Laboratório sobre Circuitos Seqüenciais e Máquinas de Estados Finitas

Prática: O exemplo da máquina de venda de refrigerantes em lata: especificação projeto e implementação em hardware do circuito usando diagramas de esquemáticos

Recursos: Sistema de CAD Foundation da empresa Xilinx, Inc, Plataforma XS40/XST1 da empresa Xess, Inc, incluindo o hardware e o software Xstools (software adicional: Espresso da Universidade de Berkeley)

Parte I - Objetivos

O **Laboratório sobre Introdução a Sistemas de CAD, Projeto com Esquemáticos e Circuitos Combinacionais** objetivou a familiarização com algumas ferramentas clássicas associadas à captura e validação de projeto de sistemas digitais, notadamente o editor de esquemáticos e o simulador lógico. O tema da aula foi a implementação de um somador de 8 bits, um **circuito combinacional**, visto que o valor de suas saídas depende única e exclusivamente do valor instantâneo de suas entradas. Conceitos importantes foram abordados, tais como o fato de esquemáticos serem compostos por três tipos de entidades fundamentais (pinos, fios ou redes e componentes), o uso de modularização e hierarquia nos projetos, o uso/utilidade de *scripts* de simulação, e a caracterização/diferenciação de diagramas de esquemáticos e macros de bibliotecas, bem como o próprio conceito de bibliotecas de símbolos esquemáticos.

Neste trabalho, a familiarização com as ferramentas continua, mas o tema envolverá a outra classe fundamental de circuitos digitais: os **circuitos seqüenciais**. Nestes, as saídas dependem não apenas do valor instantâneo das entradas, mas também de entradas passadas, ou melhor, da ordem de aparecimento destas. Esta ordem é armazenada internamente, normalmente de forma parcial, pelo circuito, e usada para definir as saídas junto com as entradas atuais. Em outras palavras, um circuito seqüencial possui uma **memória interna**, que armazena informações distintas em momentos distintos. Às informações armazenadas internamente por circuitos seqüenciais, dá-se o nome de **estado interno** do circuito, ou simplesmente **estado** do circuito. Em particular, nos interessa aqui lidar apenas com circuitos seqüenciais síncronos, ou seja, aqueles onde a troca de um estado para outro é comandada por um **sinal de relógio**. Um modelo abstrato útil para representar circuitos seqüenciais são as **máquinas de estados finitas** (em inglês, *finite state machines*, ou **FSMs**), também denominadas **autômatos finitos**, revisado nas aulas teórica. Lembre-se que a implementação de FSMs sob a forma de hardware síncrono pode ser realizada seguindo o modelo estrutural da Figura 1, onde toda a parte seqüencial consiste de um registrador de estados controlado pelo sinal de relógio, e a parte combinacional consiste de dois conjuntos de funções Booleanas, uma para gerar a saída (função λ da Figura 1) e outra para calcular o próximo estado (função δ da Figura 1), ambas dependentes das entradas primárias atuais (I_i) e do estado atual armazenado no registrador de estado (S_i). Com isto, dada uma codificação dos estados internos, basta gerar o hardware combinacional que implementa as duas funções para implementar a FSM.

Um objetivo específico deste laboratório é, a partir de uma especificação informal de uma máquina de vender refrigerantes em lata, projetar uma versão simplificada do circuito de controle desta. O trabalho limitar-se-á, por enquanto, a versões simuláveis descritas usando **diagramas de esquemáticos**. Além deste, outro objetivo específico deste laboratório é, a partir do projeto da máquina de venda de refrigerante em esquemáticos, **aprender como, a partir de um projeto validado funcionalmente chegar a um hardware que efetivamente funcione**. Isto será feito usando-se ferramentas de síntese e implementação automatizada de hardware disponíveis, bem como através da plataforma de prototipação disponível no ambiente de aula.

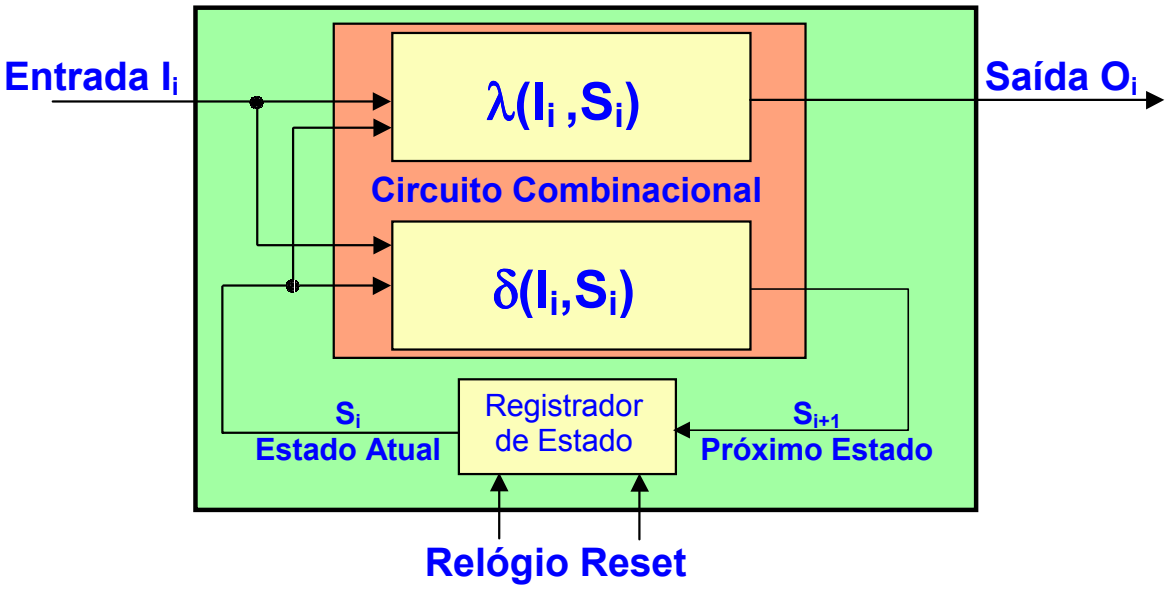


Figura 1 – Modelo estrutural para FSMs síncronas.

