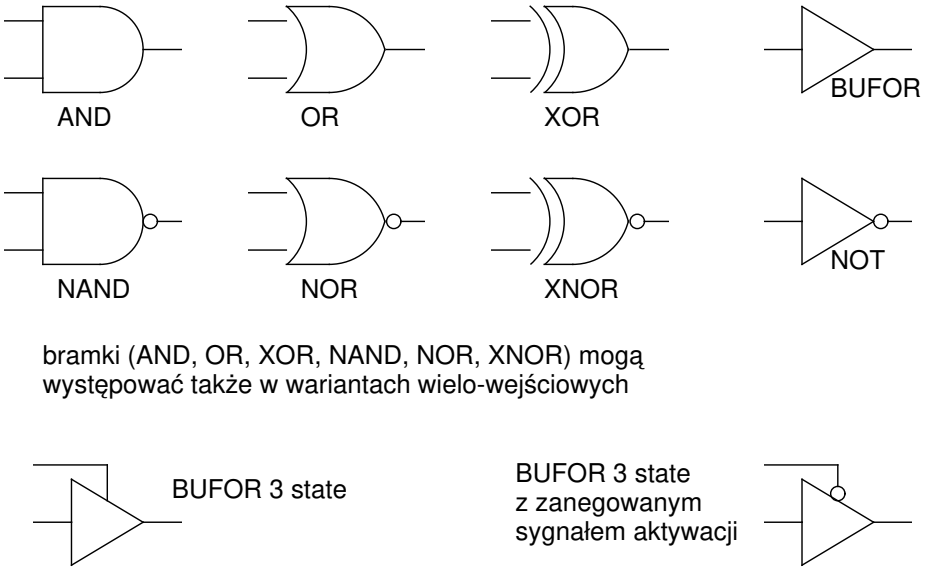
Linux i Python w Elektronicznej Sieci #09: Wprowadzenie do elektroniki cyfrowej

Projekt „Matematyka dla Ciekawych Świata”,
Robert Ryszard Paciorek
<rrp@opcode.eu.org>

2023-06-25

1 Bramki



Bramki są układami elektronicznymi realizującymi podstawowe, opisane powyżej funkcje logiczne. Obok zostały przedstawione podstawowe symbole poszczególnych bramek w wariancie dwu wejściowym, spotkać się można także z symbolami z zanegowanymi wejściami - w takiej konwencji np. bramka AND reprezentowana jest przez NOR z zanegowanymi wejściami. Bramki (z wyjątkiem buforów oraz bramki NOT), mogą występować także w wariantach wielo-wejściowych (ze względu na łączność podstawowych operacji nie ma wątpliwości co don wyniku jaki powinna dawać np. 8 wejściowa bramka OR). Na ogół w pojedynczym układzie scalonym znajduje się kilka jednakowych bramek.

Zobacz symulację działania różnych bramek logicznych: http://ln.opcode.eu.org/bramki (H oznacza stan wysoki, czyli prawdę, L stan niski czyli fałsz, klikając na H/L przy wejściach można zmieniać ich stan).

1.1 trój-stanowe

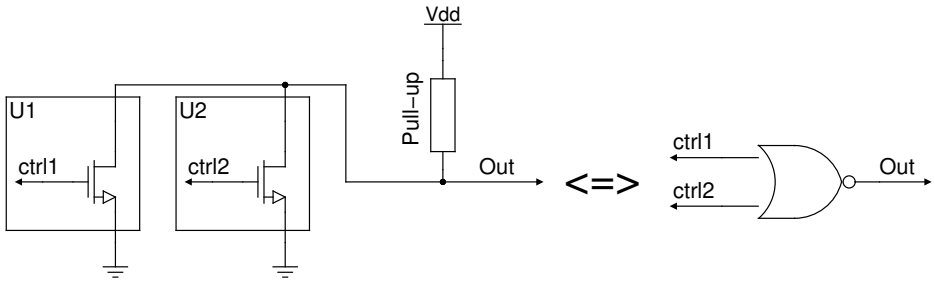
Typowa bramka wymusza (w sposób silny) na swoim wyjściu stan wysoki lub niski, co uniemożliwia bezpośrednie łączenie wyjść bramek. Bramki trój-stanowe mają możliwość skonfigurowania wyjścia w stan *wysokiej impedancji* czyli nie wymuszania żadnej jego wartości. Sterowanie załączeniem bądź wyłączeniem wyjścia (przełączeniem w stan wysokiej impedancji) odbywa się przy pomocy zewnętrznego sygnału sterującego ”output enabled” (”OE”), sygnał ten może występować w postaci prostej i zanegowanej. Pozwala to na podłączanie do jednej linii wielu bramek i decydowaniu która z nich będzie nią sterować.

1.2 open collector / drain

Są kolejnym rodzajem bramek których wyjścia można podłączać do wspólnej linii. Układy te posiadają wyjście w postaci tranzystora zwierającego linię wyjściową do masy, z tego względu samodzielnie zapewniają jedynie stan niski wyjścia (są w stanie wymusić stan niski, ale nie mają możliwości wymuszenia stanu wysokiego).

Stan wysoki musi zostać zapewniony zewnętrznym rezystorem podciągającym. Pozwala to stosować na takiej linii inny poziom stanu wysokiego niż na wejściach takiej bramki oraz pozwala na sterowanie wspólnej linii przez wiele bramek (czyli łączenie wyjść bramek, jednak w odróżnieniu od bramek trój-stanowych nie wymaga dodatkowych sygnałów sterujących).

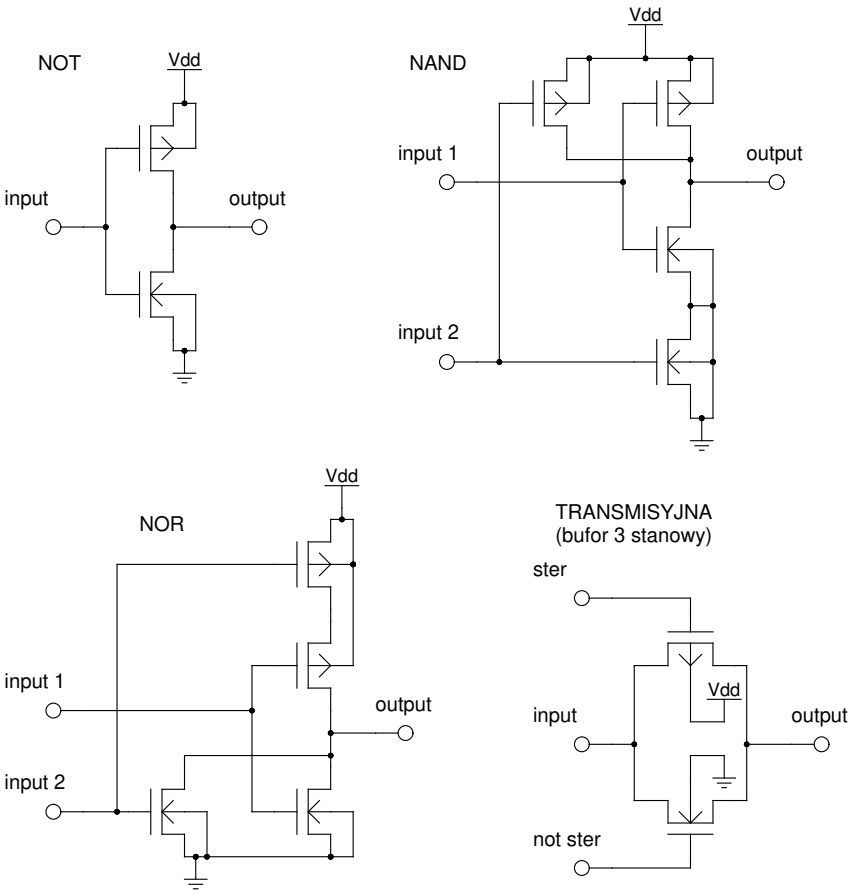
Na schemacie obok przedstawiono dwa układy (U1 i U2) typu open-drain sterujące wspólną linią wyjściową w układzie *suma na drucie*. Jeżeli jeden z podłączonych do linii układów będzie miał wewnętrzne wyjście ("ctrl$X$") w stanie wysokim to jego wyjście OC będzie zwarte do masy (negacja na tranzystorze N-MOS lub NPN), wtedy też cała linia będzie w stanie niskim.



