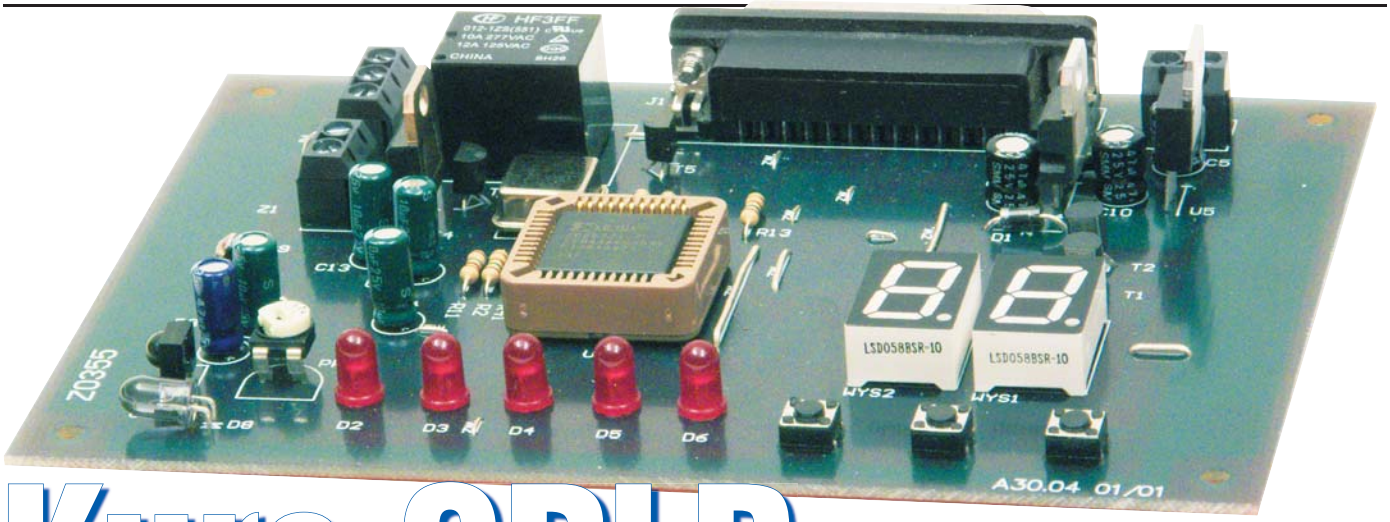


Kurs CPLD

część 2

W poprzednich odcinkach przedstawione zastały podstawowe informacje dotyczące kursu CPLD: omówiono budowę płytki testowej, pokazane zostało, jak sprawdzić poprawność montażu oraz jak pobrać i zainstalować środowisko WebPACK ISE. W tej części chciałbym zapoznać Czytelnika z metodami tworzenia własnych projektów oraz ugruntować wiedzę o podstawowych elementach wykorzystywanych do budowania układów cyfrowych i poprzeć ją stosownymi przykładami.

Zanim zaczniemy, pragnę uświadomić Czytelnikom pewną podstawową, ale być może nie oczywistą rzecz: czym jest w układach cyfrowych zero, a czym jeden? W książkach często operuje się tymi pojęciami i podaje przeróżne definicje, tabelki itp. Zadajmy sobie jednak pytanie: co to znaczy podać jedynkę logiczną na wejście układu? Odpowiedź w dużej mierze zależy od zastosowanego standardu – TTL czy CMOS? W pierwszym przypadku za jedynkę logiczną uznaje się napięcie z zakresu 2V do 5,5V, natomiast w drugim wartość odpowiadającą napięciu od 3,5 do 5V (dla zasilania 5V). W przypadku niniejszego kursu jedynką logiczną będzie napięcie zasilania (3,3V), natomiast zerem logicznym – podanie masy. Widać to również na schemacie płytki testowej – przyciski S1, S2, S3 posiadają podciąganie do napięcia zasilania (3,3V) poprzez rezystory 4,7kΩ i kiedy przycisk jest zwolniony na porty układu CPLD podawane jest to napięcie. Naciśnięcie przycisku spowoduje, że port układu programowalnego zostanie zwarty do masy, co oznacza podanie logicznego zera. Jak wspomniano wcześniej, układ XC9572XL akceptuje również napięcie 5V – w przypadku doprowadzenia takiej wartości ona również zostanie odczytana jako jedynka logiczna (ma to miejsce np. dla odbiornika podczerwieni SFH5110). Zasadniczo dopuszczalne napięcia dla każdego poziomu logicznego (tzn. wartości graniczne) są możliwe do odczytania z karty katalogowej danego elementu. Dla zastosowanego układu CPLD stosowny fragment karty podano na **rysunku 1**.

Bramki logiczne

Po tym krótkim, być może oczywistym wstępie, rozpoczniemy pracę od zbadania bramki AND (iloczynu logicznego). Jest to jeden z podstawowych elementów stosowanych w układach cyfrowych. Spotykana w literaturze tabelka prawdy oraz symbol graficzny tej bramki zostały przedstawione na **rysunku 2**. Jego analiza nasuwa prosty wniosek: podanie na wejścia dwóch jedynek sprawia, że na wyjściu też jest jedynka.

Sprawdźmy przy pomocy płytki testowej, czy rzeczywiście tak jest. Do zadawania stanów wykorzystane zostaną przyciski S1 oraz S2. Jeżeli nie będą wciśnięte, to na obu wejściach pojawi się stan wysoki, jeżeli jakiś przycisk zostanie naciśnięty, to na odpowiadajace im wejście podany zostanie stan niski. Wyjście bramki zostanie podłączone do diody LED, która będzie włączona, gdy pojawi się stan niski.

Pozwolę sobie w tym miejscu na pewną dywagację. Diody LED można wysterować na dwa sposoby: podając na port wyjściowy stan logiczny zero (metoda ta jest zalecana w układach TTL, dlatego została wybrana) lub podając logiczną jedynkę. To, który ze stanów spowoduje zaświecenie diody jest zależne od konstrukcji urządzenia. Trzeba pamiętać, że z portu wyjściowego nie wolno pobierać dowolnie dużego prądu, a jedynie taki, który nie spowoduje uszkodzenia bufora wyjściowego, stąd obecność rezystora ograniczającego, dołączonego szeregowo do diody. Dopuszczalny prąd zawiera się najczęściej w granicach kilkunastu mA, dokładną jego wartość można odczytać z noty katalogowej układu. Jest to prąd na tyle mały, że nie pozwala na wysterowanie niektórych urządzeń, np. przekaźnika. Konieczne staje się wtedy zastosowanie wzmacniacza – tranzystora, tak jak na płytce eksperymentalnej do tego kursu. Podobnie postępuje się w sytuacjach, gdy do wyjściowego portu ma być podłączone kilkanaście diod LED lub dioda dużej mocy.

Po tym kolejnym wstępnie teoretycznym przyszła pora na narysowania schematu z bramką, kompilację projektu oraz wgranie go do układu programowalnego. Posłużymy się środowiskiem WebPACK ISE 6.2, którego szczegółowa instalacja została przybliżona w poprzedniej części kursu. Nowy projekt

Rys. 1

Recommended Operation Conditions

Symbol	Parameter		Min	Max	Units
V_{CCINT}	Supply voltage for internal logic and input buffers	Commercial T_A = 0ºC to 70ºC	3.0	3.6	V
		Industrial T_A = –40ºC to +85ºC	3.0	3.6	V
V_{CCIO}	Supply voltage for output drivers for 3.3V operation		3.0	3.6	V
	Supply voltage for output drivers for 2.5V operation		2.3	2.7	V
V_{IL}	Low-level input voltage		0	0.80	V
V_{IH}	High-level input voltage		2.0	5.5	V
V_O	Output voltage		0	V_{CCIO}	V

