Barramentos e Transferências de Dados

Para que haja comunicação entre os circuitos digitais, seja interno ou externo ao µP, é necessário que eles se conectem através de fios (ou trilhas numa placa de circuito impresso ou no próprio silício do CI). Este conjunto de fios agrupados se chama barramento.

Tipos de informação

Existem 3 tipos de informação em um µP: Dados, endereço e controle.

Dados: É a informação que se quer processar (transformar), os dados são os conteúdos dos registradores e da memória. É o tipo principal de informação que os programadores têm acesso.

Endereço: É a informação de onde se encontra do dado. Pode ser um endereço de memória ou do registrador dentro do µP. Programadores avançados trabalham com endereços de variáveis como ponteiros em linguagem C, por exemplo.

Controle: Indica o que fazer com o dado. Como exemplo de um registrador, é o controle que vem da unidade de controle e indica se ele deve ser lido ou deve ser escrito um dado que está na sua entrada.

Desta forma, quanto existem também 3 tipos de barramentos: O de dados, endereço e controle.

Transferências de Dados

Em um µP existem vários registradores e outros circuitos que precisam se comunicar. Existem duas estratégias: O uso de barramentos dedicados e de um barramento único.

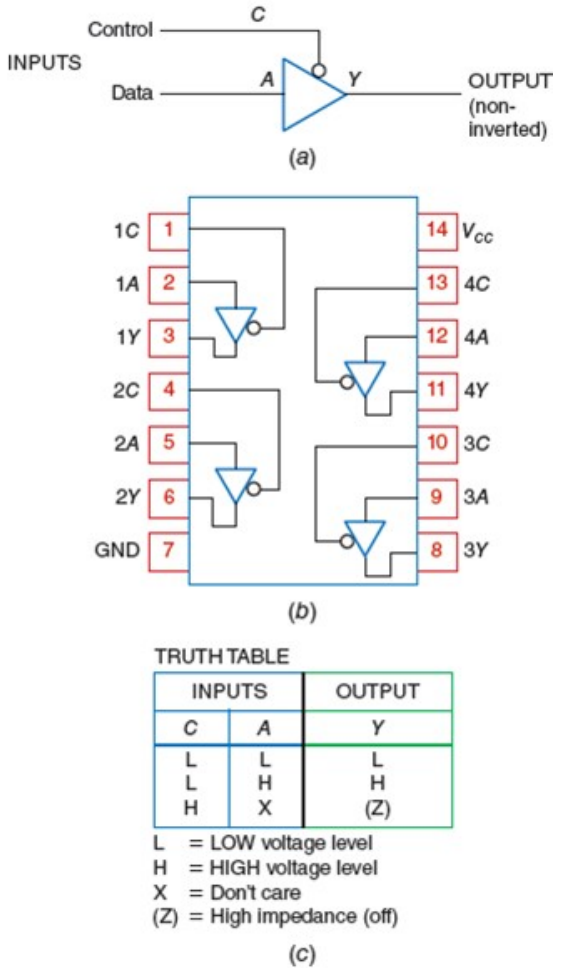
Barramentos dedicados

São linhas de comunicação que servem para um só fim, como por exemplo, conectar dois registradores. Os barramentos dedicados funcionam bem para poucos elementos no circuito mas quando este número aumenta, o número de fios cresce a ponto de inviabilizar um projeto pois dentro do CI teremos cruzamento de muitos fios o que encarece a produção deste CI.

Barramento único

Neste esquema de arquitetura, o barramento é único para todos os registradores e outros circuitos que o usarem, sendo as entradas e saídas ligadas ao mesmo barramento (para que qualquer registrador possa transferir seus dados para qualquer outro, com a restrição de que o meio só

suporta uma transferência por vez). Este esquema não funciona com os circuitos apresentados até agora pois se ligarmos duas saídas lógicas, teremos um problema lógico e elétrico (um pode ter saída 0 e outra, saída 1). Para resolver isso, usamos o ***buffer* de três estados** ou ***three state buffer***. Existe um CI com 4 *buffers* (74125) e ele é mostrado na figura abaixo:



74125 – composto de 4 buffers de 3 estados. (a) diagrama de blocos (b) pinagem do CI (c) tabela verdade

