Metodologia de projeto no nível de transferência entre registradores RTL

Atualmente, a tendência dentro de sistemas digitais é empregar abordagens com um nível maior de abstração para a construção de sistemas digitais complexos. Um sistema digital é um conjunto de vários componentes que coordenam os dispositivos de entrada e saída (I/O), além da memória para a realização do processamento de algum dado. Um sistema de digital pode ser entendido como a união do Hardware (processador principal, memória, dispositivos de I/O e barramentos) e do Software (programa que usa os componentes do hardware). Como é mostrado na Figura 1, os sistemas digitais podem ser agrupados em três principais famílias:

- Lógica padrão: engloba os componentes digitais mais básicos (portas lógicas, flip-flops, decodificadores) que estão disponíveis como circuitos integrados, alguns mais de 30 anos. Eles já vêm sendo usados há anos para projetar sistemas digitais complexos. Eles possuem diversos encapsulamentos devido a diversidade de tipos existentes. O processo de manipulação é complexo e envolve um alto custo. O hardware já vem pronto de fábrica com uma função específica. A ULSI é um exemplo de empresa que lida com este tipo de projeto.
- Circuitos integrados de aplicação específica (ASICs - *Application Specific Integrated Circuits*): É a solução moderna em termos de hardware para os sistemas digitais. São circuitos integrados projetados para implementar uma aplicação específica. São sistemas mais complexos, que demandam muito tempo de desenvolvimento e muitos recursos. Isso os torna sistemas mais caros, em contrapartida, o sistema resultante oferece um ótimo desempenho com um baixo consumo de energia. Exemplos são os PLDs (dispositivos lógicos programáveis), como o FPGA e até os PSoCs.
- Microcontroladores/Microprocessadores/processadores digitais de sinais (DSP - do inglês Digital Signal Processor): Diferente do projeto de sistemas digitais com componentes digitais básicos, esses dispositivos possuem vários tipos de blocos funcionais dentro de uma placa. Eles possuem um hardware fixo, mas com a flexibilidade de adequar o código a cada problema. O grande problema associados a esse tipo de dispositivo é a velocidade, uma vez que a solução de hardware para o seu projeto de sistema digital sempre será mais rápido do que uma solução de software.

O projeto no nível de transferência entre registradores (RTL - *register-transfer level*) ocupa um espaço importante no contexto de ASICs, principalmente nos PLDs. RTL é necessária, uma vez que, mesmos os potentes computadores que estão disponíveis no mercado, não são capazes de dado uma descrição de um circuito lógico em linguagem comum converter isso para um circuito lógico funcional (físico). A subfamília que mais se vale das linguagens

de descrição de hardware são os PLDs. A RTL pode ser entendida como uma forma de se projetar ou descrever um circuito síncrono e seu fluxo de dados.

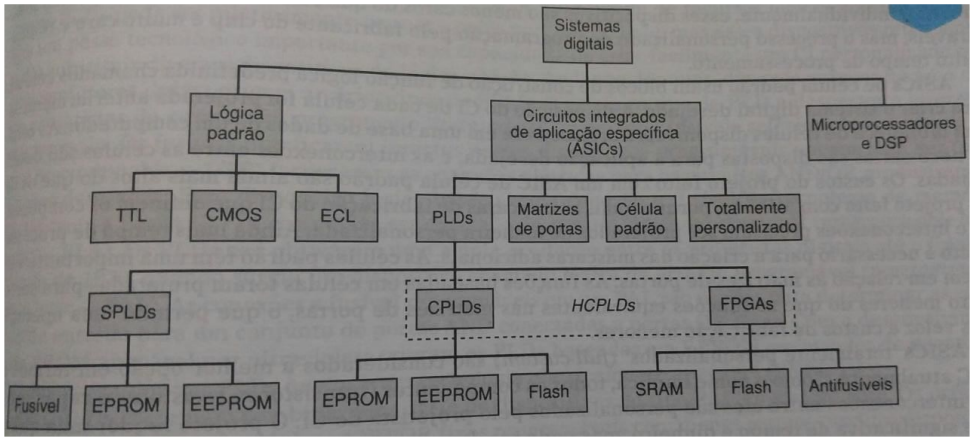
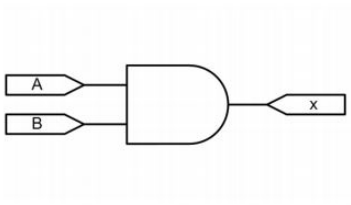