Rys. 2

p	q	Y
0	0	0
0	1	0
1	0	0
1	1	1

rozpoczyna się od wybrania z menu *Start* zakładki *Programy*, następnie *Xilinx ISE* i w końcu *Project Navigator*. Po chwili zostanie uruchomione środowisku pokazane na **rysunku 3**. Rozpoczynamy pracę od wybranie pozycji *New project* z menu *File*. W pierwszym okienku kreatora wpisujemy nazwę projektu *(Project Name),* wybieramy folder, w którym zostanie on umieszczony *(Project location)* i wybieramy pracę ze schematem (*Top-Level Module Type*). Przykład wypełnienia został pokazany na **rysunku 4**. Po kliknięciu *Dalej* ukazuje się kolejne okienko. W polu *Device Family* należy wybrać rodzinę układu programowalnego (XC9500XL CPLDs), określić konkretny układ z tej rodziny w polu *Device* (XC9572XL), następnie wybrać rodzaj obudowy w polu *Package* (PC44), dalej ustawiamy pozycję *Speed Grade* na wartość odczytaną z obudowy zakupionego układu (patrz **rysunek 5**), najprawdopodobniej będzie to –10. Resztę pozycji pozostawiamy bez zmiany, przykład wypełnienia widoczny jest na **rysunku 6**. Po kliknięciu *Dalej* kreator umożliwia dodanie pliku źródłowego, w którym tworzony będzie schemat. Skorzystamy z tej możliwości poprzez kliknięcie przycisku *New Source*, a w kolejnym, otwartym oknie zaznaczamy pozycję *Schematic* i wpisujemy nazwę dla schematu (**rysunek 7**). Po upewnieniu się, że zaznaczona jest opcja *Add to Project* klikamy *Dalej* oraz *Zakończ*. Następnie dwukrotnie *Dalej* oraz *Zakończ*. Po tych operacjach mamy utworzony gotowy projekt, w którym narysujemy układ.

Jedno słowo komentarza – po co wprowadzać do kreatora zastosowany rodzaj obudowy? Jest to niezbędne, aby skorzystać z kreatora pozwalającego przypisać sygnały do konkretnych portów.

Bardzo prawdopodobne jest, że po zakończeniu pracy z kreatorem otworzy się okno edycji schematu pokazane na **rysunku 8**. Jeżeli nie zostanie ono otwarte lub w czasie pracy zostało zamknięte, można je wywołać z okna *Project Navigator*, klikając dwukrotnie na nazwie pliku schematu (**rysunek 9**).

Zdefiniowanie układu pozwalającego zbadać zachowanie bramki AND sprowadza się do kilku kroków. Po pierwsze, należy umieścić w schemacie bramkę, która ma zostać zbadana. Przechodzimy do zakładki *Symbols* i w polu *Categories* wybieramy pozycję *Logic*, a w polu *Symbols* zaznaczamy pierwszy element – *and2* i dodajemy bramkę (**rysunek 10**). W polu *symbols* znajdują się jeszcze inne bramki, które są opisany przy pomocy następującego klucza:

nzwXbY, gdzie:

– nzw – typ bramki,

– X – liczba wejść,

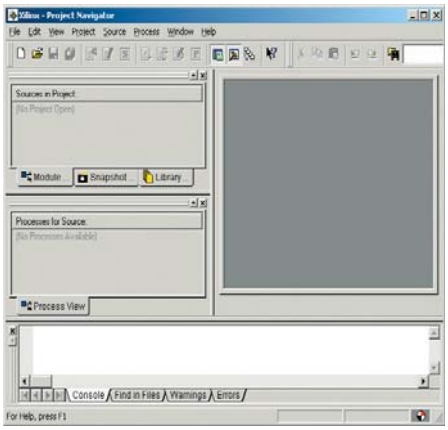
– bY – parametr *b* oznacza wejścia zanegowane, natomiast *Y* ich liczbę.

Przykładowo pozycja *or3b1* oznacza symbol bramki OR posiadającej trzy wejścia, z czego jedno jest zanegowane.

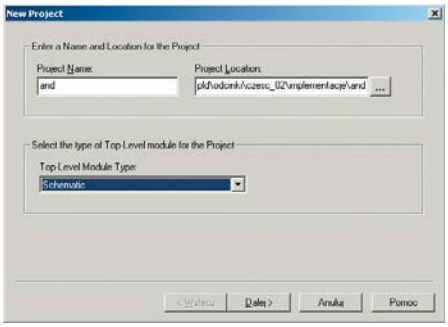
Narysowanie samej bramki to za mało i konieczne jest zdefiniowanie sygnałów wejściowych i wyjściowych. Przy okazji utworzymy markery (porty), które pozwolą wprowadzić i wyprowadzić te sygnały na zewnątrz układu CPLD. Zadania te są spełniane przez tzw. *I/O Markers* (które będę nazywał również portami ze względu na pewne analogie do portów mikrokontrolerów). Do projektu można dodać trzy typy markerów:

– *input* – oznacza port wejściowy umożliwiający wprowadzenie sygnału do układu programowalnego,

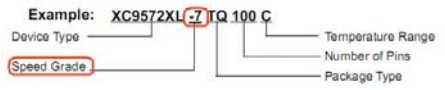
– *output* – oznacza port wyjściowy przeznaczony do wyprowadzania na zewnątrz prze-



