

Układy kombinacyjne - multiplexer

1. Cel ćwiczenia

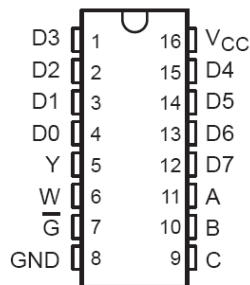
Zastosowanie multiplexera do realizacji funkcji kombinacyjnych wielu zmiennych.
Wykorzystanie pamięci ROM do realizacji funkcji logicznych wielu zmiennych.

2. Multiplexer

Jest to tzw. selektor danych. Multiplexer przekazuje informacje jednego z kilku(nastu) wejść na jedno wyjście. Wybór wyjścia następuje za pośrednictwem wejść adresowych (wejść selekcji). Mogą występować w postaci scalonej lub jako elementy wewnętrzne w układach programowalnych (umożliwiając użytkownikowi dowolną konfigurację szerokości ścieżki danych i sposobu adresowania).

2.1. Przykład multiplexera scalonego (układ 74HC151)

Układ scalony 74151 jest multiplexerem 8x1 z wejściem blokującym (G) oraz wyjściem prostym (Y) i zanegowanym (W). Układ posiada 3 wejścia adresowe (A,B,C) oraz 8 wejść danych (Dx). Wejście zezwolenia (G) jest aktywne stanem niskim [1].



SN54HC151

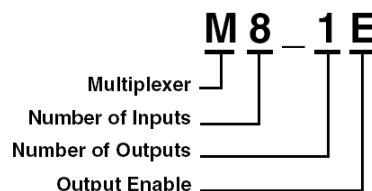
FUNCTION TABLE

INPUTS				OUTPUTS	
SELECT			STROBE \bar{G}	Y	W
C	B	A			
X	X	X	H	L	H
L	L	L	L	D0	$\bar{D0}$
L	L	H	L	D1	$\bar{D1}$
L	H	L	L	D2	$\bar{D2}$
L	H	H	L	D3	$\bar{D3}$
H	L	L	L	D4	$\bar{D4}$
H	L	H	L	D5	$\bar{D5}$
H	H	L	L	D6	$\bar{D6}$
H	H	H	L	D7	$\bar{D7}$

D0, D1 . . . D7 = the level of the respective D input

2.2. Multiplexery z biblioteki Xilinx Spartan3E

W kategorii 'Mux' w bibliotekach dla układów Spartan3E występuje kilkanaście typów multiplexerów. Najważniejszymi, z punktu widzenia laboratorium, są elementy zdefiniowane jako makra, określane nazwami zgodnymi z poniższym szablonem:



Powyższa nazwa określa multiplexer 2x1 z wejściem odblokowującym E (wszystkie zdefiniowane multiplexery są elementami działającymi na danych 1-bitowych).

symbol	opis elementu
M2_1	multiplekser 2x1,
M2_1B1	multiplekser 2x1, z zanegowanym wejściem D0,
M2_1B2	multiplekser 2x1, z zanegowanym wejściem D0 i D1,
M2_1E	multiplekser 2x1, z wejściem odblokowującym E,
M4_1E	multiplekser 4x1, z wejściem odblokowującym E, 2 wejścia selekcji,
M8_1E	multiplekser 8x1, z wejściem odblokowującym E, 3 wejścia selekcji,
M16_1E	multiplekser 16x1, z wejściem odblokowującym E, 4 wejścia selekcji.

2.3. Realizacja funkcji kombinacyjnych z użyciem multiplexera

Multiplekser realizuje dowolną funkcję przełączającą p zmiennych. Realizacja funkcji kombinacyjnej n zmiennych przy użyciu multiplexera nie przysparza żadnych problemów gdy $p = n$. W takim przypadku na wejścia selekcji S_x podajemy wektory wymuszające, a wejścia danych łączymy odpowiednio do stanu '1' lub '0', zgodnie z tablicą prawdy realizowanej funkcji.

