ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO
Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos

PMR 2433 Eletrônica Analógica e Digital

2º Semestre 2015

Experiência 3

CIRCUITOS SEQUENCIAIS & FPGA (REV. A)

A FPGA (Field Programmable Gate Array) é uma das tecnologias mais avançadas para a implementação de circuitos lógicos digitais. São dispositivos facilmente programáveis, de alta densidade, com os quais podemos implementar a um custo muito baixo sistemas digitais equivalentes a milhões de transistores.

Esta experiência é uma introdução ao ambiente de desenvolvimento Foundation Series para a programação de FPGAs. Vamos programar e testar uma FPGA utilizando a placa de desenvolvimento Digilab, da Digilent. O projeto do circuito é feito no computador com o programa Foundation Series, e o arquivo de configuração resultante é transferido para a FPGA através da porta paralela do computador.

Além desta apostila, você vai precisar do manual de usuário da placa de desenvolvimento Digilab XLA5, que será utilizada no laboratório para testar nossos projetos em FPGA. Opcionalmente, você pode obter o manual do software Foundation Series, que usaremos para projetar os circuitos. Esse manual está disponível no endereço http://sites.poli.usp.br/d/pmr2433. Não é estritamente necessário ler esse manual, mas essa leitura prévia certamente facilitará sua vida durante a aula.

Usaremos esse sistema de desenvolvimento para estudar alguns circuitos sequenciais utilizando latches e flip-flops. Portanto, faça uma revisão do capítulo 6 do livro texto.

Lembre-se: você deve fazer o PRÉ-RELATÓRIO desta experiência com antecedência e entregá-lo no início da aula. Revise o pré-relatório e a apostila antes da aula. Sua compreensão poderá ser avaliada por meio de uma ARGUIÇÃO ORAL. Traga para a aula a apostila **IMPRESSA**. Os pontos importantes das atividades devem estar **destacados ou grifados**.

PARTE I TEORIA

3.1 FPGA (Field Programmable Gate Array)

Internamente, uma FPGA é composta por células padrão que podem ser programadas individualmente para constituir funções lógicas de n entradas e m saídas, chamadas de PLB (*programmable logic block*). As m saídas de um PLB podem ser conectadas a suas próprias entradas (formando circuitos realimentados) ou a entradas de outros PLBs, através de matrizes de interconexões denominadas de SM (*switch matrix*). As SMs também podem ser programadas individualmente. Como uma FPGA possui milhares de PLBs e SMs, temos um dispositivo capaz de implementar desde funções lógicas simples a sistemas digitais bastante complexos.

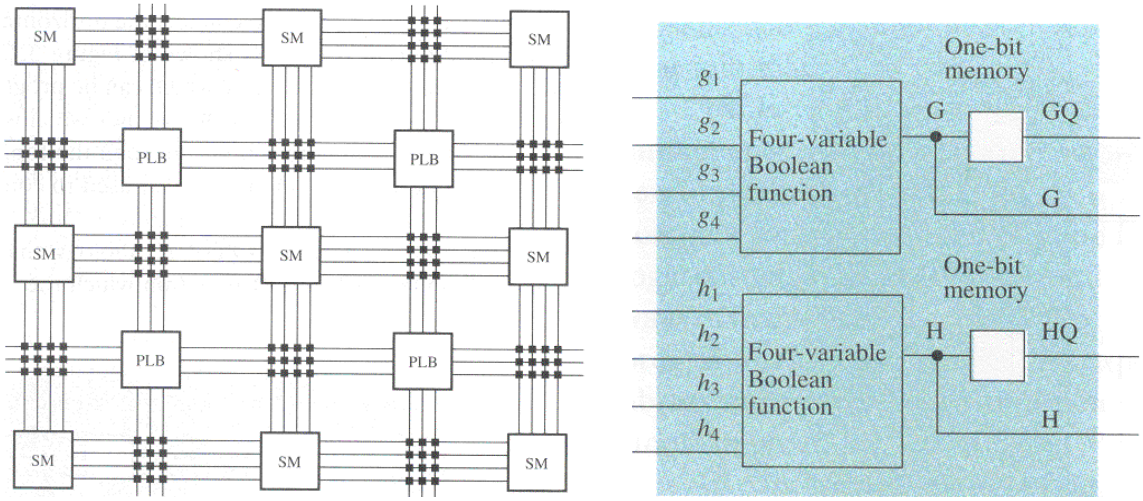
A Figura 3.1 ilustra a estrutura simplificada de uma FPGA. Neste exemplo, cada PLB contém duas memórias programáveis (do tipo RAM), cada uma contendo 16 posições de 1 bit. Essa memória é chamada LUT (*look up table*), e cada uma delas permite implementar qualquer função booleana de 4 variáveis de entrada, como se fossem tabelas da verdade eletrônicas.

Note que, neste exemplo, temos $n = 8$ e $m = 2$. Observe que a estrutura desse exemplo de PLB não permite realizar qualquer função lógica de 8 variáveis, e sim duas funções de 4 variáveis cada. Além disso, a matriz de interconexão SM não permite conectar quaisquer PLBs. Mas estas restrições podem ser superadas, interconectando-se vários PLBs.

A Figura 3.2 mostra como implementar um somador completo de 1 bit em uma PLB. Repare que a LUT superior (G) reproduz a tabela da verdade do vai-um (c_{i+1}) e a inferior (H) contém a tabela da verdade do i-ésimo bit da soma (s_i). Como são funções de três variáveis, uma das entradas de cada LUT não é utilizada. No caso da LUT G, por exemplo, a entrada g_1 foi deixada em aberto, e portanto o conteúdo da LUT deve ser o mesmo tanto para $g_1 = 0$ como para $g_1 = 1$ (na figura, isto é indicado com X na coluna g1 de endereços). Analogamente, na LUT H, a entrada h_4 não foi utilizada.

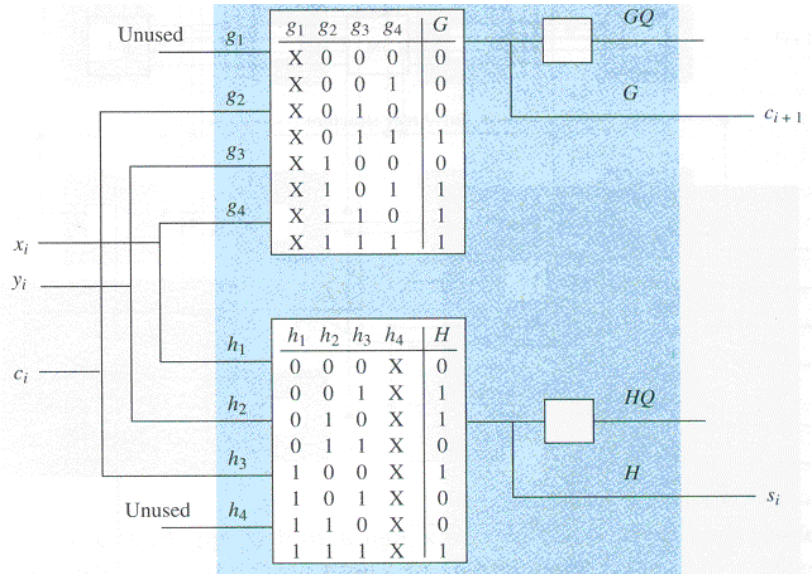
Uma FPGA tem vários pinos que podem ser configurados para servir de entradas ou de saídas para as funções programadas. A programação de PLBs, SMs e pinos é armazenada internamente em memória volátil, o

que significa que uma pastilha pode ser reprogramada milhares de vezes. Isso as torna uma excelente tecnologia para a prototipagem rápida de sistemas digitais. Projetistas de circuitos integrados, por exemplo, costumam utilizá-las para testar por partes o projeto de CIs complexos, como um microprocessador moderno.



(a) (b)

Figura 3.1 a) Estrutura matricial de uma FPGA, com PLBs interconectadas por SMs
b) Exemplo de PLB, composta por duas LUTs de 4 variáveis



g_1	g_2	g_3	g_4	G
X	0	0	0	0
X	0	0	1	0
X	0	1	0	0
X	0	1	1	1
X	1	0	0	0
X	1	0	1	1
X	1	1	0	1
X	1	1	1	1

h_1	h_2	h_3	h_4	H
0	0	0	X	0
0	0	1	X	1
0	1	0	X	1
0	1	1	X	0
1	0	0	X	1
1	0	1	X	0
1	1	0	X	0
1	1	1	X	1

Figura 3.2 Somador completo (c_{i+1} s_i) implementado com uma PLB

3.2 Programação

A programação interna de PLBs e SMs normalmente é bastante complexa, mesmo para se implementar funções lógicas bem simples. Para facilitar a sua vida, os fabricantes de FPGAs investem pesadamente no desenvolvimento de softwares integrados para a programação automática desses dispositivos. São os chamados *compiladores de silício*. A ideia é permitir que você faça o projeto utilizando portas lógicas e bibliotecas de funções, de forma hierárquica. O compilador cuida de converter seu projeto em sequências de bits que posteriormente são transferidas para a memória da FPGA.

Em linhas gerais, o ciclo de desenvolvimento com uma FPGA envolve as seguintes etapas.

- Projeto digital (*design entry*): é o projeto propriamente dito do circuito que será implementado dentro da FPGA. Em projetos mais simples, utilizam-se normalmente editores gráficos para se desenhar o diagrama lógico do circuito – é o chamado modo de captura esquemática (*schematic entry*). Em casos mais complicados, podem ser utilizadas linguagens de programação (*HDL, hardware description language*), como as linguagens ABEL, VHDL e Verilog. São linguagens que lembram C, por exemplo, onde funções lógicas são definidas por expressões algébricas (do tipo Y = A & B) e chamadas de sub-rotinas.
- Simulação funcional: simula o funcionamento do circuito projetado com portas lógicas, sem considerar os atrasos de propagação, para que se possa verificar se a lógica implementada corresponde ao esperado.

- Implementação: é quando o projeto é compilado para formar os bits de programação da FPGA. É hora de se verificar se o modelo de FPGA a ser utilizada (*target*) comporta o projeto, isto é, se o CI contém PLBs e SMs em quantidade suficiente para implementar o circuito desejado.
- Verificação: simula eletricamente o circuito implementado na FPGA, considerando tempos de atraso, para se verificar se o circuito é capaz de operar na velocidade desejada, e se problemas como *harzard* não afetam o funcionamento do circuito.

Nesta experiência, utilizaremos o software Foundation Series, da Xilinx. Trata-se de um ambiente de desenvolvimento profissional bastante complexo. Nesta aula, procuraremos aprender a usar as funções elementares do programa. Ao utilizá-lo em sala, procure ler os manuais *on-line* disponíveis no menu Help e, em algumas situações, teclando-se F1 após selecionar algum elemento das janelas. As dicas desta apostila visam dar a você um ponto de partida, mas ao se limitar apenas a elas, você acabará com uma visão muito fragmentada do poderoso sistema que terá em mãos.

3.3 Placa de Desenvolvimento Digilab

No laboratório, vamos testar os circuitos projetados utilizando a placa de desenvolvimento Digilab XLA5, da Digilent. Ela contém uma FPGA Spartan XCS10 PC84, da Xilinx, de 84 pinos, capaz de acomodar circuitos equivalentes a até 10.000 portas lógicas, aproximadamente.

A placa contém ainda vários conectores e dispositivos que permitem verificar o funcionamento do circuito programado na FPGA. Dentre eles, temos 4 displays de 7 segmentos, 8 chaves de duas posições, 8 *leds* (*Light-Emitting Diodes*), 4 botões de contato momentâneo (*push buttons*).

Veja o manual do usuário em anexo. Na página 12, a tabela indica em que pinos da FPGA esses dispositivos estão conectados, e essa informação deve ser levada em conta no projeto do circuito. Por exemplo, para acender e apagar o *led* LD1, devemos conectar o pino 69 da FPGA a uma saída do circuito projetado.

A página 13 mostra um exemplo de um circuito que controla o *led* LD1 através de um AND entre as chaves SW1 (pino 28 da FPGA) e SW2 (pino 27). Note que, para configurar um pino da FPGA como entrada, devemos incluir no projeto dois elementos: o IPAD e o IBUF. O primeiro determina o número do pino a ser usado e o segundo insere um circuito elétrico (*buffer* de entrada) para conectar o pino a entradas de portas lógicas.

Analogamente, a configuração de um pino da FPGA como saída exige os elementos OBUF (*buffer* elétrico de saída) e OPAD (definição do pino).

3.4 Contadores Síncronos

Uma aplicação bastante simples porém muito importante dos flip-flops é a construção de contadores binários. A Figura 3.3 mostra como construir um contador de um bit, usando um flip-flop tipo D com a saída invertida realimentada na entrada. A cada borda de subida do sinal C, a saída Q se inverte de 0 para 1 e de 1 para 0, sucessivamente.

