

PROJETO DE SISTEMAS DIGITAIS

Edson Midorikawa

emidorik@usp.br

Departamento de Engenharia de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo

Versão 1.3 (26/02/2012)

Objetivo: *Este documento apresenta uma metodologia de projeto estruturado de sistemas digitais, mostrando como particionar o projeto em módulos menores. Estratégias específicas para cada módulo são apresentadas, como por exemplo, projeto RTL (register transfer level) para o fluxo de dados e uso de diagramas ASM (algorithmic state machine) para o desenvolvimento da unidade de controle.*

SUMÁRIO

I. INTRODUÇÃO

II. PARTICIONAMENTO DE SISTEMAS DIGITAIS

III. Desenvolvimento de um Projeto

III.1. Projeto do Fluxo de Dados

III.2. Projeto da Unidade de Controle

III.3. Projeto do Sistema Digital

IV. Bibliografia

I. INTRODUÇÃO

Atualmente, os sistemas digitais estão amplamente difundidos em todos os lugares. Temos como exemplos de sistemas digitais desde os celulares que carregamos aonde formos, o forno de micro-ondas na cozinha de casa ou no escritório e os próprios *tablets*, *notebooks* ou computadores desktop onde editamos nossos relatórios, até outros exemplos mais “desconhecidos” como os microcontroladores que são responsáveis pelo sistema de controle de tração e pelos freios ABS de um carro moderno. Tais sistemas digitais são sistemas complexos e o desenvolvimento destes sistemas envolve uma série de atividades que devem seguir uma metodologia, que contém uma série de atividades de forma a tratar esta complexidade de forma sistemática.

Um sistema digital (SD) é um sistema com entradas e saídas, como qualquer outro sistema real (fig.1). O fato que diferencia um SD de outros diz respeito ao tipo de dados de entrada e saída que são manipulados: os dados são digitais, ou seja, são representados por um conjunto finito de sinais binários e discretos.



Figura 1 – Sistema digital geral.

Tradicionalmente, um SD era projetado como um sistema único e usando-se componentes discretos SSI e MSI com circuitos integrados TTL [Fregni & Saraiva, 1995] [Morris & Miller, 1978]. O desenvolvimento de circuitos digitais era baseado em um diagrama de circuitos conhecido como captura esquemática (fig.2).

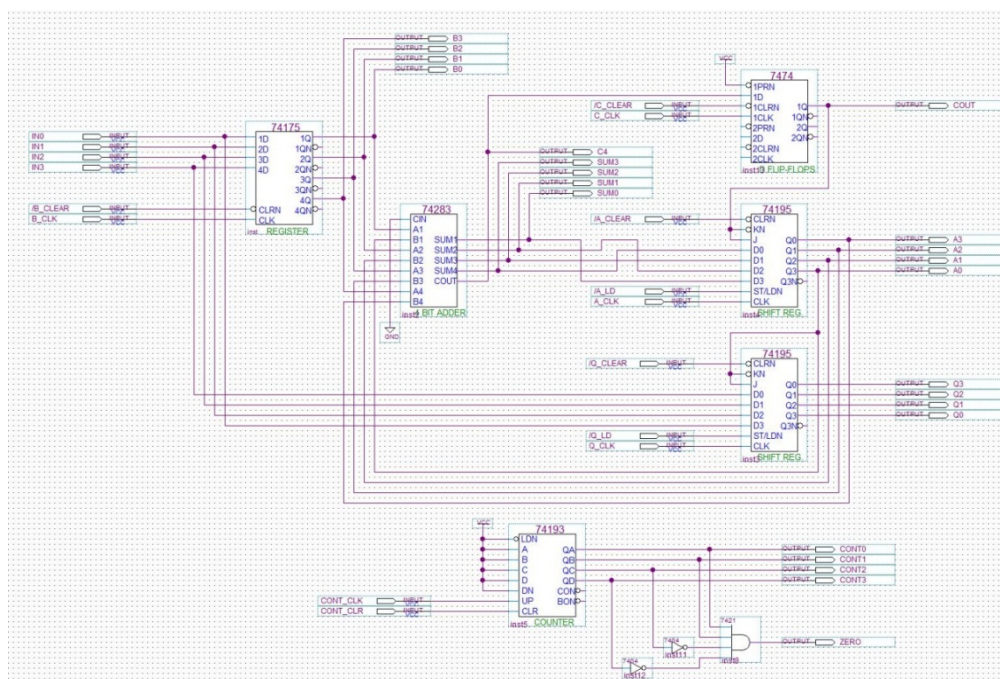


Figura 2 – Exemplo de projeto de circuito digital com captura esquemática.

Com a evolução e o aumento da complexidade de sistemas digitais, esta metodologia tradicional se mostrou limitada e novas estratégias tiveram de ser desenvolvidas. O projeto hierárquico foi uma forma de tratar a complexidade dos circuitos grandes, com a divisão do circuito em blocos ou módulos menores que são projetados em separado e depois interligados para compor o sistema completo (fig.3).

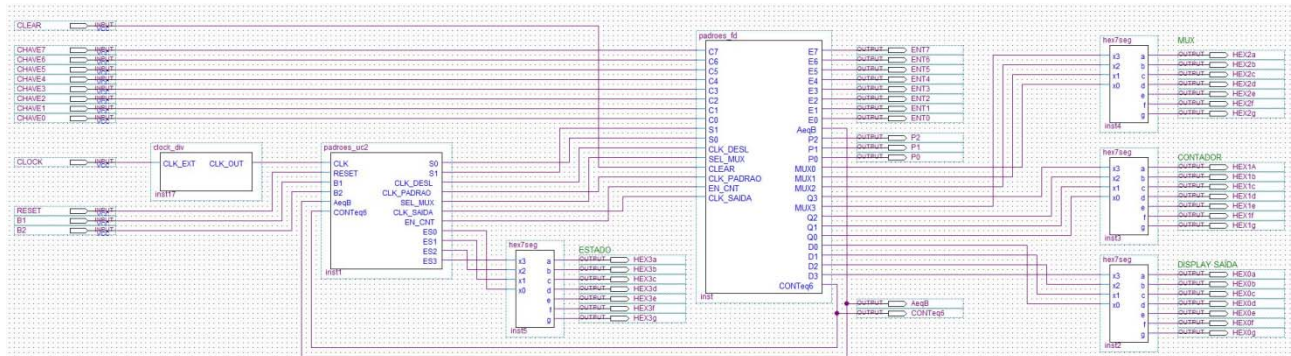


Figura 3 – Exemplo de projeto hierárquico de sistemas digitais.

Cada bloco em um projeto hierárquico é um subprojeto que também possui um diagrama de captura esquemática. Esta hierarquia pode conter vários níveis, até que a complexidade de um bloco seja adequada para um projeto tradicional usando componentes digitais básicos.

Apesar da definição de estratégias de particionamento de circuitos como forma de gerenciamento de complexidade, projetistas menos experientes mostram muita dificuldade para projetar circuitos muito complexos. Esta dificuldade se refere principalmente ao desenvolvimento e depuração de circuitos de controle e de sincronização destes sinais de controle. Torna-se importante então a aplicação de estratégias específicas para o projeto de circuitos digitais. Por exemplo, para o projeto de circuitos de controle, a literatura mostra uma série de ferramentas para a especificação, modelagem e desenvolvimento: diagramas de transição de estados [Wakerly, 2006], *statecharts* [Harel, 1987] e diagramas ASM [Givone, 2003] [Mano & Kime, 2000], entre outros. Estas ferramentas auxiliam o projetista no desenvolvimento de circuitos sequenciais, facilitando o teste e, depois, a correção do circuito projetado.

A aplicação de estratégias de auxílio ao projeto de sistemas digitais complexos é independente da forma como o circuito digital é desenvolvido. Antes da década de 1980 a captura esquemática foi muito difundida e ferramentas de software eram utilizadas pelos projetistas para o projeto e documentação, as chamadas ferramentas de CAD (*computer-aided design*) ou, como é conhecida mais atualmente, de EDA (*electronic design automation*).

Nos anos 80, a utilização de linguagens de descrição de hardware (HDL – *hardware description language*) se difundiu e começou a se tornar uma alternativa para o projeto de circuitos. As padronizações das linguagens VHDL em 1987 (IEEE Standard 1076-1987) e Verilog em 1995 (IEEE Standard 1364-1995) permitiram o uso destas linguagens de forma universal, desde a especificação de sistemas até a etapa de síntese de circuitos integrados. Atualmente várias outras linguagens de descrição de hardware estão disponíveis, incluindo várias características, inclusive extensões para suporte a sinais analógicos (*analog and mixed-signal extensions*), como por

exemplo, o VHDL-AMS, Verilog-A e Verilog-AMS. Para suporte a projetos em alto nível, ou seja, em nível de sistema (*system-level modelling*), tem-se a linguagem SystemC (IEEE Standard 1666-2005). Já a linguagem SystemVerilog (IEEE Standard 1800-2005) inclui suporte a verificação de projetos.