Gdy liczba zmiennych funkcji realizowanej jest większa od liczby wejść adresowych multiplexera, można zestawić układy w piramidę tworząc MUX dowolnej wielkości. Takie rozwiązanie jest jednak kosztowne. Korzystniejszym jest rozbięcie wektora wymuszającego X na dwie części: X_a i X_b tak, aby liczba elementów wektora X_a wynosiła p . Podział wektora ma oczywiście wpływ na realizację funkcji (złożoność układu). Podstawową więc kwestią jest wybór takiego podziału, by złożoność była najmniejsza. Zakłada się, że w skład X_b powinny wejść te zmienne, które mają w minimalnej postaci funkcji największy współczynnik nieokreśloności. (Współczynnik nieokreśloności, czyli liczba symboli x (wartość nieistotna, don't care) w zbiorze wektorów postaci minimalnej.) [2]

2.3.1. Przykład realizacji

- funkcja 3-zmiennych $Y(A,B,C) = \{0,2,3,5,7\}$

$$Y = /A/B/C + /AB/C + /ABC + A/BC + ABC$$

- v1: podział wektora: $X_a = (A,B)$, $X_b = (C)$

$$Y = /A/B (/C) + /AB (/C) + /AB (C) + A/B (C) + AB (C)$$

$$Y = /A/B (/C) + /AB (/C + C) + A/B (C) + AB (C)$$

$$Y = /A/B (/C) + /AB (1) + A/B (C) + AB (C)$$

z powyższego wynika realizacja v1:

$$D0 = /C, D1 = 1, D2 = C, D3 = C.$$

- v2: podział wektora: $X_a = (A,C)$, $X_b = (B)$

$$Y = /A/C (/B) + /A/C (B) + /AC (B) + AC (/B) + AC (B)$$

$$Y = /A/C (/B + B) + /AC (B) + AC (/B + B)$$

$$Y = /A/C (1) + /AC (B) + A/C (0) + AC (1)$$

z powyższego wynika realizacja v2:

$$D0 = 1, D1 = B, D2 = 0, D3 = 1.$$

- v3: podział wektora: $X_a = (B,C)$, $X_b = (A)$

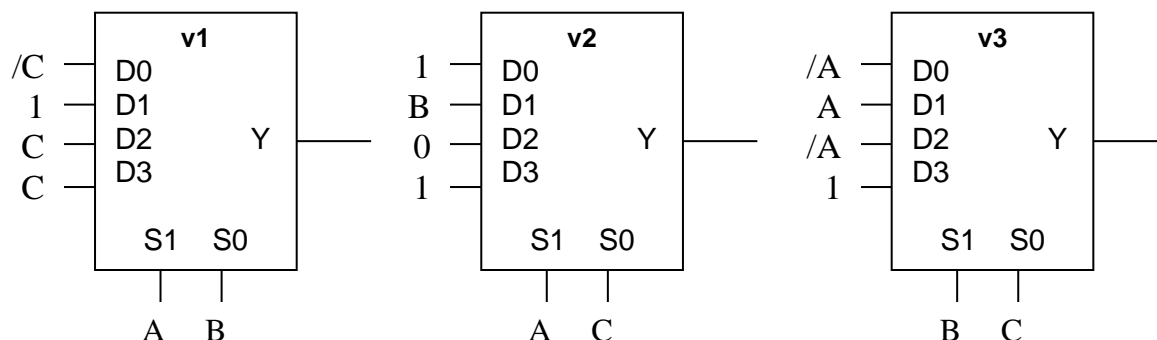
$$Y = /B/C (/A) + /BC (A) + B/C (/A) + BC (/A) + BC (A)$$

$$Y = /B/C (/A) + /BC (A) + B/C (/A) + BC (/A + A)$$

$$Y = /B/C (/A) + /BC (A) + B/C (/A) + BC (1)$$

z powyższego wynika realizacja v3:

$$D0 = /A, D1 = A, D2 = /A, D3 = 1.$$



2.3.2. Wnioski

Szukając postaci minimalnej funkcji otrzymamy: $Y = /AB + AC + /A/C$, gdzie największy stopień nieokreśloności ma zmienna B. Potwierdza to wcześniejsze twierdzenie, że w skład X_b powinny wejść te zmienne, które mają w minimalnej postaci funkcji największy współczynnik nieokreśloności (najmniejsza złożoność w realizacji v2).

Zalety:

- łatwość tworzenia i modyfikacji projektu w przypadku gdy S_x (liczba wejść selekcji) równa się liczbie zmiennych realizowanej funkcji,
- minimalizacja liczby układów scalonych użytych w projekcie.

Wady:

- synteza 'intuicyjna' w przypadku gdy S_x (liczba wejść selekcji) jest mniejsza od liczby zmiennych realizowanej funkcji logicznej.