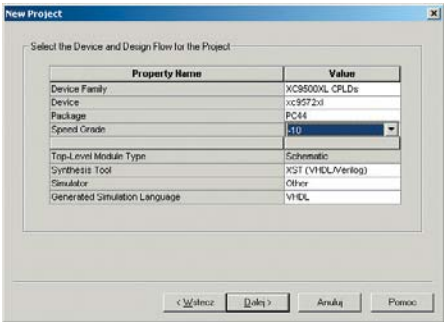
Rys. 3



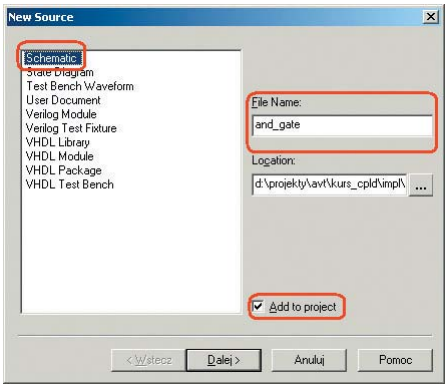
Rys. 4



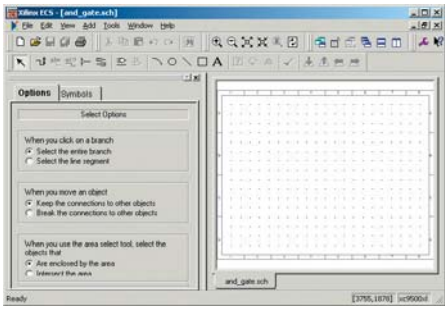
Rys. 5



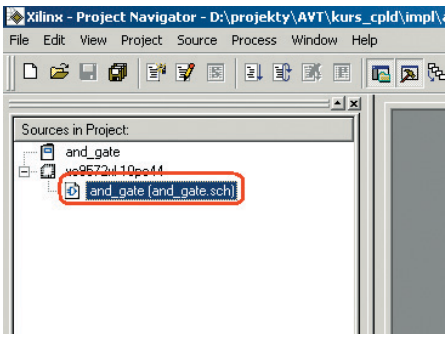
Rys. 6



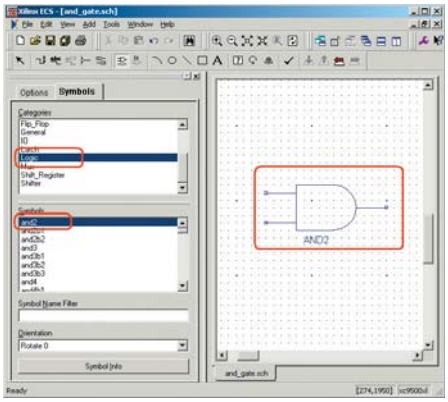
Rys. 7



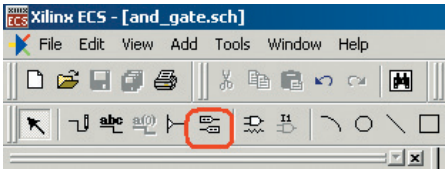
Rys. 8



Rys. 9

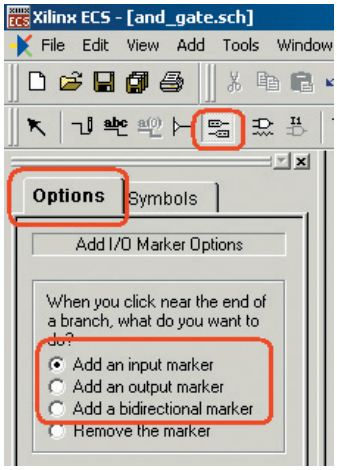


Rys. 10



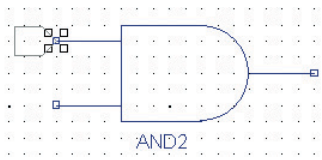
Rys. 11

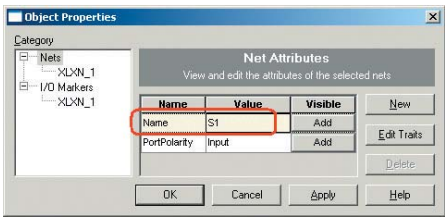
tworzonego przez CPLD sygnału cyfrowego,
– *bidirectional* – port dwukierunkowy.
W naszym prostym przykładzie będziemy potrzebować trzech portów: dwóch wejściowych pozwalających wprowadzić sygnał na wejścia bramki AND oraz portu wyjściowego pozwalającego wyprowadzić sygnał z bramki AND na zewnątrz układu. Nowy marker może zostać dodany na jeden z trzech sposobów:
– kliknięcie ikonki (**rysunek 11**),
– wciśnięcie kombinacji klawiszy: CTRL + G,
– wybranie z menu *Add* pozycji *I/O Marker*.
Po wybraniu jednej z tych możliwości, w zakładce *Options* ukażą się dostępne typy portów: *input*, *output* oraz *bidirectional* (*remove* służy do usuwania markerów) – **rysunek 12**. Zaznaczmy pozycję *input* i przesuńmy kursor na wejście bramki. W pewnym momencie pojawią się cztery kwadraciki wokół wejścia sygnalizujące poprawne połączenie, wtedy należy kliknąć i w ten sposób port zostanie dołączony do wejścia bramki (**rysunek 13**). Do drugiego wejścia bramki należy przypisać marker w sposób analogiczny. Pozostało jeszcze dodać port wyjściowy dołączony do wyjścia bramki – czynność ta odbywa się analogicznie z tym, że należy zaznaczyć opcję *output*. Klikając dwukrotnie na dodanych uprzednio portach można otworzyć okienko dialogowe i wpisać do niego dowolną nazwę portu (**rysunek 14**). Jest to bardzo przydatna cecha, z której warto korzystać. W dalszym etapie pracy będziemy przypisywać porty do fizycznych wyprowadzeń układu i właśnie te nazwy będą widoczne. W polu *PortPolarity* istnieje możliwość zmiany kierunku pracy portu (np. z wejściowego na wyjściowy, itp. Nasz port został ustawiony na etapie dodawania go do schematu, więc pozostawimy tą opcję bez zmian). Przy nazewnictwie portów będę starał się trzymać zasady: *nazwa_portu = nazwa_wyprowadzenia_układu_CPLD*. Pozwoli to od razu określić gdzie przypisać dany sygnał, ale podkreślam: nazwy te są DOWOLNE. Ukończony schemat ze zdefiniowanymi nazwami portów można zobaczyć na **rysunku 15**. Pozostaje zapisać zmiany i przystąpić do budowania projektu i programowania układu CPLD (można zamknąć edytor schematów).



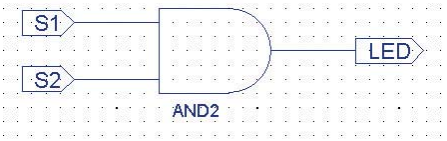
Rys. 12

Rys. 13





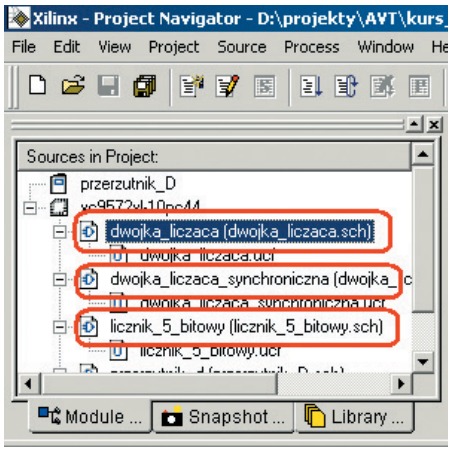
Rys. 14



Rys. 15

Rys. 16

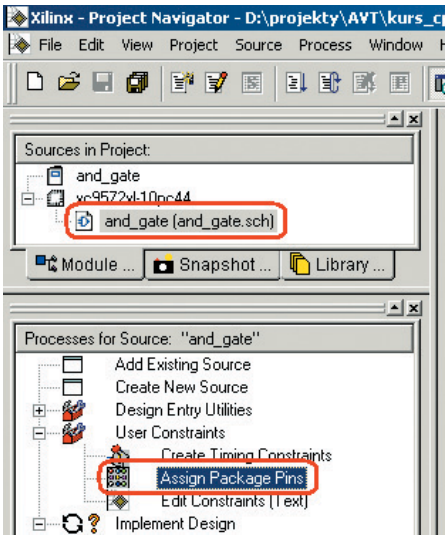
Rys. 17

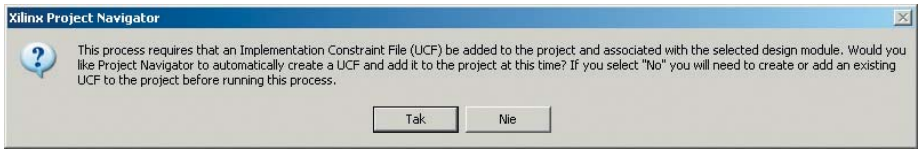


Edytor schematów – kilka uwag

Pozwolę sobie umieścić jeszcze kilka uwag dotyczących korzystanie z edytora schematów, być może okażą się pomocne w dalszej pracy.
Jeżeli w czasie rysowania schematu popełnimy błąd, to można go łatwo poprawić. Wystarczy kliknąć na niepotrzebnym elemencie, co zaznaczy go kolorem czerwonym i usunąć wciskając przycisk *Del* na klawiaturze (względnie z menu *Edit* wybrać pozycję *Delete*).
Praca z edytorem bezpośrednio po jego uruchomieniu prawdopodobnie nie będzie zbyt wygodna ze względu na małe powiększenie. Można je regulować przy pomocy ikonek na pasku menu lub poprzez klawisze skrótów:
– F8 – powiększenie widoku,
– F7 – pomniejszenie widoku.
Czasami na schemacie widoczne są przekłamania graficzne, np. widać usunięte elementy. Problem rozwiązuje odświeżenie widoku, którego dokonuje się klawiszem F5.
Czytelnicy prawdopodobnie niejednokrotnie zobaczą błąd z **rysunku 16**. Nie jest to nic poważnego, oznacza jedynie, że schemat jest nazbyt „ściśnięty". Problem rozwiązuje się poprzez rozsunięcie elementów i ponowne narysowanie połączenia lub elementu.
Należy zachować ostrożność podczas usuwania markerów ze schematów. Może być tak, że zostaje on usunięty, ale program nie pozwala dodać ponownie markera o tej samej nazwie. Dobrą praktyką jest kasowanie najpierw markera, a dopiero później połączenia, które jest z nim związane.
Warto jeszcze wspomnieć, że projekt możne zawierać kilka schematów. W naszym przypadku pozwoli to na zgromadzenie ćwiczeń dotyczących jednego elementu w pojedynczym projekcie. Dodanie drugiego, trzeciego czy kolejnego schematu przebiega analogicznie do opisanego wcześniej. Wystarczy wybrać z menu *Project* pozycję *New Source* i dodać nowy schemat. W oknie *Project Navigator* pojawi się dodatkowy schemat, który można edytować przez dwukrotne kliknięcie (**rysunek 17**). Po jego zaznaczeniu zyskujemy możliwość indywidualnego przypisania wyprowadzeń układu CPLD do markerów z danego schematu, a także wygenerowania dla niego pliku wynikowego (przeznaczonego do zaprogramowania CPLD).
Edytor schematów daje możliwość sprawdzenia, czy nie popełniono błędów podczas rysowania. Sprawdzana jest co prawda tylko poprawność połączeń, a nie błędy logiczne, jednakże umożliwia to szybsze wykrycie błędów, niż miałoby to miejsce na etapie kompilacji. Sprawdzenia takiego dokonuje się poprzez wybranie z menu *Tools* pozycji *Check Schematic*.