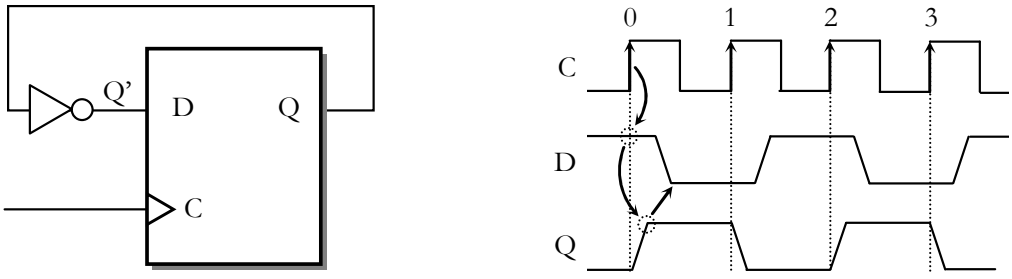
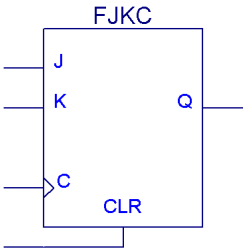


Figura 3.3 Contador de um bit

Repare que este circuito somente funciona porque há *atrasos* envolvidos. Se todos os sinais mudassem instantaneamente, o valor de Q ficaria indefinido pois D é igual a Q' e teríamos Q = Q' (estranho, não?) após a borda de *clock*. No entanto, como o sinal D leva alguns nano-segundos para se inverter após uma mudança em Q, o *próximo valor* da saída Q após a borda de clock será o inverso do *valor atual* depois de um pequeno atraso, conforme mostra a carta de tempos da Figura 3.3. Ou seja, para o instante *k*, temos $Q^{k+1} = (Q^k)'$.

Repare que o sinal C é uma onda quadrada ideal, enquanto que mudanças nos sinas D e Q aparecem como bordas inclinadas, indicando que as mudanças de nível levam algum tempo para se consumarem. Note que exageramos o atraso do sinal D com relação a Q. Como mostra a figura, instantes antes da borda 0 do *clock*, temos Q = 0 e (devido ao inversor) D = 1. Após a borda 0, a saída Q do flip-flop copia o valor de D, e depois de um certo atraso D se inverte.

Podemos cascatear vários contadores de 1 bit para formar um contador binário de *n* bits. Mas para isso vamos usar flip-flops tipo JK, que são mais adequados para esta aplicação. A Figura 3.4 ilustra um flip-flop JK com entrada de *clear* (CLR) assíncrono, que zera a saída Q quando em 1, independentemente do sinal de *clock* C. Com CLR = 0, a saída Q é atualizada na borda de subida do *clock*, conforme o valor das entradas de comando J e K. A entrada J leva a saída a 1, enquanto que K zera a saída; para JK = 00, a saída se mantém no valor atual, e para JK = 11 a saída se inverte.

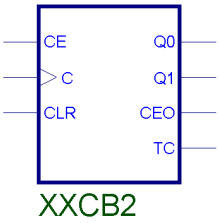


CLR	C	J K	Q^{k+1}	
1	X	X X	0	(clear assíncrono)
0	↑	0 0	Q^k	(mantém)
0	↑	0 1	0	(reset)
0	↑	1 0	1	(set)
0	↑	1 1	$(Q^k)'$	(inverte)

Figura 3.4 Flip-flop tipo JK e sua tabela característica

3.4.1 Contador de dois bits

Podemos usar esse flip-flop para construir um contador síncrono de 2 bits, denominado xxCB2, ilustrado na Figura 3.5. A entrada CLR zera o contador, assincronamente com o sinal de *clock* C. A entrada CE, chamada *count enable*, habilita a contagem quando em 1, fazendo as saídas Q_1 Q_0 contarem ciclicamente de 00 a 11 nas bordas de subida do *clock*. Na tabela, a notação "Q[1:0]" representa a palavra de dois bits Q_1Q_0 (nessa ordem), e é a notação usada pelo software de programação de FPGA *Foundation Series* para dar nomes a **barramentos** (ou *buses*) constituídos por vários bits numerados, como é o caso dos bits Q_1 e Q_0.



CLR	CE	C	$Q[1:0]^{k+1}$	CEO	TC	
1	X	X	0 0	0	0	(clear assíncrono)
0	0	X	$Q[1:0]^k$	0	$Q_1.Q_0$	(clock desabilitado)
0	1	↑	$Q[1:0]^k + 1$	$Q_1.Q_0$	$Q_1.Q_0$	(incremento módulo 4)

Figura 3.5 Contador síncrono de dois bits, com clear assíncrono (CLR) e *count enable* (CE)

As saídas TC (*terminal count*) e CEO (*count-enable output*) são saídas de *status* (ou *flags*) do contador. A saída TC quando em 1 indica que a contagem atingiu o valor máximo (ou seja, Q[1:0] = 11). O *flag* CEO também indica o fim de contagem, mas vale 1 somente se, além de termos Q[1:0] = 11, o contador também se encontra habilitado (CE = 1). Por esse motivo, a saída CEO é geralmente usada para fazer o **cascateamento** desses contadores, como se verá mais a frente.

A Figura 3.6 mostra do diagrama lógico do circuito do contador. Repare que, por ser um circuito síncrono, o sinal de *clock* é o mesmo para os dois flip-flops. Estando habilitado (CE = 1) e com CLR = 0, o primeiro flip-flop (H1) inverte a saída Q_0 a cada borda de *clock*. O segundo flip-flop (H2) inverte a saída Q_1 sempre que a anterior for 1, ou seja, quando o bit menos significativo atinge seu final de contagem. As portas AND2 geram as saídas TC e CEO, seguindo a tabela da Figura 3.5.

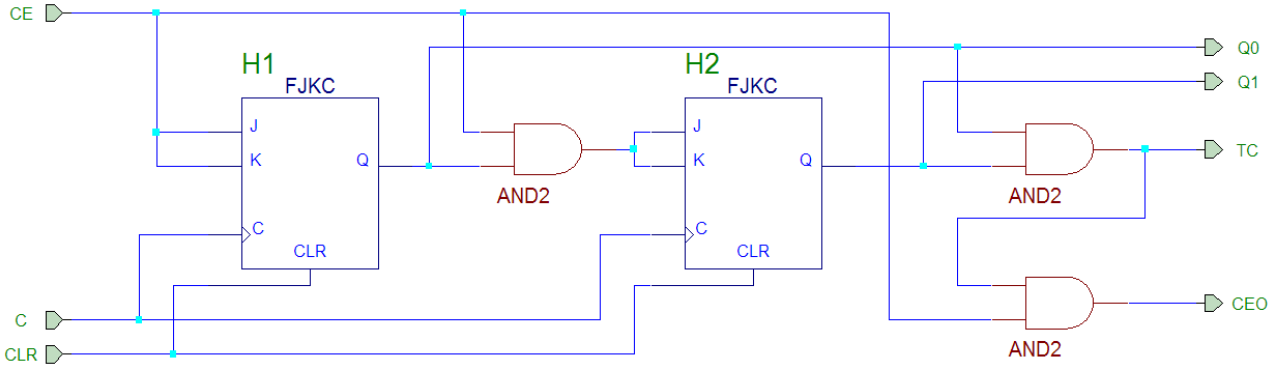


Figura 3.6 Circuito do contador síncrono de dois bits

A carta de tempos da Figura 3.7 ilustra o funcionamento do contador. Para deixá-la mais concisa, representamos as bordas de subida do sinal de *clock* por um trem de impulsos, e os bits Q_1 e Q_0 são representados pelo barramento condensado Q[1:0], no qual indicamos apenas as transições e o valor da contagem em binário.

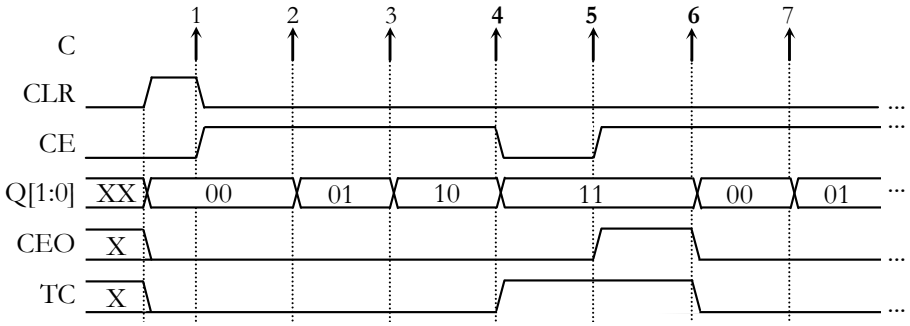


Figura 3.7 Carta de tempos do contador de dois bits

A carta retrata um fato comum em circuitos síncronos. Repare que as entradas CLR e CE também variam sincronamente com a borda 1 do sinal de *clock* C: nesse caso, deve-se considerar os níveis no momento **anterior** à borda (ou seja, CLR = 1 e CE = 0), e por isso a contagem se mantém em zero. O mesmo acontece nas bordas 4 e 5. No instante 4, o contador ainda está habilitado e incrementa Q[1:0], pois o sinal CE muda somente *após* a borda de *clock*. Analogamente, o contador ainda está desabilitado no instante 5, voltando incrementar somente depois, no instante 6. Concluindo: em circuitos síncronos, considere o valor presente nas entradas no momento *anterior* à borda de *clock*.

Veja a diferença entre as saídas CEO e TC. A contagem em Q_1Q_0 atinge 11 na borda 4 do *clock* C, ao mesmo tempo em que o contador é desabilitado (CE = 0), fazendo com que CEO permaneça em zero. A saída CEO vai para 1 somente após a borda 5, em que o contador é novamente habilitado (CE = 1). Por sua vez, a saída TC vai a 1 já na borda 4, tão logo a contagem atinge 11.

3.4.2 Estendendo para mais bits

Para construir um contador de três bits (de saídas $Q_2Q_1Q_0$), precisamos acrescentar mais um flip-flop no circuito da Figura 3.5 para gerar o bit Q_2. Qual seria a lógica de funcionamento deste bit?

Assumindo que o contador esteja habilitado (CE = 1) e começando a contagem em 000, o bit Q_2 deve ir de 0 para 1 após a contagem atingir 011 (ou 3 em decimal), e voltar para 0 após 111 (7 em decimal). Ou seja, Q_2 deve se inverter sempre que Q_1 e Q_0 forem iguais a 1.

Estendendo esse raciocínio para um número qualquer de bits, temos que o i-ésimo bit (Q_i) do contador deve se inverter sempre que o contador estiver habilitado e TODOS os bits menos significativos que ele forem iguais a 1. Ou seja, inverte-se Q_i sempre que se verificar a condição $CE.Q_{i-1}.Q_{i-2}\ldots.Q_0 = 1$, com $i > 0$.

3.5 Cascateamento de contadores síncronos

A saída CEO serve para cascatear contadores. Basta ligar a saída CEO do contador menos significativo à entrada CE do contador seguinte, como mostra a Figura 3.8. O sinal de *clock* deve ser o mesmo para todos os contadores para que funcionem sincronamente.

Fazendo a entrada CE igual a zero, o primeiro contador é desabilitado e a saída CEO_A também vai a zero, garantindo que o segundo contador também seja desabilitado. Quando CE estiver ativo e o primeiro contador (H3) atingir o final de contagem ($Q_1Q_0 = 11$), teremos CEO_A = 1 e o segundo contador (H4) será habilitado. Na borda seguinte do *clock*, as saídas Q_3Q_2 serão incrementadas juntamente com Q_1Q_0.

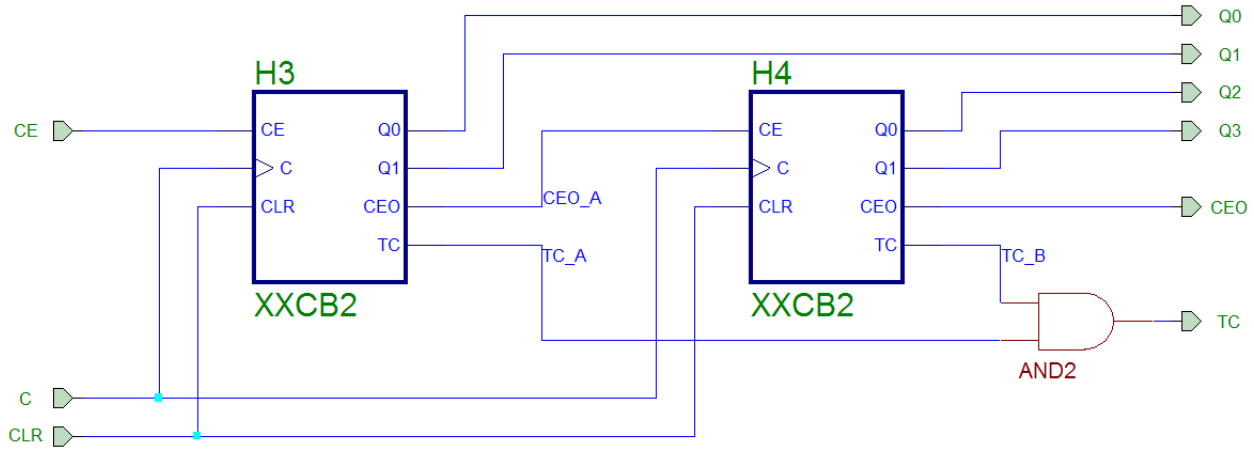


Figura 3.8 Contador de quatro bits usando dois contadores em cascata.

A carta de tempos da Figura 3.9 ilustra o mecanismo de cascateamento (usamos os mesmos sinais de entrada da Figura 3.7). Na borda 4 do *clock* C, o primeiro contador é desabilitado, desabilitando também o segundo (CEO_A = 0), e a contagem de ambos se mantém a mesma na borda 5. No instante 6 ambos estão habilitados e a contagem nos bits Q[3:0] passa de 0011 para 0100..