Figura 1: Famílias de sistemas digitais

O projeto em nível de transferência entre registradores, ou projeto RTL, é composto por uma grande variedade de abordagens. Todas as abordagens, geralmente, possuem uma semelhança, onde um projetista especifica os registradores de um circuito, descreve as possíveis transferências e operações a serem realizadas com os dados de entrada, de saída e dos registradores, além de definir o controle que especifica quando transferir e operar com os dados.

Geralmente, o projeto de circuitos de lógica padrão podem ser agrupados nos seguintes passos: (1) capturar o comportamento desejado para o circuito lógico, por meio de alguma abordagem; e (2) converter o comportamento em um circuito. Os circuitos de lógica padrão podem ser divididos em dois grupos: combinacionais (somente o valor da entrada impacta na saída) e sequenciais (o valor das entradas e o estado atual do componente definem a saída). Para esses dois tipos, a única alteração é a forma que captura-se o comportamento do circuito. Enquanto o primeiro captura por meio de uma tabela-verdade ou equação, o segundo usa uma máquina de estados finitos. O processo de projeto RTL segue os mesmos dois passos descritos para os circuitos de lógica padrão, a única alteração é a adição do conceito de máquina de estados de alto nível para capturar o comportamento do circuito RTL. Ela usa de uma lógica de alto nível para desenhar um circuito. O RTL é usado no processo de design lógico do sistema.

Pode-se usar uma linguagens de descrição de hardware (HDL, do inglês, Hardware Description Languages) para capturar o comportamento de um circuito e implementar uma RTL. HDL é usada para descrever a estrutura de sistemas eletrônicos e seu comportamento ao longo do tempo utilizando expressões padrão baseadas em texto. Ela utiliza sintaxe e semântica para deixar explícito notações que expressam simultaneidade, além de noções explícitas de tempo, a qual é um atributo primário do hardware. Ele é usado para escrever especificações executáveis para o hardware. A Figura 2 mostra um exemplo da criação da porta AND usando HDL.




```
1  ENTITY portas_2_1 IS
2    PORT ( a, b : IN BIT;
3                x : OUT BIT);
4  END portas_2_1;
5
6  ARCHITECTURE pand OF portas_2_1 IS
7  BEGIN
8    x <= a and b;
9  END pand;
```

Figura 2: Exemplo do uso de HDL

A criação da HDL e RTL visam fornecer ao projetista de hardware a capacidade de modelar, simular e testar uma peça de hardware antes mesmo que ela seja criada fisicamente. Isso é feito ao implementar a semântica subjacente das instruções de linguagens e simular o progresso de execução com o passar do tempo. Há HDLs direcionadas para suportar modelagem de eventos discretos (digital) e de tempo contínuo (analógico).

HDL visa expressar a conectividade do circuito entre uma hierarquia de blocos que estão adequadamente classificadas como uma linguagem de arquitetura do circuito (rede de ligações entre os elementos disponíveis na plataforma - *netlist*) independente de tecnologia e fabricante. Ela é utilizada no projeto auxiliado por computador (CAD - computer-aided design). O uso de HDL é focado em escrever especificações executáveis em um pedaço de hardware. Como as HDLs são independentes de tecnologia e fabricante, ela pode ser usada para troca de informações entre projetistas, fabricantes e fornecedores. Outro uso é a síntese de circuitos para CPLDs (*Complex Programmable Logic Device*) e FPGAs (*Field Programmable Gate Array*), além da criação layout e geração de máscaras para ASICs.