Rys. 18





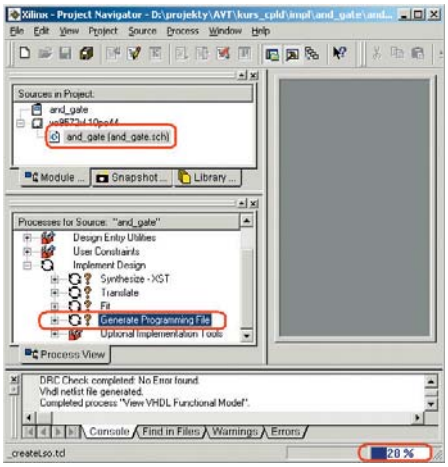
Rys. 19

Ostatnią rzeczą na którą chciałbym zwrócić uwagę, są wymuszenia stałe, takie jak 0 oraz 1. W edytorze schematów mogą być dołączone do wybranych wejść elementów lub portów wyjściowych. Znajdują się one w kategorii *General* i mają oznaczenie *vcc* (jedynka logiczna) oraz *gnd* (zero logiczne). Podłącza się je, tak jak wszystkie inne komponenty.

Bramki logiczne – ciąg dalszy

Dokończmy badanie bramki AND. W oknie *Project Navigator* zaznaczamy schemat z projektem, znajdujemy na liście pozycję *User Constraints*, rozwijamy ją i dwukrotnie klikamy na *Assign Package Pin* (**rysunek 18**). Po chwili otworzy się kreator, w którym możliwe będzie przypisanie utworzonych w edytorze schematów markerów do fizycznych wyprowadzeń układu scalonego (prawdopodobnie pojawi się komunikat pokazany na **rysunku 19** – zgadzamy się na dodanie niezbędnego do projektu pliku poprzez kliknięcie przycisku *Tak*). W nowo otartym oknie widoczna jest wybrana na etapie tworzenia projektu obudowa – do niej można przypisać zdefiniowane porty. Po najechaniu na dany port myszą, po chwili pokaże się numer wyprowadzenia oraz pełniona funkcja – **rysunek 20**. Przy odpowiednio dużym powiększeniu, program wyświetli również numeracje portów.

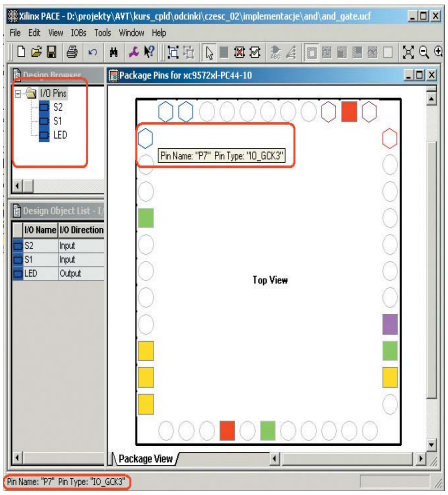
Na tym samym rysunku, w lewej jego części, widać utworzone na schemacie porty. Przypisanie sygnału do wybranego wyprowadzenia opiera się na metodzie „przeciągnij i upuść". Wystarczy kliknąć na wybranym sygnale (w lewej części okna) i przeciągnąć go do wybranego wyprowadzenia – na liście zmieni się jego ikona, a port zmieni kolor. Powstaje oczywiste pytanie: do których wyprowadzeń przypisać te sygnały? Odpowiedzi należy szukać na schemacie w artykule opisującym płytkę testową. Widać na nim, że przycisk S1 jest podłączony do wyprowadzenia 29 (tu przeciągniemy marker S1) a przycisk S2 – do portu 28. Można również wykorzystać przycisk S3 zamiast S1 czy S2. Pozostał ostatni sygnał – Czytelnik może wybrać sobie dowolną diodę LED i przeciągnąć sygnał *LED* do stosowanego wyprowadzenia. Ja zdecydowałem się na diodę LED1 (na płytce oznaczona jako D2) – czyli pin 18. Można zapisać zmiany i zamknąć kreator. Pragnę zauważyć, że raz przypisany sygnał można bez problemu zmienić – wystarczy go przeciągnąć z wyprowadzenia, do którego został przypisany, w nowe miejsce.



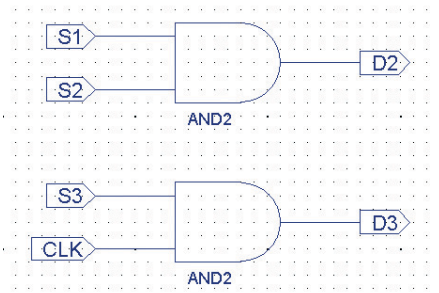
Rys. 21

Powracamy do okna programu Project Navigator i na liście po lewej stronie zaznaczamy schemat z projektem, a w poniższym oknie rozwijamy pozycję *Implement Design*. Następnie klikamy dwukrotnie na pozycji *Generate Programming File* i czekamy aż pasek postępu na dole okna osiągnie wartość 100% (**rysunek 21**). Jeżeli w schemacie nie popełniliśmy błędów, to przy kliknięte pozycji pojawi się zielony znaczek. Pozostaje wtedy rozwinąć pozycję *Generate Programming File* i dwukrotnie kliknąć na pozycji *Configure Device*, co otworzy znany z poprzednich części program iMPACT – plik wynikowy *.jed znajduje się w katalogu z projektem. Wystarczy wczytać go do programu jednokrotnie, a po ewentualnej rekompilacji jego zawartość zostanie odświeżona automatycznie w czasie programowania (trzeba będzie jedynie kliknąć *Tak* w komunikacie informującym, że zawartość pliku uległa zmianie).

Po zaprogramowaniu układu nasza praca zasadniczo kończy się – pozostaje obejrzeć jej efekty. Wszystkie diody na płytce powinny być wygaszone, natomiast po naciśnięciu S1 powinna zaświecić się dioda D2. To samo dzieje się po naciśnięciu S2 lub obu przycisków jednocześnie. Czy jest to zgodne z oczekiwaniami? Puszczenie przycisków sprawia, że na obu wejściach bramki są stany wysokie i w efekcie na wyjściu też jest stan wysoki. Dioda jest zaświecana stanem niskim, więc w obecnej sytuacji będzie wygaszona. Wciśniecie dowolnej kombinacji przycisków sprawi, że na wejście bramki AND zostanie doprowadzony przynajmniej jeden stan niski (lub dwa) i bramka da na wyjściu stan niski, a więc włączy się dioda.



Rys. 20



Rys. 22

Na podstawie tego prostego eksperymentu możemy stwierdzić, że bramka pracuje prawidłowo.

Dwie bramki AND

Zachęceni pierwszym sukcesem spróbujmy dodać drugą bramkę AND. Zacznijmy od otworzenia edytora schematu (dwukrotne kliknięcie na nazwie schematu w oknie *Project Navigator*). Zwróć uwagę, że są tu teraz dwa pliki: plik schematu z rozszerzeniem **.sch* oraz plik definiujący wyprowadzenia (ten z rozszerzeniem **.ucf*).

Po otwarciu edytora dodaj drugą bramkę i przypisz do niej porty, tak jak powyżej, według **rysunku 22**. Zapisz zmiany i powróć do *Project Navigator*. Na schemacie pojawiły się nowe sygnały, więc skorzystamy z kreatora do ich przypisania do wyprowadzeń układu scalonego – narzędzie *Assign Package Pin*. Powinno się ono bez problemu uruchomić, wyświetlając jedynie błędy informujące o braku sygnału *LED*, który został zamieniony na D2. W sporadycznych wypadkach może się okazać, że nie uda się uruchomić tego kreatora. Istnieją w takim wypadku dwa wyjścia. Po pierwsze, można usunąć plik **.ucf* z projektu (klikając na jego nazwie prawym przyciskiem myszy i wybierając pozycję *Remove*, a następnie usuwając z katalogu z projektem plik z rozszerzeniem **.ucf*) i utworzyć go ponownie. Inna opcją jest ręczna edycja pliku dokonywana poprzez kliknięcie pozycji *Edit Con-*

straints (pod *Assign Package Pin*). Jest to bardzo dobre rozwiązanie, gdy usunięcie pliku wymagałoby żmudnego przypisywania dużej liczby sygnałów od nowa. Chciałbym pokazać Czytelnikom jak ręcznie edytować ten plik. W tym celu klikamy dwukrotnie na pozycji *Edit Constraints,* w wyniku czego w sąsiednim okienku otworzy się plik tekstowy – **rysunek 23**. Widać na nim trzy linijki z przypisanymi wyprowadzeniami. Budowa pojedynczego wpisu jest bardzo prosta, gdyż składa się ze słowa kluczowego *NET*, następnie podanej w cudzysłowach nazwy sygnału, dalej słowa kluczowego *LOC*, znaku równości i na końcu, w cudzysłowach, litery *P,* za którą podane jest wyprowadzenie. Linia kończy się średnikiem. Wpis *LED* zastąpimy słowem *D2*. Następnie dodamy trzy linijki poniżej już obecnych (wielkość liter ma znaczenie!):