As seções a seguir apresentam uma metodologia de projeto com a apresentação de estratégias específicas. Estas estratégias são independentes da forma como o circuito é desenvolvido, seja na forma de esquemáticos ou com uso de HDLs. Para ilustrar os conceitos, é desenvolvido um projeto exemplo.

II. PARTICIONAMENTO DE SISTEMAS DIGITAIS

Para gerenciar a complexidade do projeto de um sistema digital não trivial, convém adotar uma estratégia do tipo “dividir para conquistar” (*divide and conquer*), onde um sistema é organizado como um conjunto de subsistemas menores interligados entre si.

O projeto de um sistema digital pode ser organizado, em um primeiro nível, particionando o SD em duas partes com funções distintas: o Fluxo de dados (FD) e a Unidade de controle (UC). O fluxo de dados compreende a parte do circuito responsável pela manipulação, processamento, armazenamento e geração de dados. A unidade de controle é responsável pelo sequenciamento das operações executadas no fluxo de dados, de forma a garantir o correto funcionamento do circuito. A figura 4 mostra um diagrama com a estruturação interna detalhada do SD, onde são apresentadas as duas partes principais e os sinais internos para interconexão entre elas.

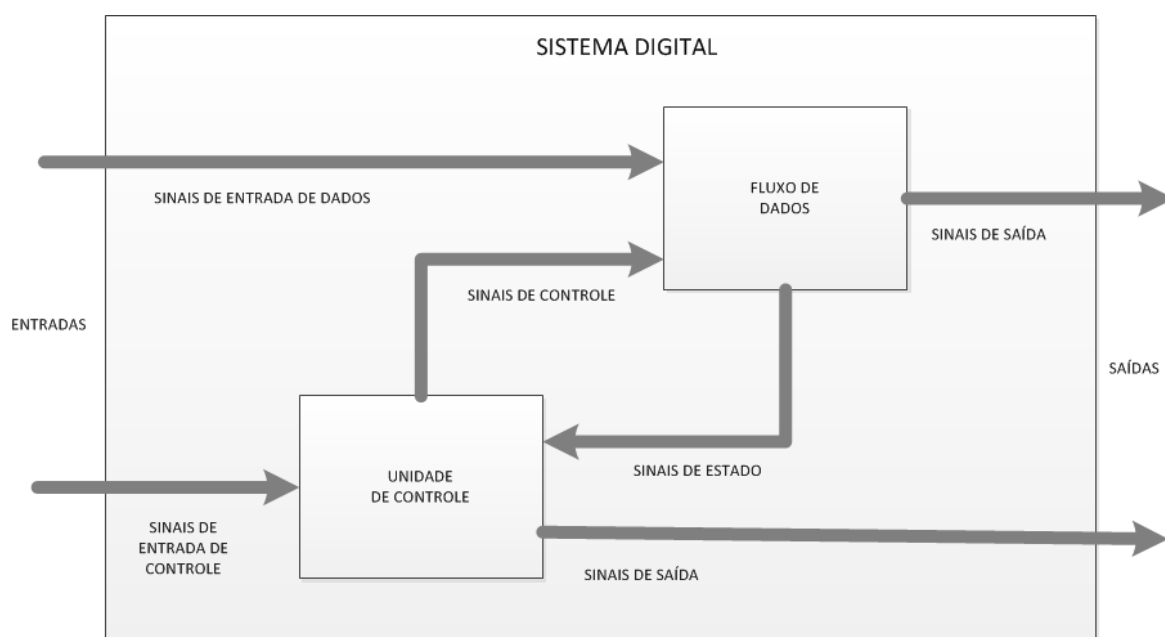


Figura 4 – Estruturação interna de um sistema digital.

Os sinais de entrada de dados são ligados ao fluxo de dados, que também gera sinais de saída. Os sinais de estado indicam o estado atual do fluxo de dados e incluem sinais como o valor atual de um determinado registrador, a saída de um comparador e a detecção de uma condição em particular.

Os sinais de estado do fluxo de dados e os sinais de entrada de controle são os responsáveis pela execução da unidade de controle. Normalmente a unidade de controle é modelada usando um circuito sequencial através de uma máquina de estados finitos. A saída da unidade de controle são sinais de controle para os componentes do fluxo de dados, como por exemplo, a seleção de função de uma ULA, a habilitação de um contador ou a habilitação de um circuito de memória RAM. Opcionalmente, a unidade de controle também pode gerar alguns sinais de saída. Por exemplo, é conveniente gerar como saída uma identificação do estado atual da máquina de estados finitos para ser usada para a depuração do circuito.

Esta organização pode ser aplicada em sistemas embarcados [Flynn & Luk, 2011], como por exemplo, em um sistema de controle automotivo, e em modernos processadores multicore [Patterson & Hennessy, 2009][Harris & Harris, 2007].

A figura 5 ilustra um diagrama com a estrutura de um processador. Nesta figura, os elementos em cor preta correspondem aos componentes do fluxo de dados e os de cor azul, aos componentes da unidade de controle. Note também os sinais de controle em azul que garantem o sequenciamento de acionamentos dos multiplexadores, comparadores, registradores, ULA e memórias, entre outros componentes.

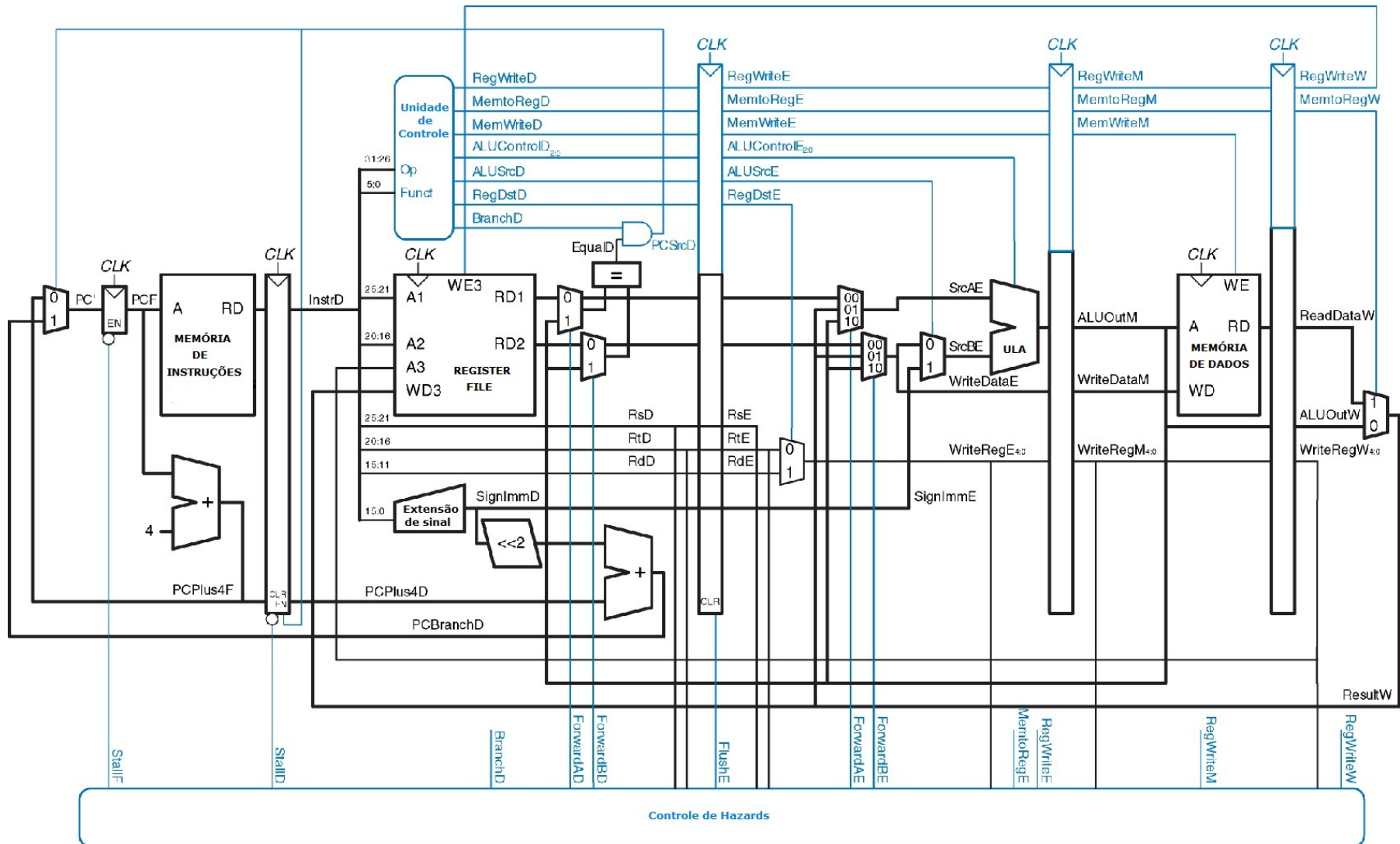


Figura 5 – Esquema da organização interna de um processador com suporte a *pipeline* e controle de *hazards*. Fonte: [Harris & Harris, 2007].

