UNIVERSIDADE DE BRASÍLIA – UNB

FACULDADE DO GAMA – FGA

Adrianne Alves da Silva

Explicação dos projetos

Atividade 1 - Mostrar dois números diferentes ao mesmo tempo nos displays de 7 seg da placa Basys 3- Os alunos devem enviar o código VHDL, compilado até o momento de baixar o código para a placa. Além disso, devem enviar um texto com as explicações do código.

A estrutura básica para realizar essa atividade conta com a implementação de um multiplexador, um divisor de clock, um decodificador BCD – Decimal (visto que o enunciado pede para mostrar dois diferentes números e hexadecimal tem a representação também de letras) e um seletor de display. Ao passo do clock dividido (que atrasa a informação em relação ao clock da placa de 100 Mhz) que controla a ativação e desativação do anodo de cada display, o mux “escolhe” os valores a serem jogados no display e o decodificador os transforma para o controle de cada segmento do display de 7 segmentos. Assim, cada vez que um número está sendo demonstrado em um display, o outro está apagado, mas essa mudança, é feita de maneira tão rápida que não é notada, para isso, o contador do clock possui tamanho n, tal que frequência da placa / (2^(n-1)) é igual a frequência desejada, que para um humano não enxergar as variações basta estar acima de 70 Hz (O divisor de clock aqui implementado possui contador de tamanho 21 para gerar uma frequência de 100Hz).

Pretende-se que através de dois números fornecidos um destine-se à um display e outro à outro display, sem que eles se repitam, por esse motivo é utilizado um multiplexador, sendo duas entradas de 4 bits, é necessário apenas uma seletora para destinar um ou outro, para isso utilizou-se a expressão lógica do multiplexador, tal que dado a seletora A e entradas I0 e I1: S = A I0 + A.I1 .

Para que os números não se repitam no display é necessário ligar um para mostrar a informação e desligar o outro mas o tempo entre cada atividade é controlado pelo divisor de clock aqui estabelecido. Através da função *rising_edge* o código incrementa o contador a cada variação de baixa para alta tensão, e pega o dígito mais significativo do vetor contador, pois este possui a frequência desejada. O clock dividido influi diretamente na seleção do display a ser ativado, assim, podemos dizer que enquanto qualificamos que um deles

recebe a divisão do clock, o outro recebe o inverso. No arquivo principal os ânodos dos demais displays são setados com 1 para que permaneçam apagados.

No arquivo principal foram realizadas as devidas ligações entre as componentes utilizadas, tal que o clock dividido está ligado a seletora de displays e ao mux, o mux envia o número escolhido ao decodificador e por fim, tanto a resposta da seletor quanto do decodificador vão para os displays.

Atividade 2 - Projeto do comparador de magnitude. Enviar a simulação do circuito no proteus ou circuit maker e as explicações do projeto.

O comparador de magnitude, assim como demais tipos de circuito possui uma tabela verdade, para a comparação de números de 1 bit, por exemplo, teríamos a seguinte tabela :

A	B	A>B	A=B	B>A
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

Tendo isso em mãos, é fácil retirar a expressão lógica para a igualdade, e maior ou menor magnitude dos números envolvidos, tal que para esse exemplo, teríamos : igualdade - not(A XOR B), A maior – A and not(B) , B maior – not(A) and B. Entretanto, se aumentarmos a quantidade de bits a retirada de uma expressão se torna mais complicada, no caso proposto, para dois números de 4 bits então, que totaliza 8 entradas, tal que a tabela obedece a ordem de 256 linhas para o número de possibilidades de combinação. Montar essa tabela é uma tarefa árdua, mas pior ainda seria simplificar a expressão obtida, por esse motivo, as tabelas foram montadas por meio do site < *http://www.32x8.com/var8.html* > que conta com tabelas de 8 variáveis. O processo de preenchimento da saída teve que ser feito da mesma maneira para os 3 casos, sendo ambos os números iguais, a > b ou b>a, e o site fornece a simplificação utilizando mapas de karnaugh, como resultado, chegou –se às seguintes expressões, sendo A o numero 1 e B o número 2 :

1 . A=B: not(A3 xor B3) and not(A2 xor B2) and not(A1 xor B1) and not (A0 xor B0), ou seja, se a comparação de todos os bits tiverem resultado de que são iguais, visto que a igualdade entre dois bits é dada por not(N1 xor N2);

2. A>B : (A3 and not(B3)) or (not(A3 xor B3) and A2 and not (B2)) or ((A3 xor B3) and not(A2 xor B2) and A1 and not(B1)) or (not(A3 xor B3) and not(A2 xor B2) and not(A1 xor B1) and A0 and not(B0));

3. B> A : (not(A3 xor B3) and not(A2 xor B2) and not(A1 xor B1) and not (A0 xor B0)) or (A3 and not(B3)) or (not(A3 xor B3) and A2 and not (B2)) or ((A3 xor B3) and not(A2 xor B2) and A1 and not(B1)) or (not(A3 xor B3) and not(A2 xor B2) and not(A1 xor B1) and A0 and not(B0));

De maneira mais simplificada, visto que deseja-se encontrar o schematic do circuito lógico, foi possível perceber da seguinte maneira :

1 . Cada igualdade dada por not(A xor B) ou seja ,

2 . A = B (igualdade de cada bit do número de posição 3 a 0) , começando do mais significativo : igualdade3 and igualdade2 and igualdade1 and igualdade0;

3. A>B poderia então dar-se como : (A3 and not(B3)) or (igualdade3 and A2 and not (B2)) or (igualdade3 and igualdade2 and A1 and not(B1)) or (igualdade3 and igualdade2 and igualdade1 and A0 and not(B0));

Vendo dessa maneira ficou menos complicado de montar o esquema obtido, e em cada saída conectou-se um LED que quando acesso identificava a magnitude de um número em relação a outro. Foram usadas em alguns casos portas lógicas com mais entradas para reduzir o tamanho do circuito, fazendo as adaptações necessárias para não alterar o resultado final. As entradas foram ligadas em switches lógicos para que pudesse ser realizada a simulação.