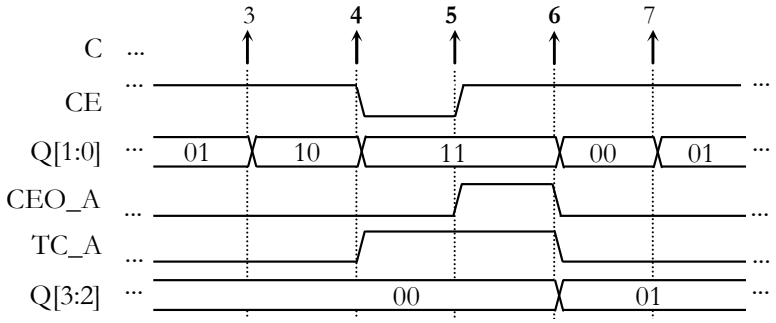


Figura 3.9 Carta de tempos resumida do contador de quatro bits em cascata

3.6 Chaves, botões e o problema de *bouncing*

Chaves e botões são dispositivos muito utilizados como entradas em circuitos eletrônicos para permitir a interação do usuário com o circuito. No entanto, eles apresentam um problema de natureza eletromecânica que pode interferir no funcionamento do circuito.

Observe a Figura 3.10. Seria de se esperar que a saída Q invertesse cada vez que o botão BTN fosse pressionado. No entanto, devido a vibrações mecânicas e contatos elétricos imperfeitos, o sinal C apresenta rápidas "oscilações" que podem durar alguns milissegundos, conhecidas como *bounce*. Isso pode resultar em bordas espúrias tanto no momento em que o botão é apertado como quando é liberado.

O *bounce* acontece com todo tipo de dispositivo com contato eletromecânico, em maior ou menor grau. Dependendo da aplicação, podem ser utilizados circuitos eletrônicos para minimizar o problema – os chamados circuitos de *debounce*.

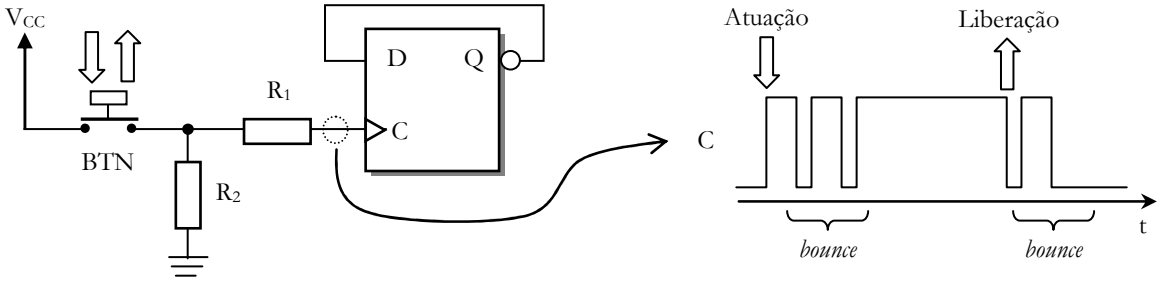


Figura 3.10 Circuito sujeito ao problema de bounce

O mais simples é o filtro RC: acrescenta-se um capacitor entre a entrada C e 0 V, como mostra a Figura 3.11. Tendo-se R_1 bem menor que R_2, o capacitor se carrega rapidamente (e exponencialmente) através de R_1 quando o botão é apertado com constante de tempo $\tau = R_1C$, mas se descarrega lentamente através de R_1 e R_2 quando o botão é liberado com $\tau = (R_1 + R_2)C$.

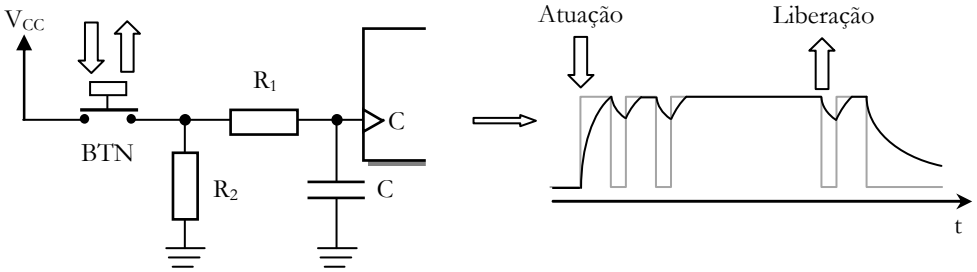


Figura 3.11 Debounce com filtro RC

O *debounce* também poderia ser feito por um circuito conhecido como *timer* ou mono-estável, que geraria um pulso de duração conhecida após ser disparado pelo botão. Veremos como construir *timers* digitais usando contadores síncronos numa próxima experiência.

Uma última observação importante: entradas sensíveis a borda não devem ser alimentadas com sinais provenientes de saídas de circuitos combinacionais. Em circuitos desse tipo, atrasos relativos entre sinais também podem gerar pulsos espúrios – efeito conhecido como *hazard* estático, visto na experiência 2. Como regra geral, sinais de clock devem ser gerados por meio de circuitos sequenciais, como latches e flip-flops.

A Figura 3.12 mostra o circuito que usaremos no laboratório (IPADs e OPADs foram omitidos), e que se encontra no arquivo TestCB4.SCH.

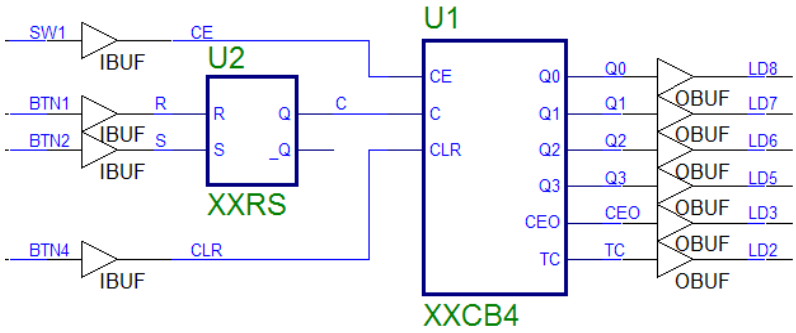


Figura 3.12 Circuito de teste de contador (TestCB4.SCH) com *clock* gerado por latch.

O símbolo XXCB4 (componente U1) representa o contador. Repare que o *clock* C não está ligado diretamente a um botão, e sim ao *latch* RS positivo XXRS (U2) para garantir que não haja *bounce* na entrada de *clock*. O botão BTN1 zera o sinal de *clock* C, enquanto que o botão BTN2 leva C a 1.

A entrada *count enable* CE pode ser controlada pela chave SW1. O botão BTN4 está ligada à entrada de *clear* assíncrono CLR e zera o contador quando pressionada. As saídas do contador podem ser observados nos leds indicados na figura.

3.7 Aplicação: Acionamento do Mostrador de Sete Segmentos

A placa Digilab tem quatro mostradores de sete segmentos. Cada segmento é formado por um *led*, um dispositivo semicondutor que estudaremos futuramente, representado na Figura 3.13 pelos triângulos com traços. Por hora, acredite que um *led* se acende quando uma corrente elétrica o atravessa no sentido apontado pelo triângulo. O terminal de entrada da corrente (base do triângulo) se chama *anodo*, e o de saída (traço) se chama *catodo*.

Há quatro terminais de alimentação (chamados de anodo comum), sendo um para cada mostrador, que são acionados através sinais A1 a A4 ligados aos pinos P44, P40, P39 e P38 respectivamente (confira na página 12 do manual da placa XLA). Os sinais CA, CB, ... CF e DP comandam cada um dos sete segmentos e o ponto decimal. Por exemplo, o segmento *a* (segmento superior) se acende sempre que fazemos CA = 0, aterrando o catodo do *led a* e permitindo que uma corrente flua através dele.

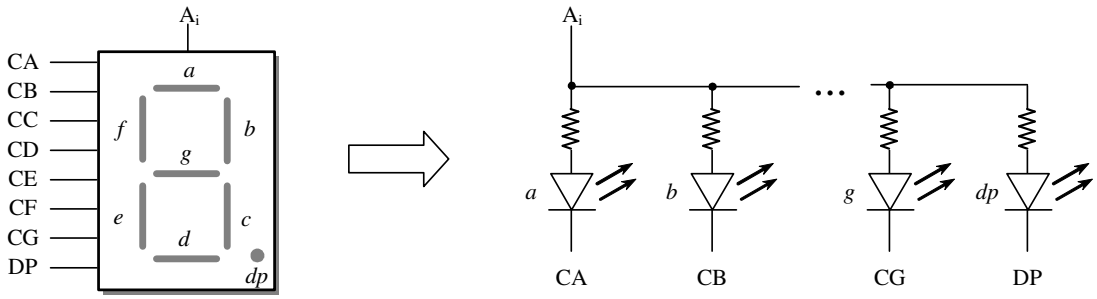


Figura 3.13 Mostrador de sete segmentos (esquerda) e esquema de ligação dos seus leds

A Figura 3.14 mostra como representar (de forma um tanto precária) dígitos hexadecimais de 0 a F. Para o dígito '0', por exemplo, devemos deixar CA a CF em zero para acender os *leds* de *a* a *f*, e fazer CG e DP iguais a um para apagar os *leds g* e *dp*; para mostrar o dígito '2' fazemos apenas CB e CC iguais a zero, e assim por diante.

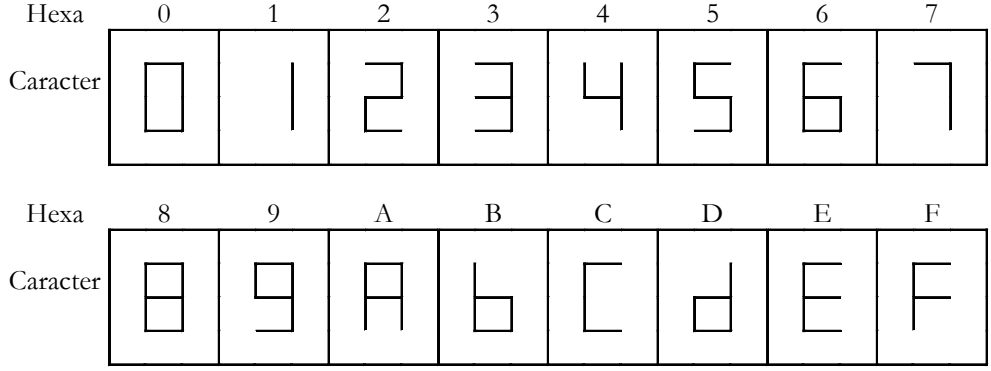


Figura 3.14 Padrão de apresentação de dígitos hexadecimais em 7 segmentos (sem o ponto decimal)

Na FPGA, os catodos CA a CF estão ligados aos pinos P51 a P45. O catodo DP (do ponto decimal) não está ligado à FPGA, mas está disponível no conector J1 da placa (confira na página 11 do manual).

Note, no entanto, que os catodos (CA a CG) são comuns aos quatro mostradores. Ou seja, ativando os quatro anodos (A1 a A4) ao mesmo tempo, os mostradores mostrarão o mesmo dígito imposto nos catodos CA a CG. Como faríamos, então, para mostrar dígitos diferentes nos mostradores?

A solução é acionar um anodo de cada vez, ciclicamente, e utilizar multiplexadores para colocar nos catodos o padrão de cada dígito, de forma sincronizada com a ativação dos anodos. Ou seja, cada mostrador piscaria um dígito diferente, um de cada vez. Se a taxa de varredura dos mostradores for alta (acima de trinta vezes por segundo, aproximadamente), ficaremos com a impressão que os mostradores estão sempre ligados, já que não conseguiremos mais observar o efeito pisca-pisca.

A Figura 3.15 mostra um circuito para mostrar quatro dígitos hexadecimais (DA a DD) nos mostradores da placa Digilent. Os dígitos entram em um multiplexador quádruplo de 4 bits (bloco H2), que a cada instante passa adiante apenas um deles. O bloco H3 faz a conversão do número em binário para o padrão correspondente para apresentação no mostrador de sete seguimentos (o ponto decimal foi omitido no esquema). O bloco H1 representa um contador cíclico de 2 bits, que gera os bits de seleção do multiplexador e também do decodificador (bloco H3), garantindo a sincronização entre o dígito selecionado no mux e o mostrador a ser ligado.

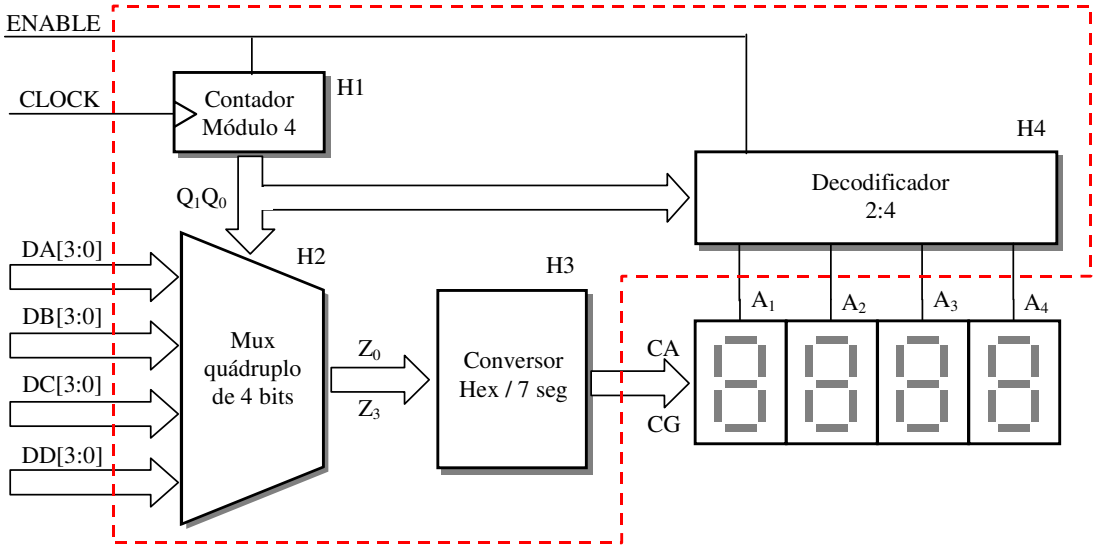


Figura 3.15 Circuito para acionamento dos mostradores de sete segmentos.

