PSI-3451 Projeto de CI Lógicos Integrados

Aula 10- Conceitos relacionados ao Fluxo de Projeto de ASICs

O objetivo desta aula é familiarizar o estudante com o fluxo de projeto de um sistema a ser implementado como circuito integrado de aplicação específica (ASIC). Para isso, a ferramenta de síntese, Leonardo Spectrum (Exemplar Logic, empresa do Grupo Mentor Graphics) é adotada na parte de exercício. São objetivos específicos desta experiência, que o aluno:

1. familiarize-se com a ferramenta Leonardo Spectrum;
2. siga e conheça o fluxo de projeto de um sistema digital a partir de uma descrição VHDL e terminando em uma lista de células com tecnologia definida e seus nós conectados (*netlist*);
3. conheça as alternativas de síntese de um circuito a partir de parâmetros definidos pelo projetista;
4. entenda a diferença em nível de abstração entre a representação comportamental inicial e a estrutural gerada.
5. verifique a correlação entre o código inicial VHDL e a implementação final com biblioteca de ASIC.

1. Fluxo de Projeto a partir de descrição RTL

--> Por que o projetista deve conhecer o fluxo de projeto? Quando qualquer pessoa planeja uma viagem ou, simplesmente, um caminhada em trilha, ela procura conhecer de antemão o trajeto de vai percorrer, os possíveis lugares que vai conhecer, as rotas alternativas, os recursos disponíveis para alimentação, etc. Desta forma, durante a realização desta viagem ou caminhada, tem-se uma noção correta de quanto foi percorrido, de quanto falta, com que recursos que pode contar ainda, etc. O mesmo ocorre para o projeto: conhecer o fluxo de projeto permite aumentar a eficiência do projeto, pois pode-se decidir pelas diversas opções e, sabendo em que ponto está dentro do fluxo, controlar a velocidade de execução das tarefas de forma mais consciente.

--> Em que implica o projeto de ASIC? Quando se usa a terminologia "ASIC", estamos entendendo que temos CIs que foram projetados e fabricados com propósito(s) específico(s), ou seja, não seriam destinados a muitos diferentes tipos de atividade. Implica em projetos curtos (menos de um ano) devido ao alto reuso de blocos pré-projetados e no projeto do leiaute do circuito a ser fabricado em silício. A fabricação é cara, mas tem-se a premissa de que um grande número de peças será fabricada, abatendo-se, assim, os custos fixos. Outras duas implementações são muito importantes, apresentando algumas diferenças em relação aos ASICs:

- circuitos dedicados de uso geral: assim como ASICs, o projeto de leiaute é realizado; entretanto, novos blocos básicos devem ser projetados minuciosamente para atender rígidas especificações de desempenho e área ocupada, aumentando o tempo de projeto. Exemplos são microprocessadores, memórias, etc.
- FPGAs: assim como ASICs, baseia-se no reuso para o projeto de circuitos de aplicação específica; entretanto, o projeto de leiaute não é necessário- com isto, o CI final é programado e não há necessidade de fabricação que é muito custoso. Usa-se esta estratégia quando um número menor de peças finais são necessárias.

Nos projetos de sistemas digitais modernos, a serem construídos como ASIC, a tarefa de síntese automática é uma das mais relevantes devido à alta complexidade dos circuitos envolvidos e das inúmeras possíveis soluções existentes para a construção do circuito final. Deseja-se sempre ganhos de eficiência em área ocupada ou velocidade do circuito, porém, o processo de otimização é muito demorado se ferramentas de CAD não estiverem presentes.

Devido à complexidade crescente dos sistemas digitais modernos, a metodologia de projeto *top-down* tem-se estabelecido como a forma mais apropriada para explorar o universo de soluções existentes no projeto de um determinado circuito. Esta metodologia baseia-se no refinamento de soluções a partir de descrições mais abstratas e genéricas (dirigindo-se progressivamente a soluções mais concretas e detalhadas). O uso de ferramentas automáticas são fundamentais devido à alta complexidade dos circuitos e à grande quantidade de elementos a serem tratados; entretanto, permite-se ao projetista guiar o processo através de restrições e especificações. A figura 1 apresenta de forma simplificada o fluxo de projeto de um sistema digital a partir de uma descrição comportamental (VHDL, por exemplo). A