III. Desenvolvimento de um Projeto

Para ilustrar os conceitos apresentados, vamos desenvolver um sistema digital exemplo com as funcionalidades descritas a seguir. Conforme ilustrado na figura 6, o sistema digital exemplo tem cinco sinais de entrada¹ e um sinal de saída [Ranzini et al, 2002]:

- IN[1..4] – entrada de dados de quatro bits;
- N1 – armazena primeiro valor de quatro bits presente na entrada IN;
- N2 – armazena segundo valor de quatro bits presente na entrada IN;
- M1 – apresenta na saída de dados OUT o primeiro valor armazenado;
- M2 – apresenta na saída de dados OUT o segundo valor armazenado;
- OUT[1..4] – saída de dados de quatro bits.

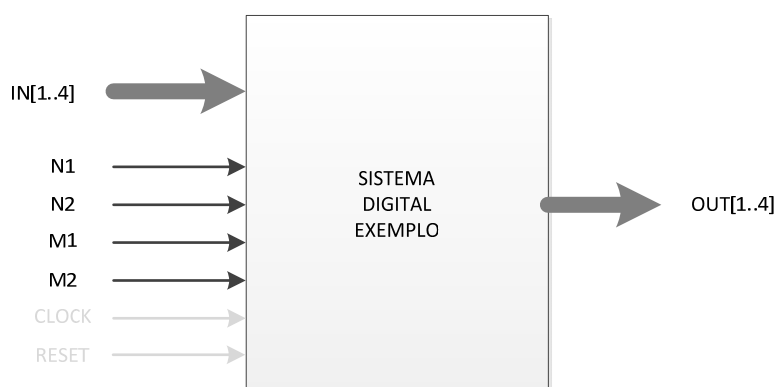


Figura 6 – Interface de entrada e saída do projeto exemplo.

O sistema digital armazena dois números internamente, que depois podem ser apresentados na saída OUT. Para que o primeiro número seja armazenado no sistema, ele deve ser colocado na entrada de dados IN e o sinal N1 deve ser ativado. De maneira análoga, o segundo número é armazenado quando o mesmo é colocado na entrada IN e o sinal N2 é ativado. Para mostrar cada um destes números na saída OUT, os sinais M1 ou M2 devem ser acionados. Caso seja acionado M1, a saída OUT deverá mostrar o primeiro número. Caso seja acionado M2, a saída OUT deverá mostrar o segundo número.

Convém ressaltar que todos os sinais são definidos como **ativo em alto**, ou seja, para acioná-lo, deve-se colocar um valor lógico ALTO ou 1.

¹ Na realidade, há dois sinais de entrada adicionais no sistema digital: o sinal CLOCK, que normalmente não é mencionado porque pode-se considerar que todo sistema digital síncrono utiliza um sinal de relógio externo para seu funcionamento, e o sinal RESET, que faz com que o sistema reinicie seu funcionamento interno.

A figura 7 abaixo mostra os resultados da simulação de uma implementação do projeto exemplo. Nas formas de onda mostradas, inicialmente é armazenado o valor 6 (acionando N1) e em seguida o valor 7 (acionando N2). Ao acionar M1, o primeiro valor é apresentado na saída. Depois, ao acionar M2, o segundo valor armazenado é mostrado. No final, o acionamento de RESET faz o sistema voltar ao estado inicial.

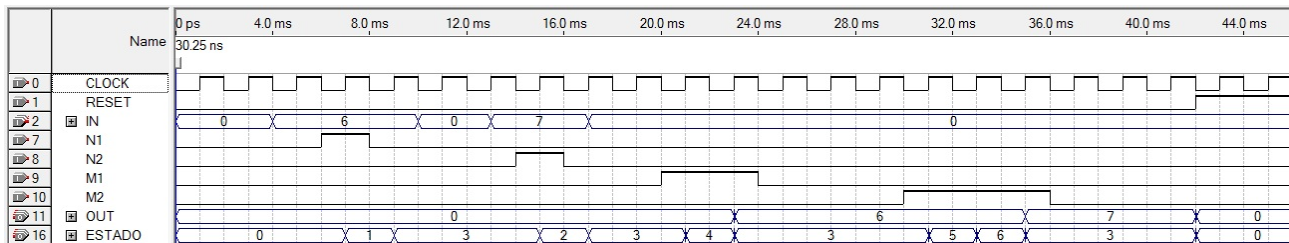


Figura 7 – Simulação do sistema digital do projeto exemplo.

O projeto do sistema digital exemplo é iniciado o particionamento do circuito com a definição do fluxo de dados e da unidade de controle (figura 8).

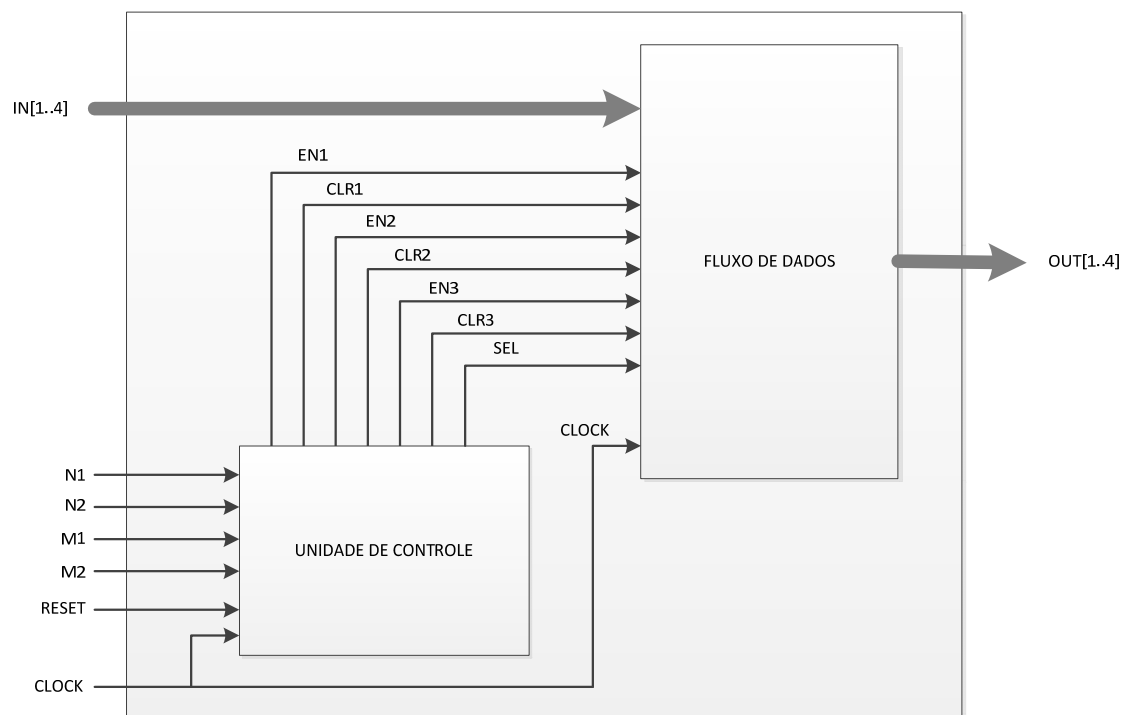


Figura 8 – Particionamento do projeto exemplo.

Na figura 8 temos a definição dos sinais de entrada de dados (sinal IN) e dos sinais de entrada de controle (sinais N1, N2, M1, M2 e RESET). Os sinais de controle são responsáveis pelo controle dos registradores internos (sinais EN1, CLR1, EN2, CLR2, EN3 e CLR3) e do multiplexador (sinal SEL) do fluxo de dados.

Neste projeto em particular, não há nenhum sinal de estado do fluxo de dados.

A seguir, nas seções seguintes, detalhamos o projeto do fluxo de dados e da unidade de controle.

III.1. PROJETO DO FLUXO DE DADOS

O fluxo de dados deve ser projetado como um circuito digital no nível de transferência de registradores. Um sistema é dito estar no nível de transferência de registradores (RTL – *register transfer level*) se as informações fluírem pelo circuito através de componentes de memória ou registro de dados (registradores). À medida que os dados fluem pelo circuito, estes podem ser manipulados por blocos combinatórios implementando uma determinada lógica (fig.9).

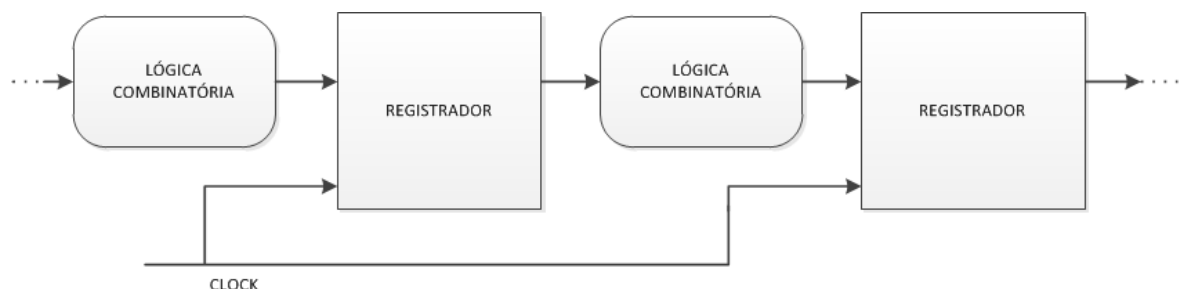


Figura 9 – Circuito no nível RTL.