O sinal de CLOCK deve ter uma frequência superior a 4 x 30 = 120 Hz. Essa é a taxa mínima necessária para que cada mostrador seja ligado pelo menos 30 vezes por segundo. O sinal ENABLE permite ligar e desligar todos os mostradores conjuntamente, desabilitando o decodificador e o contador.

Para construir esse circuito no laboratório, você poderá usar os componentes mostrados na Figura 3.16.

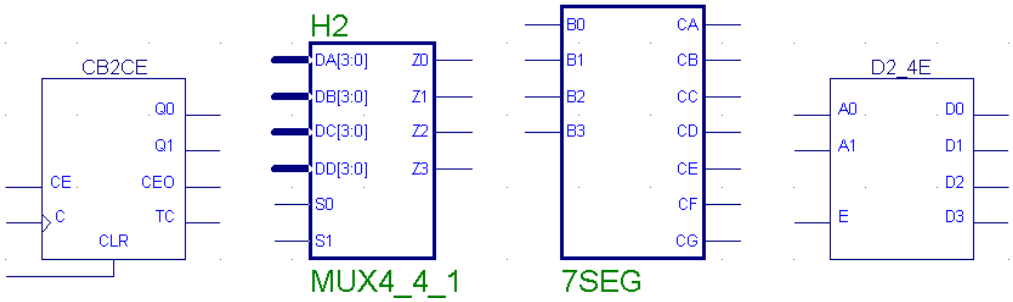


Figura 3.16 Componentes para o projeto do acionador de mostrador de sete segmentos

O CB2CE é um contador de dois bits, com entradas de Clock Enable (CE = 1 habilita a contagem) e Clear assíncrono (CLR = 1 zera o contador, independentemente do sinal de Clock). A saída TC (Terminal Count) vai a 1 sempre que a contagem em Q_1Q_0 atinge o valor máximo – neste caso, 11 em binário. A saída CEO é simplesmente um AND entre TC e a entrada CE.

As macros MUX4_4_1 e 7SEG implementam o multiplexador quádruplo e o conversor de código para sete segmentos. Os diagramas lógicos destas duas macros estão em anexo.

Finalmente, o bloco D2_4E é um decodificador 2:4 com entrada de *Enable* (E). Quando o decodificador se encontra habilitado (E = 1), a saída D_i endereçada pelas entradas A_1A_0 vai a um, enquanto as demais permanecem em zero; quando E = 0, todas as saídas permanecem em zero.

3.8 PRÉ-RELATÓRIO

ATENÇÃO: leia as atividades por completo para fazer os exercícios corretamente!
Total de exercícios: 9 (NOVE)

O *pré-relatório* é composto pelos EXERCÍCIOS contidos nesta apostila. Você deverá entregar o pré-relatório até o começo da aula de sua turma, caso contrário não poderá fazer a experiência. Os exercícios podem ser resolvidos a lápis, mas as respostas finais devem ser escritas **à caneta**. Faça-os com **antecedência**, para ter tempo para tirar dúvidas e fazer **todos** os exercícios.

A nota do pré-relatório depende também de uma **ARGUIÇÃO ORAL** que poderá ser feita em a aula. Portanto, **revise** o pré-relatório e a apostila antes da aula. A argüição oral também poderá ser feita para esclarecer partes confusas ou mal feitas no pré-relatório, por isso é importante que você o faça com **esmero**.

Leia com atenção todas as atividades da Parte II antes da aula, e não apenas os exercícios. Isso é necessário para fazer os exercícios corretamente pois muitos detalhes estão descritos nas atividades em que se inserem. Além disso, você terá uma noção das atividades a serem desenvolvidas e perderá menos tempo durante a aula.

Traga para a aula a apostila **IMPRESSA**. Os pontos importantes das atividades devem estar **destacados ou grifados**. As anotações serão avaliadas e contarão na nota final da experiência.

Ao final da apostila, encontram-se algumas dicas e respostas para alguns dos exercícios.

PARTE II PRÁTICA

LEIA tudo com atenção

NÃO tente apenas REPRODUZIR as figuras sem ler a apostila.

ATENÇÃO: há um problema de configuração no Windows Vista que IMPEDE que o primeiro projeto implementado no computador seja reimplementado uma segunda vez! Para contornar, sempre que o computador for religado ou seja feito um *login*, primeiramente carregue o projeto XLA5, clique em
Project | Clear Implementation Data e implemente, conforme descrito a seguir.

Atividade 1 Teste da placa

Anotação 1a Anote no cabeçalho do relatório: nome dos integrantes da equipe, número da bancada, número da placa XLA5 e a hora de início das atividades.

Veja o diagrama lógico XLA5 em anexo. No canto esquerdo, há três blocos, CB8 e CB16 e CD4CE, conectados em sequência. Isto forma um contador, cuja função é dividir a frequência do *clock* de entrada presente em P13. Por hora, acredite apenas que CD4CE conta ciclicamente de 0 a 9, em binário, e o bloco H4 faz a conversão para acionar os displays de 7 segmentos, que estão ligados aos pinos P45 a P51.

Atente para o restante do circuito, concentrado na parte inferior direita. Ele implementa uma lógica que acende os *leds* através das chaves e dos botões da placa. Vamos testar a placa XLA5 com esse circuito. Os arquivos necessários para este teste se encontram compactados no arquivo XLA5.ZIP, disponível no servidor, no endereço **\\ts02-00\pmr2433.**

Copie o arquivo XLA5.ZIP do servidor para o diretório C:\PMR2433_ARQ (e SOBREESCREVA se ele já existe). Carregue o software de programação: menu Start – Programs – Xilinx Foundation Series – Project Manager. Na janela de abertura (Getting Started), clique em Cancel, para iniciar o programa sem carregar nenhum projeto. Na barra de menus do Project Manager, clique em File, e em seguida em Restore Project... Entre no diretório C:\PMR2433_ARQ, e abra o arquivo XLA5.ZIP.

Na janela "Restore Project Wizard...", localize o campo "Destination" e digite C:\PMR2433*seugrupo*, onde *seugrupo* é o nome de guerra do seu grupo. **Atenção**: ao dar nomes a elementos, arquivos e diretórios dentro do Foundation, não use nomes com mais de 8 caracteres, nem acentos ou espaços em branco!

Deixe o box "Open restored project" selecionado e clique em Avançar. Clique em "Yes to All" para ignorar todos os arquivos já existentes. O projeto será descompactado e carregado pelo programa.

Abra o editor esquemático, dando um duplo clique no arquivo XLA51.SCH que aparece na sub-janela "Files" (à esquerda). Você verá o diagrama lógico do circuito de teste (anexo XLA5). Vamos agora programar a

FPGA com este projeto. Retorne à janela principal do programa "Project Manager", e clique no menu **Project – Clear Implementation Data**. Isso força o programa a passar por todos os passos da implementação (e vai convenientemente ativar o bug do Windows em cima do desse projeto de teste, que não poderá mais ser alterado). Caso contrário, se alguém já tiver implementado o projeto XLA5 antes de você, o programa detectará que não é necessário fazer nada e o bug do Windows aparecerá quando você for trabalhar no seu projeto!

Ainda na janela do "Project Manager", clique agora no botão Implementation que aparece na janela "Flow" à direita, e em seguida no botão "**RUN**" na janela que se abre (não clique em "OK", pois isto apenas fecha a janela).

O programa vai implementar o projeto (ou seja, "compilar" a descrição do circuito) para gerar o arquivo de configuração (ou programação) da FPGA. Acompanhe os passos dessa transcrição:

- Translate e Map: converte seu projeto em funções lógicas que podem ser implementadas nos PLBs
- Place and Route: decide em quais PLBs da FPGA as funções lógicas serão programadas e faz a programação das interconexões entre elas (SMs).
- Configure: gera o arquivo binário (extensão "bit") que será enviado à FPGA para programá-la.

VERIFIQUE: se o programa não passou pelas fases citadas acima, você clicou no botão errado!

Conecte o cabo da porta paralela do PC na placa e ligue a fonte de alimentação. ATENÇÃO: NÃO deixe coisas metálicas sob a placa! Isso pode causar um curto-circuito entre os pontos de solda.

Agora, verifique se a chave SW9 está na posição PROG e transfira o arquivo "bit" gerado para a FPGA da placa. Isso é feito com o programa **C:\Xilinx4\XLA5\ConfigXP.pif** (há um link na área de trabalho).

Esse aplicativo deve ser executado com as prerrogativas de administrador para poder acessar a porta paralela do computador. Para isso, clique sobre ele com o botão da direita do mouse e clique em seguida em "Executar como Administrador". Na janela "Configure Digilab" que o programa abre, clique no menu Control – Configure Device from File, e selecione o arquivo C:\PMR2433*seugrupo*\XLA5\XLA5.bit.

Exercício 1 (Lembre-se: os resultados finais dos exercícios devem ser escritos **à caneta!**) No circuito XLA5 descrito nesta atividade, determine a equação lógica que faz o *i*-ésimo *led* LD_i acender, em função das chaves (SW_i) e dos botões (BTN_i).

Verifique se o circuito está funcionando corretamente. Em funcionamento normal, os displays mostrarão os dígitos de 0 a 9, ciclicamente. Acione os leds através das chaves e dos botões e verifique se isso condiz com o previsto no exercício. Verifique se todos os botões, chaves e leds estão funcionando.

ATENÇÃO: Ao contrário do painel LSD utilizado nas experiências anteriores, uma chave da placa Digilab envia à FPGA o nível lógico 0 quando está para frente, e 1 quando está para trás. O botão envia 1 quando é pressionado.

Anotação 1b Verifique se a resposta no seu pré-relatório condiz com o funcionamento dos leds observado nesta atividade. Caso contrário, comente os erros detectados.

Atividade 2 Restaurando os arquivos de projeto

O disco de seu computador contém alguns arquivos que serão utilizados nessa experiência, tais como macros prontas e esquemáticos incompletos. Copie o arquivo 2433_E3.ZIP do servidor (**\\ts02-00\pmr2433)** para o diretório C:\PMR2433_ARQ (mesmo que ele já exista).

Volte à janela do programa Project Manager. Clique no menu File – Restore Project... Selecione o arquivo 2433_E3.ZIP no diretório C:\PMR2433_ARQ. Na janela "Restore Project Wizard..." , campo "Destination", digite C:\PMR2433*seugrupo*, onde *seugrupo* é o nome de guerra do seu grupo.

Deixe o box "Open restored project" selecionado e clique em Avançar. Clique em "Yes to All" para ignorar todos os arquivos já existentes. O projeto será descompactado e carregado pelo programa. Não vamos trabalhar com esse projeto. Ao invés disso, vamos copiá-lo com outro nome para o seu diretório de trabalho. Clique no menu File | Copy Project. O campo Source Project, já deve estar preenchido com C:\PMR2433*seugrupo*\2433_E3.PDF. No campo Destination Directory, digite c:\PMR2433*seugrupo*. No campo Destination Name, digite algo como *xx*FF, onde *xx* são as 2 primeiras letras do nome *seugrupo*. Clique em **OK**.

Podemos agora carregar o projeto copiado no Project Manager. Clique no menu File | Open Project, e localize o arquivo *xx*FF.PDF no diretório C:\PMR2433*seugrupo*. Selecione o arquivo e clique em Open. Estamos prontos para começar a experiência.

Anotação 2a Acrescente o nome do diretório de trabalho no cabeçalho do relatório.

Atividade 3 Latch RS negativo

Vamos construir uma macro para implementar um latch RS negativo. Uma macro representa um circuito por um símbolo, tal como um componente encapsulado, que pode ser incluido várias vezes em outros projetos.

Dispare o editor de esquemáticos: no menu Tools, selecione Design Entry | Schematic Editor (ou clique no ícone correspondente na sub-janela Flow). O editor cria o arquivo de nível hierárquico mais alto do projeto, *xx*FF1.sch. O latch será uma macro, de nível hierárquico inferior (a famosa caixa-preta).

Para criar a macro, no menu Tools, selecione Symbol Wizard. Esse módulo define as interfaces de entrada e saída da macro, e cria um símbolo (a cara da caixa-preta). Passe pela janela de apresentação do módulo e avance para a janela Design Wizard – Contents. No campo Symbol Name, digite *xx*NRS (*xx* são as iniciais do seu grupo) e selecione Schematic no campo Contents. Clique em Next.

Na janela Design Wizard – Ports, defina os sinais _R e _S de entrada e os sinais Q e _Q de saída (o traço indica que os sinais são ativos em 0). Clique em New, e digite _R no campo Name, e selecione Input no campo Direction. Repita esse processo para criar a entrada _S.

Em seguida, crie a saída Q (clique em New, digite Q no campo Name e selecione Output no campo Direction) e _Q. Neste ponto, a definição de sua macro deve ter o aspecto mostrado na Figura 3.17. Avance para a janela seguinte com Next.

Na janela seguinte, Attributes, digite no campo Longer Comment algo como "Latch RS negativo" (deixe o campo Shorter em branco). Clique em Next, e na janela seguinte em Finish. O programa constrói o esqueleto da macro, já com os pontos de entrada e saída rotulados.