Parte II - Especificação da Máquina

A máquina deve ser capaz de fornecer dois tipos de refrigerantes, denominados MEET e ETIRPS. Estes estão disponíveis para escolha pelo usuário a partir de duas teclas no painel com o nome dos refrigerantes. Ambos refrigerantes custam R$1,50 e existe na máquina uma fenda para inserir moedas com uma mecânica capaz de reconhecer moedas de R$1,00, R$0,50 e R$0,25, e capaz de devolver automaticamente qualquer outro tipo de moeda ou objeto não reconhecido. Além disso, durante a compra, o usuário pode desistir da transação e apertar a tecla DEV que devolve as moedas inseridas até o momento. Somente após acumular um crédito mínimo de R$1,50 o usuário pode obter um refrigerante. A devolução de excesso de moedas é automática sempre que o valor inserido antes de retirar um refrigerante ultrapassar R$1,50. Para evitar situações esdrúxulas tenham de ser consideradas, tais como aquelas onde o usuário gera simultaneamente mais do que um evento de entrada, vamos supor que existe um codificador de prioridade que produz um código de uma única tecla (ou nenhuma tecla) sendo apertada a cada instante do tempo. Vamos supor também que existem circuitos para evitar que o fato de uma tecla ficar muito tempo apertada seja interpretado como vários apertos consecutivos da mesma tecla. As duas observações finais são hipóteses simplificadoras da especificação. No trabalho de implementação, deve-se assumir a existência externa do codificador de prioridade e dos circuitos de tratamento de tecla. Uma terceira hipótese simplificadora consiste em ignorar a composição exata das moedas inseridas na máquina, atendo-se apenas ao montante total inserido.

Parte III - Projeto da Máquina

O projeto inicial consiste em definir a interface do circuito de controle da máquina com o mundo externo e definir o comportamento deste circuito segundo uma máquina de estados finita. A interface pode ser depreendida a partir da especificação. Primeiro, cada tecla do painel corresponde a uma entrada de um bit. Tecla apertada será bit em ‘1’ e tecla não apertada será bit em ‘0’. Isto vale para as teclas de pedido de refrigerante (MEET e ETIRPS) e para a tecla de devolução do dinheiro (DEV). O circuito de controle deve ser síncrono, e suponha que seu relógio será muito rápido comparado com as ações do usuário. Logo, cada uma destas teclas, uma vez apertada, irá gerar um sinal que fica muitos ciclos de relógio ativado. Outro ponto de entrada é a informação de moedas inseridas na fenda. Suponha que o circuito eletromecânico de reconhecimento de moedas gera um pulso de curta duração (longo apenas o suficiente para ser detectado pela próxima borda de relógio, e curto o bastante para não estar ativo em duas bordas consecutivas. Isto simplifica o

projeto do controlador, constituindo-se em mais uma hipótese simplificadora) para cada moeda reconhecida como válida. Assim, isto corresponde a três outras entradas do circuito de controle, denominadas M025, M050, M100, cada uma destas entradas recebendo um pulso em '1' cada vez que moedas válidas de R$0,25, R$0,50 e R$1,00 são inseridas, respectivamente. As saídas da máquina correspondem a pulsos de devolução de moedas (D025, D050 e D100) e pulsos de liberação de refrigerante (LMEET e LETIRPS). Logo a interface do circuito de controle está completamente definida, tendo 8 entradas e 5 saídas. Esquematicamente, mostra-se o diagramde blocos resultante desta discussão na Figura 2:

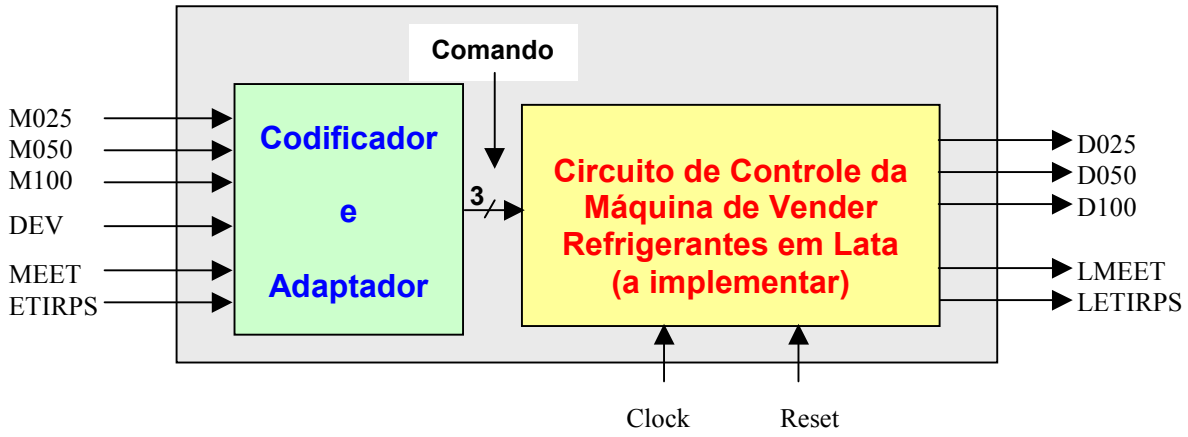


Figura 2 – Diagrama de blocos da máquina de venda de refrigerantes, mostrando a interface com o mundo externo.

Note-se na Figura 2 a separação do circuito Codificador e Adaptador, este responsável por realizar todas as tarefas das hipóteses simplificadoras estabelecidas aqui e na **Parte II**. A saída do Codificador e Adaptador é um vetor de 3 bits que contém um **Comando**. Assuma a seguinte codificação para este vetor: **Comando**=000 – nenhuma tecla apertada, **Comando**=001 – chegou moeda de R$0,25, e assim por diante, na ordem de aparecimento das entradas na Figura 2, de cima para baixo, terminando com **Comando**=110 – apertada a tecla de Pedido de lata de refrigerante ETIRPS. O último código, **Comando**=111, nunca deve aparecer, constituindo-se assim em uma condição de inespecificação, ou *don't-care*. Este último pode e deve ser usado para simplificar o projeto do circuito.

O próximo passo é definir a tabela de transição de estados para o circuito de controle. Para fazer isto é necessário definir que informações serão usadas para indicar o estado da máquina. No caso, usar o valor total inserido até o momento é uma escolha adequada. Assim, a cada instante do tempo, podem ter sido inseridas moedas para perfazer qualquer valor entre R$0,00 e R$1,50, sem esquecer que sempre que o valor total exceder R$1,50 moedas em excesso são imediatamente devolvidas. Assim, a moeda de mais baixo valor (R$0,25) determina que precisaremos de 7 estados ao todo para representar qualquer combinação de moedas inseridas, inclusive nenhuma. Para gerar a tabela de transição de estados, basta definir as ações associadas a cada codificação do vetor **Comando**. Não se esqueça de considerar a ação do circuito quando nenhuma das entradas está ativa (entrada **Nenhum**). Assim, temos a Tabela 3, onde se mostra a tabela de transição de estados, onde as posições desta estão apresentando o próximo estado seguido da geração de sinais de saída. **Atenção** à convenção usada na Tabela 3: **se nenhum sinal de saída aparece, isto significa que todos são gerados com valor '0'** (por exemplo, quando se está no estado S000 e a entrada é M025). Note ainda que muitas entradas da tabela não estão preenchidas.