O circuito descrito na figura 9 é basicamente um **circuito sequencial síncrono**, onde um sinal global de relógio (*clock*) gerencia o fluxo de informações pelo sistema digital. O bloco de lógica combinatória pode incluir um circuito com portas lógicas ou outros componentes mais complexos, como, multiplexadores, decodificadores, ULAs, etc. Já os módulos registradores incluem registradores, propriamente ditos, ou simplesmente *flip-flops* ou ainda registradores de deslocamento.

A identificação dos componentes do fluxo de dados pode ser feita a partir do algoritmo ou descrição detalhada do funcionamento do circuito digital. A partir da identificação dos componentes (por exemplo, contador), pode-se verificar quais operações são executadas por estes componentes (por exemplo, zerar, incrementar, reiniciar, etc).

Um circuito no nível RTL é estruturado como um conjunto de caminhos para o fluxo de informações pelo sistema digital completo. A ordenação correta e o processamento específico são determinados pela unidade de controle que aciona os sinais de controle dos vários componentes combinatórios do fluxo de dados (por exemplo, função da ULA) e também dos elementos registradores (por exemplo, habilitação da saída *tri-state* de um registrador).

Os recursos disponíveis no fluxo de dados implementam funções diferentes no hardware [de Micheli, 1994], que podem ser classificados em:

- Os **recursos funcionais** processam dados. Eles implementam funções aritméticas ou lógicas e podem ser agrupadas em duas subclasses. A primeira são os recursos primitivos que foram projetados cuidadosamente uma vez e usados frequentemente. São exemplos desta classe as unidades lógicas e aritméticas e as funções lógicas padrão, como os codificadores e decodificadores. A segunda classe inclui recursos específicos da aplicação, pois incorporam circuitos que executam uma tarefa particular. Um exemplo desta classe é o circuito de tratamento de interrupções de um processador.

- Os **recursos de memória** armazenam dados. São exemplos os registradores e as memórias EPROM e RAM. O requisito para armazenar informação é importante no sequenciamento de operações de um sistema digital.
- Os **recursos de interface** realizam a transferência de dados. Estes recursos incluem vias de dados que compõem um meio de comunicação essencial em um fluxo de dados. Interfaces com circuitos externos incluem ainda pinos de E/S e circuitos de interfaceamento.

O fluxo de dados do projeto exemplo pode ser estruturado conforme o diagrama de blocos da figura 10. Os sinais de entrada são ligados aos registradores R1 e R2. As saídas destes registradores são conectadas na entrada de um multiplexador. E a saída deste multiplexador é ligada na entrada do registrador R3.

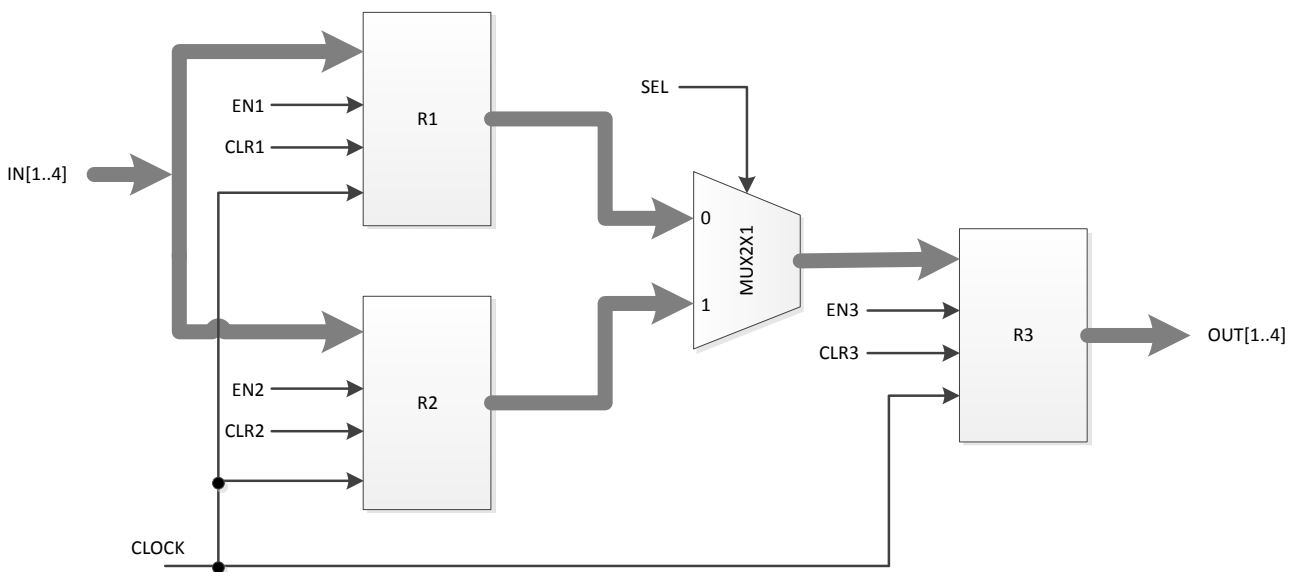


Figura 10 – Esquema do fluxo de dados do projeto exemplo.

Observe, assim, que os sinais de controle necessários para controlar a operação dos componentes do fluxo de dados são para o controle do armazenamento dos dados nos registradores (EN1, EN2 e EN3), para a reinicialização dos registradores (CLR1, CLR2 e CLR3) e para a seleção de entrada do multiplexador (SEL).

A implementação do fluxo de dados pode ser realizada usando componentes MSI da família 74xxx. Desta forma, considerando-se o circuito integrado 74173 para ser usado para os componentes R1, R2 e R3 e o circuito integrado 74157 para o componente MUX2x1, podemos desenvolver o diagrama esquemático da figura 11.

Observe que como o projeto considera todos os sinais de controle como ativos em alto, há a necessidade da inversão dos sinais de habilitação dos registradores (EN1, EN2 e EN3) para a controle dos componentes 74173.

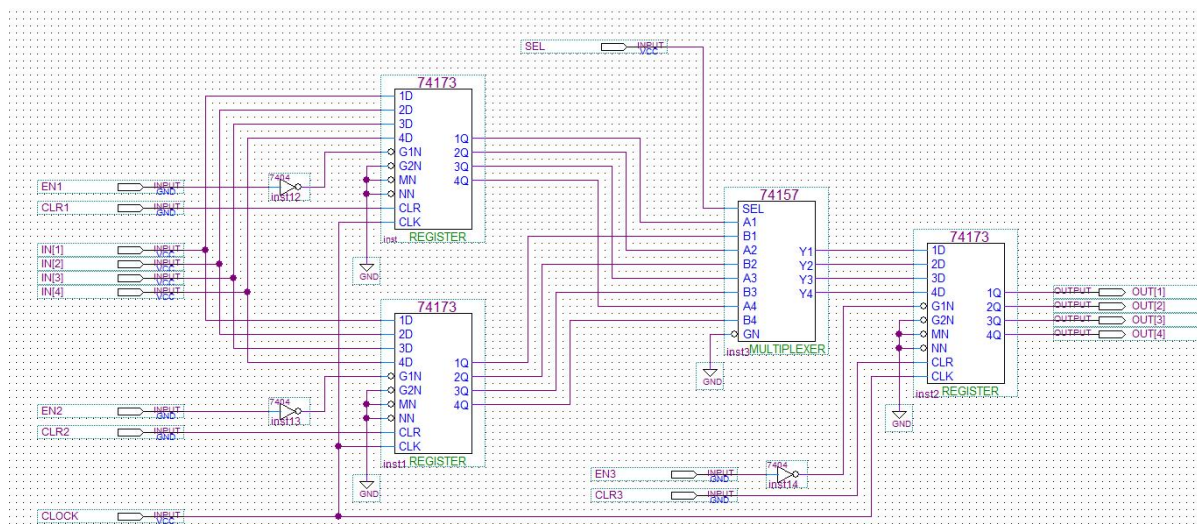


Figura 11 – Diagrama esquemático do fluxo de dados do projeto exemplo.

Depois de compilar o projeto do fluxo de dados, podemos gerar o símbolo do circuito no Altera Quartus II, conforme ilustrado na figura 12.

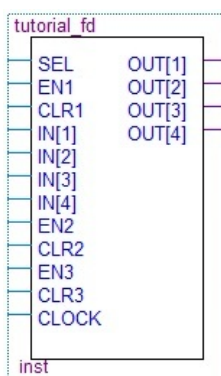


Figura 12 – Símbolo do componente correspondente ao projeto do fluxo de dados.

Uma vez desenvolvido o projeto do fluxo de dados, realiza-se a simulação do projeto. As formas de onda da figura 13 mostram uma possível simulação, com a entrada do valor 6 em R1 e do valor 7 em R2 e, posteriormente, a saída destes valores na saída. Ao final, realiza-se uma reinicialização do circuito. A simulação é conduzida com o acionamento correto dos sinais de controle.