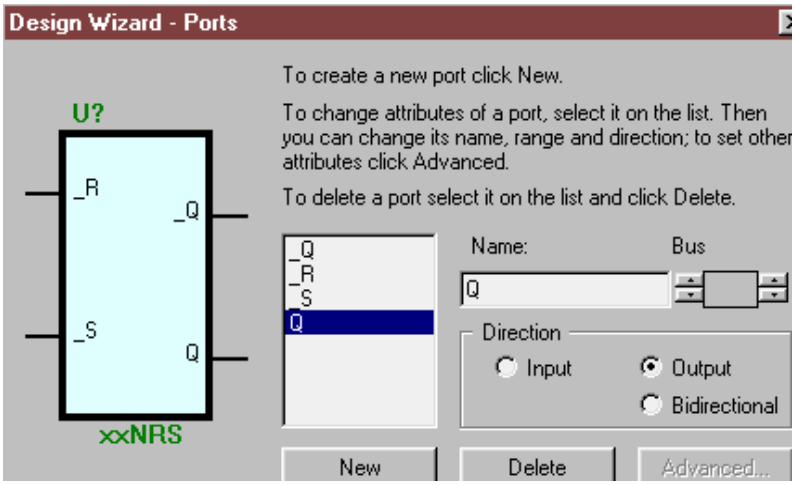


Figura 3.17 Definição da macro *xx*NRS

Complete o esquema, conforme mostra a Figura 3.18, e salve o arquivo (menu File | Save). Uma dica: insira primeiro todos os símbolos (no caso, dois NANDs), acionando o modo de inserção através do menu Mode | Symbols, ou clicando no ícone de um AND que aparece na barra lateral esquerda. Você verá uma extensa lista de nomes mneumônicos dos símbolos. Digitando no campo inferior dessa janela, fica mais fácil para encontrar o símbolo desejado. Selecione e inclua-o no projeto, clicando na posição desejada. Para incluir outras cópias, clique em um símbolo já posicionado e posicione a nova cópia.

Ao incluir um símbolo, estamos *instanciando* a macro correspondente a ele, ou seja, estamos fisicamente inserindo no nosso projeto os componentes do circuito representado pela macro. Esses componentes serão repetidos tantas vezes quanto a macro for instanciada no projeto.

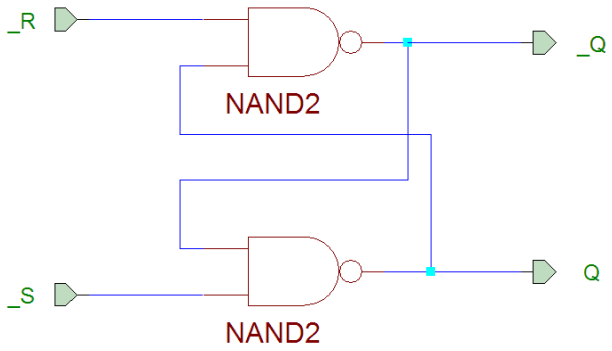


Figura 3.18 Macro *xx*NRS

OBSERVE: a saída Q deve ser a da porta NAND ligada à entrada _S!

Complete as conexões ativando o menu Mode | Draw Wires. Clique num terminal de um símbolo e depois em outro para ligá-los. Clicando em posições intermediárias, é possível definir uma rota. Para inserir uma derivação em um fio, basta clicar sobre ele no ponto desejado.

ATENÇÃO: para conectar o terminal de um símbolo ao terminal de outro, não basta sobrepô-los ao posicionar os símbolos. Uma conexão somente se dá através de um fio. Por isso, posicione os símbolos mantendo uma certa distância entre os terminais a serem conectados.

Verifique se está tudo em ordem. Clique no menu Options | Integrity Test for Current Sheet. Se houver algo errado, aparecerão mensagens na tela (Console) do Project Manager. Nesse caso, corrija, teste de novo, e salve. IMPORTANTE: se precisar alterar a macro mais tarde, **NUNCA** a abra com o comando File | Open... – use o comando **File | Open macro**, ou clique sobre ela com o botão da direita e clique em Hierarchy Push.

Volte à folha *xx*FF1. Insira nela a macro *xx*NRS: clique no menu Mode | Symbols, ou clique no ícone correspondente na barra esquerda. O símbolo *xx*NRS está no fim da lista, debaixo da hierarquia do nosso projeto (*xx*FF).

A Figura 3.19 mostra o diagrama parcial do circuito que usaremos para testar o latch. Atente para a ordem dos pinos de entrada e saída nas faces laterais do símbolo U1. Se você não incluiu os sinais na ordem descrita anteriormente, provavelmente eles estarão diferentes.

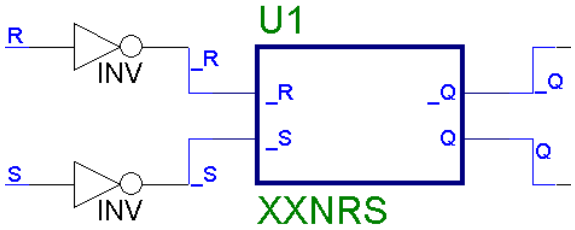


Figura 3.19 Circuito de teste do *latch* xxNRS *(incompleto)*

Complete as ligações do circuito de teste. As entradas R e S devem ser ligadas a botões e as saídas Q e _Q a leds, conforme a tabela abaixo:

ENTRADAS		SAÍDAS	
R	BTN1	Q	LD1
S	BTN2	_Q	LD2

Exercício 2 Faça o diagrama lógico do latch RS negativo xxNRS e sua tabela característica (ou da verdade). Em seguida, complete a carta de tempos da Figura 3.20: desenhe os sinais _R, _S, Q e _Q.

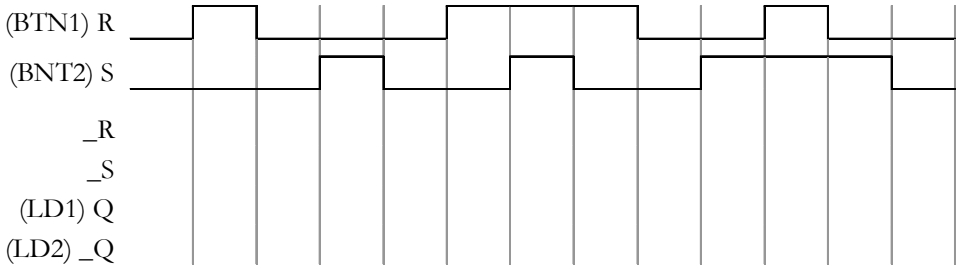


Figura 3.20 Sinais para teste do latch xxNRS

Exercício 3 Faça o diagrama lógico completo do circuito de teste do latch descrito nesta atividade. Inclua os IPADs e IBUFs para ligar os botões às entradas R e S. Inclua os OBUFs e OPADs para ligar aos sinais Q e _Q aos leds. Anote nos IPADs e OPADs o nome dos pinos da FPGA aos quais estão ligados.

Anotação 3a Caso seu projeto tenha sido aprovado pelo professor, NÃO é necessário refazer o DL. CASO CONTRÁRIO, refaça-o corretamente (**a mão livre!**) no relatório antes de começar a edição do circuito.

Lembre-se que é necessário mapear cada IPAD e OPAD em um pino da FPGA. Clique duas vezes em um deles para abrir a janela Symbol Properties. Na parte inferior da janela (Parameters), selecione LOC no campo Name, e digite a identificação do pino no campo Description (por exemplo, P59). Clique em Add, caso contrário isso não terá efeito. Veja a Figura 3.21. Clicando sobre a marca preta à direita de LOC, você pode ajustar a forma como esse parâmetro será exibido no desenho: uma marca exibirá apenas o valor do parâmetro (P59), duas exibirá "LOC=P59", e nenhuma marca fará com que o programa não exiba o parâmetro.

Os identificadores dos fios, tais como "BTN1" e "LD1", podem ser inseridas clicando-se duas vezes sobre os fios. Isto abre a janela "Net Name". Digite o nome e clique em OK.

VEFIRIQUE: se na barra de título da janela do Schematic Editor contar algo como "Schematic Editor –

[Non-Project – xxNRS.SCH]", você NÃO está editando a macro! Volte ao começo desta atividade e RELEIA-A até este ponto.

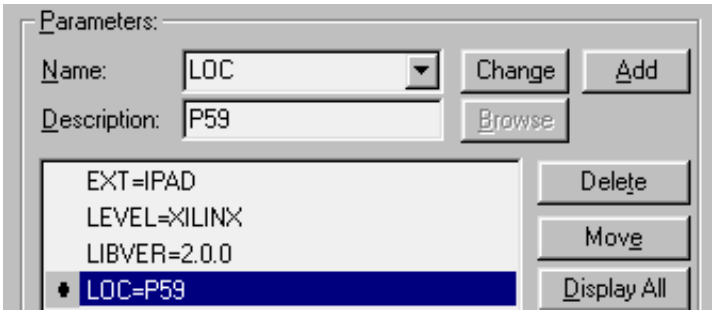


Figura 3.21 Configuração de um IPAD

Por fim, salve o arquivo. Clique no botão Implementation do programa Project Manager para gerar a implementação do circuito. Descarregue o arquivo bit (c:\pmr2433*seugrupo**xx*FF*xx*FF1.bit) na placa, com o programa ConfigXP, conforme descrito na Atividade 1.

Teste o funcionamento do latch acionando os botões. Lembre-se que os botões em repouso fornecem nível lógico 0 e que estão ligados às entradas _R e _S do latch por meio de inversores. O botão ligado a S seta o latch, acendendo o led de Q e apagando o led de _Q. Já o botão de R causa o reset do latch, apagando Q e acendendo _Q. Se isto não estiver acontecendo, revise o seu circuito!

Anotação 3b Caso sua carta de tempos do pré-relatório tenha sido aprovado pelo professor, NÃO é necessário refazê-la. CASO CONTRÁRIO, refaça-a (a mão livre!) corretamente no relatório.

Anotação 3c Teste o *latch* usando os botões para reproduzir as formas de onda dos sinais R e S mostrados na Figura 3.20. Compare com o previsto no pré-relatório

Anotação 3d Descreva o que acontece com _R, _S, Q e _Q quando os dois botões são pressionados ao mesmo tempo. Em seguida, tente soltar os botões ao mesmo tempo. O estado final é sempre o mesmo em várias tentativas? Discuta a razão do comportamento observado.

Atividade 4 Contador síncrono de quatro bits

Vamos implementar em uma macro o contador de quatro bits projetado no pré-relatório.

Inicialmente, encerre o simulador e o Schematic Editor. Na janela do Project Manager, remova o arquivo xxFF1.SCH da hierarquia do projeto. Para isso, clique sobre ele para selecioná-lo e clique no menu Document | Remove, confirmando em seguida (o arquivo não é apagado e permanece no disco; apenas é retirado da lista de arquivos que compõe o projeto). Em seguida, clique no menu Document | Add... e selecione o arquivo TestCB4.SCH (que deve estar no seu diretório de projeto c:\pmr2433*seugrupo**xx*FF).

Usaremos o circuito contido nesse arquivo para testar o contador. A Figura 3.12 mostra esse circuito parcialmente, onde o símbolo xxCB4 representa o contador. Abra o arquivo no Schematic Editor.

Abra em seguida a macro XXCB4: para isso, clique sobre ela com o botão da direita e clique em Hierarchy Push, ou clique no menu File | Open macro... e selecione essa macro (lembre-se: **NÃO** use o comando File | Open...).

VERIFIQUE: se no tab Files da janela do Project Manager constar outros arquivos com extensão ".sch" na hierarquia do projeto (que deveria se chamar algo como *xx*FF), além do arquivo "TestCB4.sch", você DEIXOU DE REMOVER o arquivo anterior da hierarquia. Volte e RELEIA esta atividade até este ponto.

Exercício 4 Projete o contador síncrono de quatro bits xxCB4, cujo símbolo é mostrado na Figura 3.12. Use quatro flip-flops tipo JK e portas lógicas adicionais (NÃO USE dois contadores de dois bits em cascata!). Faça o diagrama lógico do circuito.

Anotação 4a Caso seu projeto tenha sido aprovado pelo professor, NÃO é necessário refazer o DL. CASO CONTRÁRIO, refaça-o (**a mão livre**!) no relatório antes de começar a edição do circuito.

Complete e modifique esse circuito seguindo o seu projeto, e salve. Feche o Schematic Editor, implemente o contador e configure a FPGA (usando o programa ConfigXP).

Teste o contador usando o circuito contido no arquivo TestCB4.SCH. As entradas e saídas estão conectadas na placa XLA conforme a tabela a seguir. Mantenha a chave SW1 em 1, pressione alternadamente os botões BTN1 e 2 para gerar o sinal de *clock* e acompanhe as saídas nos leds. Coloque a chave SW1 em 0 algumas vezes antes de fazer o *clock* subir, e verifique se o contador fica desabilitado corretamente.

ENTRADAS		SAÍDAS	
CE	SW1	C (clock)	LD1
Reset C	BTN1	TC	LD2
Set C	BTN2	CEO	LD3
CLR	BTN4	Q[3:0]	LD5 a LD8

Em particular, teste o contador com os sinais CLR e CE mostrados na Figura 3.22: desabilite o contador quando a contagem atingir 3 (Q[3:0] = 0011) e o valor máximo 15 (Q[3:0] = 1111). Note: quando Q[3:0] chegar a 1111, as saídas TC e CEO (leds LD2 e 3) devem ir para 1; fazendo CE = 0 (chave SW1), apenas CEO deve zerar.