Parte do trabalho de implementação consistirá em escolher uma codificação dos estados internos. Perceba que esta escolha pode afetar o tamanho do hardware gerado. Cada codificação distinta pode gerar um hardware combinacional distinto. É importante salientar que existem codificações corretas e incorretas. Uma maneira simples de garantir a corretude de uma codificação é simplesmente atribuir códigos distintos a estados distintos. No entanto, se 2 estados têm o mesmo

comportamento em relação a entradas e estado atual, eles são ditos equivalentes, e não é necessário que os dois existam, podendo-se reduzi-los a um único estado, ou, equivalentemente, a um único código que os represente.

Tabela 3 – Tabela de transição de estados que descreve o comportamento da máquina de venda de refrigerantes.

Estado Atual	Comando						
	Nenhum=000	M025=001	M050=010	M100=011	DEV=100	MEET=101	ETIRPS=110
S000		S025			S000	S000	
S025		S050			S000, D025=1		
S050						S050	
S075							
S100	S100			S150, D050=1			
S125							
S150							

Tarefa 1: Pare aqui a leitura deste texto e execute as atividades numeradas de 1 a 6 descritas na **Parte VI** ao final deste texto. Somente após concluir estas atividades, prossiga na Segunda Fase deste laboratório, que inclui as **Partes IV, V** e atividades restantes da **Parte VI**.

Parte IV – Adaptação do Projeto da Máquina de Venda de Refrigerantes para Implementação na Placa de Prototipação XS40/XSt1 da Empresa Xess Inc.

Descreve-se a seguir, passo a passo, as ferramentas a usar e o processo para completar e adaptar do projeto obtido ao final da realização das Tarefas propostas até a **Parte III**, com vistas à implementação em hardware da máquina de venda de refrigerantes sobre uma plataforma específica disponível.

1. Plataforma XS40/XSt1 da Xess (Figura 4)

Trata-se de um Plataforma de Hardware adequada para uso educacional, formada por duas placas. Estas contêm um dispositivo de hardware reconfigurável do tipo FPGA (Field-Programmable Gate Array), memória, microprocessador (na placa XS40) e diversos recursos para realizar Entrada e Saída: displays de 7 segmentos, leds, conectores para teclado e vídeo, codificadores/decodificadores de áudio, e área de prototipação para montagens de hardware (sobretudo na placa inferior, a XSt1). Para trabalhar nesta e nas partes seguintes deste trabalho, utilize uma da plataformas XS40/XSt1 disponíveis no ambiente de aula.

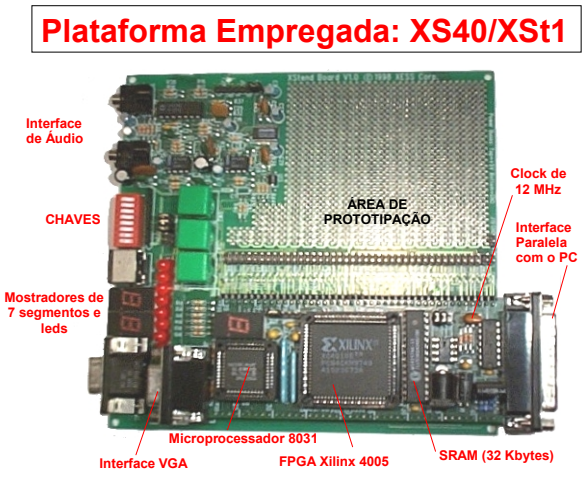


Figura 4 – Plataforma XS40/XSt1.

2. Para a implementação do circuito da máquina de vender refrigerantes na plataforma XS40/XSt1, são necessários alguns blocos adicionais de hardware, que adaptem os sinais e formatos de entrada e saída para utilização por seres humanos, devido às características físicas dos dispositivos de entrada e saída que queremos usar. Vamos agora estudar estes blocos:
   - Debounce: este é o nome do circuito que "limpa" o sinal binário de uma chave mecânica, para evitar erros tais como interpretar que apertar uma vez a tecla seja considerado como apertar diversas vezes. Abaixo segue uma explicação mais detalhada do porquê este circuito é necessário. No laboratório, mostrar-se-á na prática como

este problema ocorre, usando equipamento de teste e medida tal como osciloscópio. Seguem a justificativa e um pouco de discussão sobre os fenômenos físicos por trás do conceito.

- Fenômenos eletrônicos são rápidos. Fenômenos mecânicos são muito mais lentos. Considere-se uma tecla do tipo push-button, que possui uma mola impulsionando-a para o estado aberto, conforme a desenho da Figura 5. Quando não há forças além daquela da mola agindo, a chave está aberta e não passa corrente no resistor R1, fazendo com que a tensão na saída seja igual à tensão de alimentação do circuito, no desenho 5Volts. Isto se deve à famosa lei de Ohm V=RI, que relaciona tensão, corrente e resistência num circuito elétrico. R no caso de chave aberta é infinita (R=R1 + ∞ = ∞), pois a resistência de um circuito série de dois resistores é a soma das resistências individuais. Uma chave ideal funciona ora como uma resistência infinita (chave aberta) ora como uma resistência nula (chave fechada). Quando a chave está aberta não pode passar corrente por R1, logo não pode haver queda de tensão sobre esta, pois a lei de Ohm diz R1*0=0V. Assim, para garantir que o circuito respeite a lei das malhas (soma das tensões sobre todos os elementos de um circuito é 0), toda a tensão está aplicada sobre a chave, e a saída é uma tensão de 5 Volts em relação ao pólo negativo da fonte de alimentação. Quando uma força mecânica (gerada pelo dedo do usuário, por exemplo) atua sobre a chave, fechando-a, ela passa a agir como um fio sem resistência, e a saída está na tensão 0 Volts em relação ao pólo negativo da fonte de alimentação.

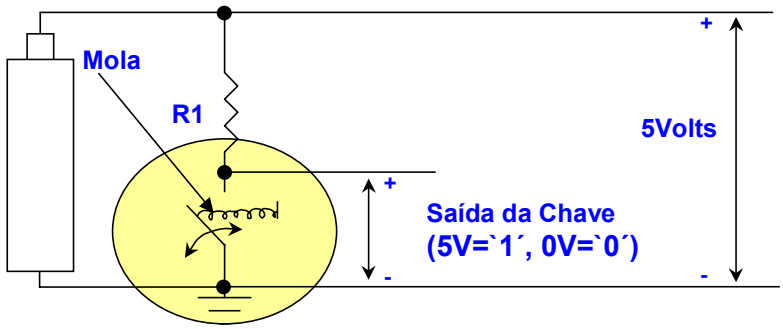


Figura 5 – Diagrama elétrico de uma chave mecânica.