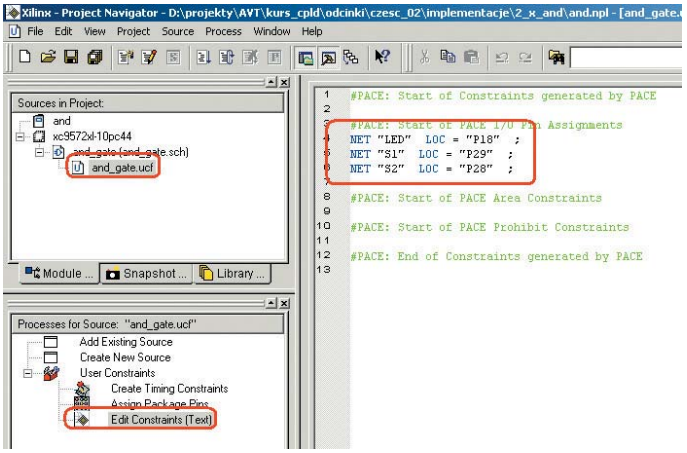
NET „S3" LOC = „P27" ;
NET „CLK" LOC = „P7" ;
NET „D3" LOC = „P20" ;

Spojrzenie na schemat płytki testowej pozwala ustalić, że wejścia nowej bramki zostały przypisane do przycisku S3 oraz sygnału *CLK* (pochodzącego z przestrajanego generatora). Wyjście jest podawane na diodę LED (D3). W tym projekcie działanie S1, S2 oraz D2 jest identyczne (wciśniecie dowolnej kombinacji pozwala włączyć diodę LED). Inaczej sytuacja wygląda z diodą D3, która powinna migotać (jeżeli nie, należy pokręcić potencjometrem – zbyt duża częstotliwość generatora nie pozwoli zaobserwować migania). Interesujące jest zachowanie diody D3 po naciśnięciu przycisku S3 – przestaje ona migać. Przerwij na chwilę dalsze czytanie i spróbuj samodzielnie odpowiedzieć na pytanie: dlaczego tak się dzieje i jaki z tego płynie wniosek?

Odpowiedź jest prosta. Bramka AND daje na wyjściu zawsze zero, gdy na przynajmniej jednym wejściu występuje zero. Wciskając S3 podajemy właśnie stan niski, a na drugim wejściu może być albo stan niski (dwa stany niskie dają zero na wyjściu) albo stan wysoki, co też daje stan niski na wyjściu. Jest to bardzo użyteczna właściwość, pozwalająca „odłączyć" sygnał taktujący od układu. Rozwiązanie to spotkać można we wszelkiej maści częstotliwościomierzach. W ramach ćwiczenia zachęcam do wykonania następujących badań:

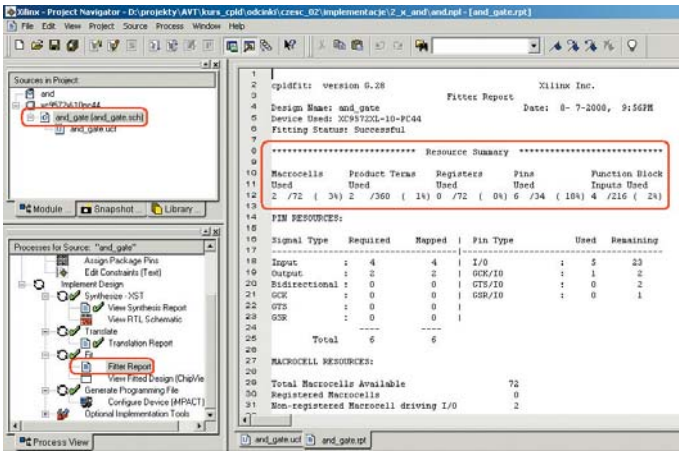
– spróbuj opracować taki układ, który będzie przepuszczał sygnał zegarowy w momentach, gdy S3 jest wciśnięty,
– zrealizuj powyższe ćwiczenie, ale z bramką OR,
– sprawdź, czy istnieje możliwość realizacji tego zadania z bramką Ex-OR (w bibliotece elementów nazywa się ona *XOR*),
– zbadaj również pozostałe bramki (NAND, NOR, itp.), również te z trzema wejściami oraz z zanegowanymi wejściami i narysuj dla nich tabelki prawdy. Czy są one zgodne z tym, co podaje literatura?

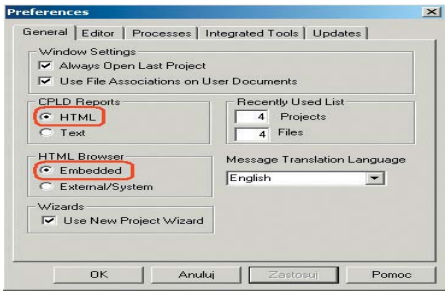
Na tym kończy się opis środowiska WebPACK ISE. Dalej skupimy się na tworzeniu układów bez szczegółowego opisu ich rysowania i kompilacji. Wspomnę jeszcze tylko o generowaniu raportów, które są bardzo użytecznym narzędziem



Rys. 23

Rys. 24





Rys. 25

pozwalającym określić ***ile*** jeszcze można zmieścić w układzie CPLD. Wygenerowany raport widoczny jest na **rysunku 24**. Można z niego odczytać, że do zaimplementowania projektu z dwom bramkami AND wykorzystane zostało 3% zasobów. Wystarczy dwukrotnie kliknąć na pozycji *Fitter Report* (rysunek 24). Domyślnie generowany jest on jako plik HTML, który czasami nie jest wyświetlany prawidłowo. Jeżeli otrzymany raport jest „pusty", proponuję przełączyć program, aby generował go w trybie tekstowym. Sprowadza się to do wybrania z menu *Edit* pozycji *Preferences*... W otwartym okienku zmieniamy pozycję *HTML* na *Text* (**rysunek 25**) i upewniamy się, że zaznaczona jest opcja *Embedded* (raport otwierany będzie w oknie *Project Navigator*).

Asynchroniczny przerzutnik RS

Najprostszym chyba układem, jaki można zbudować w oparciu o bramki jest przerzutnik RS. W literaturze można spotkać dwie wersje takiego przerzutnika: zbudowanego w oparciu o bramki NOR lub NAND. Różnią się one aktywnymi stanami, tzn. w przypadku bramek NAND ustawianie i zerowanie wyjścia odbywa się przy pomocy stanów niskich (odwrotnie niż dla NOR). Zbudujemy wersję opartą o bramki NAND. Załóżmy również, że zastosu-

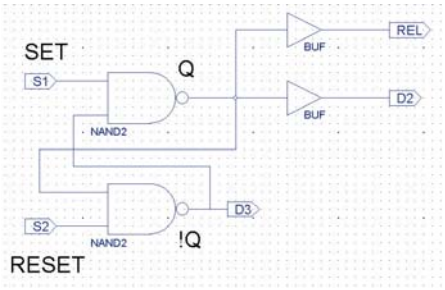
jemy ten prosty układ do sterowania przekaźnika, co umożliwi budowę funkcjonalnego, aczkolwiek prostego sterownika. Naciśnięcie jednego przycisku włączy przekaźnik, a drugiego – wyłączy. Dołożymy do tego również dwie diody LED (na wyjście Q oraz !Q, gdzie ! oznaczać będzie symbol negacji). Z narysowaniem przerzutnika nie będzie problemu, gdyż bez problemu można go odszukać w książce lub Internecie, jednak jak podłączyć dwa porty wyjściowe go wyjścia Q? Napotkamy tutaj na dwie trudności. Pierwszą z nich może być nieudana próba podłączenia drugiego portu, który po prostu nie daje się narysować... Jeżeli Czytelnik wymyśli sposób na obejście tego problemu (wystarczy narysować fragment połączenia za pomocą narzędzia CTRL+W i na jednym końcu dodać port), to okaże się, że nie można przypisać mu innej nazwy... Na szczęście istniej prostsze wyjście z tej sytuacji – wystarczy użyć bufor (element *Buf* dostępny w kategorii *Buffer*). Po jego dodaniu do schematu, możliwe jest dodanie drugiego portu. Schemat tak utworzonego przerzutnika widoczny jest na **rysunku 26**.