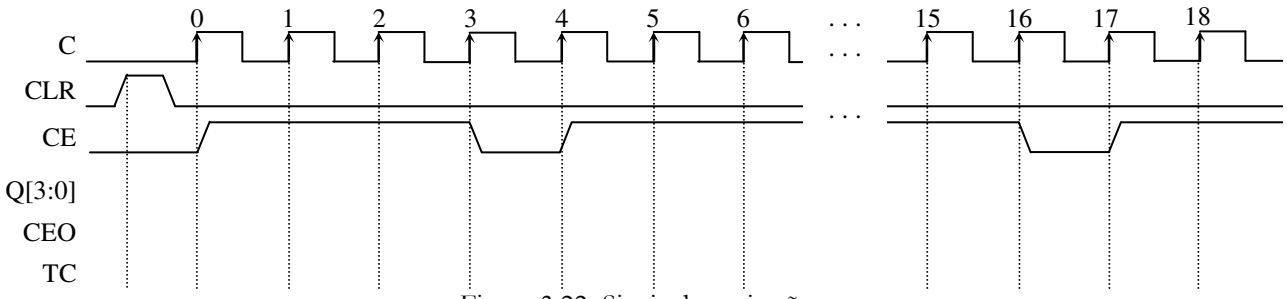


Figura 3.22 Sinais de excitação.

Exercício 5 Complete a carta de tempos da Figura 3.22: desenhe (a mão livre!) os sinais Q[3:0], CEO e TC. NÃO é necessário desenhar todos os clocks entre 0 e 18 – indique a continuidade por reticências. Desenhe as formas de onda das saídas Q[3:0] de forma **condensada** e com capricho! Elas devem estar sincronizadas com o sinal de clock, caso contrário serão de pouca utilidade. LEMBRE-SE: considere o estado de um sinal de entrada no instante imediatamente anterior a uma borda – por exemplo, na borda 0 de clock, o sinal CE ainda está em 0. Antes de ter-se CLR = 1, as saídas Q[3:0] e CEO são desconhecidas, mas com certeza tem-se TC = 0, uma vez que CE = 0 nesse intervalo.

Exercício 6 Determine o valor de contagem nas saídas Q[3:0] **após** as bordas de clock 3, 4 e 5, e **após** as bordas 16, 17 e 18.

Anotação 4b Descreva o que acontece com as saídas CEO e TC no teste descrito acima nas seguintes situações: a) contagem abaixo de 1111; b) contagem em 1111 e CE = 1; c) contagem em 1111 e CE = 0. Compare com o previsto na carta de tempos feita no pré-relatório

Anotação 4c Anote a hora atual e mostre o contador funcionando para o professor. Caso encontre algum erro no projeto do pré-relatório, aponte-o e descreva o efeito causado.

Atividade 5 Controlador do Display de 7 Segmentos

Vamos criar uma macro para implementar o acionador de mostrador de sete segmentos.

Comece ativando a ferramenta Symbol Wizard (menu Tools do Schematic Editor), e dê ao símbolo um nome como *xx*DsDrv (não esqueça de definir Contents como Schematic). A Figura 3.23 mostra o símbolo resultante ao final da definição dos pinos.

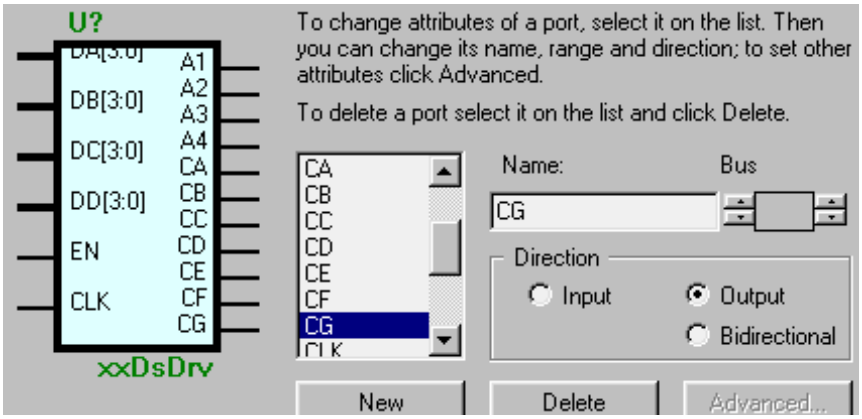


Figura 3.23 Definição do símbolo xxDsDrv

As entradas DA, DB, DC e DD são barramentos (*buses*), de quatro bits cada. Para defini-las, clique em New na janela Ports, digite o nome do bus no campo Name (por exemplo, DA), e ajuste a dimensão do bus para [3:0] no campos Bus (ao lado do campo Name). Defina as demais entradas e saídas da macro: habilitação EN e *clock* CLK. As saídas são os sinais de ativação dos anodos (A1 a A4) e dos catodos (CA a CG) dos mostradores.

As linhas grossas indicam barramentos (buses). Os barramentos de entrada podem ser ligados diretamente aos barramentos da macro MUX4_4_1, como mostra a Figura 3.24. Para desenhar um barramento no Schematic Editor, clique em Mode | Draw Bus (ou clicando no ícone correspondente).

As macros MUX4_4_1 e 7SEG já estão disponíveis no disco do seu computador, como parte da biblioteca *xx*FF, e podem ser instanciadas (isto é, inseridas) dentro do seu circuito. Complete o esquema do circuito e salve. Dica: os barramentos de entrada podem ser ligados diretamente aos barramentos da macro MUX4_4_1, como mostra a Figura 3.24.

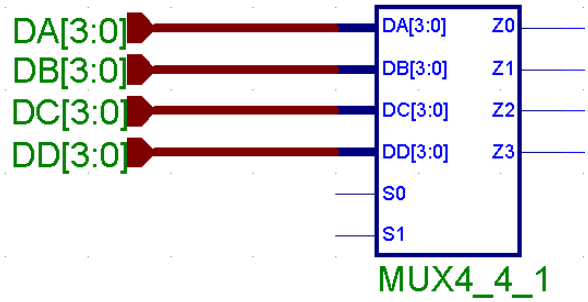


Figura 3.24 Ligação entre barramentos

Nota: para impor entradas em 0 ou 1, existem os "componentes" **GND** e **VCC** na biblioteca.

Vamos testar o acionador utilizando o circuito de um cronômetro, mostrado na Figura 3.25. O sinal de clock é gerado pelo símbolo OSC4, cuja saída F490 fornece um sinal de 400 Hz. Esse sinal será utilizado para fazer a varredura do mostrador, e portanto cada mostrador piscará a uma taxa de 100 vezes por segundo.

O contador CB2CE divide a frequência do sinal por 4, resultando em uma base de tempo de um centésimo de segundo. À frente deste, temos quatro contadores CD4CE de quatro bits que contam ciclicamente de 0 a 9 (são os chamados contadores de década ou contadores BCD). Portanto, cada um deles divide a frequência por 10. Como o primeiro contador é acionado a cada 0,01 s, temos no barramento DD a contagem decimal em centésimos de segundo. Analogamente, em DC temos a contagem em décimos de segundo, e assim por diante.

Note como o cascateamento dos contadores é feito. O sinal de clock de 400 Hz é conectado em todas as entradas C, mas apenas o contador CB2CE (de módulo 4) está habilitado continuamente. A saída CEO deste contador habilita o primeiro contador CD4CE. Como vimos na Parte I, essa saída CEO vai a um somente nos períodos em que a contagem passa pelo máximo (3, neste caso) e portanto o primeiro contador CD4CE é incrementado apenas a cada 4 ciclos de clock. Pelo mesmo princípio, o segundo contador CD4CE é incrementado a cada 40 ciclos, o terceiro a cada 400 ciclos, e o último a cada 4000 ciclos.

Exercício 7 Reveja o circuito com dois contadores em cascata da Figura 3.8 (Parte I). O que aconteceria se o cascateamento fosse feito de forma errada, ligando-se a saída **TC** (e não CEO) do primeiro contador (H3) ao CE do segundo (H4)? Refaça a carta de tempos da Figura 3.9, substituindo o sinal CEO_A por TC_A e determine o valor de contagem em Q[3:0] **após** as bordas de *clock* 4, 5 e 6.

O botão BTN4 está ligado às entradas CLR (Clear) dos contadores. Dessa forma, o cronômetro é zerado sempre que pressionamos esse botão.

No Project Manager, remova do projeto o arquivo utilizado na atividade anterior: clique sobre o nome do arquivo na árvore do projeto e clique no menu Document | Remove, e confirme.

Você não precisará construir todo o circuito da Figura 3.25. No diretório do projeto já existe um arquivo parcialmente feito. No Schematic Editor, use o menu File | Open para abrir o arquivo CronSemi.SCH no diretório C:\PMR2433*seugrupo**xx*FF. Em seguida, salve-o com outro nome, por exemplo *xx*Cron.SCH (onde *xx* são as iniciais do seu grupo), no diretório do seu projeto, usando o menu File | Save as. Para incluir o arquivo no projeto, clique no menu Hierarchy | Add Current Sheet to Project.

Complete o esquemático com a instanciação da macro *xx*DsDrv que você criou.

- Para inserir um bus, clique em Mode | Draw Bus (ou no ícone correspondente). Se desejar terminar o bus sem conectá-lo, clique duas vezes – na janela que se abrir, digite o nome, ajuste a dimensão e selecione "Terminal Marker" como "None" (isso indica que não se trata de um bus de entrada ou saída de uma macro).
- Para conectar sinais (pinos) ao bus, clique no menu Mode | Draw Bus Taps. Em seguida clique num pino para o programa fazer a ligação com o bus que estiver em frente.

Notas importantes (veja a Figura 3.25):

- **Nota 1**: é pelo nome do bus que o programa define os sinais que o compõe. Se o bus estiver sem nome, o programa atribuirá nomes genéricos para os sinais. NÃO DEIXE um bus sem nome: se estiver sem, clique duas vezes sobre ele para definir o nome.
- **Nota 2**: as vias também deve ter nome (por exemplo, DA3) para que o programa saiba como ligá-las aos sinais do bus. Para isso clique duas vezes no fio e digite o nome da via no campo Net Name.

Implemente o circuito e transfira para a FPGA. Verifique o funcionamento do circuito.

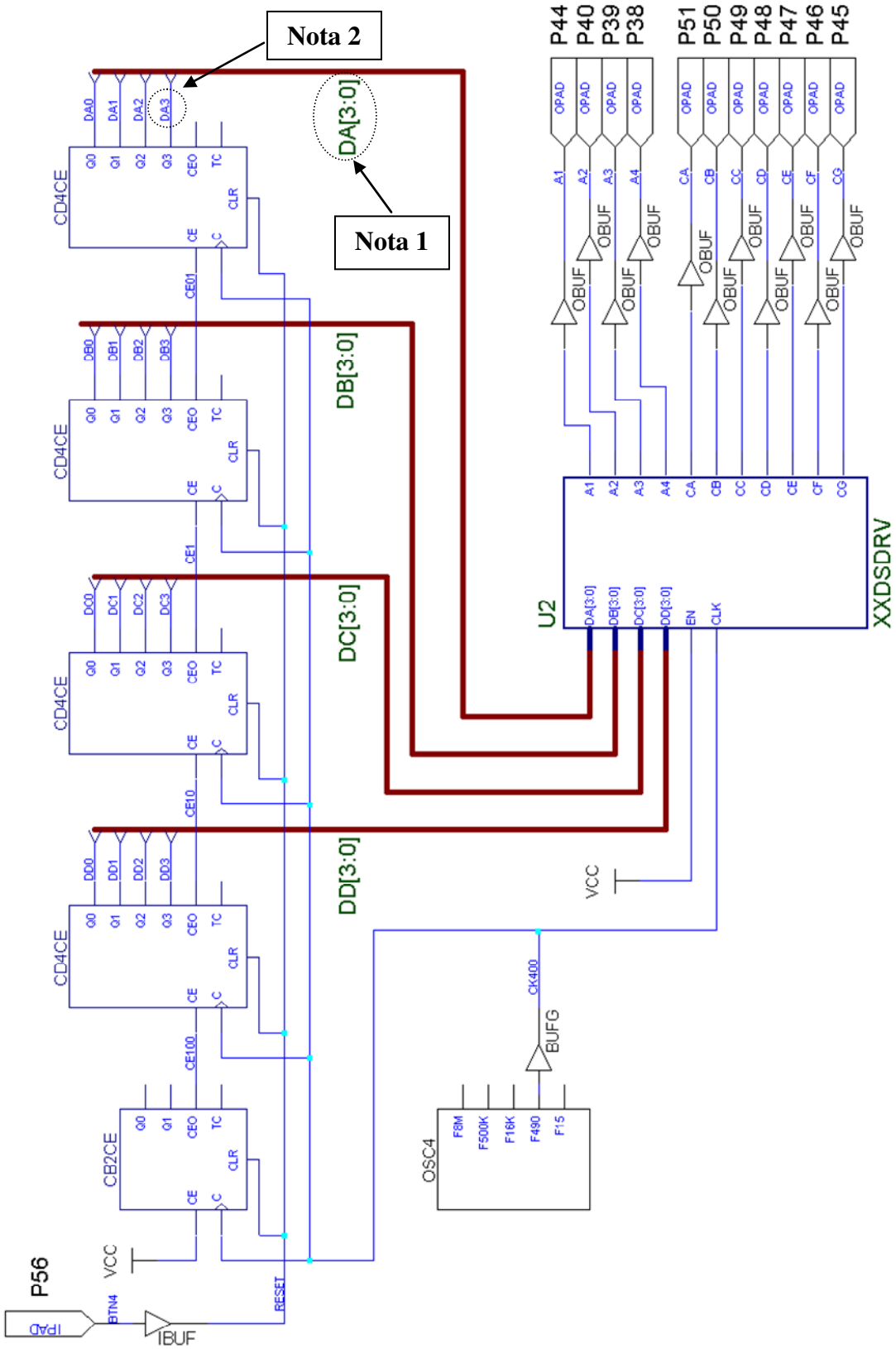


Figura 3.25 Circuito do cronômetro

Anotação 5a Caso seu projeto tenha sido aprovado pelo professor, NÃO é necessário refazer o DL. CASO CONTRÁRIO, refaça-o (a mão livre!) no relatório antes de começar a edição do circuito.