- Assim, quando uma tecla push-button é pressionada (o raciocínio é análogo para qualquer outro tipo de contato mecânico), há idealmente uma transição da tensão de 1 para 0, ou seja, uma borda descida. Contudo a realidade é mais complicada, pois o contato dos dois pólos da chave não é perfeito, sempre podendo haver um ou mais "repiques" do contato, provocando não uma, mas diversas transições 0-1, 1-0 até estabilizar-se a saída no valor de chave apertada (0). Medindo com equipamento suficientemente sensível (e.g. osciloscópio) a relação Tensão versus Tempo na saída da chave, chega-se a um gráfico similar ao da Figura 6, supondo que o contato inicial dos pólos da chave ocorreu no instante de tempo 100 microssegundos.

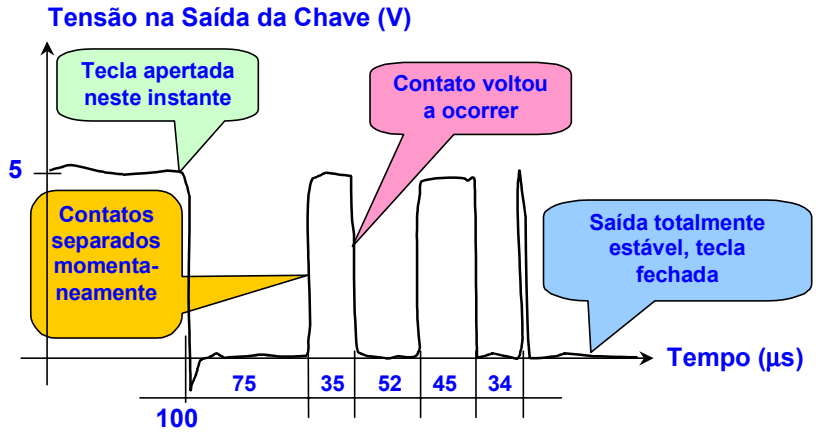


Figura 6 – Exemplo de Gráfico de Tensão versus Tempo típico de uma chave mecânica.

Note-se que ao invés de uma transição 0-1, foram obtidas 4 transições. Se este sinal é o relógio de um registrador sensível à borda de descida, seriam executadas 4 escritas neste último ao invés de 1 apenas, que é o que se pretende apertando a tecla apenas uma vez. Note-se também que o tempo ao longo do qual se distribuem as transições indesejadas (241 microssegundos, ou quase ¼ de segundo) é muito grande em relação ao tempo necessário para o sistema digital efetuar operações (na ordem de nanossegundos, ou seja, um bilionésimo de segundo, ou seja, 1/1.000.000.000 segundos). O circuito que recebe esta entrada "suja", elimina as transições espúrias e produz uma única transição a cada aperto de tecla se denomina

debounce (em inglês, **bounce** significa picar ou repicar, ou seja debounce significa retirar os repiques). Desenhe um diagrama de tempos da saída do circuito debounce para a entrada da Figura 6.

- Conversor *bin_7seg*: Trata-se de um circuito combinacional, que recebe como entrada um certo número de bits (entre 1 e 4bits, para ser exato), correspondendo a um valor binário representável como um dígito hexadecimal entre 0H e FH (p/ 4 bits) ou entre 0 e 7 ou 0 e 3 ou 0 e1 (p/ 3, 2 e 1 bits, respectivamente), e produz como saídas 7 sinais para acionar um mostrador de 7 segmentos, acendendo os segmentos que identificam o número binário representado na entrada. Estes dois circuitos serão fornecidos na homepage da disciplina, através de sua descrição em VHDL. Use-os inicialmente como células de biblioteca do tipo caixas pretas e insira-os no seu projeto na medida do necessário. O processo de inserção é detalhado mais adiante.

3. Tarefa 2: criar um novo projeto a partir de uma cópia de seu projeto original da máquina de vender refrigerantes realizado na Fase anterior deste laboratório (Note que seu projeto tem de estar já simulado **ok** antes de usá-lo aqui). Use o Comando **File →Copy Project** do Gerenciador de Projeto do CAD Foundation, criando uma cópia do seu projeto com outro nome, por exemplo **refr_hw**. **Faça isto agora**!
4. Tarefa 3: Crie um macrossímbolo de sua máquina de venda de refrigerantes. Instancie-o em uma folha vazia. Crie 1 pino de saída para cada saída do seu circuito e conecte-os às saídas pertinentes da máquina através de *buffers* do tipo ***OBUF*** (do inglês, *output buffer*) da biblioteca XC4000X. Estes elementos conectam sinais de saída aos pinos externos do circuito integrado e tem como função Booleana associada a função identidade, ou seja, funcionam logicamente como fios (eles servem para adaptar níveis de corrente e tensão elétrica dentro e fora do circuito). O resultado deve parecer-se, *mutatis mutandis*, com a Figura 7, onde o mesmo foi feito para um somador de 8 bits.

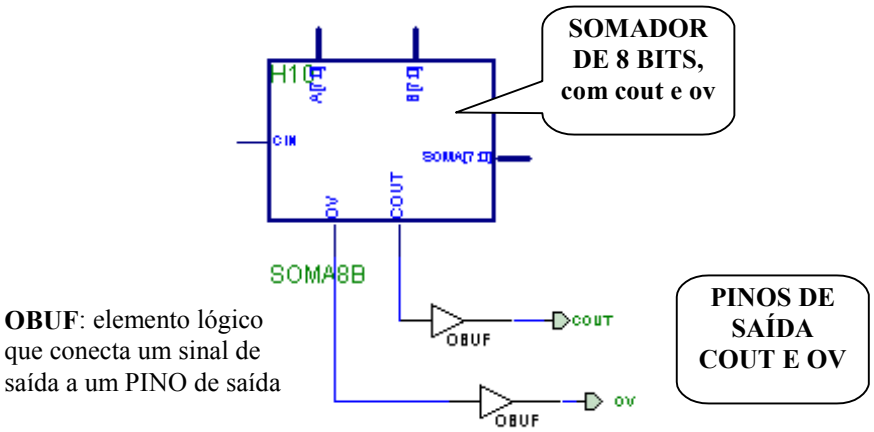
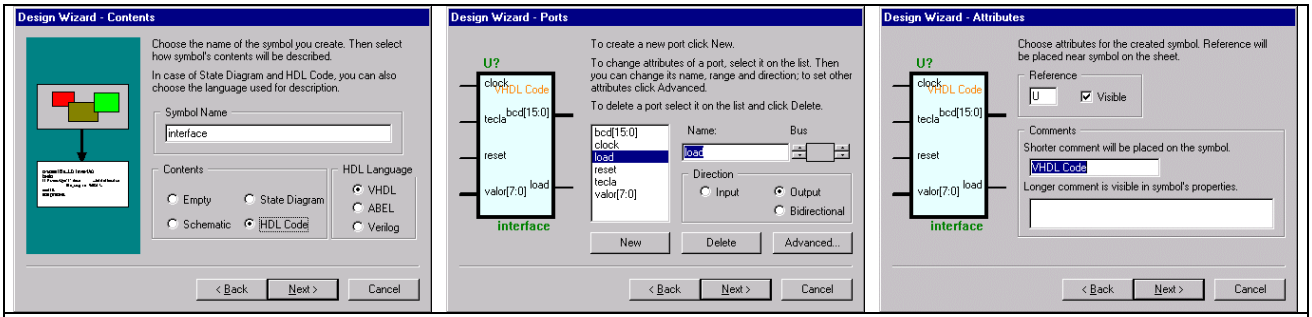


Figura 7 – Instanciamento de um somador de 8 bits e sinais de saída Cout e Ov.