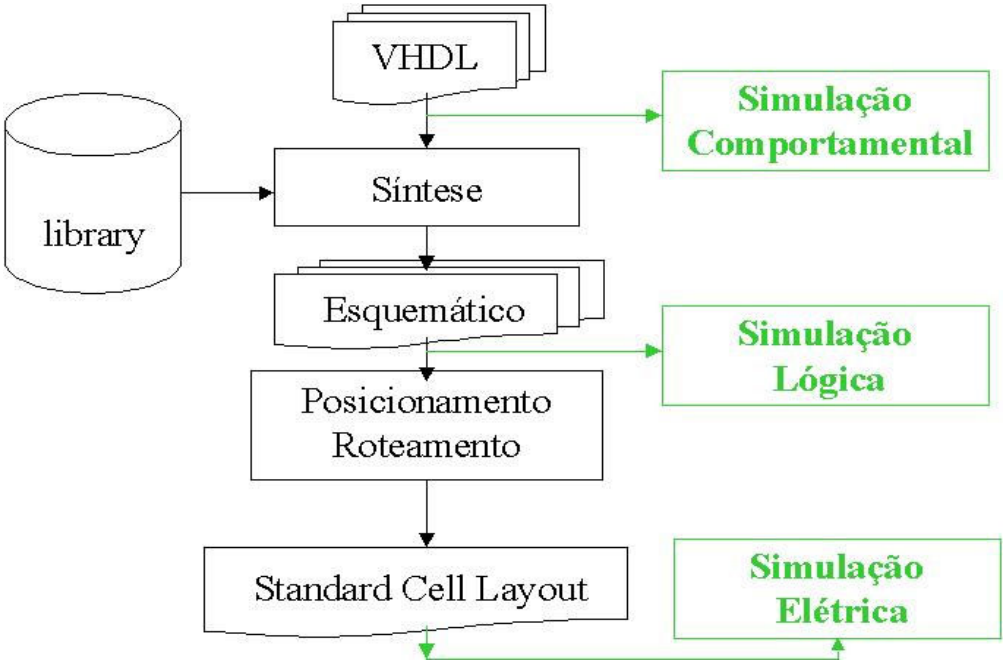


Figura 1. Fluxo de projeto (figura de Fernanda G de Lima, Inst. Inform. - UFRGS, 2001)

Duas etapas são evidenciadas: a) geração de um esquema elétrico (ou esquemático) em que uma lista de componentes (células, normalmente portas lógicas) com tecnologia definida são interligadas; b) a geração de um leiaute final a partir da descrição do item a), através das tarefas de posicionamento e roteamento. Nesta disciplina, o foco será no item a), conhecida como etapa de síntese lógica, cujos elementos principais são, de acordo com a figura 1:

- descrição comportamental: neste contexto, ela corresponde a de uma máquina de estados finitos (nível de transferência entre registradores, RTL). Significa que ou tratamos de um circuito combinatório, ou descrição de circuitos onde os passos de controle (clock) estão definidos explícita ou implicitamente;
- sistema de síntese lógica: algoritmos que manipulam a representação inicial e, em conjunto com uma biblioteca de células, otimizam a implementação do circuito em forma de um *netlist* de células contidas na biblioteca;

- biblioteca de células: conjunto de células contendo informações variadas para síntese e simulação. Uma célula construída para com uma determinada tecnologia pode apresentar descrição de função lógica, capacitância, tempos de atrasos, consumo de potência, leiaute, modelos de simulação (nível de portas, de chaves e de circuito), etc. Podem existir, portanto, células com mesma funcionalidade, porém diferenciados por outros parâmetros.
- simuladores: ferramentas de verificação da funcionalidade do circuito através de sua relação entrada-saída.

A síntese lógica consiste de duas sub-etapas, ambas quase que totalmente automatizadas: uma de otimização independente da tecnologia e, outra, de mapeamento tecnológico; ambas são descritas a seguir.