Zobacz symulację lininii z bramkami trójstanowymi (stan wysokiej impedancji symulowany za pomocą przełącznika) oraz linii open-colektor: http://ln.opcode.eu.org/ster

1.3 budowa wewnętrzna

Przedstawiony powyżej układ sumy na drucie jest bardzo prostą (jedno tranzystorową) realizacją bramki logicznej realizującą funkcję logiczną NOT OR (z punktu widzenia wejść *ctrl1* i *ctrl2* oraz wyjścia *Out*). W podobny sposób można zrealizować bramkę AND (negując wejścia, np. przy pomocy jednego traznzystora). Jeszcze bardziej uproszczoną realizację można uzyskać stosując diody pozwalające na wpływanie prądu do węzła (funkcja OR) lub wypływanie z niego (funkcja AND).



Po prawej przedstawione zostały schematy ideowe inwertera, dwóch podstawowych bramek (NOR i NAND) oraz bramki transmisyjnej (bufora 3 stanowego) w technologii CMOS.

Działanie tych bramek (za wyjątkiem transmisyjnej) polega na otwieraniu tranzystorów podłączonych do napięcia które chcemy otrzymać na wyjściu, a zamykaniu prowadzących do napięcia przeciwnego. W szczególności bramka NOT stanowi półmostek H pomiędzy stanem wysokim a stanem niskim.

Dzięki zastosowaniu tranzystorów PMOS polaryzowanych Vdd oraz NMOS polaryzowanych GND obie gałęzie operują na tym samym sygnale wejściowym (nie jest wymagana jego negacja). Szeregowe łączenie tranzystorów zapewnia że należy otworzyć oba aby otworzyć daną drogę, a równoległe że otwarcie danej drogi powodowane jest otwarciem pojedynczego tranzystora. Dzięki zastosowaniu technologi MOS i podłączaniu wejść bramki tylko do bramek tranzystorów wejścia praktycznie nie pobierają prądu (istotnym wyjątkiem jest chwila zmiany sygnału).

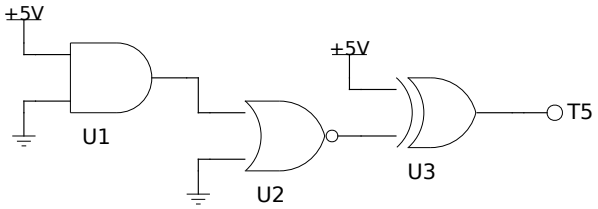
Działanie bramki transmisyjnej polega na przepuszczaniu lub nie (w zależności od stanu wejścia sterującego) sygnału z wejścia na wyjście. Bramka taka nie regeneruje sygnału. Ponadto w uproszczonym (jedno tranzystorowym) rozwiązaniu prowadzi ona do degradacji sygnału wartość w przybliżeniu równą

napięciu progowemu tranzystora. Dlatego też na ogół występuje wraz z bramką NOT (bufor 3 stanowy z negacją) lub dwiema szeregowo połączonymi bramkami NOT (bufor 3 stanowy bez negacji).

Zobacz symulację budowy bramek: NOT (http://ln.opcode.eu.org/not), NAND (http://ln.opcode.eu.org/nand) i NOR (http://ln.opcode.eu.org/nor).

Zadanie 1.0.1

Podaj wartość napięcia (względem GND) w punkcie T5. Przyjmujemy iż użyte bramki działają na poziomie napięć 5V (prawda) / 0V (fałsz). Odpowiedź krótko uzasadnij.



2 Przerzutniki i rejestry

2.1 przerzutniki i ich budowa

RS Flip-flop (RS Latch) jest podstawowym układem służącym do zapamiętania jednego bitu informacji. Posiada on dwa wejścia (set i reset) i dwa wyjścia (Q i NOT Q), wejścia mogą reagować na stan wysoki (oznaczane jako S i R) lub niski (oznaczane jako wejścia zanegowane $\overline{S}$ i $\overline{R}$), jedno z wyjść może być jedynie wewnętrzne (nie wyprowadzone na zewnątrz układu). Podanie stanu wysokiego na wejście S (niskiego na $\overline{S}$) powoduje wystawienie stanu wysokiego na wyjściu Q, a podanie stanu wysokiego na wejście R (niskiego na $\overline{R}$) powoduje wystawienie stanu niskiego na wyjściu Q. Stan na wyjściu Q nie zmienia się po zmianie wejść S i R na stan niski (zostaje zapamiętany).

Zobacz i przeanalizuj symulację działania zatrzasku RS: http://ln.opcode.eu.org/rs z wejściami zanegowanymi.

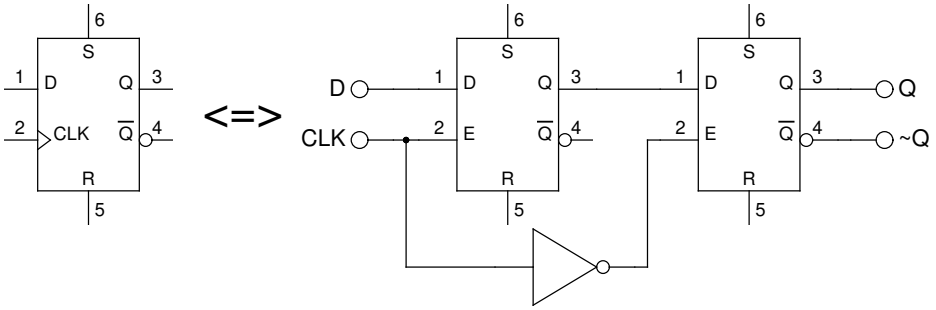
2.2 zatrzask a przerzutnik

Zatrzask jest elementem reagującym na poziom sygnału na wejściu ”enable” (E). W przypadku nie zanegowanego wejścia E, jeżeli jest ono w stanie wysokim sygnał na wyjściach (Q i NOT Q) jest funkcją sygnałów wejściowych, natomiast stan niski wejścia E blokuje zmianę sygnału wyjściowego (zostaje on zapamiętany).

Przerzutnik jest elementem reagującym na zbocze sygnału na wejściu ”clock” (CLK). W zależności od konstrukcji może reagować na zbocze narastające, opadające albo na oba (wtedy na jednym realizuje odczyt wejść a na drugim zmianę stanu wyjść).

2.3 zatrzask i przerzutnik D

Posiada jedno wejścia sygnałowe ”data” (D) oraz wejście ”enable” (E) w przypadku zatrzasku lub wejście ”clock” (CLK) w przypadku przerzutnika. Może także posiadać asynchroniczne (niezależne od stanu wejścia E / CLK) wejścia reset i set (zanegowane



lub proste). Obniżenie sygnału E lub zbocze sygnału CLK powodują zapamiętanie (i wystawienie na wyjściu Q) stanu wejścia D.