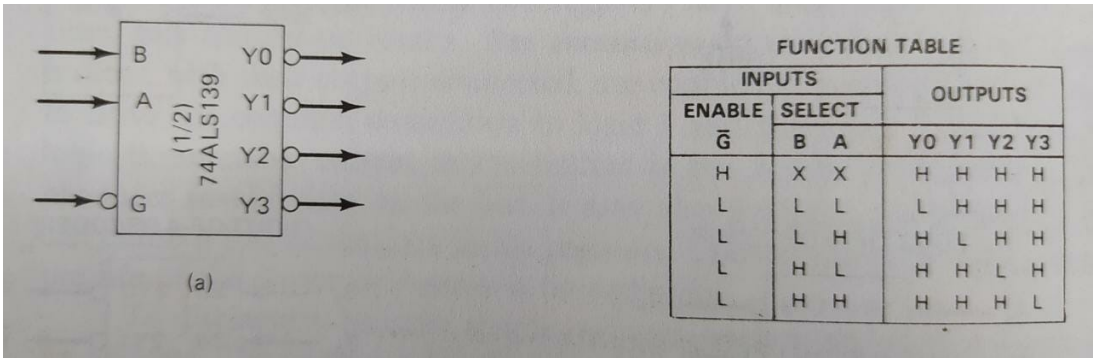
Este buffer funciona como uma chave controlada eletronicamente. Isto significa que a chave pode ser fechada (saída = entrada) ou aberta (alta impedância – Z na figura (c)). O controle C, pode ser também chamado de *ENABLE* (ou habilita em português) e dependendo do seu nível lógico, abre ou fecha a chave. Veja no exemplo do 74125 que o controle habilita a chave ser fechada em nível lógico 0. O pequeno círculo indica isso: a ação vai ser tomada em 0.

Este buffer deve ser colocado nas saídas dos registradores pois se torna o controle de saída dos registradores

Decodificadores

Os decodificadores são circuitos que selecionam uma saída de acordo com um código de entrada. Eles são importantes em sistemas microprocessados para que se associe a noção de endereço ao sistema.

A figura abaixo mostra um dos decodificadores existentes no CI 74139. Ele é um decodificador de duas saídas (Y0 a Y3) e com um código de 2 bits de entrada (A, B). Além disso ele tem um sinal G que funciona como habilita, ou seja, o decodificador só seleciona uma saída se G estiver habilitado. Lembrando que as saídas têm círculos no diagrama de blocos, e isso faz com que só uma saída poderá ter 0 (saída selecionada) e todas as outras terão 1. Da mesma forma o G tem um círculo, isso significa que o decodificador estará habilitado somente se G for 0. A tabela verdade mostra as opções. O X na tabela verdade indica que as opções podem ser 0 ou 1 e a saída é a mesma, servindo para diminuir o tamanho da tabela.