Po zaprogramowaniu układu pozostaje sprawdzić, czy pracuje on zgodnie z założeniami. Naciśnięcie przycisku S1 wyłącza diodę D2 (stan wysoki na wyjściu) i włącza przekaźnik, natomiast przycisk S2 wyłącza diodę D3 (stan wysoki na !Q) i wyłącza przekaźnik (stan niski na wyjściu Q). Otrzymaliśmy w ten sposób najprostszy element pamiętający. Naciśnięcie jednocześnie S1 oraz S2 jest zabronione. Nie spowoduje to uszkodzenia układu, jednakże jego zachowanie będzie bezsensowne – wyjście Q oraz !Q będą miały ten sam stan, pozornie przecząc prawom logiki boolowskiej – wyjście zanegowane musi mieć stan przeciwny do wyjścia niezanegowanego.

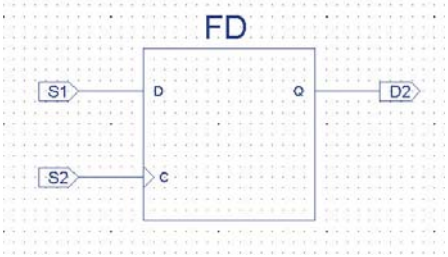
Jeżeli przekaźnik nie działa, być może zasilanie było podłączone odwrotnie? Taka sytuacja może prowadzić do uszkodzenia tranzystora T5 i konieczna będzie jego wymiana.

Przerzutnik D

Popularnym elementem stosowanym w technice cyfrowej jest przerzutnik typu D. Jego działanie jest bardzo proste i sprowadza się do przeniesienia poziomu z wejścia na wyjście w momencie podania zbocza narastającego na wejście zegarowe. Używając mniejszej liczby żargonowych pojęć można powiedzieć, że poziom obecny na wejściu pojawi się na wyjściu w sytuacji, gdy wejście zegarowe zmieni swój stan z niskiego na wysoki.

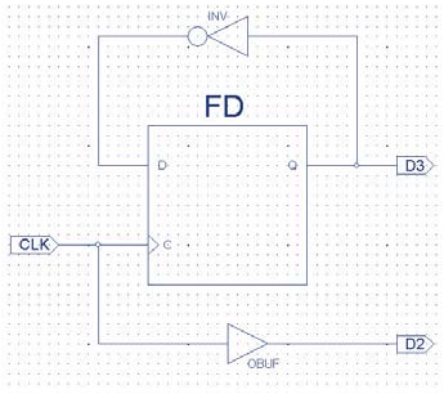


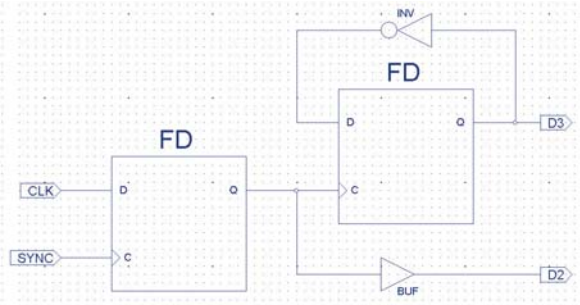
Rys. 26



Rys. 27

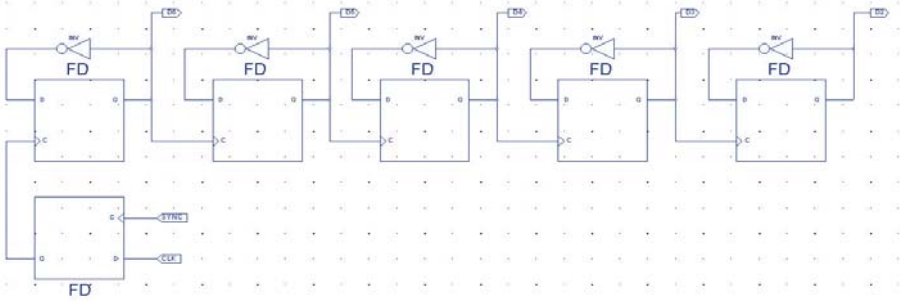
Rys. 28





Rys. 29

Rys. 30

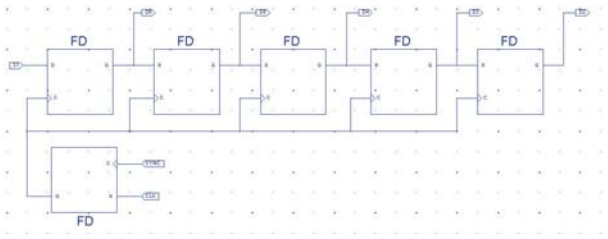


Sprawdzenie pracy przerzutnika D jest możliwe po narysowaniu bardzo prostego schematu – **rysunek 27**. Naciśnięcie przycisku S1 (wejście danych) nic nie zmienia – pomimo zmiany na wejściu danych wyjście pozostaje bez zmian. Chcąc zmienić stan diody należy podać na przerzutnik sygnał zegarowy, konkretnie zbocze narastające. Następuje to poprzez naciśnięcie S2 (stan niski na wejściu zegarowym) i następnie przez jego puszczenie (stan niski zamienia się w stan wysoki – otrzymujemy zbocze narastające). Jeżeli w momencie zwalniania S2 wciśnięty był S1, to na wyjście przeniesiony zostanie poziom niski – dioda D2 zostanie włączona. W przeciwnym przypadku na wyjściu pojawi się stan wysoki – dioda D2 będzie wygaszona. Przerzutnik ma symbol FD i można go znaleźć w kategorii *Flip_Flop*.

Przerzutnik D – dwójka licząca

Przyjrzyjmy się teraz innym zastosowaniom przerzutnika D. Ciekawym układem jest tzw. dwójka licząca. Budowa takiego układu sprowadza się do sprzężenia wejścia przerzutnika z zanegowanym wyjściem. Efektem jest podzielenie częstotliwości podanej na wejście zegarowe przez dwa. Zbudujmy układ z **rysunku 28**. Sygnał pochodzący z przestrajanego generatora (pin 7) został podany na diodę D2 oraz wejście zegarowe przerzutnika. Wyjście przerzutnika zostało również wyprowadzone – na diodę D3 (pin 20). Układ ten powinien sprawić, że dioda D3 będzie migać dwa razy wolniej. Okazuje się niestety, że tak nie jest. Wszechobecność zakłóceń sprawia, że na wejściu CLK pojawiają się bardzo krótkie szpilki. Są one widoczne właśnie w tym układzie i objawiają się tym, że dioda D3 czasami nie zmienia swojego stanu, chociaż powinna. Rozwiązaniem jest synchronizacja wejścia CLK przy pomocy generatora kwarcowego i... przerzutnika D! Na **rysunku 29** przedstawiono bardziej rozbudowany układ. Pierwszy przerzutnik pełni rolę synchronizacyjną. Sygnał CLK pochodzi z przestrajanego generatora, natomiast sygnał SYNC z generatora kwarcowego (pin 5). Bardzo krótkie impulsy zostaną pominięte, przeniesienie poziomu logicznego z wejścia na wyjście następuje tylko przy zboczu narastającym pochodzącym z generatora. Ten bardzo prosty przykład dobitnie pokazuje, jak ważne są szczegóły niewidoczne gołym okiem i często powodują one nieprawidłowe działania układu. Nie ma to, oczywiście, nic wspólnego z „magią" :).

Czy proste rozwiązanie z rysunku 28 nie ma



Rys. 31

-racji bytu? Zamiast odpowiadać na to pytanie, zachęcam do prostego eksperymentu. Załaduj ponownie przykład z rysunku 28, połącz wyprowadzenie 7 (CLK) oraz 10 (masa) w podstawce kondensatorem 100pF i zobacz czy coś się zmieniło. Uważaj, aby nie zrobić zwarcia!