Zobacz symulację zatrzasku typu D zbudowanego z bramek NAND: http://ln.opcode.eu.org/zatrzask (możesz zmieniać stan wejścia D klikając na nie, zegar zmienia się automatycznie). Zwróć uwagę iż przy wysokim stanie sygnału zegara (enable) stan wyjścia odpowiada stanowi wejścia (zatrzask jest przeźroczysty), natomiast przy niskim stanie zegara wyjście nie reaguje na zmiany stanu wejścia i znajduje się w takim stanie w jakim było wejście w chwili opadającego zbocza sygnału zegarowego.

Zobacz symulację przerzutnika D złożonego z dwóch zatrzasków: http://ln.opcode.eu.org/przerzutnik. Zauważ że w żadnej fazie sygnału zegarowego nie jest on przeźroczysty (wyjście Q nie zależy od obecnego stanu wejścia D). Zwróć uwagę że sygnał wejściowy zostanie zapamiętany i wystawiony na wyjście Q w momencie opadającego zbocza sygnału zegarowego.

2.4 rejestry

Mianem rejestru n-bitowego określa się zespół n przerzutników (rzadziej zatrzasków), często z uwspólnionym sterowaniem (sygnały clock, set, reset, etc) służący do zapamiętania n-bitowej wartości (liczby). W zależności od sposobu zapisu i odczytu można wyróżnić:

2.4.1 rejestry równoległe

Posiadają taką samą liczbę wejść jak i wyjść, sygnał na i-tym wyjściu jest bezpośrednio powiązany z sygnałem z i-tego wejścia (jest sygnałem zapamiętanym z tego wejścia).

Zobacz symulację rejestru równoległego zbudowanego z przerzutników typu D: http://ln.opcode.eu.org/rejestr1 (stan wszystkich wejść zostanie zapamiętany i przepisany na wyjścia w chwili narastającego zbocza zegara).

2.4.2 rejestry szeregowe serial-input

Z kolejnymi sygnałami zegarowymi odczytywany jest stan wejścia szeregowego, a stan poprzedni przenoszony jest do kolejnego przerzutnika w ramach rejestru. W ten sposób po n cyklach zegara n-bitowy rejestr ma zapisaną nową zawartość. Często posiada zespolony z nim rejestr równoległy zapobiegający zmianie stanu wyjść w trakcie ładowania danych z wejścia szeregowego przepisanie danych z rejestru przesuwnego do rejestru odpowiedzialnego za sterowanie wyjściami sterowane jest osobnym sygnałem zegarowym.

Zobacz symulację prostego rejestru z wejściem szeregowym (bez zatrzasku/rejestru wyjściowego): http://ln.opcode.eu.org/rejestr2. Zauważ że stan wyjść zmienia się na bieżąco w trakcie szeregowego wpisywania wartości do rejestru.

Zobacz symulację rejestru z wejściem szeregowym i rejestrem wyjściowym: http://ln.opcode.eu.org/rejestr3. Zauważ, że stan wyjść zmienia się na skutek osobnego sygnału, który może zostać wygenerowany po zakończeniu szeregowego zapisu do rejestru.

2.4.3 rejestry szeregowe paraller-input serial-output

Z kolejnymi sygnałami zegarowymi na wyjście szeregowe wystawiany jest stan kolejnego z rejestrów wejściowych. Wariant asynchroniczny posiada osobny sygnał powodujący odczyt wejść do rejestru (sygnał działa jak ”enable” w zatrzaskach). Wariant synchroniczny posiada sygnał decydujący o tym czy na zboczu zegara dokonywany jest odczyt wejść czy też przesuwanie zawartości rejestru umożliwiający odczyt z wyjścia szeregowego.

2.4.4 liczniki

Z kolejnymi sygnałami zegarowymi zwiększana jest o jeden wartość rejestru. Prostszy w budowie licznik asynchroniczny ma większe (i w dodatku rosnące wraz z bitowością licznika) ograniczenia dotyczące szybkości zliczania od licznika synchronicznego, ze względu na opóźnienie z jakim dochodzi zliczany sygnał (CLK) do kolejnych stopni licznika.

Zadanie 2.0.1

Znajdź dokumentację (kartę katalogową) do rejestru przesuwnego, który kupiłeś/aś (CD4094 lub 74HC595). Odczytaj, jakie jest największe napięcie zasilania tego układu (*supply voltage*, w sekcji *Absolute maximum ratings*), oraz do których pinów należy podłączać napięcie zasilania (w sekcji *pin configuration*).

3 Transmisja - sterowanie linią

3.1 bufory

Bufor jest to układ przekazujący sygnał logiczny z wejścia na wyjście. Bufor może służyć do:

- regeneracji sygnału,
- uniemożliwieniu wprowadzenia sygnału z jego strony wyjściowej na wejściową,
- decydowania o jego przepuszczeniu lub nie (trój-stanowy),
- decydowania o kierunku przepuszczenia sygnału (dwa trój-stanowe albo trój-stanowy dwukierunkowy),
- konwersji na linię open-collector / open-drain,
- negacji sygnału (niektóre bufory realizują funkcję NOT).

3.2 enkodery

Enkoder "n to m" jest to układ o n wejściach, który na swoim m bitowym wyjściu wystawia numer (typowo) wejścia o najwyższym numerze, które znajduje się w stanie niskim. Możliwe są też warianty wystawiające numer pierwszego (a nie ostatniego) w kolejności wejścia lub wybierające wejście ze stanem wysokim.

Jako że wejścia numerowane są zazwyczaj od zera do 2m to układ najczęściej posiada dodatkowe wyjście informujące że którekolwiek z wejść jest w stanie aktywnym. Typowo numer wystawiany jest w postaci NKB, ale możliwe są inne kodowania.

Układ pozwala na redukcję ilości wejść potrzebnych do obsługi n-bitowego sygnału w którym tylko jeden bit może być ustawiony lub w którym można pozwolić sobie na obsługę kolejnych linii z kasowaniem ich bitu (np. wektor przerwań z priorytetyzacją).

3.3 dekodery

Dekoder "m to n" jest układem o działaniu przeciwnym do enkodera. Aktywuje on wyjście o numerze odpowiadającym wartości na m-bitowym wejściu adresowym. Typowo posiada także wejście zezwolenia na aktywację wyjść, które może zostać użyte do podłączenia informacji że którekolwiek z wejść enkodera było w stanie aktywnym lub do podłączenia sygnału danych z multipleksowanej linii celem jej demultipleksacji.

Przykład użycia enkodera i dekodera do obsługi matrycy przełączników (klawiatury) można zobaczyć na symulacji: http://ln.opcode.eu.org/matrix.

3.4 (de)multipleksery cyfrowy

Multiplekser cyfrowy (jednokierunkowy) na wyjście kopiuje stan wskazanego (poprzez adres podany na wejście adresowe) wejścia. W przypadku braku sygnału "enable" w zależności od rozwiązania wyjście pozostanie w stanie niskim lub wysokiej impedancji.

Demultiplekser cyfrowy (jednokierunkowy) to zazwyczaj układ dekodera w którym na wejście enabled podany jest sygnał z multipleksowanej linii. Nie wybrane wyjścia pozostają w stanie niskim lub wysokim (zależnie od użycia nieodwracającego lub odwracającego dekodera). Cyfrowe demultipleksery z 3 stanowym wyjściem są rzadkością. Demultipleksację można rozwiązać także przy pomocy odpowiednio sterowanych (np. z dekodera adresu) buforów trój-stanowych lub dwu-wejściowych multiplekserów.