Além de descrever a arquitetura, a HDL pode ser usada para expressar projetos comportamentais e abstração de projetos RTL, o qual é um modelo de fluxo de um circuito ao longo do tempo. Nesses casos, o sintetizador (conversor de projeto de descrição de hardware para projeto físico) decide a arquitetura e layout das portas lógicas.

O sintetizador é uma ferramenta EDA (*Electronic design automation*), a qual é uma categoria de ferramentas de software para design de sistemas eletrônicos, como circuitos integrados e PLDs. Essas ferramentas geralmente traduzem um arquivo HDL para um ASIC ou um FPGA, além de realizar a otimização lógica do circuito.

De forma geral, o projeto RTL pode ser realizado de diversas formas, conforme dito. Isso porque ele é um processo criativo em sua maior parte. Todavia, a forma mais comum e popular segue os seguintes passos:

1. Obtenha uma máquina de estados de alto nível: Consiste em capturar o comportamento desejado do sistema na forma de uma máquina de estados de alto nível. A máquina é dita de alto nível pois as condições de transições e as ações dos estados são mais do que simplesmente operações booleanas envolvendo os bits de entrada e de saída.
2. Criação de um bloco operacional: Uma vez com a máquina de estados de alto nível criada no passo anterior, cria-se um bloco operacional capaz de realizar as operações que envolvem os dados.

3. Conecta-se o bloco operacional a um bloco de controle: O bloco operacional criado no passo anterior, conecta-se o bloco de controle, além das entradas e saídas booleanas.
4. Obtenha a máquina de estados finitos (FSM) do bloco de controle: A máquina de estados de alto nível é convertida na máquina de estados finitos do bloco de controle. Para que isso ocorra, as operações que envolvem dados são substituídas por sinais de controle, que são ativados pelo bloco de controle. Diferente da máquina de estados de alto nível, a FSM deve ter suas entradas e saídas restritas a tipos booleanos.

Além dos quatro passos citados, pode haver a introdução de um quinto passo, o qual envolve a escolha da frequência do relógio do sistema. Caso o desempenho seja peça fundamental do sistema, pode-se optar por frequências mais altas. O primeiro passo consiste na captura do comportamento do circuito, e os demais convertem esse comportamento em circuito. A implementação da FSM do bloco de controle pode ser feito na forma de um circuito sequencial.

Por exemplo, o projeto de uma máquina de refrigerante pode ser enquadrada nesses quatro passos. Antes mesmo do primeiro passo, primeiro inicia-se o processo de entender a máquina. A máquina de refrigerante deve possuir um detector de moedas, o qual detecta a "chegada" de uma moeda e o seu valor, além de outra entrada que indica o custo do refrigerante desejado. Em termos mais baixo nível, há três entradas: c (detecta uma moeda, 1 - moeda detectada, 0 - caso contrário); a (o valor da moeda detectada - pode ser 8 bits); e s (valor do refrigerante - pode ser de 8 bits). A saída da máquina consiste em liberar ou não o refrigerante, a qual pode ser descrita como uma variável binária d (1 - liberar o refrigerante, 0 - não faz nada). Não há troco. O primeiro passo parte desta descrição para obter-se o comportamento desejado do sistema. E consiste basicamente em gerar a máquina de estados, conforme mostra a Figura 2a. Do estado inicial só pode-se partir para o estado "esperar" e este espera até o valor fornecido ser maior ou igual ao total do refrigerante, conforme novas moedas entram ela vai para o estado somar e volta a esperar. Uma vez que o total fornecido é igual ao valor do refrigerante, muda-se de estado e vai para o "fornecer" e libera o refrigerante, após isso, volta-se para o início e zera-se todas as variáveis. Nesse caso não há somente estados booleanos, mas sim mais complexas.