Przerzutnik D – licznik (dzielnik) n-bitowy

Przerzutnik D jest bardzo interesującym elementem. Można na nim zbudować również licznik asynchroniczny o dowolnej pojemności. Opisana powyżej dwójka licząca ma tę właściwość, że dzieli sygnał zegarowy przez dwa. Łącząc kaskadowo np. 5 takich dwójek otrzymuje się podział częstotliwości przez dwa do potęgi piątej, czyli przez 32. Na wyjściu Q kolejnych przerzutników sygnał zmienia się dwukrotnie wolniej w stosunku do poprzedniego przerzutnika. Ta właściwość sprawia, że układ taki liczy w naturalnym systemie dwójkowym. Na **rysunku 30** pokazany został bardziej złożony schemat zawierający pięć dwójek liczących oraz przerzutnik służący do synchronizacji sygnału zegarowego.

Po przypisaniu wyprowadzań stosownie do nazw markerów, dioda D6 jest najmniej znacząca (LSB), a D2 najbardziej (MSB). Czy zauważasz pewną „nieprawidłowość"? Świecąca dioda odpowiada jedynce, podczas gdy powinno być odwrotnie. Oznacza to ni mniej ni więcej, że wyjścia licznika są zanegowane, chociaż nie robimy tego jawnie – na schemacie nie ma bramek NOT przed diodami. Odpowiada za to pewna charakterystyczna cecha liczenia w systemie binarnym, mianowicie po zanegowaniu kolejnych liczb otrzymujemy... zliczanie od tyłu! Pojawia się więc wniosek, że licznik utworzony z przerzutników D zlicza od tyłu.

Zachęcam Cię, w ramach ćwiczenia, do budowy licznika liczącego w przód. Możesz się pokusić o samodzielne rozwiązanie tego problemu lub odnalezienie w książce bądź Internecie gotowego rozwiązania (np. na przerzutnikach J-K).

Rejestr przesuwający – przerzutniki D

W oparciu o przerzutniki D można stworzyć również rejestr przesuwający sygnał wejściowy w takt sygnału zegarowego. Wprowadzane zera i jedynki będą przemieszczać się przez kolejne komórki rejestru (kolejne przerzutniki). "Poruszające się" bity można łatwo zaobserwować, gdyż wyjścia wszystkich przerzutników są podłączone do diod LED. Zbudujmy układ w oparciu o **rysunek 31**. Przycisk S1 pozwala wprowadzić do układu logiczne zero (zaświecić diodę), które następnie „wędruje" przez rejestr w takt przestrajanego generatora. Po zwolnieniu S1 do wejścia rejestru trafiają jedynki (zgaszenie diody).

Rejestr przesuwający

Zamiast budować od podstaw rejestr przesuwający można wykorzystać gotowy element z biblioteki. Znajduje się on w kategorii *Shift_Register*. Przykładowy schemat rejestru przedstawionego wcześniej pokazano na **rysunku 32**. Charakterystyczną jego cechą jest magistrala oznaczona pogrubioną linią. Ten przykład warto zrealizować, by przyswoić sobie zasady obowiązujące przy rysowaniu magistral.

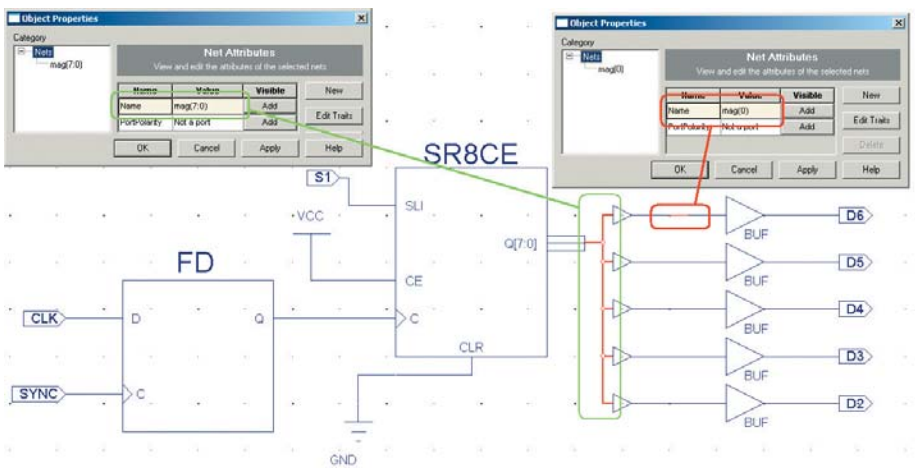
Najpierw dodajmy do schematu 5 elementów *Bus Tap* (z menu *Add* lub paska menu), które pozwolą wyprowadzić pojedynczą linię z magistrali, która zawiera ich osiem (bo tyle jest wyjść rejestru: Q7, Q6,...,Q0). Łączenie wyjścia rejestru z wejściami *Bus Tap* jest identyczne jak rysowanie normalnych połączeń – linia zostanie automatycznie zamieniona na magistralę. Po narysowaniu magistrali warto kliknąć na nią dwukrotnie, co otworzy okienko dialogowe. W polu *Value* przy pozycji *Name* należy podać nazwę i rozmiar magistrali według wzoru:

nazwa_mag(nr_bitu_MSB:nr_bitu_LSB).

Zwracam uwagę, że pośrodku jest dwukropek. W tym przykładzie wpiszmy tam: *mag(7:0)*. W tym momencie określamy, że na magistrali są linie *mag7*, *mag6*,...,*mag0* podłączone odpowiednio do wyjść rejestru *Q7*, *Q6*,...,*Q0*. Dalszy etap pracy z magistralą sprowadza się do narysowania fragmentu linii (CTRL + W) wychodzącej z *Bus Tap*, wstawienia buforów i podłączenia do nich portów wyjściowych (tu podłączone zostaną diody LED). Teraz należy kliknąć kolejno na wszystkich liniach dodanych przed buforami i wprowadzić w polu *Name* nazwy sygnałów, które mają zostać pobrane z magistrali, czyli dla D6 będzie to *mag(0)*, ..., D2 – *mag(4)*. Po tych zabiegach można skompilować projekt i go uruchomić. Dziaa identyczne jak rejestr na przerzutnikach D.

Pytanie: co by się stało, gdyby nazwą magistrali było *mag(0:7),* zamiast *mag(7:0)*? Powiem, że nie będzie błędu i projekt da się skompilować i uruchomić. Z jakim efektem? Cóż, sprawdź sam:). Czy potrafisz określić, co się stało?

Rys. 32



Licznik dwójkowy

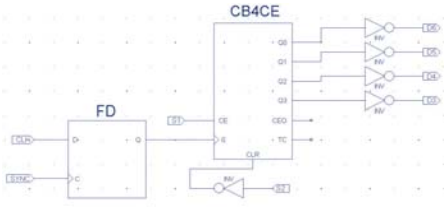
W bibliotece elementów znajdują się gotowe liczniki, więc nie trzeba ich budować samodzielnie na przerzutnikach. Na **rysunku 33** znajduje się przykładowy schemat. Zawarto tu jednak dwie subtelne modyfikacje: wykorzystano wejścia *CE* oraz *CLR*. Pierwsze z nich musi mieć stan wysoki, aby licznik liczył. Wciśnięcie S1 spowoduje podanie stanu niskiego i zliczanie ulegnie zatrzymaniu. Po zwolnieniu S1 będzie ono kontynuowane od chwili zatrzymania. Takie wstrzymywanie zliczania bywa nazywane bramkowaniem.

Wciśnięcie S2 wymusza na wejściu *CLR* stan wysoki (ze względu na obecności bramki NOT). Efektem będzie wyzerowanie licznika i zliczanie od zera.