Exercício 8 Analise o diagrama da macro 7SEG em anexo, que converte um número binário de 0 a 15, codificado nos 4 bits de entrada $B_3B_2B_1B_0$, para o padrão de acionamento de um mostrador de 7 segmentos com anodo comum. Uma saída C_i em 0 faz o led *i* correspondente acender, e vice-versa. Inspecionando o circuito, indique que leds serão apagados quando nas entradas $B_3B_2B_1B_0$ tivermos:
a.1) 0010 a.2) 0111 a.3) 1101

Exercício 9 Faça o diagrama lógico do circuito de acionamento do mostrador de sete segmentos, mostrado na Figura 3.15. O circuito a ser implementado é o que se encontra delimitado pela linha tracejada; o display de 7 segmentos não faz parte dele, já que é um componente separado. No seu projeto, utilize os elementos mostrados na Figura 3.16. A entrada CLR do contador deve ser ligada à 0 V (**GND**), já que não será usada.

Atividade 6 Run / Stop

Modifique o circuito do cronômetro para incluir mais dois botões: Run (BTN1) e Stop (BTN2).

Ao se pressionar o botão Run, o cronômetro deve disparar. Pressionando-se Stop, a cronometragem se detém; nessa condição, pressionando-se Run novamente o cronômetro continua a partir da contagem em que parou.

Atenção: o botão Clear (BTN4) somente pode zerar o cronômetro quando este estiver parado. Dica: use um latch para controlar a habilitação dos contadores (pino CE).

Anotação 6a Faça o diagrama lógico do circuito modificado de forma **resumida:** não inclua PADS e BUFS; havendo componentes repetidos com ligações iguais, desenhe somente o primeiro e o último com reticências entre eles. No entanto, indique os botões e leds da placa XLA usados nas entradas e saídas do circuito.

Anotação 6b Anote a **hora atual** e mostre o **circuito funcionando** para o professor.

Atividade 7 Finalização

Deixe a bancada em ordem e limpa. Falhas nesse procedimento serão penalizadas.

Anotação 7a *Check list* – Verifique e anote no relatório cada um dos itens abaixo. ESCREVA os nome dos itens e não apenas uma seqüência de meros "sim" e "não".

- **Equipamentos** Liste todos os equipamentos usados. Todos estão desligados? Em especial, certifique-se que o COMPUTADOR e o MONITOR estejam DESLIGADOS (NÃO deixe o computador em STAND-BY).
- **Multímetro** Os cabos das pontas de prova do multímetro estão arrumados**?** Deixe o multímetro sobre o **tampo baixo da bancada**, para que possamos conferir facilmente se está desligado.
- **Osciloscópio** Os cabos das pontas de prova do osciloscópio estão arrumados?
- **Cabos** Liste todos os cabos de conexão usados. Os cabos foram recolocados nos lugares de origem? (O cabo paralelo da placa deve ser deixado conectado ao computador)
- **Placa XLA** a placa foi embalada no plástico anti-estático e guarda da na caixa **com sua fonte**?
- **Empréstimos** Usou alguma coisa de outra bancada? O quê? Foi devolvido?
- **Defeitos** Encontrou algum defeito? Preencheu a Comunicação de Defeito?
- **Limpeza** A bancada está limpa?

Equipamentos com defeito devem ser deixados na própria bancada. Cabos, cabinhos e outros acessórios com defeito devem ser entregues ao professor, juntamente com a Comunicação de Defeito.

Entregue o relatório feito em sala e o seu pré-relatório

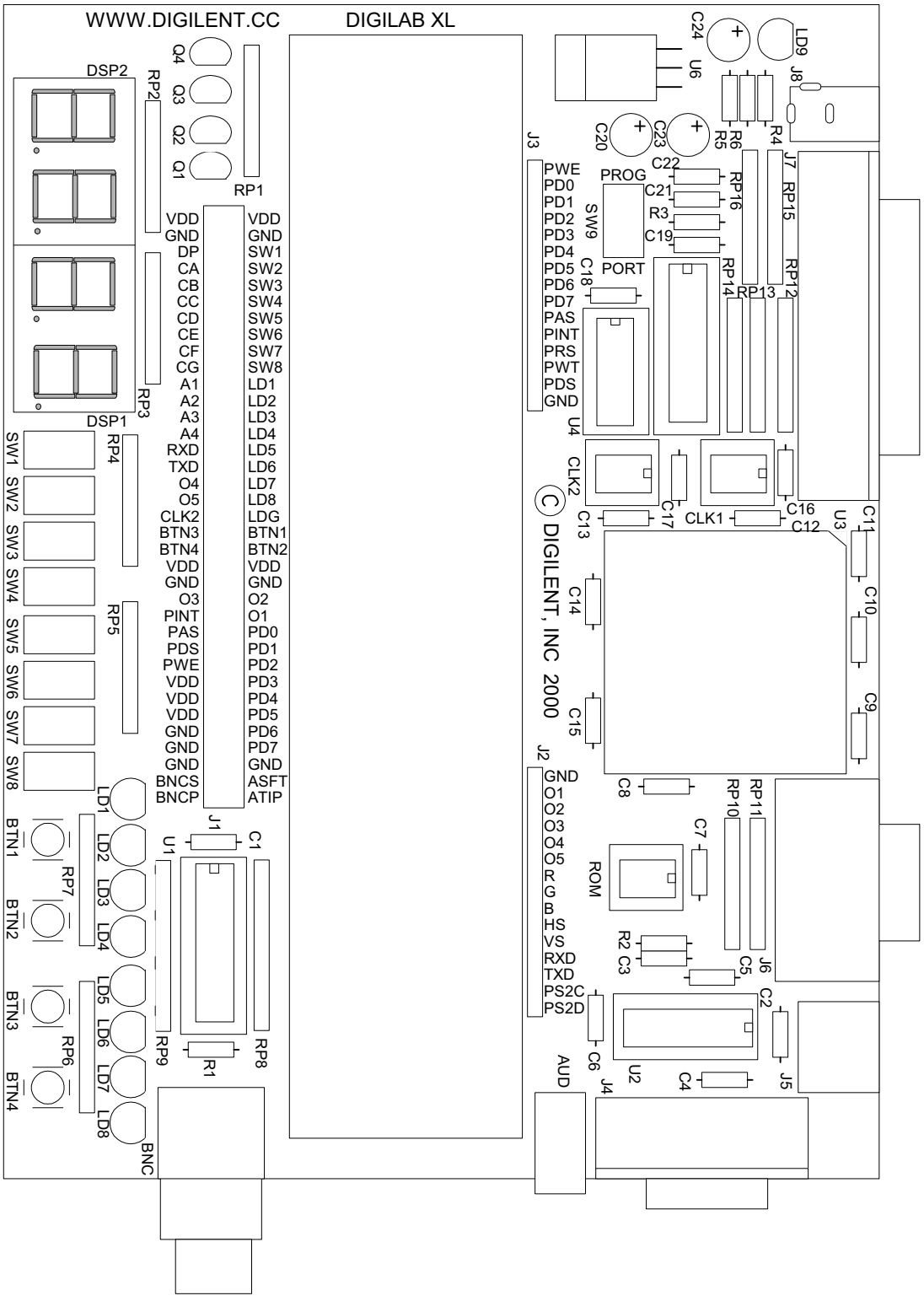
Respostas e dicas de alguns exercícios

Exercício 4 Flip-flop de Q2: J=K = CE.Q1.Q0; flip-flop de Q3: J=K = CE.Q2.Q1.Q0.

Exercício 6 Contagem **após** alguns instantes de clock (Q[3:0]):
3: 0011 4: 0011 5: 0100 16: 1111 17: 1111 18: 0000

Exercício 7 Q[3:2] seria incrementado incorretamente nas bordas de *clock* 5 e 6.

Board Graphic



FPGA

The Digilab board can accommodate a Xilinx Spartan XCS05, XCS10, XCS05XL, and XCS10XL FPGA in the 84-pin PLCC socket (the XL parts require a 390 ohm R6 for 3.3V operation). Any of these SRAM-based FPGA's may be programmed using a parallel cable or an SPROM (see above). The parallel cable provides an inexpensive programming solution that is compatible with the Xilinx CAD-tool cable detection software, so that the FPGA can be programmed without leaving the Xilinx environment. Refer to the Xilinx Spartan data sheet (http://www.xilinx.com/partinfo/spartan.pdf) for technical data regarding the FPGA's.

The table on the right shows all FPGA pin connections. In this pin-list, gray boxes indicate dedicated pins that are not available for use. Italicized names indicate dual-purpose pins; for these pins, the Xilinx function is shown first followed by Digilab's assignment in parenthesis.

Some FPGA signals, including the LED drive signals and the unassigned (or open) signals are available on the J1 prototyping connector. Care should be taken to ensure that these signals are not simultaneously driven by both the FPGA and by other drivers. If the FPGA is loaded in the U3 socket and external circuits must drive these signals, it would be best to tri-state the FPGA signals.

The parallel port connector can be used as the FPGA programming port or as a parallel port. When downloading a circuit that drives the parallel port data signals, ensure that the signals are not driven until SW9 has been moved to the PORT position. See the parallel port demo project on the Digilent web site.

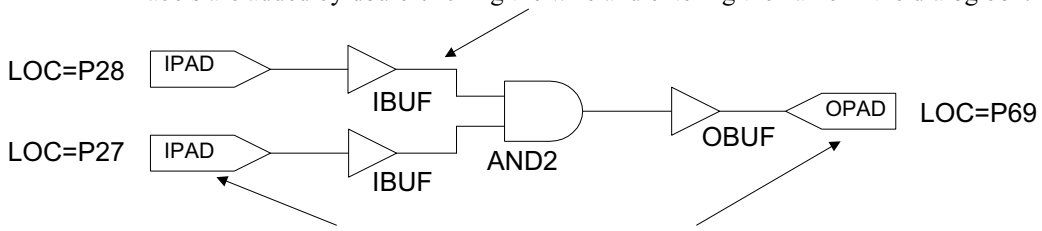
Pin #	Function	Pin #	Function	Pin #	Function
1	**GND**	29	O1	57	BTN3
2	**Vdd**	30	**M1_NC**	58	BTN2
3	PWE	31	**GND**	59	BTN1
4	PD0	32	**MODE**	60	LD8
5	PD1	33	**Vdd**	61	LD7
6	PD2	34	**M2_NC**	62	LD6
7	PD3	35	CLK2	63	**Vdd**
8	PD4	36	O2	64	**GND**
9	PAS	37	O3	65	LD5
10	PRS	38	A4	66	LD4
11	**Vdd**	39	A3	67	LD3
12	**GND**	40	A2	68	LD2
13	CLK1	41	*INIT (O4)*	69	LD1
14	PDS	42	**Vdd**	70	LDG
15	PWT	43	**GND**	71	*DIN (O5)*
16	PD5	44	A1	72	*DOUT (RXD)*
17	PD7	45	CG	73	**CCLK**
18	PD6	46	CF	74	**Vdd**
19	SW8	47	CE	75	*TXD (PINT)*
20	SW7	48	CD	76	**GND**
21	**GND**	49	CC	77	R
22	**Vdd**	50	CB	78	G
23	SW6	51	CA	79	B
24	SW5	52	**GND**	80	HS
25	SW4	53	**DONE**	81	VS
26	SW3	54	**Vdd**	82	PS2C
27	SW2	55	**PROG**	83	PS2D
28	SW1	56	BTN4	84	PINT

Any CAD-tool-designed circuit that requires fewer than about 5K – 10K gates can be programmed into the Xilinx FPGA. However, the circuit description must first be transformed into the format required by the FPGA. This transformation proceeds in several steps, typically beginning with an EDIF, VHDL, or Verilog file format and ending with a Xilinx "bit" file format. Xilinx (of course) produces a tool that accomplishes this transformation, which is available in the Xilinx Alliance and Foundation products (see the Xilinx web site). Although other methods of transforming files may be available, only the Xilinx solution has been used with the Digilab board. Although the use of the Xilinx tools is beyond the scope of this document, Xilinx has several good tutorials and helpful documentation available at their web site.

All signals on the Digilab board that connect the buttons, switches, and LEDs to the J1 connector are connected to the Xilinx FPGA chip as well. Any circuit implemented in the FPGA can use the buttons and switches as inputs and the LEDs as outputs. When the Digilab board was fabricated, the buttons, switches, and LEDs were connected to particular pins on the FPGA (see the table in the previous section for all FPGA pin definitions). To connect an FPGA-based circuit to these devices, you must include information in your schematic to "map" circuit inputs and outputs to particular FPGA pins. Mapping is accomplished by including special components in your schematic called IPADs, IBUFs, OPADs and OBUFs. These components exist solely to allow you to define physical pin connections, and so they only need be used in circuit schematics that you intend to download.