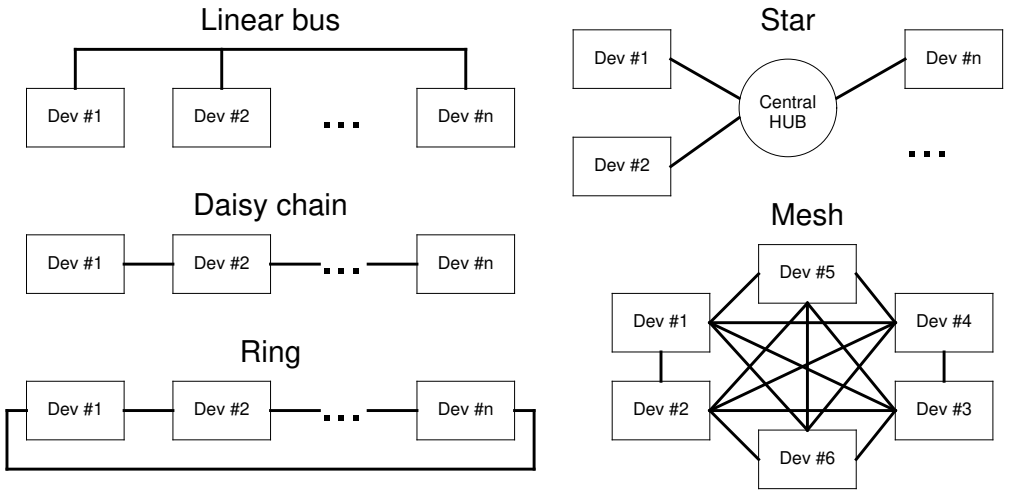
3.5 (de)multipleksery analogowy

Multiplekser analogowy (dwukierunkowy) działa na zasadzie przełącznika łączącego wskazane wejście z wyjściem, dzięki elektrycznemu "zwarciu" (na ogół rezystancja takiego zwarcia to kilkadziesiąt omów) wejścia z wyjściem transmisja może odbywać się w obu kierunkach. Pozwala to także na wykorzystanie tego samego układu w roli multipleksera i demultipleksera.

4 Topologie i typy transmisji

W zależności od układu fizycznych połączeń komunikujących się urządzeń wyróżnia się różne topologie sieci. Na schemacie poniżej przedstawione zostały główne topologie połączeń:

- **magistrala** (linear bus) – wszystkie urządzenia są podłączone do jednej linii (wspólnego medium transmisyjnego), okablowanie nie wyróżnia punktu centralnego
- **łańcuch** (daisy chain) – struktura okablowania podobna jak w magistrali, ale medium transmisyjne jest podzielone (połączenie n urządzeń składa się z n-1 łączy punkt-punkt pomiędzy urządzeniami)
- **pierścień** (ring) – topologia daisy chain w której końce są połączone, uodparnia to na pojedyncze uszkodzenie
- **gwiazda** (star) – wszystkie podłączenia biorą początek w węźle centralnym, w zależności od konstrukcji węzła centralnego może być realizowana w oparciu o współdzielone medium lub połączenia punkt-punkt
- **krata** (mesh) – każde urządzenie ma bezpośrednie połączenie punkt-punkt do każdego innego urządzenia (połączenie n urządzeń wymaga n(n-1)/2 połączeń punkt punkt)

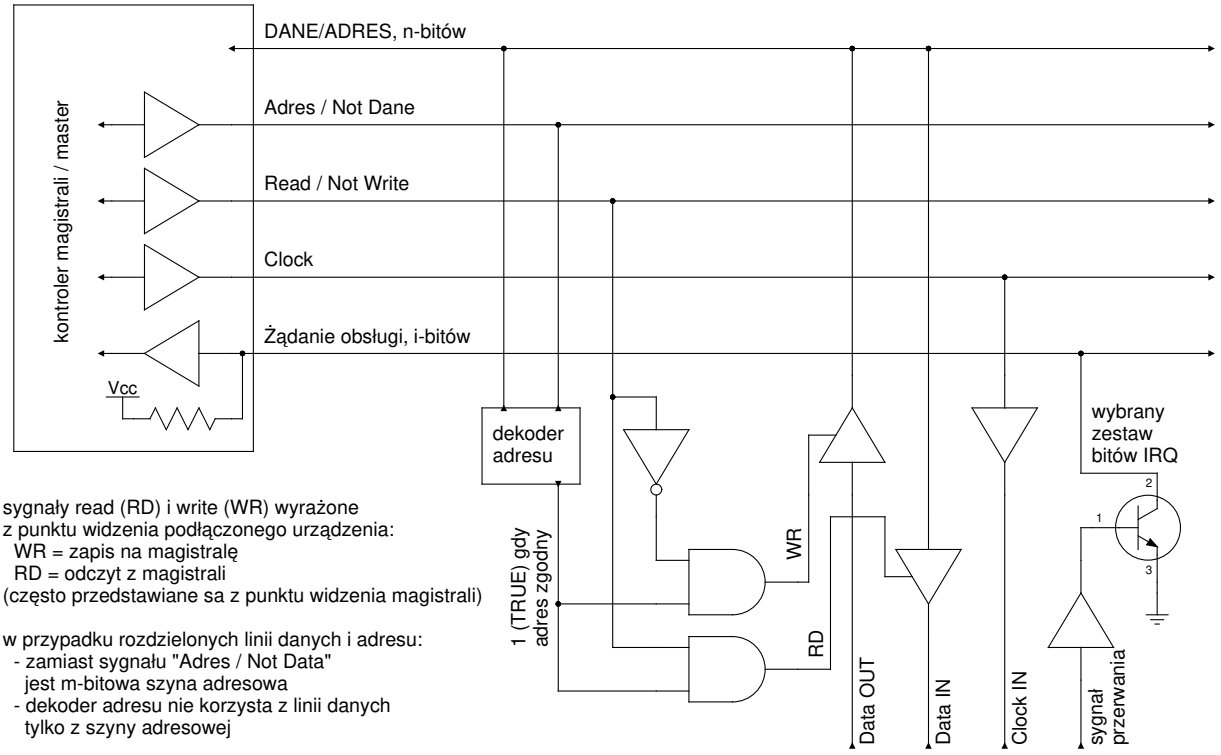


Ponadto występują topologie mieszane złożone z opisanych powyżej: gwiazda wielokrotna (tzn. taka gdzie niektóre z węzłów stanowią punkty centralne kolejnych gwiazd), magistrala lub ring pomiędzy punktami centralnymi gwiazd, gwiazda w której w węzłach występują magistrale lub pierścienie, itd.