3. Podstawowe elementy pamięciowe RAM/ROM

Bloki pamięciowe służą do trwałego lub czasowego przechowywania informacji zapisanej w postaci binarnej. Najprostszymi elementami pamięciowymi są zwykłe przerzutniki, które możemy traktować jako pamięci jednobitowe.

Pamięci o dostępie swobodnym można ogólnie podzielić na statyczne (SRAM) i dynamiczne (DRAM). Z punktu widzenia łatwości sterowania najwygodniejsze w użyciu są pamięci SRAM. Praca z pamięcią statyczną polega na podaniu sygnału adresowego, a następnie w zależności od wartości sygnałów sterujących, odczytaniu lub zapisaniu zawartości szyny danych do komórek pamięci wskazywanych przez adres.

Pamięci ROM mają organizację podobną do organizacji pamięci RAM. Komórkami tych pamięci są pojedyncze tranzystory, które przewodzą lub nie. Patrząc z zewnątrz na pamięć ROM, jej działanie nie różni się od działania zwykłego układu kombinacyjnego. W obu przypadkach istnieje wejście (w przypadku pamięci są to linie adresowe), na które podawane są słowa binarne i wyjście, na którym pojawiają się odpowiedzi w postaci innych słów binarnych. Każdemu słowu wejściowemu jednoznacznie przyporządkowane jest słowo wyjściowe. Tak więc pamięć ROM może być traktowana jako jeden ze sposobów realizacji funkcji kombinacyjnej. Koncepcja wykorzystania pamięci do realizacji funkcji kombinacyjnych jest szeroko stosowana w układach typu FPGA, gdzie do realizacji funkcji stosowane są bloki LUT (Look-Up Table).

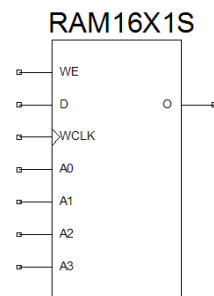
Podsumowując: dowolny układ kombinacyjny może być zrealizowany przy użyciu pamięci stałej ROM, w której zapisane są tablice funkcji przełączających opisujących ten układ. Na wejścia adresowe pamięci podane są zmienne wejściowe, a wyjścia pamięci są wyjściami układu. W ćwiczeniu można wykorzystać jednoportową pamięć synchroniczną RAM o organizacji 16x1b (RAM16X1S) lub pamięć asynchroniczną ROM o organizacji 16x1b (ROM16X1).

3.1. RAM16X1S

Pamięć RAM16X1S jest jedną z kilkunastu dostępnych typów pamięci RAM na platformie Spartan3. Jest to synchroniczna pamięć statyczna jednoportowa o organizacji 16 słów 1-bitowych.

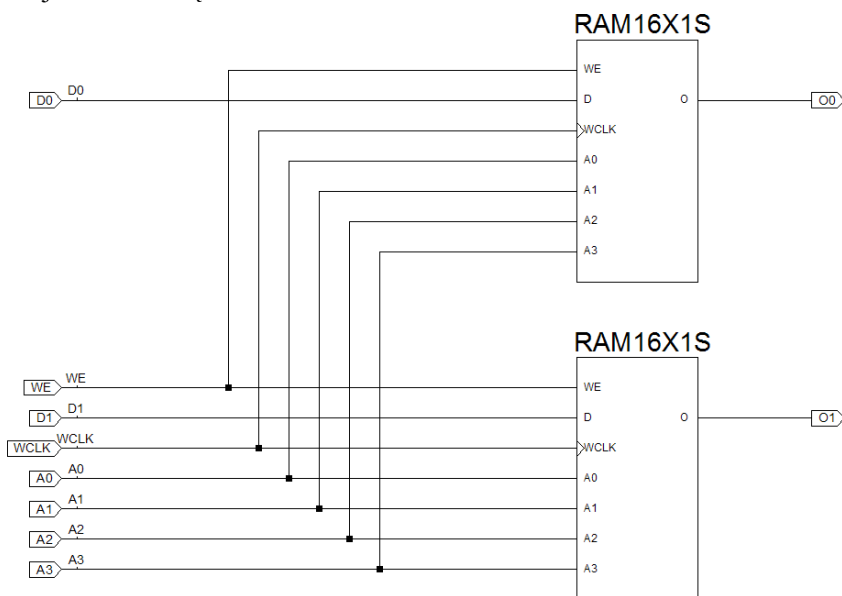
Inputs			Outputs
WE (mode)	WCLK	D	O
0 (read)	X	X	Data
1 (read)	0	X	Data
1 (read)	1	X	Data
1 (write)	↑	D	D
1 (read)	↓	X	Data