No passo 2, o bloco operacional é criado pensando nas necessidades do circuito (Figura 2b). Por exemplo, para armazenar o valor de *tot* da Figura 2a, precisa-se de um registrador, um somador para realizar o cálculo *tot* + *a*, e por fim um comparador conectado a *tot* e *s* para realizar a verificação se *tot*<*s*. O próximo passo, consiste em conectar o bloco operacional a um bloco de controle, onde todas as entradas e saídas do bloco de controle são sinais de apenas um bit. Por fim, no último passo obtém-se o FSM do bloco de controle. Esse passo consiste em criar uma tabela de estados, a qual inclui um registrador de estado de dois bits e criar um circuito para as saídas do bloco operacional criado na Figura 2a. Os mesmos blocos criados no passo 2 são utilizados, contudo, há a adição do bloco operacional para realizar as operações. A Figura 2c ilustra esse contexto.

A escolha por máquina de estados de alto nível são usadas ao invés das FSM, se dá pelas suas vantagens. A FSM lida somente com entradas e saídas booleanas de somente um bit,

enquanto as máquinas de estados de alto nível oferecem uma entrada com mais bits e operações (atribuição, soma). Outro ponto é que as FSMs não permitem o armazenamento local de dados (o único dado armazenado é o valor do próprio estado). Além disso, as ações nos estados e as condições para as transições das máquinas de estados de alto nível envolvem operações com dados, ao passo que uma FSM permite somente equações expressões booleanas. Pode-se entender as máquinas de estados de alto nível, ou FSM com dados (FSMD), como uma extensão das FSMs. Há outros tipos de máquinas de estados, como as máquinas algorítmica de estados, ou ASM (*Algorithmic State Machine*) que se assemelham a fluxogramas com a inclusão da noção de tempo. Ela é mais restritas que as máquinas de estados comuns, uma vez que restringem as transições de tal forma que a computação se assemelhe um algoritmo (uma sequência ordenada de instruções). O uso das HDLs acabaram por substituir o de ASMs.

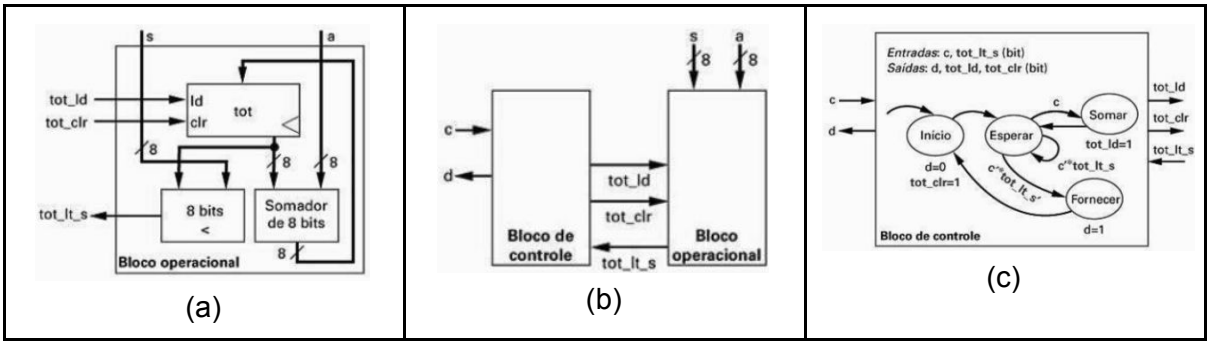


Figura 2: Etapas de projeto RTL

A máquina de estados de alto nível utiliza uma abstração muito grande para descrever o sistema. Esse tipo de visão não é condizente com FSM. Então o processo de criação de um bloco operacional visa implementar os armazenamentos de dados e computação envolvendo tipos de dados não booleanos. Isso permite a substituição da máquina de estados de alto nível por uma FSM, a qual controla apenas o bloco operacional. O processo de criação de um bloco operacional pode ser dividido em passos mais simples:

- Fazer com que os dados de entrada e saída se tornem entradas e saídas do bloco operacional.
- Acrescentar um registrador ao bloco operacional para cada função que exige armazenamento. Além disso, geralmente acrescenta-se um registrador para cada uma das saídas de dados.
- Examinar cada estado e cada transição, acrescentado e conectando novos componentes de bloco operacional para que as novas computações com dados sejam implementadas. Se caso for necessário, acrescenta-se multiplexadores à frente das entradas de cada componente (CPLDs e FPGAs). Ao fazer isso, um componente pode ser compartilhado entre diversos sinais que usam o mesmo componente em estados diferentes.