1.1 Síntese Lógica Independente de Tecnologia

De uma maneira geral pode-se definir a síntese lógica independente de tecnologia como a tarefa de gerar, a partir de uma descrição comportamental RTL, um *netlist* intermediário de componentes simbólicos, também conhecidos como **primitivas** que tenha a mesma funcionalidade lógica da descrição original. Portanto, não há necessidade de se saber, nesta etapa, qual é a tecnologia ou dispositivo alvo. Cada ferramenta de síntese realiza esta tarefa de acordo com as suas primitivas e seus algoritmos implementados. A partir destas primitivas, técnicas de mapeamento, também específicas da ferramenta utilizada, utiliza bibliotecas de células de tecnologia "reais", obtidas com as *foundries*.

Nos projetos de sistemas digitais modernos, a serem construídos como ASIC (também para FPGA), uma teoria sólida de álgebra Booleana e uma série de algoritmos poderosos são utilizadas para que a tarefa acima seja cumprida com sucesso. Pode-se dividir a etapa de síntese lógica independente da tecnologia nas seguintes subtarefas principais, como indicado na figura 2 (veja que são tarefas já conhecidas pelo(a) aluno(a) em cursos básicos de lógica digital:

- captura da máquina de estados finitos: a partir de uma descrição RTL (VHDL, por exemplo) uma máquina de estados simbólica é extraída e otimizada (figura 3.a), minimizando o número de estados e, portanto, o número de registradores (flip-flops);
- assinalamento ou designação de estados: corresponde à tarefa de definir, dentro de critérios pré-estabelecidos, uma codificação em bits otimizada para os estados da máquina de estados finitos- critérios de (redução de) área são os mais comuns. Tem-se, costumeiramente, como resultado desta etapa a geração de uma máquina de Huffman (figura 3.b) que é um módulo digital composto de um conjunto de equações Booleanas e registradores (flip-flops) que servem para sincronizar o circuito e definir os estados;
- minimização lógica multinível: A minimização lógica consiste, basicamente, na decomposição das equações Booleanas do circuito em uma rede Booleana multinível (figura3.c), onde, cada nó contém uma equação Booleana de poucas entradas. Novamente critérios de área são as mais relevantes na criação de uma rede Booleana otimizada. Nesta etapa, os registradores já gerados são mantidos.
- mapeamento a células primitivas: o objetivo desta etapa é a preparação do circuito para o mapeamento tecnológico posterior (próxima seção). Para isso, as equações dos nós da rede Booleana multinível são mapeadas a descrição de portas lógicas da biblioteca de **primitivas**. Uma forma bastante comum é a realização de um circuito só com NANDs (e os registradores) (figura 3.d), apesar de não ser utilizada pelo ***Leonardo Spectrum***.