Wyróżnić można także typy transmisji:

- **simplex** – umożliwia tylko transmisję jednokierunkową
- **half-duplex** – umożliwia transmisję dwukierunkową, ale tylko w jedną stronę równocześnie
- **full-duplex** – umożliwia pełną transmisję dwukierunkową (równoczesne nadawanie i odbiór)

4.1 Magistrala równoległa

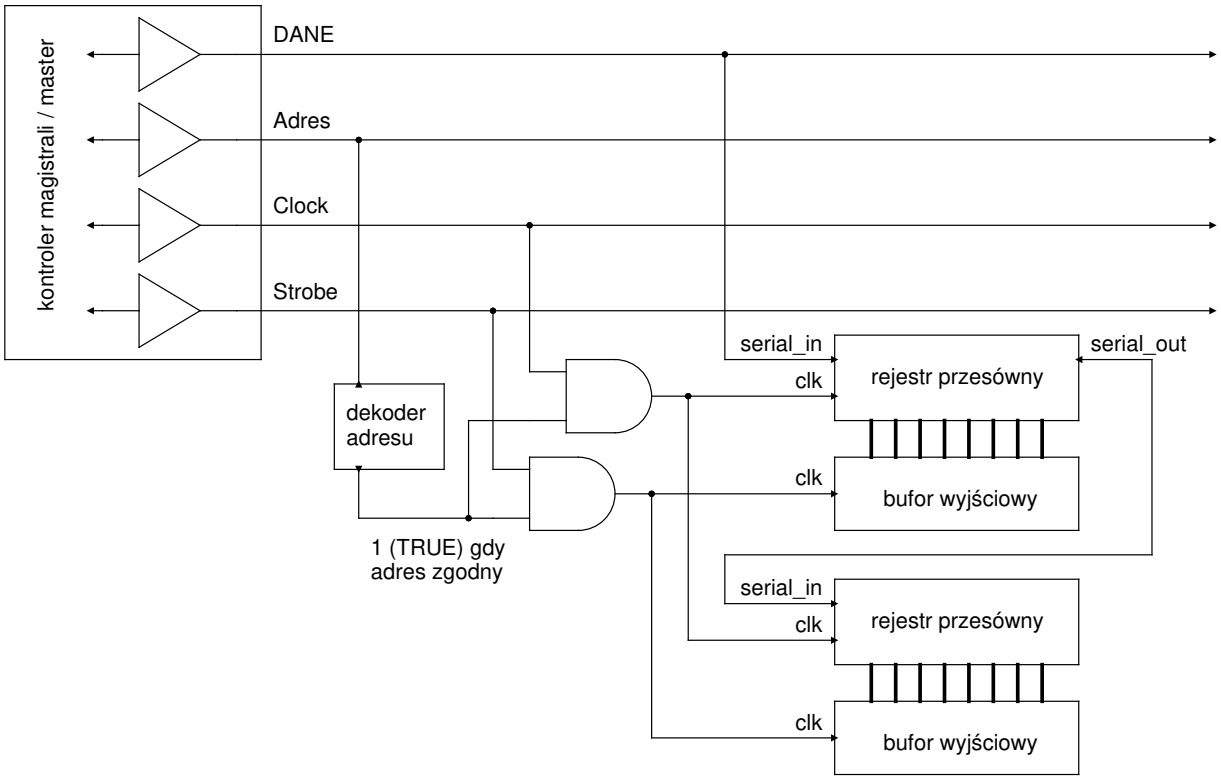


Magistrala równoległa jest zespołem linii, wraz z układami nimi sterującymi, umożliwiającym równoległe przesyłanie danych (w jednym czasie / takcie zegara na magistrali wystawiane / przesyłane jest całe n-bitowe słowo). Można wyróżnić szyny sterującą (kierunek przypływu, żądania obsługi, etc), adresową (adres układu który ma prawo nadawać) i danych (przesyłane dane). Często szyna adresowa współdzieli linie transmisyjne z szyną danych. Do realizacji magistrali (celem umożliwiania podłączenia wielu układów) stosuje się zazwyczaj bufory trój-stanowe, a do zapewnienia współdzielonej szyny żądania obsługi (interrupt request) często układy typu open-collector.

Typowy układ realizacji magistrali half-duplex ze współdzielonymi liniami danych i adresu przestawiony został na schemacie zamieszczonym obok. Zadaniem dekodera adresu jest ustalenie czy wystawiony na magistrali adres (w trakcie wysokiego stanu linii "Adres / Not Dane") jest adresem danego urządzenia i zapamiętanie tej informacji do czasu wystawienia nowego adresu. Informacja ta jest wykorzystywana do sterowania dwukierunkowym buforem trój-stanowym (jako sygnał enable). O kierunku działania bufora decyduje sygnał "Read / Not Write". Przy magistralach o ustalonym protokole transmisyjnym sterowanie kierunkiem może być zależne od wykonywanej komendy (po ustawieniu adresu urządzenie odczytuje z magistrali polecenie i w zależności od niego steruje kierunkiem bufora - odczytuje lub zapisuje dane na magistralę). Zastosowanie kilku linii typu OC do odbierania żądań obsługi pozwala (na podstawie tego które z tych linii znalazły się w stanie niskim na identyfikację urządzenia lub grupy urządzeń, z której niektóre zgłaszają żądanie obsługi.

W przypadku prostych urządzeń wejścia / wyjścia zamiast buforu dwukierunkowego może być umieszczony np. jednokierunkowy bufor (lub n-bitowy rejestr) z wyjściami trój-stanowymi, który wystawia dane na magistralę w oparciu o sygnał zapisu na magistralę (WR) oraz zegar (clock) albo n-bitowy rejestr do którego zapisywane są dane z magistrali w oparciu o sygnał RD i Clock.

4.2 Magistrala szeregowa



W magistrali szeregowej dane przesyłane są bit po bicie w pojedynczej linii. Podobnie jak w magistrali równoległej oprócz linii danych mogą występować także linie sterujące. Prostą realizację magistrali szeregowej zapewniają rejestry przesuwne.

Przykładowy układ realizacji magistrali simplex (jednokierunkowej) z rozdzielonymi szynami danych i adresową został na schemacie zamieszczonym obok. W prezentowanym przykładzie oprócz adresu master wystawia trzy sygnały - dane, zegar i strobe. Z każdym taktem zegara na linii danych wystawiany jest kolejny bit który jest wczytywany do zespołu rejestrów. Sygnał strobe służy do przepisania wartości z rejestrów przesuwnych do rejestrów wyjściowych, takie rozwiązanie zapobiega zmianom wyjść w trakcie przesyłania nowych danych poprzez szynę szeregową, jest ono jednak opcjonalne.

W zależności od konstrukcji dekodera adresu szyna adresowa może być równoległa (w najprostszym przypadku - przez całą transmisję do danego urządzenia jego adres musi być wystawiony na szynie a dekoder jest układem bramek NOT i wielowejściowej bramki AND) lub szeregowa (w takim wypadku powinna posiadać własny zegar lub sygnał informujący o nadawaniu adresu z taktami zegara głównego, a dekoder powinien być wyposażony w rejestr przesuwny do odebrania i przechowywania aktualnego adresu z magistrali). Natomiast jeżeli magistrala byłaby oparta tylko na połączonych szeregowo rejestrach (wyjście serial-out do wejścia serial-in) to szyna adresowa nie jest potrzebna, ale konieczne może być każdorazowe wpisanie wszystkich wartości na szynę (czas zapisu rośnie z ilością podłączonych rejestrów).

Zadanie 4.2.1

Zapoznaj się z dokumentacją układu 74HC574 i opisz sposób jego użycia (wraz z sposobem sterowania) jako modułu podłączonego do 8 bitowej magistrali równoległej w roli układu wejściowego oraz w roli układu wyjściowego.

Zadanie 4.2.2

Zapoznaj się z dokumentacją układu 74HC595 i opisz sposób jego użycia (wraz z sposobem sterowania) w roli układu wyjściowego podłączonego do magistrali szeregowej.

5 Układy programowalne

5.1 układy o programowalnej strukturze (PLD)

Są to układy w których programowany jest układ bramek, przerzutników, itp. "umieszczanych" w kości oraz ich połączeń.