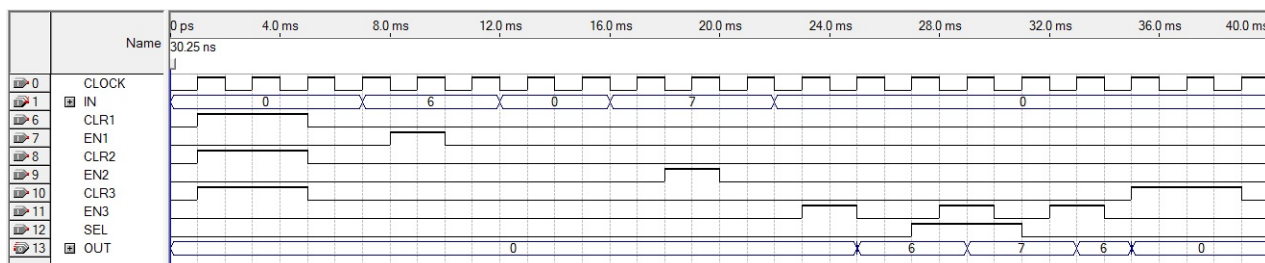


Figura 13 – Formas de onda da simulação do fluxo de dados.

Uma vez desenvolvido o fluxo de dados, é possível partir para o projeto do fluxo de dados, responsável pelo acionamento correto dos sinais de controle.

III.2. PROJETO DA UNIDADE DE CONTROLE

A unidade de controle deve organizar o funcionamento correto do sistema digital garantindo o correto sequenciamento de operações realizadas pelo fluxo de dados. Uma forma para desenvolver a unidade de controle é a partir de um diagrama de transição de estados ou do diagrama ASM correspondente.

III.2.1. Diagrama ASM

Um **diagrama ASM** (*algorithmic state machine*) é um diagrama similar à máquina de estados finita. Seus principais elementos são: [Givone, 2003] [Mano & Kime, 2000]

- bloco de estado,
- bloco de decisão e
- bloco de saída condicional.

O bloco de estado (fig.14) representa um estado no diagrama ASM [Givone, 2003]. Ele tem apenas uma entrada e apenas uma saída. Normalmente é atribuído um nome ao estado correspondente e dentro do símbolo são apresentados os comandos executados no estado. A entrada do bloco de estado pode vir de outro bloco de estado, de um bloco de decisão ou de um bloco de saída condicional. E a saída do bloco de estado pode ser ligada a outro bloco de estado ou bloco de decisão.

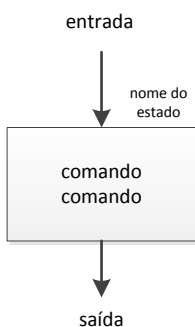


Figura 14 – Bloco de estado do diagrama ASM.

O bloco de condição (fig.15) apresenta a verificação de uma condição booleana e duas alternativas de próximo estado para continuação do fluxo de execução. A condição apresentada deve ser uma expressão booleana em função dos sinais de estado do fluxo de dados. Dependendo da avaliação desta expressão booleana o fluxo de execução prossegue na saída com condição verdadeira ou na saída com condição falsa. A entrada do bloco de decisão pode vir de um bloco de estado ou outro bloco de decisão. E cada saída pode ser ligada a outro bloco de decisão, bloco de estado ou bloco de saída condicional.

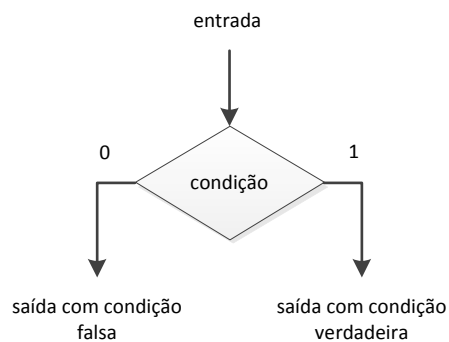


Figura 15 – Bloco de condição do diagrama ASM.

O componente final do diagrama ASM é o bloco de saída condicional. O símbolo apresentado na figura 16 apresenta o símbolo deste bloco, que contém uma entrada e uma saída. Normalmente a entrada do bloco de saída condicional vem de um bloco de condição, e é usado em modelos de máquina de estados do tipo Mealy. Dentro do símbolo é apresentado um conjunto de comandos a serem executados na transição de estados.

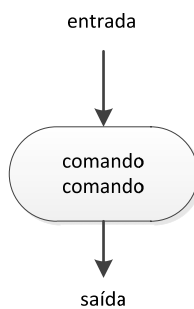


Figura 16 – Bloco de saída condicional do diagrama ASM.

A seguir apresentamos um diagrama ASM que modela o circuito da unidade de controle do projeto exemplo (figura 17). Inicialmente podemos desenvolver um diagrama ASM com comandos em alto nível, sem o detalhamento dos sinais internos da interface com o fluxo de dados.

Uma vez que o diagrama ASM da unidade de controle for desenvolvido e o fluxo de dados for definido, pode ser gerado um diagrama ASM mais detalhado, com a especificação direta dos sinais de controle e dos sinais de estado. Neste caso, os blocos de estado contêm os sinais de controle que devem ser ativados no estado correspondente e os blocos de condição testam o valor do sinal de estado. A figura 18 mostra o diagrama ASM detalhado da unidade de controle do projeto exemplo.

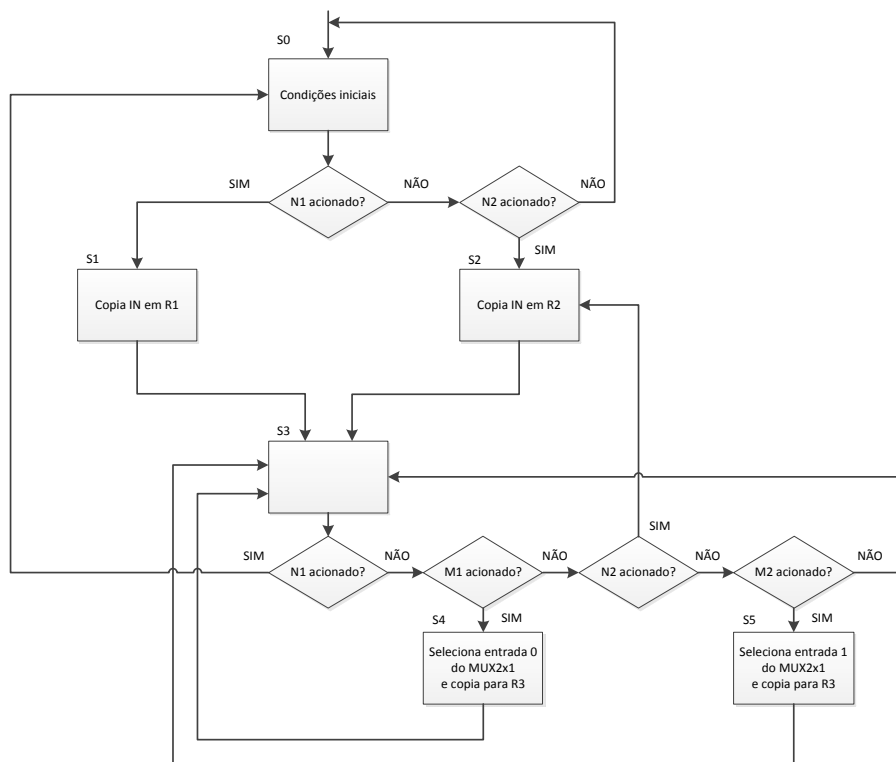


Figura 17 – Diagrama ASM de alto nível da unidade de controle do projeto exemplo.