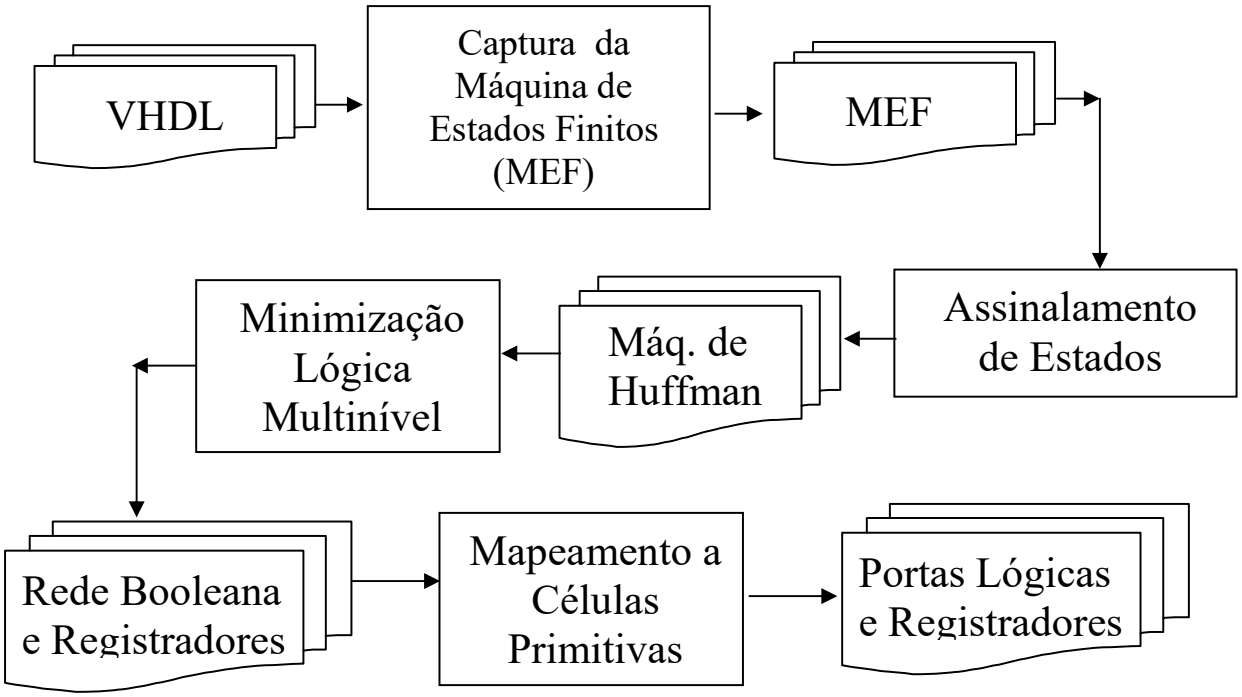


Figura 2. Síntese Independente de Tecnologia

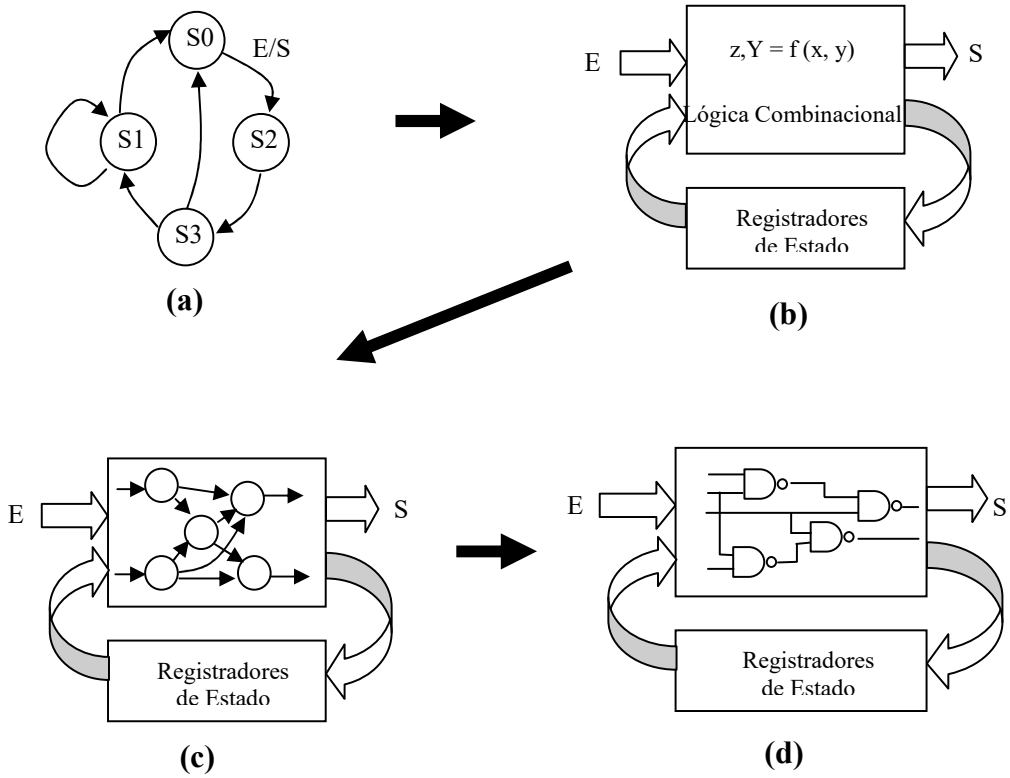


Figura 3. Os formatos intermediários da síntese independente de tecnologia

1.2 Mapeamento Tecnológico

O mapeamento tecnológico pode ser definido como a tarefa de gerar, a partir de uma rede de primitivas, um *netlist* de componentes com tecnologia definida (por exemplo, como usaremos na parte prática, células construídas com tecnologia CMOS de 0,35 micron de uma determinada *foundry*). Quando se faz o mapeamento tecnológico, pelo menos duas restrições são abordadas: o tempo de atraso permitido para a lógica do circuito, ou seja, a lógica entre a saída de um e a entrada de outro (ou do mesmo) flip-flop, ou entre uma entrada e um flip-flop, ou entre um flip-flop e uma saída do circuito. Este tempo determina a frequência máxima de operação do circuito. A outra restrição é a área ocupada, que, normalmente, é levada em conta quando a restrição de tempo já é possível de ser satisfeita. Em outras palavras, o atraso é um requisito que deve ser atendido para o valor estipulado enquanto a redução de área é sempre a máxima. Outros requisitos como consumo de potência pode ser importante para certos tipos de aplicação.