Program dla takich układów tworzony jest w Hardware Description Language (najczęściej VHDL lub Verilog) i zamiast wykonywanego kodu opisuje strukturę układu logicznego (połączenia bramek, tablice prawdy, etc), która następnie jest programowana w fizycznej kości.

Najprostszym przykładem układu o programowalnej strukturze logicznej jest układ pamięci 2^n bitowej z n-bitową szyną adresową adresującą pojedyncze bity - pozwala on na realizację dowolnej funkcji logicznej o n wejściach i pojedynczym wyjściu.

Do kategorii tej zaliczają się układy typu:

- SPLD
  - PLE - programowalna matryca bramek OR
  - PAL i GAL - programowalna matryca AND z dodatkowymi bramkami OR (często także obudowana rejestrami i multiplekserami na wyjściach)
  - PLA - programowalne matryce AND i OR
- CPLD
- FPGA - programowalny element pamięciowy (możliwość zdefiniowania dowolnej - na ogół 4 wejściowej - funkcji w każdym elemencie logicznym, programowalne połączenia między elementami logicznymi i pinami, itd)

5.2 systemy procesorowe

Są to systemy realizujące ciąg instrukcji pobieranych z jakiejś pamięci.

System taki składa się z procesora odpowiedzialnego za interpretację i wykonywanie kolejnych instrukcji oraz pamięci z której pobierane są instrukcje i dane (może to być jedna pamięć, mogą to być rozdzielone pamięci). Do kategorii tej zaliczają się zarówno typowe systemy komputerowe, systemy obliczeniowe jak i różnego rodzaju programowalne mikrokontrolery.

Procesor pracuje w cyklach rozkazowych, w ramach których przetwarza pojedynczą instrukcję. Cykl taki może trwać od 1 do kilku lub więcej cykli zegarowych i typowo składa się z następujących kroków:

1. pobranie instrukcji z pamięci - realizowane jest poprzez wystawienie na szynę adresową zawartości *licznika programu* (zawierające adres instrukcji do wykonania) oraz wygenerowanie cyklu odczytu z pamięci, po wykonaniu odczytu danych następuje ich zapamiętanie w *rejestrze instrukcji* oraz zwiększenie wartości *licznika programu* o jeden;
   (zawartość rejestru *licznika programu* po resecie procesora określa skąd pobierana będzie pierwsza instrukcja, pod takim adresem zazwyczaj umieszczana jest jakaś pamięć typu ROM lub flash)
2. dekodowanie instrukcji - układ dekodera (np. oparty o PLA) dokonuje zdekodowania instrukcji znajdującej się w *rejestrze instrukcji* i konfiguracji procesora w zależności od jej kodu i (opcjonalnie) jej argumentów; może to być np.:
   - odpowiednie ustawienie multiplekserów pomiędzy rejestrami a jednostką ALU oraz wystawienie odpowiedniego kod operacji dla ALU (celem wykonania operacji arytmetycznej na wartościach rejestrów)
   - wystawienie zawartości wskazanego rejestru na szynę adresową, podłączenie wskazanego rejestru do szyny danych oraz skonfigurowanie operacji odczytu/zapisu (celem wykonania odczytu lub zapisu wartości rejestru z/do pamięci)
3. wykonanie instrukcji - realizacja wcześniej zdekodowanej instrukcji zgodnie z ustawioną konfiguracją procesora

Instrukcje skoku polegają na załadowaniu nowej wartości do *licznika programu*, w przypadku skoków warunkowych jest to uzależnione od stanu *rejestru flag*, które ustawiane są w oparciu o wynik ostatniej operacji wykonywanej przez ALU.

Przedstawiony model działania jest przykładowym i w rzeczywistym procesorze może to wyglądać odmiennie - np. długość instrukcji może być większa niż długość słowa używanego przez procesor / szerokość szyny danych co rozbudowuje fazę pobierania instrukcji z pamięci, mogą występować instrukcje bardziej złożone (np. operacje wykonywane z argumentem pobieranym z pamięci a nie rejestru), może także występować więcej faz (np. poprzez wydzielenie faz dostępu do pamięci, czy zapisywania wyników działania instrukcji). Procesor może także działać potokowo, czyli nakładać na siebie kolejne fazy wykonywania różnych instrukcji (np. w czasie wykonywania jednej instrukcji realizować pobieranie kolejnej).

5.2.1 Mikrokontrolery

Mikrokontroler jest układem typu "System on Chip" zawierającym w jednym układzie procesor, pamięć RAM, układy wejścia-wyjścia (np. GPIO, porty szeregowe typu USART, SPI, I2C, przetworniki ADC), pamięć typu Flash (dla programu).

6 Projektowanie

Reguły DRY i KISS, o których była mowa przy omawianiu bibliotek w pythonie, mają zastosowanie także w elektronice, a zwłaszcza projektowaniu układów elektronicznych i różnego rodzaju systemów automatyki. Przy projektowaniu elektroniki trochę trudniej niż w programowaniu jest zachować regułę DRY (zwłaszcza przy tworzeniu projektów płytek drukowanych *PCB*), jednak należy dążyć do tego – wydzielać powtarzające się elementy do pod-schematów, modularyzować tworzony projekt, itd.

Kurs ten poświęcony jest głównie elektronice cyfrowej, a ta współcześnie opiera się na wykorzystaniu układów programowalnych. Zatem dalsze rady będą dotyczyły głównie projektowania systemu/urządzenia opartego na jakimś mikrokontrolerze (mimo to wiele z nich można zastosować także przy projektowaniu innych układów elektronicznych i nie tylko).
Przy tworzeniu projektów tego typu systemów/urządzeń należy mieć szczególnie na uwadze:

- trzymanie się standardów i modułowość:
  - należy stosować standardowe, popularne, otwarte protokoły komunikacyjne, takie jak I^2C, RS-485/Modbus, Ethernet/IP
  - należy dokonywać podziału system na moduły funkcjonalne i określamy interfejsy pomiędzy nimi, zasadę tę należy stosować rekurencyjnie do wszystkich większych/bardziej złożonych jego elementów
  - należy unikać projektowania modułów „na miarę" (lepiej mieć n+2 jednakowych modułów niż n każdy innego rodzaju)
  - gdy oczekujemy większej niezawodności to należy zastosować redundancję – np. podwójne układy zasilania i komunikacji (dwa porty RS-485 lub dwa porty Ethernet) w każdym z urządzeń (modułów) składających się na system
  - protokoły komunikacyjne, prędkość transmisji, etc należy dobierać z sporym zapasem (rzędu nawet 50-70%), przewidując pojawienie się kolejnych rejestrów, funkcji, itp.
  - należy przewidzieć rezerwę miejsca w modułach (np. dostępnych wejść)
- dokumentacja, wersjonowanie:
  - należy stosować wersjonowanie nie tylko kodu, ale również schematów, projektów PCB, dokumentacji, etc związanych z naszym projektem
  - należy także oznaczać wersje związane z wykonanymi prototypami (w taki sposób aby można było je jednoznacznie powiązać z fizycznie wykonanym prototypem) i trzymać je co najmniej do zakończenia życia tych prototypów