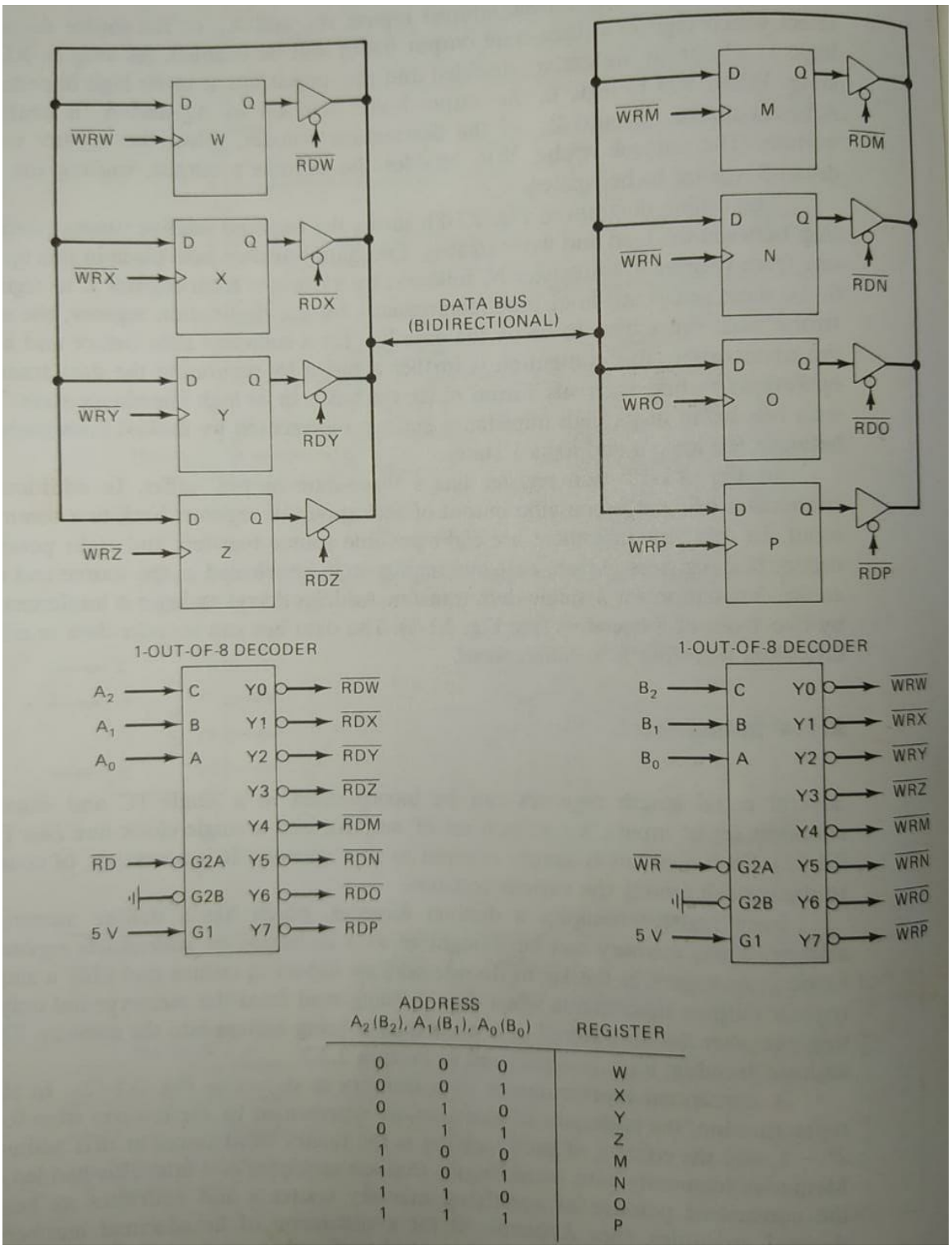
FUNCTION TABLE

INPUTS			OUTPUTS			
ENABLE	SELECT					
$\bar{G}$	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

Decodificador de 4 saídas e sua tabela verdade

Transferências entre registradores

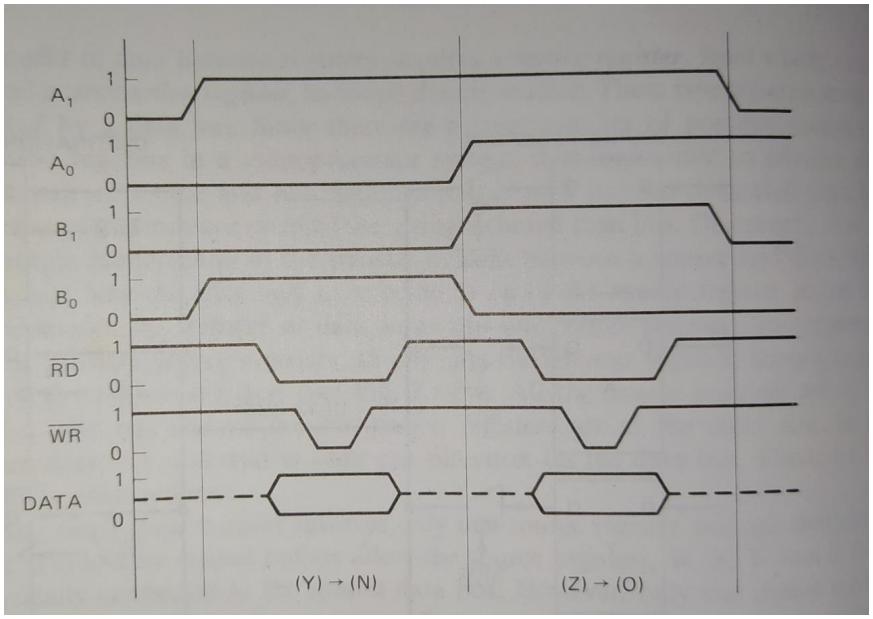
A figura abaixo mostra 8 registradores ligados a um barramento único. Todas as entradas e saída de dados vão para o mesmo barramento. Temos agora dois decodificadores de 8 saídas e eles têm a mesma função dos de 4 saídas apresentado anteriormente. Os controles G1, G2B e G2A são controles paralelos e são necessários quando for preciso acrescentar outros sinais de controle vindo de outras fontes. No nosso caso somente precisamos de um sinal de controle que vem da unidade de controle e esses sinais são o $\overline{WR}$ (escrita) e $\overline{RD}$ (leitura), devendo os outros controles serem habilitados (G1 com 5V – lógico 1 e G2B aterrado – lógico 0). Uma outra forma de se indicar que uma ação ocorre com 0 é barrar o sinal como o $\overline{WR}$ e $\overline{RD}$. Cada decodificador cuida de uma operação específica e basta que a unidade de controle coloque o endereço e os sinais de leitura e escrita corretamente que estes serão enviados para o controle correto do registrador (escrita no clock e leitura no habilita do *buffer* de 3 estados). A figura ainda mostra o endereço associado a cada registrador.



Registradores ligados a um barramento único com entradas e saídas controladas por *clock* e controle do *buffer* de 3 estados respectivamente. Os decodificadores indicados enviam os sinais para o registrador correto

A figura abaixo mostra os sinais enviados pela unidade de controle para uma transferência de dados do registrador Y para o N e na sequência de Z para O. Quando $\overline{RD}$ é 0, os buffers de 3 estados deixam o dado disponível para o barramento. Somente neste momento em que o dado está

disponível, que um outro registrador pode registrar o que está na entrada de dados. É por isso que o sinal de $\overline{WR}$ deve subir antes que o $\overline{RD}$ suba para 1. Como o *clock* é disparado por transição de subida, esta transição deve acontecer enquanto o $\overline{RD}$ estiver 0.



Sinais enviados pela unidade de controle para os decodificadores da figura anterior