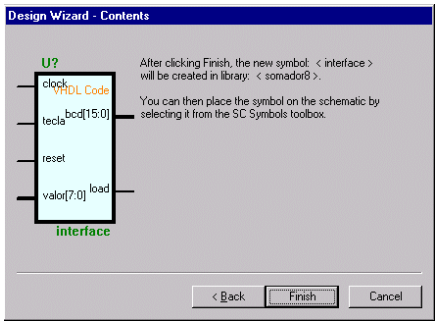
5. Tarefa 4: Agora, faça a inserção de dois novos blocos, do tipo VHDL, no seu projeto. Para cada um, crie um símbolo vazio, via opção Hierarchy → New Symbol Wizard, e insira neste símbolo a descrição VHDL dada, uma para o **debounce** e outra para o módulo **bin_7seg**. Estas descrições contêm todo o hardware necessário para realizar operações de entrada e saída na máquina de vender refrigerantes. Ignore por enquanto o texto VHDL dado. Este assunto será abordado apenas na segunda metade do curso. Um exemplo de detalhamento deste procedimento aparece abaixo, para outro tipo de circuito. Siga os mesmos passos para seu caso, *mutatis mutandis* (em latim, mudando o que tem de ser mudado). Abaixo aparece a criação de um bloco chamado interface. O módulo **debounce** tem três entradas de 1 bit (**tecla**, **reset**, **clock**) e uma saída de um bit (**teclaf**, significando tecla filtrada). O módulo **bin_7seg** possui uma entrada de quatro bits (**valor[3:0]**) e uma saída de 7 bits (**setseg[6:0]**). O nome do módulo criado tem de ser exatamente o mesmo do arquivo VHDL correspondente, ou seja, *debounce* e *bin_7seg*.



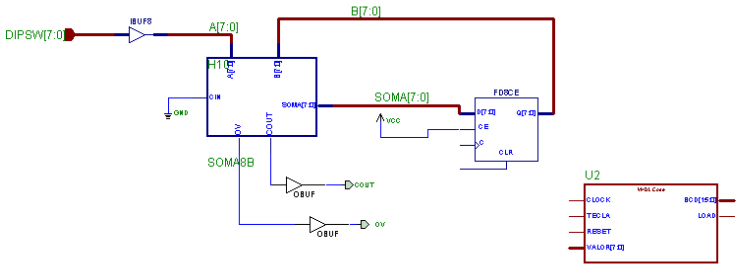
1. Nomeie o nome do bloco como *interface*
2. Indique que o conteúdo é HDL
3. Indique VHDL como a HDL que será utlizada.

1. Defina os pinos de entrada: clock, reset, tecla e valor (indicar no quadrado bus o intervalo 7:0)
2. Defina os pinos de saída: load e bcd (indicar intervalo 15:0)

Clique next para avançar.



Clique Finish para terminar.



No esquemático em edição, instancie o módulo, e verifique se corresponde ao esperado.

Figura 8 - Criação de um novo símbolo VHDL vazio.

6. Tarefa 5: Insira o código VHDL dado no bloco interface (até então vazio). Selecione o botão para percorrer a hierarquia (botão "H"), clicando duas vezes na instância do módulo criado(**debounce** e/ou **bin_7seg**). O editor de VHDL é aberto com um esqueleto do circuito.
   - apague todo o conteúdo do arquivo, ou seja, o esqueleto do circuito.
   - abra os arquivos VHDL disponíveis na homepage da disciplina, usando um editor de texto.
   - para cada arquivo, selecione todo o arquivo, copie-o e cole-o no arquivo VHDL em edição.
   - salve o arquivo VHDL.
7. Ao sair do editor de VHDL, será feita automaticamente a verificação sintática e a síntese de hardware do módulo VHDL. Poderá levar alguns minutos. Paciência ...
8. Tarefa 6: Insira um módulo **bin_7seg** no seu esquemático, e um módulo **debounce**. **debounce** deve ter sua saída *teclaf* conectada ao pino de clock de sua máquina, sua entrada *tecla* conectada ao um pino externo tal como *Tclk* através de um buffer de entrada, claro! (*Tclk* significa Tecla de clock, e vai ser um dos botões verdes da plataforma de hardware, a tecla SPARE). Os pinos de controle do debounce, por outro lado, devem ser ligados aos pinos externos *Treset* (Tecla de Reset, o botão verde do meio, na plataforma de hardware) e Ck12MHz (o pino conectado ao clock de 12MHz da plataforma de hardware (ambos via buffers). **Responda à seguinte questão**: Existem 5 entradas no circuito; porque pede-se aqui então que apenas a entrada de Clock tenha seu circuito de debounce? Não há problemas similares para os sinais de Reset e os 3 bits de comando? A saída do módulo **bin_7seg** deve ser ligada ao estado de sua máquina. Se o estado de sua máquina não for uma saída do seu módulo, altere sua macro para disponibilizá-lo como uma saída. Apesar disto não ser necessário ao funcionamento, será uma boa ilustração enxergar a troca de estados internos. Encare isto como uma informação útil para a depuração do circuito.