O mapeamento tecnológico é realizado com uma biblioteca de células das mais variadas de um ***design kit*** (que contém uma ou mais tecnologias). Os ***design kits*** devem conter informações de área, capacitância, parâmetros para cálculo de tempo de atraso e consumo de potência. Além disso, é desejável que modelos de simulação (com especificação de tempos de atraso) acompanhem o pacote para que o comportamento do circuito possa ser verificado (veja a figura 4).

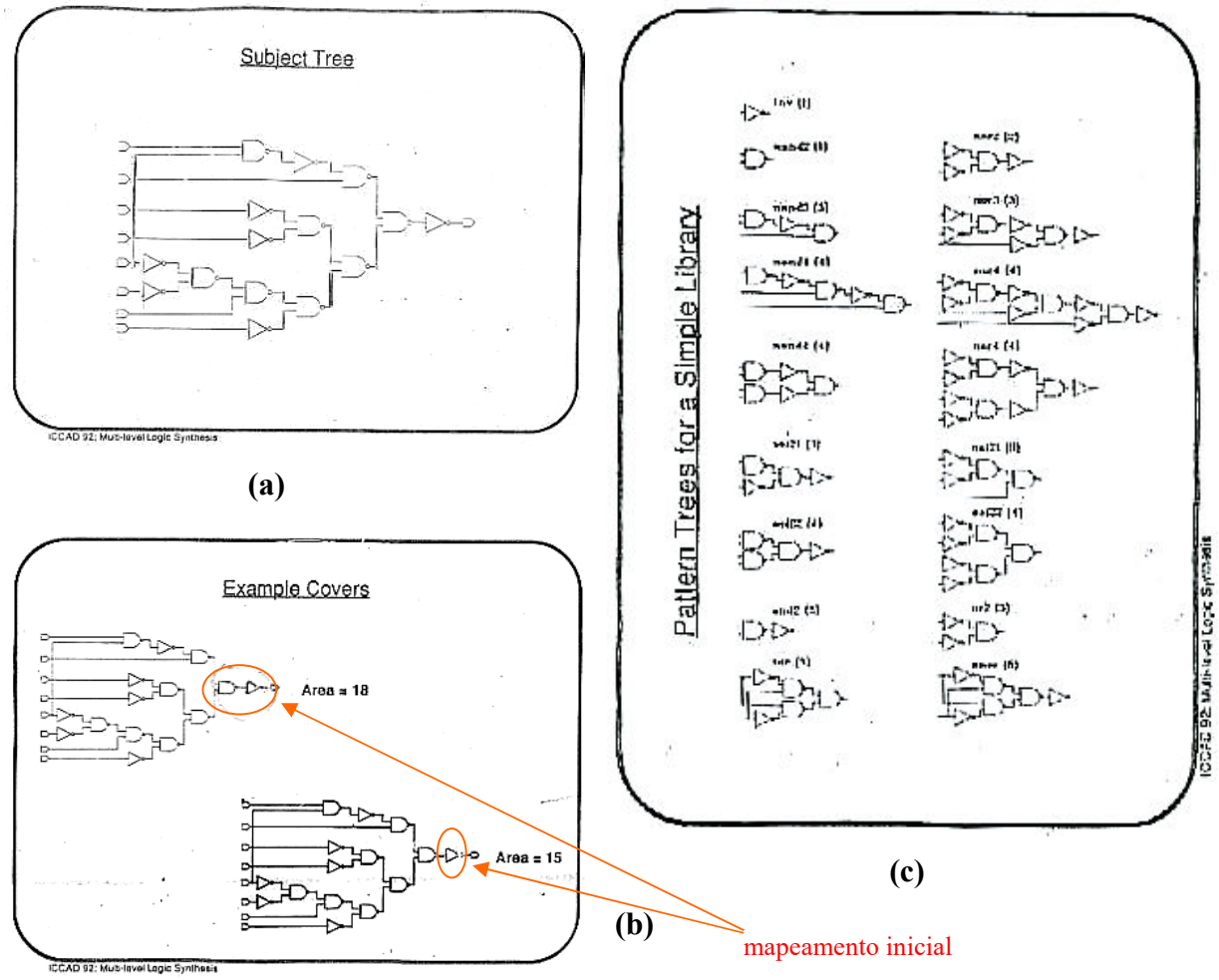


Figura 4. Mapeamento tecnológico

Tipicamente, a entrada das ferramentas de mapeamento tecnológico é um esquema (*netlist*) de células simbólicas (primitivas). As ferramentas de mapeamento tecnológico incorporam algoritmos que fazem o mapeamento de configurações destas primitivas para possíveis configurações de células da biblioteca tecnológica. A principal estratégia algorítmica existente é a programação dinâmica através da qual o espaço de soluções é explorado adequadamente. A figura 4 apresenta um pequeno exemplo com dois possíveis mapeamentos topológicos de um mesmo circuito. A figura 4.a mostra o circuito original já descrito exclusivamente por primitivas do tipo NAND (lembrar que inversor é uma NAND cujas entradas estão em curto). A figura 4.b mostra duas possíveis coberturas do esquema e suas respectivas áreas. A figura 4.c mostra uma biblioteca simplificada de células, sendo que cada uma delas é também descrita através de redes equivalentes de NANDs (para fins de mapeamento).

2. O Leonardo Spectrum

Atenção: esta seção é apenas uma breve apresentação da ferramenta ***Leonardo Spectrum***. Para maiores detalhes, o aluno deve utilizar a documentação que acompanha o software.

O Leonardo Spectrum é um software proprietário da Exemplar Logic, subsidiária da Mentor Graphics. O Leonardo é uma ferramenta de síntese lógica que permite realizar a primeira parte do fluxo de projeto descrito na figura 1, se usado em conjunto com um simulador VHDL (ModelSim, por exemplo). Se utilizado para implementação em FPGAs, as tarefas de posicionamento e roteamento podem ser também realizadas. Para