Data – zawartość komórki wskazywanej przez adres A(3:0)



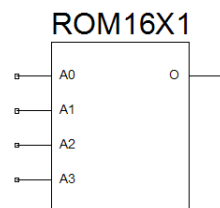
Przy stanie wysokim na wejściu WE (Write Enable) każde narastające zbocze zegara WCLK powoduje zapis danych z wejścia D do komórki wskazywanej przez adres A(3:0), gdzie A0 jest najmłodszym bitem adresu. Aby zapewnić poprawny zapis, adres i dane muszą być stabilne przed przyjściem narastającego zbocza WCLK. Synchroniczny odczyt z pamięci najwygodniej realizować opadającym zboczem zegara WCLK. Zawartość pamięci może być inicjalizowana dowolnym słowem 16-bitowym na etapie projektowania urządzenia. Do tego celu służy parametr INIT określany z poziomu schematu lub języka opisu sprzętu.

Z podstawowych elementów pamięci można realizować blok pamięć o większej pojemności rozszerzając adres lub ścieżkę danych. Poniżej przykład rozbudowania szerokości słowa – pamięć 16x2bity złożona z elementów 16X1S. Każda z zastosowanych pamięci posiada własny wektor inicjalizujący o postaci 'XXXX', gdzie X jest cyfrą heksadecymalną. Domyślnie elementy pamięciowe inicjalizowane są zerami.



3.2. ROM16X1

Pamięć ROM16X1 jest jedną z kilku podstawowych pamięci ROM dostępnych na platformie Spartan3E. Jest to blok pamięci asynchronicznej – po podaniu adresu na wejścia A(3:0) na wyjściu O pojawiają się dane odczytane z odpowiedniej komórki. Czas dostępu do danych jest rzędu nanosekund. Bloki ROM należy inicjalizować na etapie projektowania – w przypadku braku definicji parametru INIT występuje błąd.



4. Zadania

A)

Realizacja funkcji 5-zmiennych przy pomocy multipleksera:

- znajdź postać minimalną funkcji **F2/5** (metoda dowolna) i wyznacz współczynnik nieokreśloności zmiennych.
- przy pomocy układu **M8_1E** (jeśli to konieczne wykorzystaj dodatkowe bramki logiczne) zbuduj układ realizujący funkcję **F2/5**.
- przygotuj schemat do testowania wykorzystujący generator wymuszeń i podłączenie analizatora stanów (wg przykładu z lab0),
- dokonaj symulacji behawioralnej zaprojektowanego urządzenia, wyniki symulacji zamieść w karcie projektu,

B)

Realizacja funkcji 4-zmiennych przy pomocy pamięci:

- przy pomocy układów **RAM16X1S** lub **ROM16X1** (jeśli to konieczne wykorzystaj dodatkowe bramki logiczne) zbuduj układ realizujący funkcję kodera wg tablicy **D2**,
- jeśli w projekcie wykorzystano pamięć RAM, zbuduj układ sterujący zapisem danych do pamięci (+), w przypadku zastosowania pamięci ROM wykorzystaj parametr INIT do inicjalizacji zawartości pamięci na etapie generacji pliku bitowego,
- przygotuj schemat do testowania wykorzystujący generator wymuszeń i podłączenie analizatora stanów (wg przykładu z lab0),
- dokonaj symulacji behawioralnej zaprojektowanego urządzenia, wyniki symulacji zamieść w karcie projektu,

Dokonaj pomiarów układu wybranego przez prowadzącego przy pomocy analizatora stanów logicznych. Wyniki pomiarów (w postaci elektronicznej) wraz z kartą projektu przedstaw prowadzącemu zajęcia.

- [1] Texas Instruments, SN54HC151 SN74HC151 8-line to 1-line data selectors/multiplexers, SCLS110E, December 1982.
- [2] Wiesław Traczyk, Układy cyfrowe - podstawy teoretyczne i metody syntezy, wydanie II, WNT Warszawa 1986.