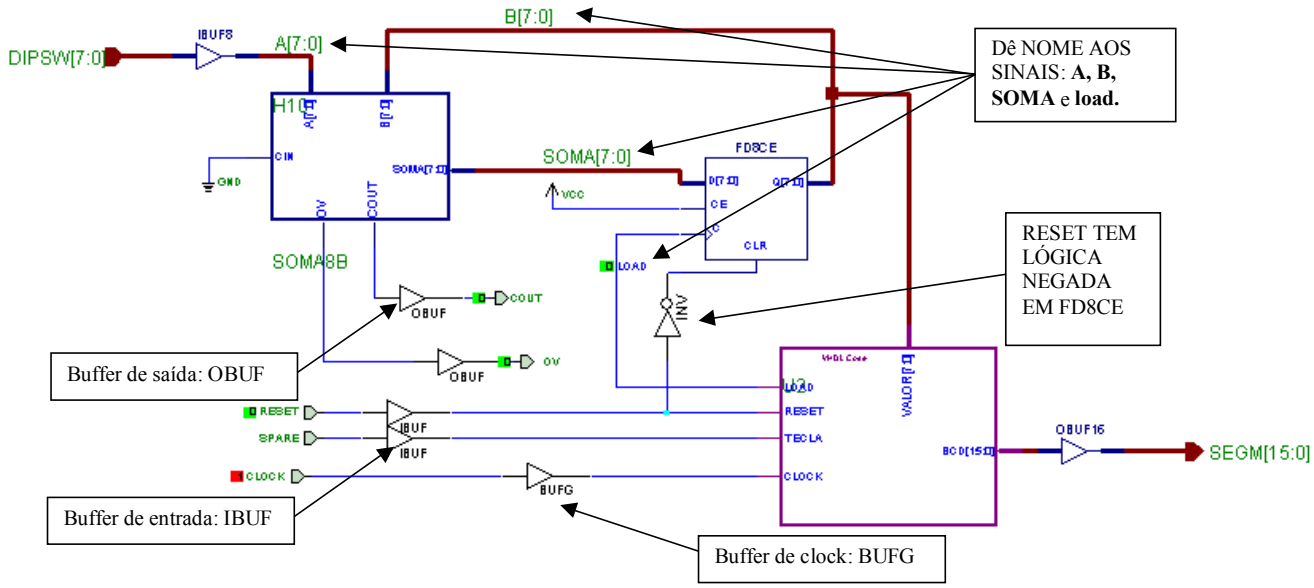


Figura 9 – Um exemplo de circuito completo, com buffers e terminais adequados.

9. Tarefa 7: Realize as conexões finais. ATENÇÃO: cuidado para ligar *buffers* de entrada em pinos de entrada e de saída em pinos de saída. Além disso, atentar que o *buffer* para o sinal de relógio **Clock** denomina-se **BUFG**, não use **IBUF**. Exemplos corretos para que vocês nele se baseiem aparecem abaixo. Note que cada entrada e cada saída é isolada do mundo externo via *buffers* adequados.

10. Agora estamos prontos para iniciar a síntese e implementação automatizada do circuito em hardware, exceto por uma parte. O circuito deve ser implementado sobre o FPGA da placa, um *chip* de 84 pinos, dos quais apenas alguns pinos serão usados. Elementos como teclas e chaves da placa XS40/XST1 servem para gerar entradas de dados (o comando C[2:0]) e de controle (clock e reset), enquanto que leds podem mostrar as cinco saídas do circuito e um display pode mostrar o estado atual da máquina. Estes dispositivos de entrada e saída estão conectados a pinos específicos na placa. O problema que surge assim é: Como fazer com que os programas de síntese saibam a que pinos ligar cada uma das entradas/saídas do circuito? As entradas são identificadas por nome, enquanto que os pinos do FPGA são identificados por outros nomes (no caso, o fabricante do FPGA usou a nomenclatura P1 a P84 para este chip). A solução para o problema vem na forma de um arquivo especial, chamado de **arquivo de restrições do usuário** (em inglês, user constraints file), com o mesmo nome que o projeto (e. g. refri_hw) e extensão .ucf. Ver-se-á a seguir como realizar isto, bem como o restante dos passos da implementação física do hardware.

Parte V – Síntese, Implementação e Teste da Máquina de Vender Refrigerantes

Uma vez editada a descrição abstrata do projeto (esta fase se chama tradicionalmente de *fase de projeto* ou, mais detalhadamente, *fase de captura do projeto*), passa-se a executar a *fase de validação de projeto*, através de simulação funcional (componentes considerados sem atrasos). Para se chegar ao produto final, é necessário passar ainda pela fase onde a descrição abstrata é traduzida para uma descrição física (a chamada *fase de síntese*). Após a síntese, procede-se à implementação do hardware (*fase de implementação*, que pode ser feita via fabricação do "chip" ou, como será o caso aqui, pela configuração de um dispositivo especial, através do processo de "descarga" (em inglês, *download*) da implementação sobre o FPGA da placa XS40/XSt1). Finalmente, deve-se proceder ao teste do circuito implementado (no que se denomina *fase de teste*).

A fase de síntese para o sistema que utilizamos é normalmente realizada de forma muito simples, pois o sistema de CAD (Computer Aided Design) Foundation possui um ferramental automatizado poderoso e de interface com o usuário amigável. No entanto, algumas tarefas devem ser cumpridas antes de realizar a síntese. Estas tarefas são aquelas dependentes do tipo de dispositivo físico a ser utilizado. Até a fase de validação, todo o processo de projeto foi independente da tecnologia de implementação (exceto a escolha do circuito integrado XC4005XLPC84-3, que não afetou nenhum passo acima, podendo ter sido qualquer outra). Agora, deve-se fornecer as informações específicas do circuito integrado, bem como da plataforma de implementação onde este circuito se encontra (placa e dispositivos a ele conectados, tais como cristal de relógio, chaves dipswitch e push-button, leds e mostradores de 7 segmentos). A mais fundamental destas informações consiste na associação de sinais de entrada e saída (E/S) do circuito aos pinos de entrada e saída do FPGA. Vejamos como se faz esta associação.

1. Existem dois métodos básicos para fazer a associação pinos E/S do projeto aos pinos E/S de FPGA no CAD Foundation para esquemáticos:
   - Método dependente do projeto – Dentro do esquemático de topo (no nível mais alto da hierarquia), usar componentes do tipo PAD (em inglês, *pad* significa almofada, é um jargão técnico usado para designar cada ponto do chip onde se soldam fios microscópicos que conectam algum ponto deste ao mundo externo através dos pinos do encapsulamento) da biblioteca XC4000X, tais como IPAD, OPAD, no lugar dos conectores de entrada e saída usados acima. Após, associar um número de pino do FPGA (no nosso caso, os pinos do XC4005PC84-3 são numerados de P1 a P84) a cada PAD. Este é o método padrão da ferramenta de esquemáticos, que não usaremos aqui;
   - Método independente de projeto – Inserir a relação entre E/S do projeto e do FPGA como uma tabela no arquivo **<nome do projeto>.ucf**, produzido automaticamente durante a criação dos arquivos de projeto. A vantagem deste método está na portabilidade dos arquivos de projeto isolando a região de projeto dependente de implementação num arquivo específico (conforme dito antes, UCF é um acrônimo para User Constraints File, ou Arquivo de Restrições do Usuário) . Este é o método que escolhemos aqui.
2. Recupere o arquivo "lab03011.ucf" disponível na homepage da disciplina.. Insira todo o conteúdo deste arquivo texto ao final do arquivo .ucf disponível no seu projeto e salve o seu .ucf. Adicione o arquivo assim obtido ao seu projeto (opção de menu **Document → Add** do Gerenciador de Projeto do Foundation). O arquivo "lab03011.ucf" possui as definições de todos os pinos necessários a este projeto específico. Ele foi montado com base nas tabelas do arquivo xst40mdl.pdf, que contém a descrição completa do modelo do usuário da placa XS40/XSt1, e

está contido nos manuais da plataforma XS40/XSt1 disponíveis no laboratório. **Observe que se você usou nomes de terminais no seu projeto que não correspondem exatamente aos nomes usados no arquivo .ucf, ou você deve editar seu esquemático, ou editar o arquivo de restrições, para compatibilizá-los**.