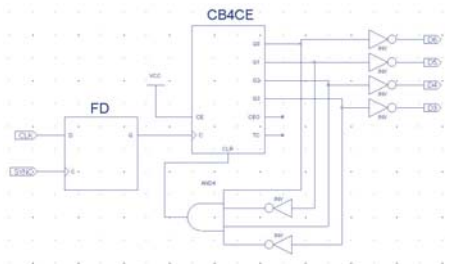
Licznik modulo N

Licznikiem modulo n określa się licznik, którego wyjścia przyjmują n stanów. Co zrobić kiedy potrzebny jest nietypowy licznik, np. 5? W takich sytuacjach wykorzystuje się bram-

Rys. 33



kę AND oraz dowolny licznik dwójkowy z wejściem resetującym. Określamy, który stan (liczba na wyjściu) ma się pojawić jako ostatni i zapisujemy na kartce w formie binarnej następny, już niepożądany. Wstawiamy licznik do schematu i wyjścia, na których jest zero (w niepożądanym stanie) podłączamy przez inwertery do bramki AND. Pozostałe wyjścia (te z jedynkami) wprowadzamy bezpośrednio na bramkę AND. Wyjście bramki podajemy na wejście resetujące licznika.



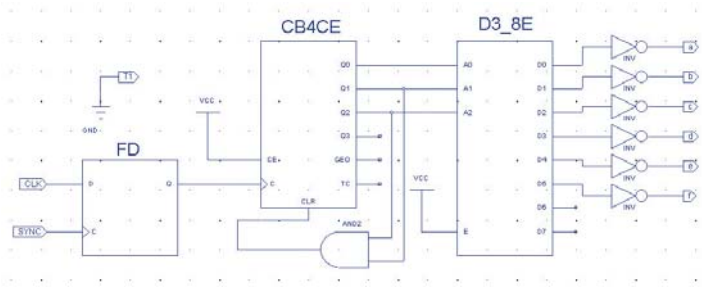
Rys. 34

Załóżmy, że chcemy otrzymać licznik modulo 5, na wyjściu którego pojawi się pięć stanów: 0, 1, 2, 3, 4, po czym zliczanie ma rozpocząć się od początku. W tym celu wykorzystamy poznany już licznik 4-bitowy i 4-wejściową bramkę AND. Wyjścia licznika zostały podłączone do diod LED i zanegowane. Ostatnim dopuszczalnym stanem jest 4, więc liczba 5 powinna resetować licznik. Ma ona następującą reprezentację w systemie binarnym: 0101. W związku z tym dwa wyjścia zostaną doprowadzone do bramki przez negator, a dwa bezpośrednio. Idea działania jest prosta – pojawienie się liczby 0101 spowoduje, że zera zamienią się w jedynki (bo są podawane przez negatory) a jedynki zostaną bezpośrednio wprowadzone na bramkę. W wyniku tego na wejściach bramki będą 4 jedynki i bramka poda na wyjście impuls (pojawi się na chwilę jedynka) i licznik ulegnie zresetowaniu – zliczanie rozpocznie się od początku. Zliczanie odbywa się w naturalnym kodzie binarnym, jedynce odpowiada zaświecona dioda. Schemat takiego układu pokazano na **rysunku 34**.

Okazuje się jednak, że można pójść na skróty. Jeżeli zliczanie odbywa się w naturalnym kodzie binarnym (tzn. stany następują kolejno: 0, 1, 2, 3, 4, 5, itd.), to nie trzeba do bramki doprowadzać „zer" przez negatory. Wystarczy podłączyć linie, na których pojawią się jedynki w pierwszym, niedozwolonym stanie (w przykładzie tym stanem jest 5). Zachęcam do wymiany bramki na 2-wejściową - urządzenie też pracuje prawidłowo.

Dlaczego taka oszczędność jest możliwa? Ponieważ te dwie jedynki są unikalne – w żadnym z poprzednich stanów nie pojawiają się jednocześnie jedynki na wyjściu Q0 oraz Q2 licznika.

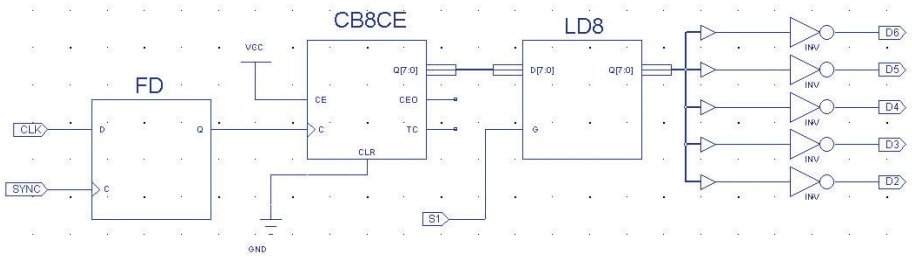
Rys. 35



Dekodery typu 1 z N

Dekoder jest elementem, który posiada n wejść adresowych, 2^n wyjść oraz wejście aktywująca *E*. Na wejścia adresowe zadaje się adres w postaci liczby binarnej i na jednym odpowiednim wyjściu pojawia się logiczna jedynka, a na pozostałych - zera. Przykład wykorzystania tego elementu pokazano na **rysunku 35** – jest to dekoder *1 z 8*. Wejścia adresowe zostały dołączone do licznika modulo 6, więc na wyjściach dekodera sekwencyjnie przemieszcza się jedynka logiczna. Wyjścia zostały zanegowane i dołączone do poszczególnych segmentów wyświetlacza 7-segmentowego. Obecność inwertera sprawia, że „wędruje" zero a nie jedynka i w efekcie włączony jest tylko jeden segment wyświetlacza. Całość tworzy efekt biegającej po wyświetlaczu kreski, a szybkość można regulować potencjometrem. Zwracam uwagę na pewien szczegół – port *T1* dołączony do masy. Jest on przypisany do tranzystora T1 i podanie na jego bramkę logicznego zera włącza wyświetlacz. Gdybyś dodał, Czytelniku, drugi port przeznaczony dla tranzystora T2 kreska „biegałaby" po obu wyświetlaczach, co jest łatwe do sprawdzenia.

Rys. 36



Zatrzaski

Zatrzask jest elementem pamiętającym – pozwala zachować stan podany na wejście. W przykładzie pokazanym na **rysunku 36** widać znany już licznik oraz dodany do niego zatrzask (element LD8 z kategorii *Latch*). Kiedy na wejściu *G* panuje stan wysoki, Latch staje się „przezroczysty" – dane podane na wejście są natychmiast przenoszone na wyjście. W podanym przykładzie widać wtedy, że licznik liczy, o czym świadczy zmieniający się stan diod LED. Po naciśnięciu S1 na wejściu *G* pojawia się stan niski i zatrzask przechodzi w stan pamiętania, pokazując stan licznika z chwili, gdy naciśnięto S1. Zatrzask przechowuje w ten sposób informację i zachowuje się jak komórka pamięci. Puszczenie S1 sprawia, że przestaje on pamiętać i prezentuje wyjście licznika. Warto zauważyć, że w przeciwieństwie do licznika z bramkowaniem, tutaj informacja na wejściu zatrzasku zmienia się. Element ten można było spotkać, np. w systemach mikroprocesorowych wyposażonych w pamięć zewnętrzną – zatrzask przechowywał fragment adresu pozwalając zaoszczędzić porty procesora. Przy pomocy portu 8-bitowego, zatrzasku i linii sterującej możliwe było podanie na wejścia adresowe pamięci EPROM adresu 16-bitowego.

Podsumowanie

W tej części kursu pokazane zostały elementarne układy spotykane w technice cyfrowej. Ich praktyczna przydatność jest niewielka. Mam jednak nadzieję, że po lekturze i wykonaniu ćwiczeń, Czytelnicy oswoją się z układami CPLD, a najmłodsi będą mieli lepszy pogląd na działanie układów cyfrowych. Mam nadzieje, że obsługa środowiska WebPACK ISE nie jest już „czarną magią" i Czytelnicy będą w stanie przetestować pomysły swoje i z literatury. Eksperymenty są najlepszą drogą do zdobywania wiedzy i znajdowania odpowiedzi na wyłaniające się pytania.

Chciałbym prosić Czytelników o wysyłanie swoich uwag i propozycji na mój adres mailowy (**jakub.borzdynski@elportal.pl**) Będę potwierdzał odbiór wszystkich odebranych maili. Nie obiecuję, że na wszystkie odpowiem, jednakże na pewno będę czytał całą korespondencję i wyciągał wnioski. Takie sprzężenie zwrotne pozwoli mi lepiej przygotowywać kolejne partie materiału. Jeżeli w ciągu trzech dni nie otrzymasz potwierdzenia, wyślij maila ponownie, ale tym razem na adres: **jotbeage@interia.pl**.

Jakub Borzdyński
jakub.borzdynski@elportal.pl