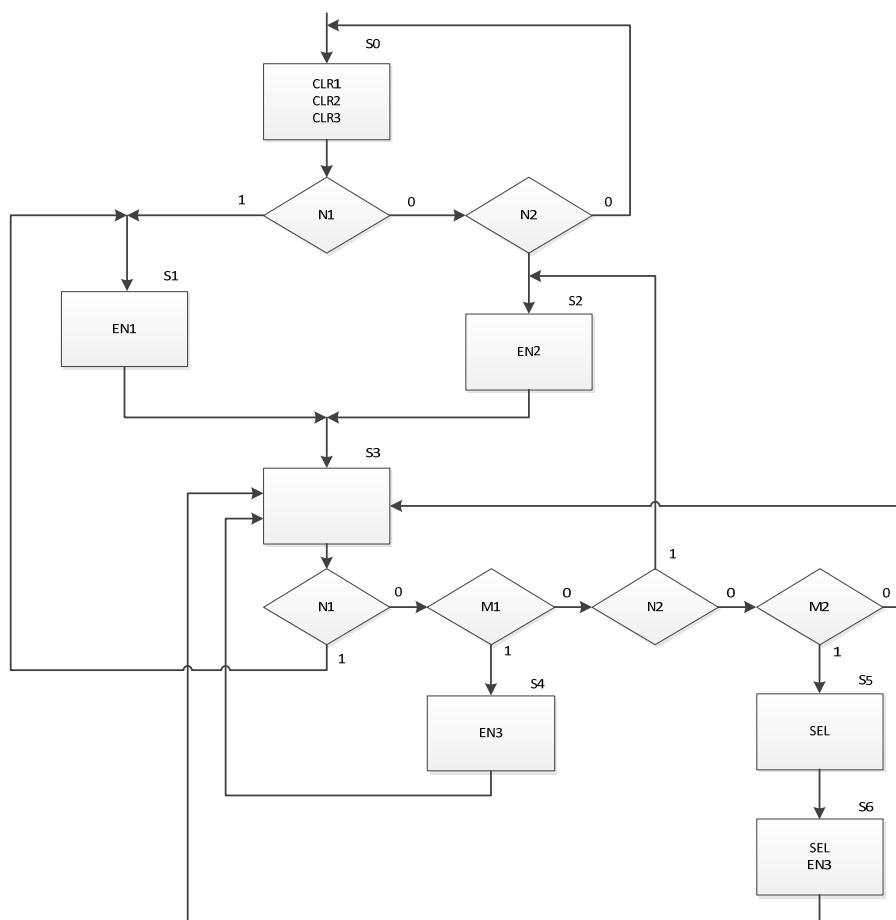


Figura 18 – Diagrama ASM do projeto exemplo.

Note que o estado S5 do primeiro diagrama ASM precisou ser dividido em dois estados no diagrama ASM detalhado, por causa dos requisitos de tempo do registrador R3 (tempo de preparação ou *setup time*). Assim, antes de copiar o dado para o registrador R3, é necessário acrescentar um estado anterior para selecionar a entrada do multiplexador.

III.2.2. Alternativas de Implementação da Unidade de Controle

A implementação da unidade de controle pode ser realizada de várias formas. A primeira alternativa é usar a **linguagem de descrição de hardware AHDL** (Altera HDL), como mostrado na figura 19.

```

SUBDESIGN tutorial_uc
(
  CLKUC: INPUT;
  RES: INPUT;
  N1,N2,M1,M2: INPUT;
  CLR1,CLR2,CLR3,EN1,EN2,EN3,SEL: OUTPUT;
  E1,E2,E3: OUTPUT;
)

VARIABLE
UC: MACHINE OF BITS (CLR1,CLR2,CLR3,EN1,EN2,EN3,SEL,E3,E2,E1)
  WITH STATES
  % Estado      Saida      %
  % atual        %          %
  (
    s0=          B"1110000000",
    s1=          B"0001000001",
    s2=          B"0000100010",
    s3=          B"0000000011",
    s4=          B"0000010100",
    s5=          B"0000001101",
    s6=          B"0000011110"
  );

BEGIN
  UC.CLK = CLKUC;
  UC.RESET = RES;

  TABLE
  %      Estado      Entradas      =>      Próximo %
  %      atual        %          estado %
  UC,
  s0,    0, 0, X, X    =>    UC;
  s0,    1, X, X, X    =>    s0;
  s0,    0, 1, X, X    =>    s2;
  s1,    X, X, X, X    =>    s3;
  s2,    X, X, X, X    =>    s3;
  s3,    0, 0, 0, 0    =>    s3;
  s3,    1, X, X, X    =>    s1;
  s3,    0, 1, X, X    =>    s2;
  s3,    0, 0, 1, X    =>    s4;
  s3,    0, 0, 0, 1    =>    s5;
  s4,    X, X, X, X    =>    s3;
  s5,    X, X, X, X    =>    s6;
  s6,    X, X, X, X    =>    s3;
  END TABLE;
END;

```

Figura 19 – Código AHDL da unidade de controle.

Convém destacar que, além dos sinais de controle e de estado do projeto exemplo, foram acrescentados três sinais de saída, E3, E2 e E1. Estes sinais identificam o estado atual da máquina de estados e são importantes no acompanhamento da execução da unidade de controle e devem ser usados na depuração do circuito.

A linguagem AHDL permite várias alternativas para designar os estados [RANZINI et al., 2002]. Adotaremos a designação de cada estado pelo valor das saídas da UC, que devem ser geradas em cada estado. Por exemplo, o estado S0 será caracterizado como sendo aquele em que: CLR1=1, CLR2=1, CLR3=1, EN1=0, EN2=0, EN3=0, SEL=0, E3=0, E2=0 e E1=0.

Uma vez caracterizados os estados, é necessário descrever as transições entre os mesmos, ou seja, descrever o diagrama ASM através de uma tabela de transições. Também há alternativas, em função da estratégia adotada na construção do ASM (Mealy ou Moore). Adotou-se aqui a máquina de Moore.

As construções desta linguagem formam as seções do arquivo texto, a saber:

- a) denominação do projeto – nesta seção, iniciada pela palavra **SUBDESIGN**, é especificado o nome do projeto, que deverá ser o mesmo declarado no início da descrição da UC.
- b) declaração de entradas e saídas – nesta seção, delimitada por parênteses, são declarados todos os sinais de entrada e de saída do diagrama. As declarações de entradas são finalizadas por **:INPUT;** e as de saída por **:OUTPUT;**.
- c) denominação dos estados – nesta seção, iniciada pela palavra **VARIABLE**, os estados do diagrama ASM são definidos em função das saídas da UC. Primeiramente deve ser descrita a ordem dos bits que definirão cada estado. Isto é feito através da seguinte sentença:

UC: MACHINE OF BITS (.....) WITH STATES

Entre parênteses devem ser colocados os nomes das saídas do diagrama. Por exemplo, para o diagrama da figura 19:

(CLR1,CLR2,CLR3,EN1, EN2, EN3,SEL,E3,E2,E1)

Em seguida, cada um dos estados deve ser descrito em função das saídas do diagrama, de acordo com a ordem estabelecida na construção anterior. Isto é feito da seguinte maneira:

(nome_do_estado = B'valores_das_saídas',);

Para a descrição da figura 19, o estado S0 seria descrito como **S0=B'111000000'**.

- d) declaração da máquina de estados – nesta seção, delimitada pelas palavras **BEGIN** e **END;** são descritos os dois sinais de controle da máquina de estados que representa o diagrama ASM (CLK e RESET) e a tabela de transições entre os estados. Esta tabela é feita baseada nos caminhos existentes entre cada um dos estados do mesmo, em função das entradas a eles associadas (caso existam), e deve ser delimitada pelas palavras **TABLE** e **END TABLE;**. Por exemplo, na figura 19, para que a UC passe do estado S0 para o estado S1, é necessário que N1=1 e independente do valor de N2, M1 e M2. Portanto, a linha que representa este caminho é :

S0, 1,X,X,X => S1;

Outra forma para a implementação da unidade de controle é através do uso de um circuito com componentes digitais básicos com uso de um **diagrama de captura esquemática**. A figura 20 apresenta uma tabela de conversão dos blocos do diagrama ASM para o circuito digital correspondente [Mano & Kime, 2000].

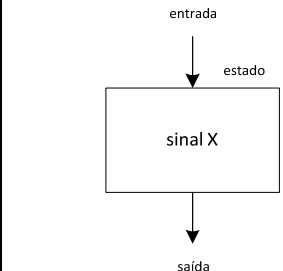
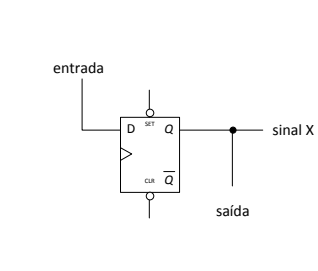
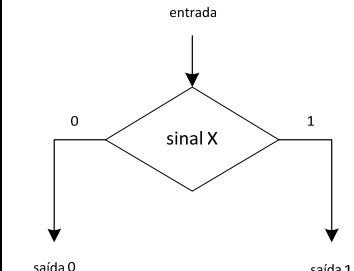
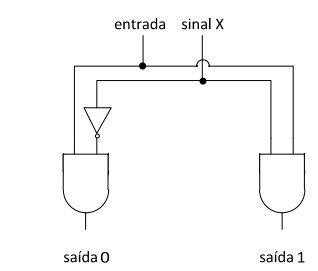
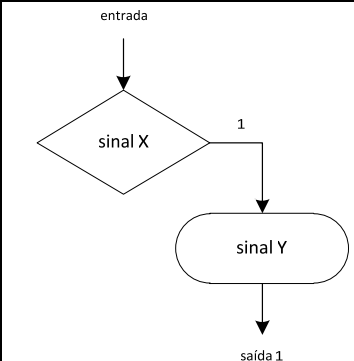
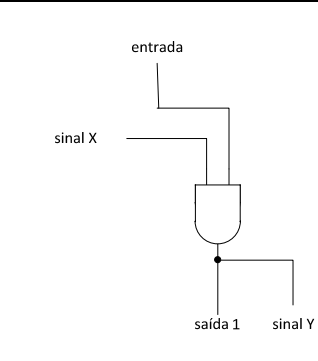
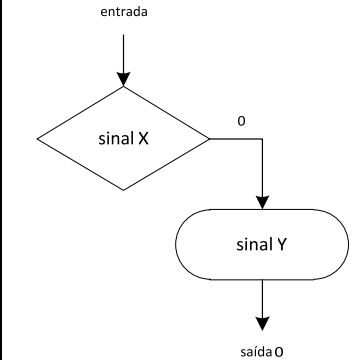
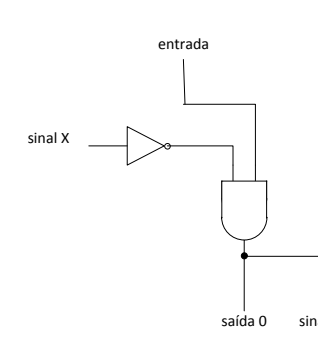
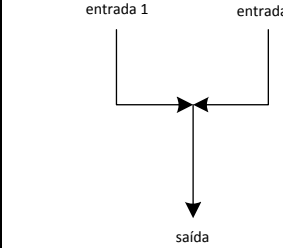
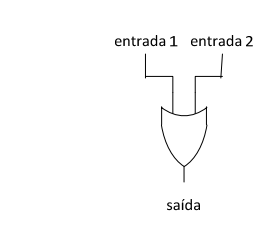
	Bloco ASM	Circuito correspondente
Estado		
Decisão		
Saída condicional		
Saída condicional		
Junção		