3. Para escolher o método de associação de pinos E/S do projeto aos respectivos pinos do FPGA, no gerenciador de projeto Foundation abra a janela de diálogo de opções de síntese, escolhendo **Implementation → Implementation Options…**. A seguir, clique no botão **Edit Options** da linha **Implementation** do quadro **Program Option Templates**. Agora, escolha a orelha **Translate** e lá marque a caixa **Create I/O Pads from Ports**, se ela já não estiver marcada. Esta opção anuncia para o CAD Foundation que a associação de pinos deve ser feita relacionando-se os terminais do nível mais alto de sua hierarquia de projeto aos pinos mencionados para estes no arquivo.ucf. A Figura 10 mostra como deve aparecer a tela após a escolha. Aceite as opções assim feitas, clicando nos botões **OK** até todas as janelas de diálogo estarem fechadas.

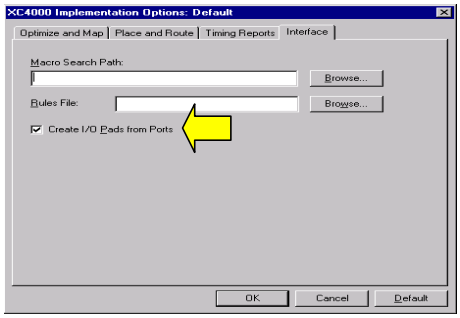


Figura 10 – Janela de escolha de método de associação de pinos E/S de projeto e FPGA.

4. Agora tudo está pronto para a síntese. Para tanto, basta clicar no botão Implementation da janela de fluxo de projeto (orelha Flow) do gerenciador de projetos. Se tudo estiver correto, surge uma janela de diálogo **Implement Design**. Clique no botão Run e aguarde o final da síntese. Se ocorrerem erros, estude as mensagens e resolva-os, em geral são causados por bobagem, tais como erro de sintaxe no arquivo .ucf. Se necessitar reexecutar a chamada da síntese automática depois que esta foi pelo menos uma vez concluída, você pode optar por clicar na opção **Overwrite current version**, para poupar espaço em disco. Cinco passos têm lugar durante a fase de síntese:
   - *Translate* - tradução dos dados de projeto do formato independente de tecnologia para o formato proprietário Xilinx;
   - *Map* – particionamento do seu projeto em um conjunto de pedaços que caibam cada um nos blocos físicos do FPGA, os chamados Configurable Logic Blocks ou CLBs e os Input/Output Blocks, ou IOBs;
   - *Place&Route* – Posicionamento dos CLBs/IOBs em posições físicas específicas do chip em questão e conexão via fios que têm de ser conectados entre dois CLBs dois IOBs e entre CLBs e IOBs;
   - *Timing* – Cálculo dos valores precisos de atrasos associados a fios e blocos de todo o circuito, visando uma ressimulação do projeto com atrasos detalhados. Este passo pode dar mais segurança que a implementação final vai funcionar (não usado aqui, reservado para cursos mais avançados);
   - *Configure* – Geração do arquivo binário ***<nome do seu projeto>.bit,*** a ser usado na configuração do FPGA para que este funcione como o hardware projetado por você.
5. Se a síntese tiver ocorrido sem problemas, surge a janela de diálogo abaixo. Clique **OK** nela.
6. Caso deseje, você pode agora observar a implementação física do projeto através do editor de "layout" de FPGAs Use a opção de menu **Tools → Implementation → FPGA Editor** do gerenciador de projeto Foundation. Lá pode-se observar e editar cada detalhe do mapeamento, do posicionamento e do traçado de rotas realizado. Não edite seu projeto nesta ferramenta, ele pode não funcionar mais depois disto!!!

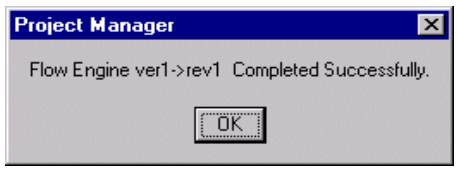


Figura 11 – Janela de diálogo final da implementação.

7. Agora vem a fase de implementação, mais simples que todas as anteriores. Para implementar seu projeto sobre o FPGA da placa XS40/XSt1 existe um conjunto de ferramentas que não está integrado com o CAD Foundation, e que deve ser executado separadamente (esta é uma característica particular da plataforma que empregamos). O único arquivo de todo o seu projeto que interessa é o arquivo binário gerado no penúltimo passo da síntese acima, ***<nome do seu projeto>.bit***. A versão mais atual deste arquivo está sempre disponível diretamente no diretório de mesmo nome que o nome de seu projeto. O objetivo da implementação é descarregar os conteúdos deste arquivo, via porta paralela do PC, para o FPGA, assim configurando-o. Mais uma vez há duas maneiras de fazê-lo:
   - *Via Interface Gráfica* – verifique se o programa GXSLOAD.EXE executa em sua máquina. Caso afirmativo, basta executá-lo. Surge uma janela para a qual basta arrastar o arquivo *.bit* desejado e a configuração é realizada. A janela do GXSLOAD tem a aparência da Figura 12;

Figura 12 – Janela da interface gráfica para configuração do FPGA da placa XS40/XSt1.