O passo de criação do bloco de controle é bem imediato. O processo consiste em criar-se um bloco de controle, com as entradas e as saídas booleanas do sistema, e em seguida conectá-lo às entradas e as saídas de controle do bloco operacional. O passo para a obtenção do FSM do bloco de controle também é direta, uma vez que o bloco operacional foi criado corretamente. Todas as computações presentes nas ações e condições são

substituídos por valores apropriados nos sinais de controle do bloco operacional. Dessa forma, a FSM terá os mesmos estados e transições que a máquina de estados de alto nível inicial, todavia com todas entradas, saídas e transições de apenas um bit.

Normalmente, o projeto RTL pode ser dividido em duas categorias: projetos com predomínio de controle ou com predomínio de dados. Ambas as categorias são muito mais descritivas, e não devem ser usadas rigorosamente para classificar os projetos. Pode haver projetos que englobam ambos os tipos. O projeto com predomínio de controle é onde o bloco de controle contém a maior parte da complexidade do projeto. O projetista destina a maior parte de seus esforços principalmente na definição do comportamentos dos estados do sistema, isto é, foca no bloco de controle. Uma vez que o projetista já definiu o comportamento, ele poderá obter imediatamente o bloco operacional a partir desse comportamento de estados. Um projeto com predomínio de controle responde tipicamente a entradas externas em um intervalo preciso de tempo, e geralmente tem um bloco operacional simples. O projeto com predomínio de dados é onde o bloco operacional contém a maior parte da complexidade do projeto. Nesse cenário, o projetista concentra-se principalmente no instanciamento e na interconexão dos componentes do bloco operacional. Uma vez que o projetista definiu o bloco operacional, é possível definir o comportamento dos estados do bloco de controle. Esse tipo de projeto, geralmente, envolve um grande paralelismo em seu bloco operacional. Os projetistas frequentemente desconsideram o primeiro passo do método de projeto RTL neste tipo de projeto.

De forma genérica, o resultado do projeto RTL é um processador, o qual possui um bloco operacional e um bloco de controle. Dentro de ambos, há registradores que necessitam de um sinal de clock. A frequência do clock determinará quão rápido o sistema irá executar as tarefas. Entretanto, o projetista não pode colocar uma frequência de clock arbitrariamente elevada. Os projetistas devem escolher a frequência de acordo com o “caminho” (sucessão de operações até terminar uma operação) crítico mais longo do circuito.

Conforme a tecnologia evolui em consonância com a complexidade dos sistemas digitais projetados, o comportamento dos sistemas digitais torna-se progressivamente mais difícil de ser compreendido. Pensando nisso, alguns projetistas utilizam linguagens de programação, como C, C++ ou Java, ou linguagens de descrição de alto nível, como VHDL ou Verilog, para primeiro obter uma descrição correta e de alto nível do comportamento do sistema desejado. Geralmente, todas essas linguagens podem ser simuladas para averiguar o comportamento, todavia, não podem ser sintetizadas (criar o sistema físico). Após esse passo, o projetista converte essa descrição em linguagem de programação ou HDL em um projeto RTL. O resultado da “conversão” de uma descrição de um sistema, feita com uma HDL, em RTL é conhecida como projeto em nível comportamental. É nesse momento que ocorre a captura do comportamento do circuito.

Atualmente, há diversas ferramentas que sintetizam linguagens HDL, como VHDL, Verilog ou SystemC (linguagens atualmente usadas), automaticamente. Essas ferramentas automatizadas seguem um método. A sintetização RTL de HDLs tendem a produzir estados extras, os quais podem e tendem a ser combinados em um passo de otimização. Cada comando (atribuição, if-then-else, laços de repetição) é convertido em uma máquina de estados, contudo, cada um com suas especificações.