Figura 20 – Tabela de conversão de bloco ASM para circuito digital.

A figura 21 apresenta o resultado da conversão do diagrama ASM da figura 18 em um circuito digital usando a tabela da figura 20.

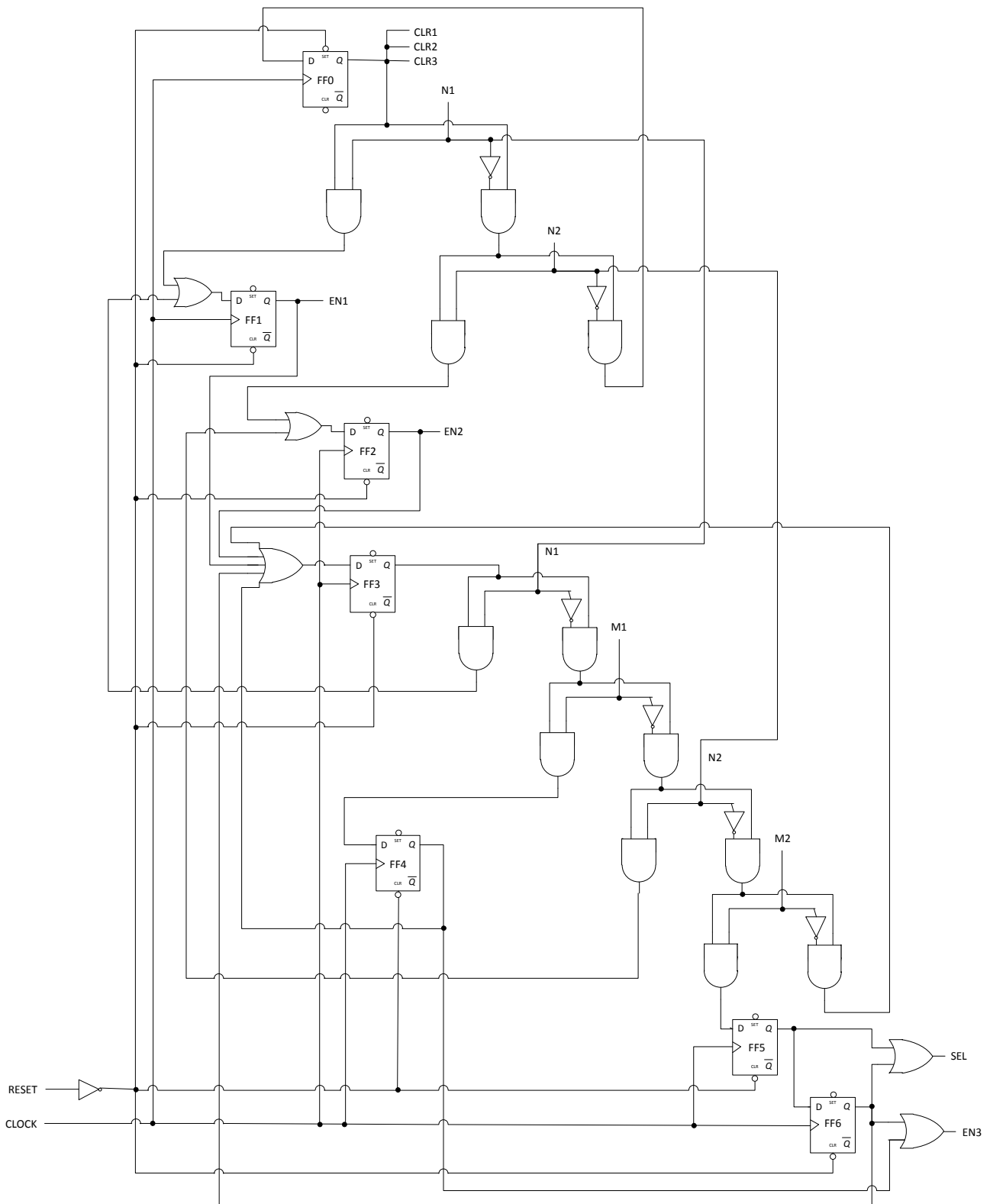


Figura 21 – Circuito da unidade de controle do projeto exemplo projetado a partir da conversão do diagrama ASM.

Convém ressaltar o sinal RESET do circuito, que ativa somente o *flip-flop* FF0 correspondente ao estado inicial do diagrama ASM. Os outros *flip-flops* são zerados. Esta codificação de estados corresponde a chamada codificação um *flip-flop* por estado ou *one-hot*.

A terceira forma de implementação da unidade de controle é definir o circuito que implementa o **diagrama de transição de estados**. Seja o diagrama de transição de estados da figura 22.

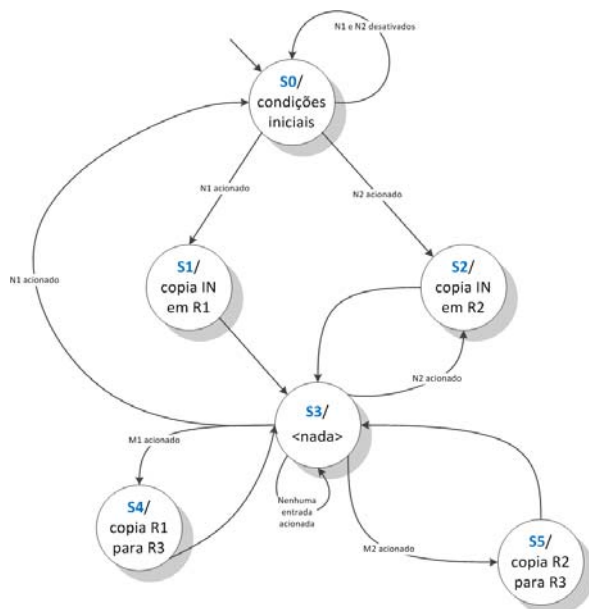


Figura 22 – Diagrama de transição de estados da unidade de alto nível.

Da mesma forma que o diagrama ASM, podemos refinar o diagrama de transição de estados a ponto de especificar os sinais de saída de cada estado e as condições com base nos valores dos sinais de entrada. A figura 23 abaixo mostra o diagrama de transição de estados assim obtido.

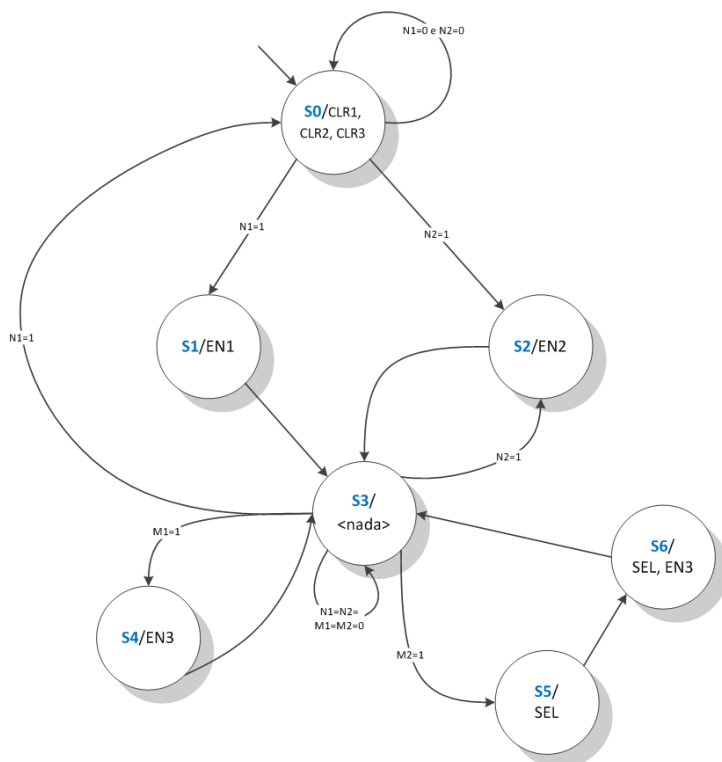


Figura 23 – Diagrama de transição de estados da unidade de controle do projeto exemplo.