   - *Via comandos DOS* – O fabricante XESS (URL - http://www.xess.com/FPGA) da placa XS40/XSt1 disponibiliza o software de programação em sua versão DOS, bem como a interface gráfica do item anterior. O programa não gráfico denomina-se XSLOAD.EXE, e aceita como parâmetro o nome do arquivo *.bit* a ser usado na configuração do FPGA da placa. Cuidado contudo deve ser tomado pois o programa espera que a variável de ambiente XSTOOLS_BIN_DIR esteja apontando para o diretório contendo os arquivos de configuração do fabricante, tipicamente em C:\XSTOOLS\BIN ou em \XSTOOLS. Verifique se esta variável está disponível na sua máquina, senão use o comando DOS SET para criá-la ou alterá-la para seu valor correto. Após isto, basta executar a linha DOS: **XSLOAD *<nome do seu projeto>.bit***

8. Finalmente, realizada a fase de implementação vem a fase de teste do circuito. As informações para executar a contento esta fase são agora abordadas:
   - *Configuração das teclas* – (Use a Figura 4 para localizar os elementos da placa XS40/XSt1. Olhe a placa colocando os displays de 7 segmentos mais próximos de você.) Os três bits da palavra de comando *C[2:0]* são entrados usando as 3 teclas mais à direita do conjunto de dip-switches (8 chaves brancas num suporte vermelho. Posição OPEN indica chave aberta. Responda: Chave aberta corresponde a entrada 0 ou a entrada 1? ver Figura 5). As teclas verdes grandes possuem nomes **SPARE**, **RESET** e **PROGRAM**. Atenção: não aperte esta última tecla (a que está mais à direita), a menos que você deseje desconfigurar ("desimplementar") seu projeto do FPGA. A tecla **SPARE** (tecla verde mais à esquerda) corresponde ao sinal *Tclock* de seu projeto, enquanto que a tecla do meio **RESET**, corresponde ao *Treset* do seu projeto. Comece inicializando seu circuito por esta tecla.
   - *Mostradores e LEDs* – o mostrador esquerdo da placa XSt1 (situado ao lado do conector de teclado) mostrará o estado atual da máquina (entre 0, correspondente a S000 e 6, correspondente a S150). Quando você pressiona RESET, este deve mostrar 0. Dos 8 leds disponíveis acima dos mostradores (numerados de D8 a D1), apenas os cinco mais à esquerda são usados, sendo D8 o sinal D025, D7 o sinal D050 e assim por diante até LETIRPS.

- *Obs*: Se você quiser saber o mapeamento dos pinos do FPGA para pinos de dispositivos da placa XS40/XSt1, existem manuais da placa disponíveis no mesmo armário que as placas, no laboratório. O uso de placas e manuais fora do horário de aula é possível através de preenchimento de protocolo junto à portaria do LAPRO.

9. Exercite o seu circuito, verificando a correção dos resultados da operação da máquina. Calcule manualmente o valor do resultado e confira usando o circuito. Use os dados de seu script de simulação, por exemplo.

Parte VI – A Fazer e Entregar

Execute todas as Tarefas propostas acima. Elabore um relatório contendo todas as suas simulações, bem como a resposta a todas as questões que eventualmente aparecem no texto. Entregue este relatório no prazo especificado na homepage da disciplina. Procure mostrar as operações que o grupo fez ao professor, em aula, o que contribuirá para a nota de participação em aula. A seguir, segue a lista de atividades solicitadas para as diferentes tarefas acima:

1. Complete a tabela de transição de estados corretamente.
2. Verifique se existem estados equivalentes na sua implementação. Caso existam, simplifique a máquina ao máximo antes de prosseguir.
3. Escolha uma codificação de estados. Ela pode ser arbitrária, desde que obedeça ao critério de corretude, conforme mencionado acima.
4. Gere uma implementação em diagrama de esquemáticos do circuito. Para tanto, use o modelo estrutural da Figura 1 deste Laboratório e implemente as funções combinacionais. Perguntas a responder:
    1. Quantos bits tem a sua codificação de estado (escolhida no item 3 acima)?
    2. Qual o número de entradas e saídas de cada uma das funções combinacionais (função de saída e função próximo estado)? Esta resposta determina quantas funções Booleanas deverão ser efetivamente implementadas e de quantas variáveis cada uma delas depende.
    3. Como se implementam estas funções em hardware?

Observação:

Para projetar a lógica da máquina de refrigerantes, várias funções devem ser implementadas em portas lógicas, tipicamente, uma para cada bit de saída, e cada uma destas dependendo de todas as entradas. Isto pode ser complexo. Uma forma de automatizar o processo de gerar estas portas lógicas, evitando a manipulação de muitos mapas de Karnaugh grandes, bem como uma outra forma de descrever máquinas de estados podem ser aprendidas através do Laboratório Extra disponível na homepage. Use-o caso queira aprender como simplificar seu trabalho aqui. Este laboratório extra é opcional, e não vale nota.

5. Simule o circuito, gerando um script de simulação realista, testando algumas condições, tais como tentativa de tirar refrigerante sem crédito suficiente, tentativa de solicitar devolução de moedas com crédito nulo, etc.
6. Responda: a máquina implementada é uma máquina de Mealy ou de Moore? Porquê?
7. Siga as instruções das Partes IV e V.
8. Responda às questões colocadas no meio do texto acima (**Partes IV e V**).
9. Suponha que você quisesse mostrar, a título de realimentação para o usuário, qual comando está sendo executado a cada instante pela máquina (C[2:0]), usando, por exemplo, os 3 leds mais à direita na placa. Modifique seu projeto para que isto ocorra e gere o novo hardware.
10. A entregar até a data limite deste laboratório:
    1. disquete contendo os projetos completos funcionando, contendo no mínimo, os itens abaixo
        1. script de simulação
        2. resultado da simulação realizada (forma de onda)
        3. diagrama de esquemáticos do projeto
    2. arquivo texto com as respostas às questões colocadas acima, bem como o desenvolvimento da implementação do hardware.

Observação Importantíssima:

Cada projeto criado no sistema de CAD Foundation e mais tarde no ambiente de simulação Active-HDL cria uma quantidade enorme de arquivos de forma automática. Para tornar possível a

correção adequada dos trabalhos, é indispensável que toda a hierarquia de projeto seja entregue ao professor. Para fazer isto de forma correta, os ambientes possuem um utilitário de arquivamento e compactação de projetos que deve ser usado. Quando o projeto estiver pronto para ser entregue, use a opção de menu **File →Archive Project...** do sistema Foundation (no sistema Active-HDL é a opção de menu **Design → Archive Design...**) para transformar toda a hierarquia de projeto num arquivo único com a denominação **<nome_do_seu_projeto>.zip**. Este arquivo deve ser entregue ao professor, quando se solicitar a entrega do projeto. A entrega de projetos incompletos ou em outro formato implicará a não avaliação do trabalho, com a conseqüente perda da nota do trabalho. Não use utilitários externos de compressão/descompressão, pois a hierarquia pode não ser salva de forma correta!!!

11. A mostrar em aula, assim que estiver pronto (até a mesma data limite do item anterior): a máquina de vender refrigerantes executando na placa.

No seu relatório coloque as dificuldades encontradas para realizar este estudo dirigido, tais como instruções pouco claras, erros de digitação, passos omitidos, etc.

Percentuais de nota atribuídos às Tarefas:

Item Avaliado	Percentual
Projeto, simulação e documentação (incluindo tabela de transição completa, codificação de estados e equações Booleanas)	**40%**
Implementação em hardware – com demonstração em aula	**30%**
Respostas às questões do texto - todas	**30%**