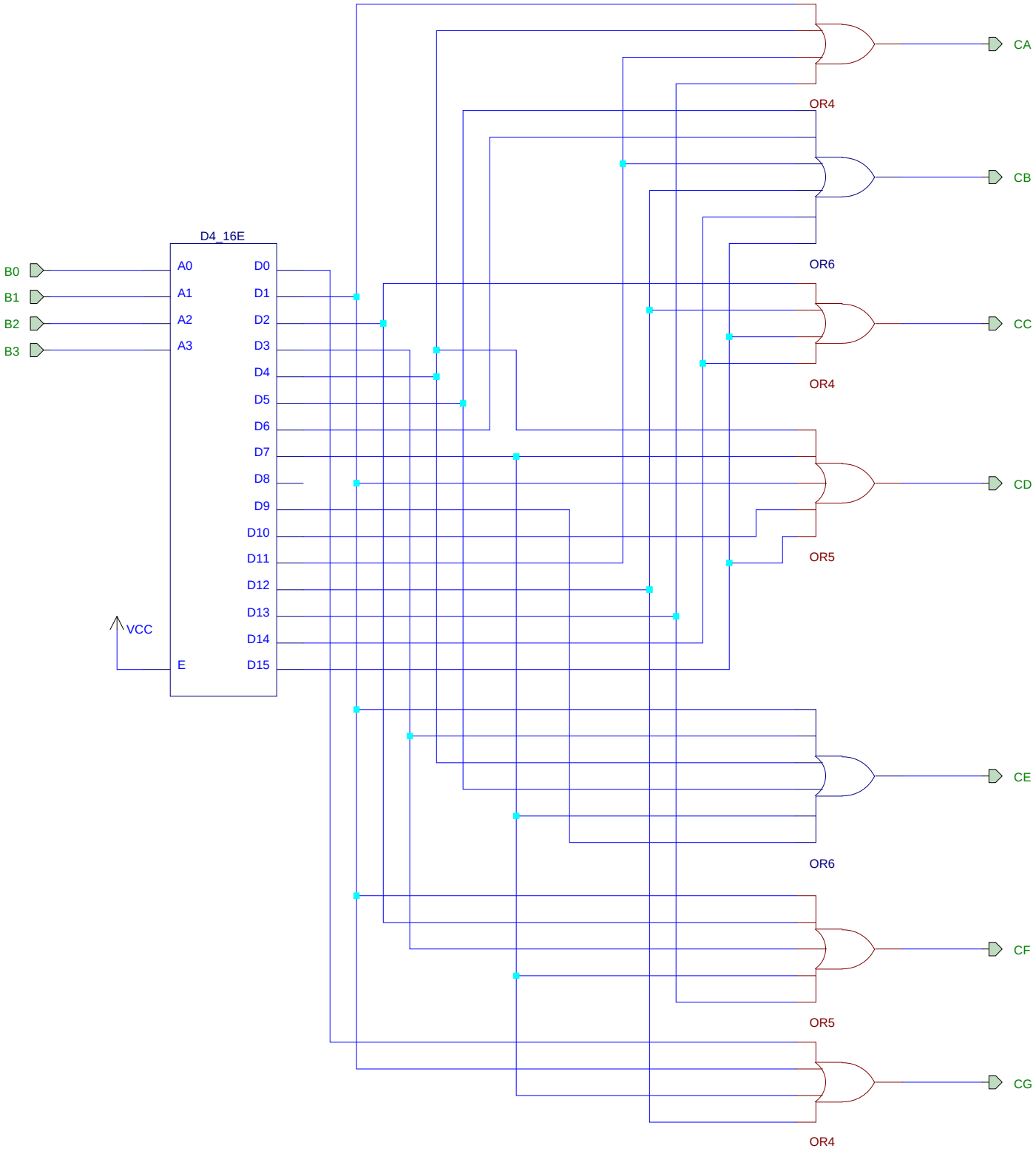
Once you have a complete and error-free schematic, you may add IBUFs and IPADs to all inputs, and OBUFs and OPADs to all outputs. Then, the IPADs and OPADs can be connected to particular pins by double-clicking the pads and entering the "LOC" parameter and pin number in the appropriate fields (Name and Description, respectively). In the example circuit below, two switches (SW1 on pin P28 and SW2 on pin P27) are connected via an AND gate to LED1 (LD1 on pin P69). If this circuit were downloaded to the FPGA, then LD1 would illuminate whenever SW1 and SW2 were asserted.

For simulation purposes, labes can be added to wires between the IBUF's and the circuit.
Labels are added by doule-clicking the wire and entering the name in the dialog box.

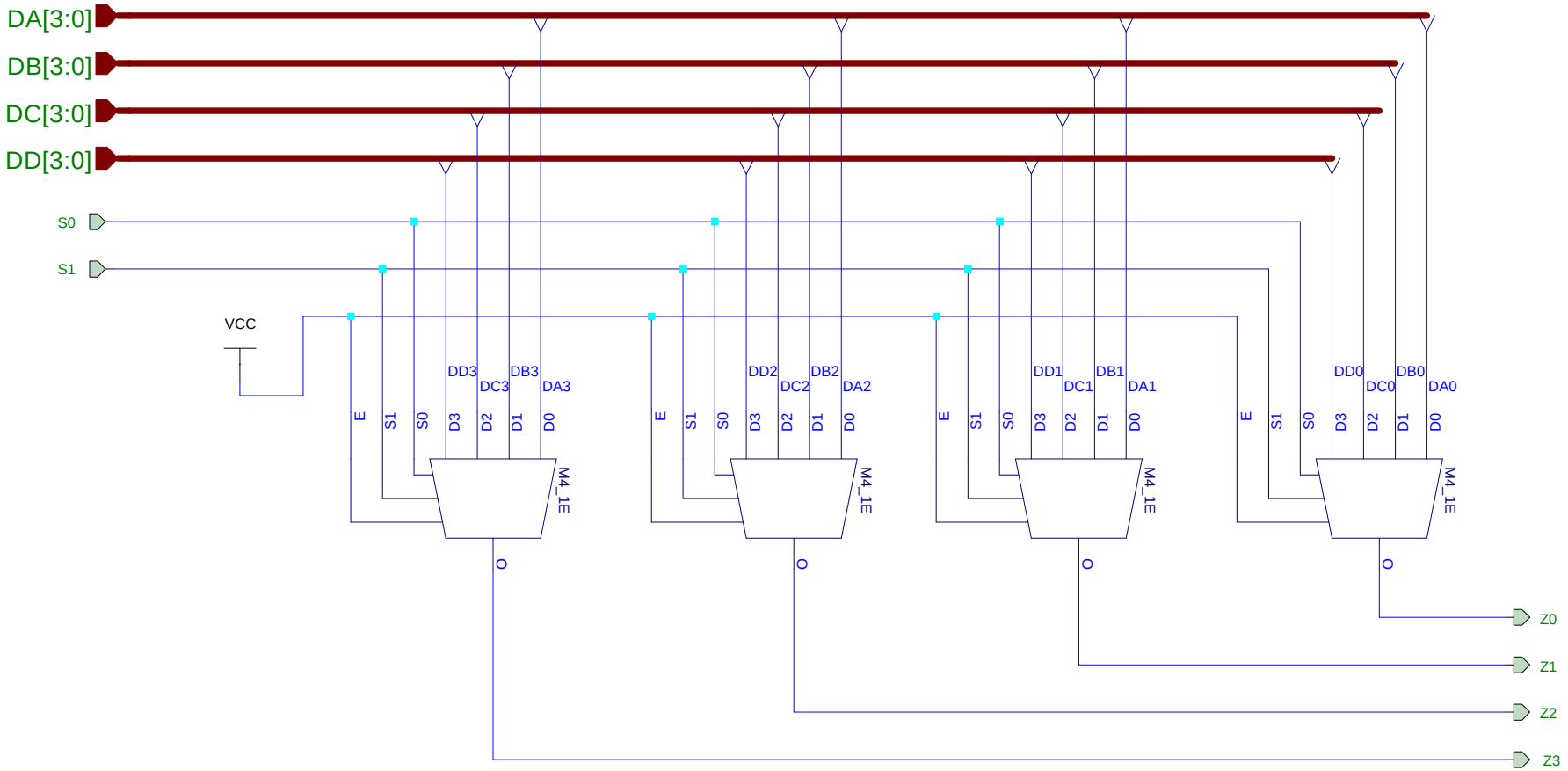


"LOC" parameters are added by double-clicking the pad symbol and entering **LOC** in the Parameters Name feild and **Pnn** in the Parameters Description feild, and then pressing ***Add*** and ***OK***.

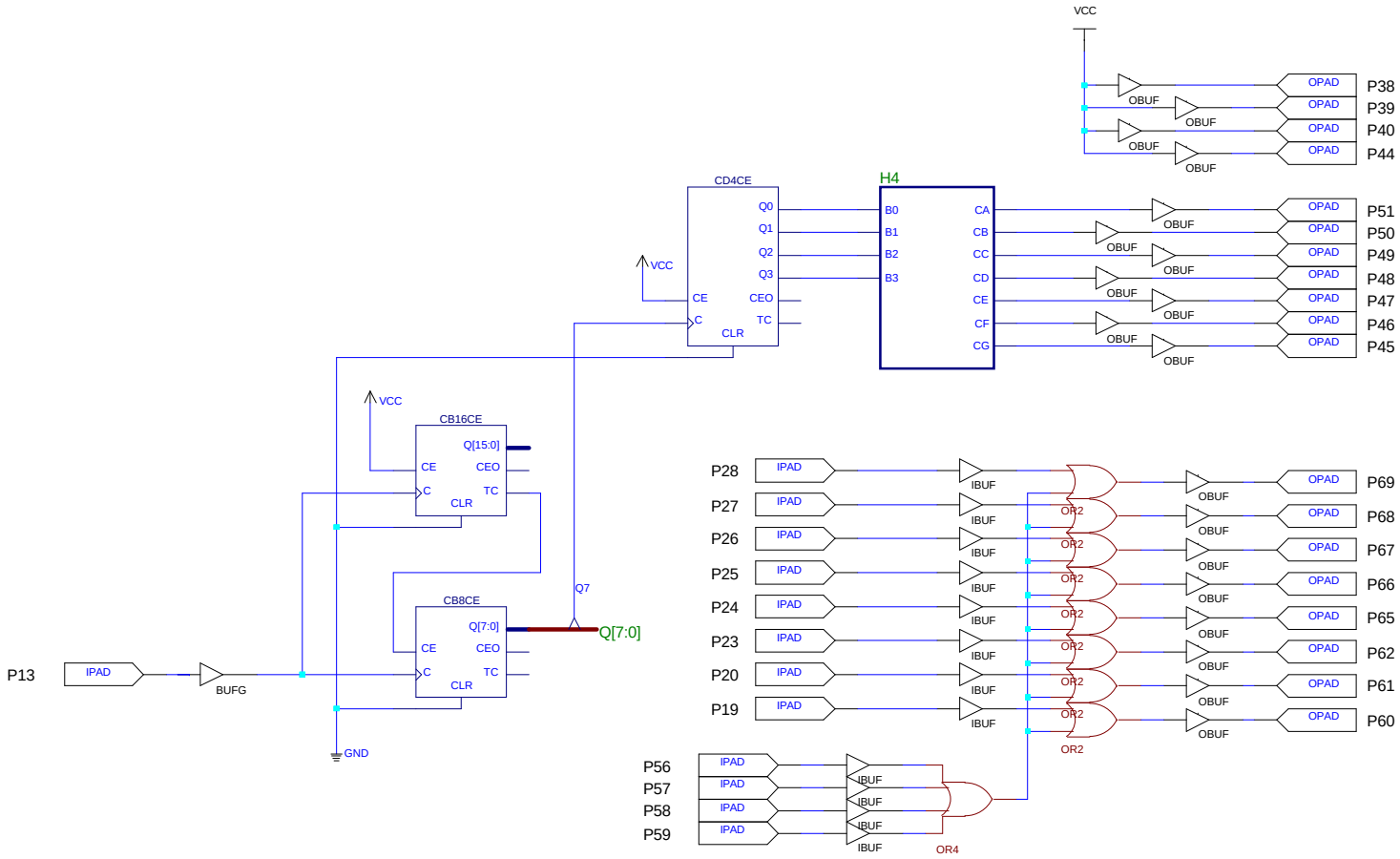
Once all IPADs, IBUFs, OPADs, and OBUFs have been added and edited with pin locations, you can begin the implementation process by choosing the "Implementation" button from the Xilinx main screen. In the first dialog box that appears, choose Yes to update the netlist from the schematic editor. In the second dialog box, make sure the device is S10PC84 and speed is 3 before proceeding; the version and revision names can use the defaults. Press the Run button, and then wait for the status window showing Translate, Map, Place & Route, Timing, and Configure processes to terminate. Before proceeding, make sure that the Digilab board is powered on and connected to the PC via the parallel cable, and that SW9 is in the PROG position. Then select the Programming option from the Xilinx main window, and "hardware debugger" from the subsequent dialog box. The cable should be auto-detected; if not, manually choose the parallel cable in the Cable → communications dialog box.



Celso M. Furukawa	Project: PMR2433
Escola Politécnica da USP	Macro: 7SEG
PMR	Date: 8/21/00
Date Last Modified: 9/10/02	



Celso M. Furukawa	Project: PMR2433
Escola Politécnica da USP	Macro: MUX4_4_1
PMR	Date: 5/5/02
Date Last Modified: 9/10/02	



Xilinx Corporation	Project: XLA5
2100 Logic Drive	Sheet: XLA51
San Jose, CA 95124	Date: 8/21/00
Date Last Modified: 4/18/02	