Aqui cabe detalhar o fato que também que foi necessário incluir um estado adicional (estado S6) devido a estrutura do módulo fluxo de dados. Devido ao fato da entrada do registrador R3 ser proveniente da saída de um multiplexador, precisamos selecionar uma das entradas deste componente antes de acionar a operação de carga do registrador.

A implementação desta máquina de estados pode ser realizada na forma de um circuito gerado a partir da metodologia de projeto de circuitos sequenciais, como apresentado em [Wakerly, 2006], ou usando uma linguagem de descrição de hardware, como o VHDL ou Verilog.

Seja a forma que for escolhida para a implementação da unidade de controle, se for obedecida a interface especificada e sua funcionalidade, a interligação com o fluxo de dados deve ser realizada sem problemas.

Sob o ponto de vista de depuração, é importante a sugestão de adicionar na saída alguns sinais que indicam o estado atual da unidade de controle. Além de possibilitar acompanhar seu funcionamento, pode ser usado também para detectar erros no sequenciamento de estados.

Ao finalizar a implementação do circuito da unidade de controle no software Altera Quartus II, o símbolo do módulo da figura 24 deve ser gerado.

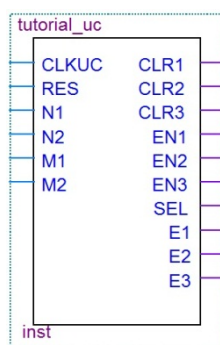


Figura 24 – Símbolo gerado para a unidade de controle.

Em seguida, a correção do circuito deve ser verificada com a realização de simulações do projeto da unidade de controle. A figura 25 mostra a saída de uma possível simulação.

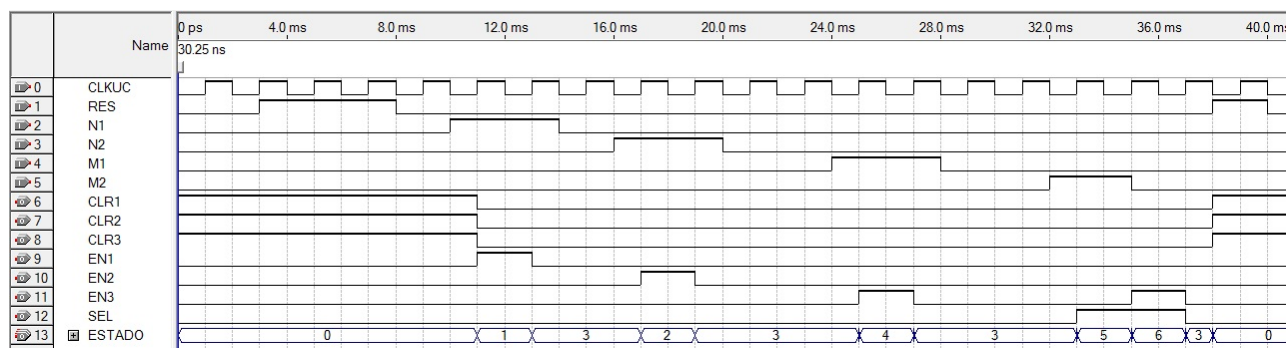


Figura 25 – Simulação da unidade de controle do projeto exemplo.

Da figura 25 pode ser observada a correta geração dos sinais de controle a partir das entradas N1, N2, M1 e M2. A visualização do estado atual da máquina de estados auxilia bastante no entendimento do funcionamento do projeto desenvolvido.

III.3. PROJETO DO SISTEMA DIGITAL

Terminado o desenvolvimento dos módulos fluxo de dados e unidade de controle do projeto do sistema digital, resta a integração destas partes para concluir o sistema.

No nosso projeto exemplo, a partir dos projetos das duas partes, podemos criar um projeto adicional, chamado `tutorial_sd` no Altera Quartus II, conectando os símbolos do fluxo de dados e da unidade de controle, conforme mostrado na figura 26.

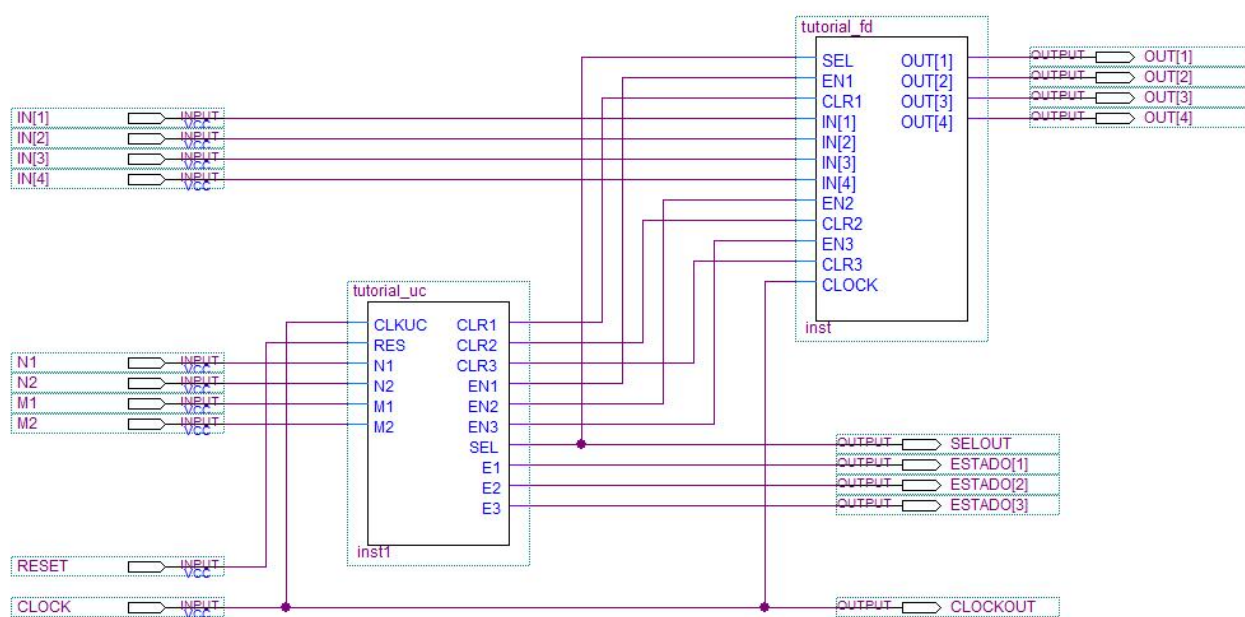


Figura 26 – Projeto do sistema digital a partir do fluxo de dados e da unidade de controle.

A simulação do projeto completo desenvolvido foi apresentada na figura 7.

IV. BIBLIOGRAFIA

- [Chu, 2006] CHU, P.P. **RTL Hardware Design Using VHDL: coding for efficiency, portability, and scalability**. Wiley, 2006.
- [de Micheli, 1994] DE MICHELI, G. **Synthesis and Optimization of Digital Circuits**. McGraw-Hill, 1994.
- [Flynn & Luk, 2011] FLYNN, M.J. & LUK, W. **Computer System Design: System-on-Chip**, Wiley, 2011.
- [Fregni & Saraiva, 1995] FREGNI, E. & SARAIVA, A.M. **Engenharia do Projeto Lógico Digital: conceitos e prática**. Edgard Blücher, 1995.
- [Givone, 2003] GIVONE, D.D. **Digital Principles and Design**. McGraw-Hill, 2003.
- [Harel, 1987] HAREL, D. *Statecharts: a Visual Formalism for Complex Systems*. **Science of Computer Programming**, vol. 8, pp. 231-274, 1987.
- [Harris & Harris, 2007] HARRIS, D.M. & HARRIS, S.L. **Digital Design and Computer Architecture**. Morgan Kaufmann, 2007.
- [Lala, 2007] LALA, P.K. **Principles of Modern Digital Design**. Wiley, 2007.
- [Mano & Kime, 2000] MANO, M. M.; KIME, C. R. **Logic and computer design fundamentals**. 2nd edition, Prentice-Hall, 2000.
- [Morris & Miller, 1978] MORRIS, R.L. & MILLER, J.R. **Projeto com Circuitos Integrados TTL**. Editora Guanabara Dois, 1978.
- [Patterson & Hennessy, 2009] PATTERSON, D.A. & HENNESSY, J.L. **Computer Organization and Design: the hardware/software interface**. 4th edition, Morgan Kaufmann, 2009.
- [Ranzini et al., 2002] RANZINI, E.; HORTA, E.L.; MIDORIKAWA, E.T. **Projeto de circuitos com MAX+PLUS II**. Apostila de Laboratório Digital, 2002.
- [Wakerly, 2006] WAKERLY, J.F. **Digital Design: Principles and Practices**. 4th edition, Prentice-Hall, 2006.