  - należy umieszczać identyfikator wersji na tworzonych płytkach PCB
  - należy tworzyć dokumentację, np. samo użycie modbus nie rozwiązuje kwestii dokumentacji komunikacji – konieczna jest rzetelna tabela rejestrów
- zachowanie prostoty:
  - jeżeli używany mikrokontroler ma wbudowaną funkcję podciągania wejść – używać jej zamiast zewnętrznych rezystorów pull-up / pull-down, jeżeli ma tylko pull-up to przyciski robić jako zwierane do masy aby móc z niej skorzystać
  - pamiętać że często (wbrew początkowej intuicji) sterowanie czymś przy pomocy wyjścia cyfrowego (zwłaszcza gdy wymagany jest do tego zewnętrzny tranzystor) prostsze jest od strony masy, czyli poprzez odłączanie/podłączanie do sterowanego układu masy (a nie dodatniego bieguna zasilania)
  - warto także ograniczyć liczbę wykorzystywanych rodzin i modeli mikrokontrolerów – w przypadku nie seryjnej produkcji oszczędności z zastosowania np. bardzo małego mikrokontrolera zostaną zatarte „kosztami" związanymi z trudnościami w jego użyciu (brak pinów do debugowania, mała pamięć programu, itd.) oraz związanymi z rozwijaniem projektów na różnych mikroprocesorach (bo do kolejnego był już za mały)
- łatwość diagnostyki i serwisu:
  - należy zapewnić reset układów peryferyjnych wraz z resetem mikrokontrolera (np. poprzez użycie jednego z pinów GPIO mikrokontrolera w tej roli), dotyczy to także przypadków gdy naszymi urządzeniami peryferyjnymi są inne mikrokontrolery
  - należy zapewnić łatwą możliwość przeprogramowania mikrokontrolera (bez potrzeby rozmontowywania układu, czy też wyjmowania z niego mikrokontrolera), a jeżeli aktualizacja wbudowanego oprogramowania odbywa się standardowo w jakiś wyżej poziomowy sposób to należy zapewnić zabezpieczenie przed awarią w jej trakcie, np. poprzez umieszczanie go na wymiennej karcie SD lub dostęp (np. poprzez UART) do sprzętowego bootloadera danego mikrokontrolera, lub zewnętrzny dostęp do programowania odpowiedniej pamięci
  - warto zapewnić 2-3 diody sygnalizacyjne informujące o stanie pracy / awarii naszego urządzenia
  - należy zapewniać możliwości naprawy i modyfikacji poszczególnych elementów systemu – przede wszystkim poprzez zapewnienie dostępu do nich (a gdy będzie on nie możliwy lub bardzo trudny poprzez położenie zapasowych przewodów), wykonywanie połączeń w łatwo dostępnych miejscach, itd.
  - należy zachowywać kompatybilność wsteczną zawsze wtedy gdy tylko to jest możliwe (nowe urządzenie, nowa wersja muszą pracować w istniejącej sieci, nowa wersja musi w prosty sposób móc zastąpić poprzednią)
  - należy konsekwentnie trzymać się określonych interfejsów i protokołów, jest to szczególnie ważne w niskopoziomowych (trudno aktualizowalnych, debugowalnych, występujących w dużej liczbie egzemplarzy) urządzeniach
- nie tworzenie „potworków" (bo to się będzie mściło):
  - należy unikać sztucznego ograniczania funkcjonalności tworzonego urządzenia (co jest nagminnie czynione w urządzeniach powszechnie dostępnych na rynku), na przykład:
    - jeżeli urządzenia ma złącze Ethernet i używa je np. do wystawienia jakiegoś WWW to należy udostępnić pełną funkcjonalność monitoringu/konfiguracji tego urządzenia przez to TCP/IP z użyciem standardowych protokołów (np. Modbus TCP)a nie wymagać do tego osobnego modułu
    - jeżeli urządzenia ma system operacyjny (np. Linuxa) należy zapewnić możliwość pełnego dostępu do niego (z prawami root'a) – także użytkownikowi, jeżeli je od nas kupił to on jest jego właścicielem

- z drugiej strony należy jednak unikać upychania funkcjonalności do granic możliwości (lepiej pozwolić „zmarnować się" kilku nóżkom mikrokontrolera niż zrezygnować z czytelności, powtarzalności czy możliwości diagnostycznych na rzecz np. obsłużenia kilku dodatkowych IO)
- należy unikać oszczędzania kilka złotych minimalizując ponad miarę rozmiar PCB czy eliminując jakieś złącza lub je ograniczając (na złączach oprócz sygnałów należy wystawiać też potrzebne zasilania/masy w odpowiednich ilościach)
- należy walczyć z z problemem plątaniny kabli już na etapie założeń projektowych poprzez:
  * stosowanie modularności – np. 1 sterownik Ethernet/IP (lub co najmniej RS-485/Modbus) na niezbyt dużą grupę wejść (np. jeden panel przycisków) / wyjść (np. jedną grupę przekaźników)
  * nie oszczędzanie miejsca na PCB i funduszy na gniazdka przyłączeniowe (nie robić przylutowywanej na stałe wiązki kabli, pamiętać o masach i zasilaniach)
  * nie oszczędzanie miejsca na PCB na otwory montażowe (najlepiej na wszystkich modułach wykonywać je w identycznych miejscach - z myślą o przyszłej obudowie, a nie tam gdzie popadnie)
  * w przypadku wykonywania większych układów, złożonych z kilku sterowników rozważyć stosowanie korytek grzebieniowych do ukrycia plątaniny kabli
  * w przypadku stosowania magistrali równoległej rozważyć multipleksowanie linii adresowej i danych
- należy pamiętać że źle zastosowane technologie, które mają służyć ułatwieniu serwisowania i obsługi systemu (koryta kablowe, stelaże/szafy rack 19", obudowy z mocowaniem na szynę DIN / TH-35) mogą przynieść odwrotny skutek, a liczy efekt w postaci dobrze zaprojektowanego, czytelnego, dobrze działającego, serwisowalnego, rozbudowywalnego urządzenia/systemu a nie konkretnie zastosowane technologie

7 Wykład wideo[1]

- *Bramki logiczne* – http://video.opcode.eu.org/09.01.mkv
- *Zatrzaski, przerzutniki i rejestry* – http://video.opcode.eu.org/09.02.mkv
- *Transmisja danych* – http://video.opcode.eu.org/09.03.mkv
- *Układy programowalne* – http://video.opcode.eu.org/09.04.mkv
- *Mikrokontrolery STM32* – http://video.opcode.eu.org/09.05.mkv

8 Literatura dodatkowa

- *Kurs elektroniki w serwisie* forbot.pl (https://forbot.pl/blog/kurs-elektroniki-dla-poczatkujacych-id5151).
- *Kurs elektroniki w serwisie* robotykadlapoczatkujacych.pl (http://robotykadlapoczatkujacych.pl/kurs-elektroniki-dla-poczatkujacych/).
- *Doświadczenia elektroniczne w circuitjs* (https://www.falstad.com/circuit/polish/e-index.html) - zbiór symulacji układów elektronicznych wraz z opisami.
- *Vademecum informatyki praktycznej* (http://vip.opcode.eu.org/) - zbiór materiałów na temat elektroniki i programowania, zawierający także dość rozbudowaną listę literatury dodatkowej.