Dentro do contexto de sintetização de uma HDL, o próximo passo após o RTL comportamental, é o RTL estrutural, o qual gera uma rede de ligações entre os elementos disponíveis na plataforma (*netlist*). Normalmente, o estados extras criados no passo de RTL comportamental tendem a gerar netlists maiores em área e mais lentas no desempenho quando comparado com esquemas tradicionais de layout de Sistemas Digitais. Geralmente, um projeto de projetista experiente usando captura esquemática tende a superar seu equivalente logicamente sintetizado. A vantagem dos circuitos logicamente sintetizados está relacionado a produtividade em áreas problemáticas para síntese RTL: circuitos de alta velocidade, baixa consumo de energia e assíncronos.

Considerando somente o contexto da RTL, a síntese passa pelo processo de criar o RTL comportamental e o estrutural, além do passo de converter o RTL comportamental para o RTL estrutural (netlist). A síntese envolve a conversão do circuito para módulos padrão primitivos, tais como somadores, comparadores, registradores, entre outros. Nesse ponto, a captura do comportamento do sistema já é feito pela HDL. O próximo passo envolve a síntese lógica do RTL estrutural e é onde entra a dependência da plataforma, já que se emprega o uso de portas lógicas, LUTs (*Look Up Table* - tecnologia de FPGA), flip-flops, entre outros. No caso do uso de HDL, consiste no processo de geração de uma descrição tipo netlist (lista de componentes e suas interligações) a partir de uma descrição feita com HDL, por exemplo, Verilog, VHDL e SystemC.

O TLM (*Transaction Level Model* - Modelo de Nível de Transição) é um tipo de modelagem usado para facilitar o processo de transição entre as especificações do sistema e os estágios do RTL. O projeto com TLM pode envolver modelos sem ligação com o tempo, o qual permite uma exploração da arquitetura com o hardware/software, e modelos temporais, os quais possibilitam uma indicação da performance do sistema. O modelo criado geralmente é uma descrição comportamental. Ele descreve o fluxo de dados e as operações necessárias.

Quando as HDLs começaram a ser usadas para síntese, houve um impulsionamento e elas que até então eram segundo plano para o design digital, entraram em evidência e com elas, projetos no nível de transferência entre registradores. A função das ferramentas de síntese é compilar os arquivos de origem HDL, em uma descrição de netlist fabricável em termos de portas e transistores.

O avanço das tecnologias fomentam a evolução e o uso de HDLs, os quais fazem com que PLDs evoluam, principalmente os FPGAs. Com a evolução desses componentes, sistemas baseados em SoCs também evoluem. E a partir destes, sistemas mais complexos e completos também surgem e evoluem, como os sistemas ciber-físicos (Cyber Physical Systems - CPS) e Internet-das-coisas. Os CPSs são a convergência entre computação, comunicação e controle, sendo a integração da computação com processos físicos, e são umas das bases da Internet-das-coisas. Acaba-se por utilizar os sistemas descritos com HDL e sintetizados com RTL para interagir com processos físicos, os quais adicionam novas propriedades em sistemas. Hoje em dia, os projetistas usam RTL e HDL para projeto de quase todos os circuitos integrados (CI). Isso mudou a forma de projeto de CIs, que, antigamente, partia da lógica padrão (*gate-oriented*).

O projeto no nível de transferência entre registradores vem ocupando um espaço importante dentro do cenário de sistemas digitais. Eles vêm possibilitando a criação de sistemas cada vez mais complexos, além de diminuir a complexidade de projeto de sistemas digitais, uma vez que utilizam um nível maior de abstração. Sistemas que antes eram complexos e difíceis de serem implementados, tiveram sua complexidade reduzida.

Disponível em: https://github.com/plsilva/Files_DigitalSystems

/home/pedro/Desktop/Concursos/UFOP-Professor/Sistemas Digitais - Projetos de Otimização e HDLs - VAHID.pdf
Seção 5 - Página 242
Seção 6.5 - Página 363
Seção 9.5 - Página 493

http://www.cl.cam.ac.uk/teaching/1011/SysOnChip/socdam-notes1011.pdf