1. Filmy posiadają napisy wgrane do kontenera multimedialnego jako osobny strumień – napisy mogą być włączone lub wyłączone w odtwarzaczu. W wielu filmach dużo dzieje się "na dole ekranu", dlatego polecamy odtwarzać filmy z napisami umieszczonymi poniżej filmu, np. przy pomocy polecenia: `vlc --video-filter='croppadd{paddbottom=120}' --sub-margin=-10 PLIK.mkv`

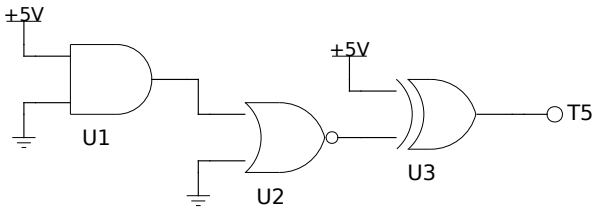
- *Linux i Python w Elektronicznej Sieci* (https://ciekawi.icm.edu.pl/lpes) - strona domowa kursu *LPES*, zawierająca nagrania i skrypty do innych wykładów, skrypty ćwiczeniowe, itd.
- *OpCode.eu.org* (http://vip.opcode.eu.org/) - strona internetowa autora kursu *LPES*, zawierająca różne materiały z szeroko rozumianej inżynierii komputerowej i elektronicznej (część materiałów pokrywa się z zawartością skryptów z tego kursu, ale nie wszystkie)

9 Zadania

Poniższe zadania znajdują się także w odpowiednich rozdziałach skryptu. Zostały jednak zamieszczone zbiorczo także w tym miejscu dla wygody czytelnika.

Zadanie 1.0.1

Podaj wartość napięcia (względem GND) w punkcie T5. Przyjmujemy iż użyte bramki działają na poziomie napięć 5V (prawda) / 0V (fałsz). Odpowiedź krótko uzasadnij.



Zadanie 2.0.1

Znajdź dokumentację (kartę katalogową) do rejestru przesuwnego, który kupiłeś/aś (CD4094 lub 74HC595). Odczytaj, jakie jest największe napięcie zasilania tego układu (*supply voltage*, w sekcji *Absolute maximum ratings*), oraz do których pinów należy podłączać napięcie zasilania (w sekcji *pin configuration*).

Zadanie 4.2.1

Zapoznaj się z dokumentacją układu 74HC574 i opisz sposób jego użycia (wraz z sposobem sterowania) jako modułu podłączonego do 8 bitowej magistrali równoległej w roli układu wejściowego oraz w roli układu wyjściowego.

Zadanie 4.2.2

Zapoznaj się z dokumentacją układu 74HC595 i opisz sposób jego użycia (wraz z sposobem sterowania) w roli układu wyjściowego podłączonego do magistrali szeregowej.

10 Rozwiązania zadań

Poniżej zamieszczone są przykładowe rozwiązania „głównych” zadań z tego skryptu wraz z komentarzami. Wiemy że zajrzenie do nich już przy pierwszej trudności jest kuszące, mimo to rekomendujemy przynajmniej podjąć uczciwą, co najmniej kilkunastominutową na każde z zadań, próbę rozwiązania tych zadania bez zaglądania do odpowiedzi.

Pamiętaj!: Samodzielne rozwiązanie problemu (wraz z wszystkimi trudnościami po drodze i popełnionymi błędami) jest dużo bardziej kształcące od nawet wielokrotnego przepisania gotowego rozwiązania, jednak nawet jednokrotne przepisanie rozwiązania jest bardziej kształcące od wielokrotnego przekopiowania go.

Rozwiązanie zadania 1.0.1

Analizujemy stany logiczne kolejnych bramek:
U1: 0V and 5V => 0V
U2: not (0V or 0V) => 5V
U3: 5V xor 5V => 0V Zatem w T5 mamy stan niski, czyli 0V.

Rozwiązanie zadania 2.0.1

Rozwiązanie zadania 4.2.1

74HC574 to 8bitowy rejestr z wyjściem trójstanowym.
Użycie jako układ **wejściowy**: peryferium steruje sygnałami D0...D7 i sygnałem CP (CLK), do szyny danych podłączone wyjścia Q0...Q7, dekoder adresu wraz z sygnałami sterującymi szyny steruje wejściem OE celem wystawienia danych z rejestru na szynę przy dokonywaniu odczytu z niej przez urządzenie nadrzędne.
Użycie jako układ **wyjściowy**: peryferium używa wyjść Q0...Q7 (np. do podłączenia kontrolek LED), OE w stanie niskim (wyjścia aktywne), D0...D7 podłączone do szyny danych, dekoder adresu wraz z sygnałami sterującymi szyny steruje wejściem CP (CLK) celem dokonania zapisu z szyny do rejestru przy wykonywaniu takiej operacji na magistrali przez urządzenie nadrzędne.

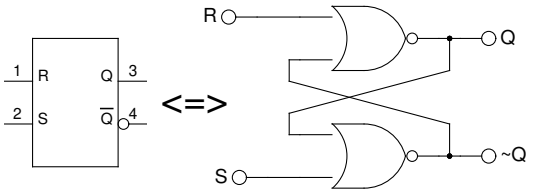
Rozwiązanie zadania 4.2.2

74HC574 to 8bitowy rejestr z wejściem szeregowym i wyjściem równoległym wyposażonym w rejestr wyjściowy z wyjściami 3 stanowe.
Układ nadrzędny (magistrala) steruje sygnałami DS (dane szeregowe), SHCP (zegar danych szeregowych) oraz STCP (zerag buforu wyjściowego) w sposób opisany w rozdziale poświęconym magistrali szeregowej. Wyjścia układu podłączone do peryferium (np. kontrolek LED). Wejście OE w stanie niskim (wyjścia zawsze aktywne). Wejście MR w stanie wysokim (reset nie aktywny).

11 Zadania praktyczne

Zadanie 11.0.1

Na schemacie przedstawiono dwubramkową budowę przerzutnika RS w wariancie z wejściami nie zanegowanymi (zastosowanie bramek NAND w miejsce NOR spowodujwe zanegowanie wejść). Zbuduj taki układ i sprawdź jego działanie.



Nie demontuj układu - będzie przydatny w zadaniu 11.0.3!

Zadanie 11.0.2

Spróbuj zbudować własną bramkę logiczną w oparciu o tranzystory NPN i PNP. Pamiętaj że w odróżnieniu od przypadku pokazanego na schemacie (w treści skryptu), gdzie zastosowane były tranzystory NMOS i PMOS, w przypadku użycia tranzystorów bipolarnych wymagane jest stosowanie rezystora na bramce.
Wskazówka: zacznij od zbudowania bramki NOT, gdyż ona jest najprostsza – to po prostu półmostek H. Później, po sprawdzeniu działania, możesz skomplikować układ - przerobić go na NAND lub NOR.

Zadanie 11.0.3

Podłącz do kolejnych wyjść układu rejestru przesuwnego z buforem wyjściowym (np. CD4094 lub 74HC595) 4 diody LED (pamiętaj o rezystorach). Zapisz do rejestru i ustaw na wyjściach taką wartość aby świeciły się dwie pierwsze i ostatnia dioda, użyj w tym celu ręcznego manipulowania sygnałami:

- wejścia szeregowego (SERIAL IN), służącego do wprowadzania danych
- zegara danych (CLOCK, CLK), determinującego chwilę odczytu kolejnego bitu z wejścia szeregowego
- zegara wyjść (STROBE), determinującego chwilę przepisania danych z rejestru przesuwnego do rejestru wyjściowego

Wskazówka 1: zapoznaj się z dokumentacją posiadanego układu, ustal nazewnictwo używane do określania poszczególnych sygnałów (może się różnić nawet w zależności od producenta układu) oraz numery nóżek układu z nimi związane (mogą się różnić w zależności od modelu / wariantu obudowy).
Wskazówka 2: do podawania sygnału zegtara danych użyj wyjścia układu zbudowanego w zadaniu 11.0.1