isso, o dispositivo FPGA para a qual o circuito será mapeado deve ser escolhido como alvo de tecnologia. O fluxo de projeto do Leonardo Spectrum para implementação em ASIC segue o fluxo descrito na figura 5, sendo que cada etapa será detalhada a seguir. Observe que, sendo uma ferramenta comercial, várias etapas intermediárias apresentadas na seção anterior não são visíveis ao usuário. Este tem acesso apenas a "pontos de pilotagem", ou seja, definição de uma série de restrições com as quais os algoritmos do software buscam soluções.

- Definição de condições de operação: Inicialmente, o usuário deve definir variáveis de temperatura e tensão de alimentação. Estas variáveis são utilizadas para o cálculo de um fator de escalamento que ajusta os tempos de atraso das células das bibliotecas.
- Definição de condições do verificador de regras de projeto (*design rule checker*): Não confundir com o DRC do editor de leiaute. Algumas condições específicas que se verificam em uma linha como o seu máximo fanout, a máxima capacitância, e o máximo tempo de subida ou descida, são definidas. O DRC verifica se o circuito atende a estas restrições em uma otimização ao final do fluxo de projeto e, em caso negativo, o DRC aplica replicação de lógica, inserção de buffers, etc. para resolver o problema.
- Carregamento de bibliotecas: Inicialmente uma das bibliotecas de tecnologia que o software reconhece é carregada. O *design kit* que adotamos para esta experiência é ASIC Design Kit (ADK) fornecido pela Mentor Graphics dentro do seu programa universitário. Este *design kit* contém cinco bibliotecas de células com tecnologia diferenciada. Duas da AMI com 1,2 e 0,5 um e três da TSMC com 0,35um, 0,25um e 0,18um respectivamente. É relevante citar que esta biblioteca é utilizada na etapa de mapeamento tecnológico.
- Definição do estilo da síntese da máquina de estados finitos: O Leonardo suporta alguns estilos de codificação de máquina de estados finitos: *binary* (mínimo número de flip-flops), *gray* (apenas uma saída de flip-flop varia por ciclo), *random* (codificação aleatória), *one-hot* (um flip-flop por estado).

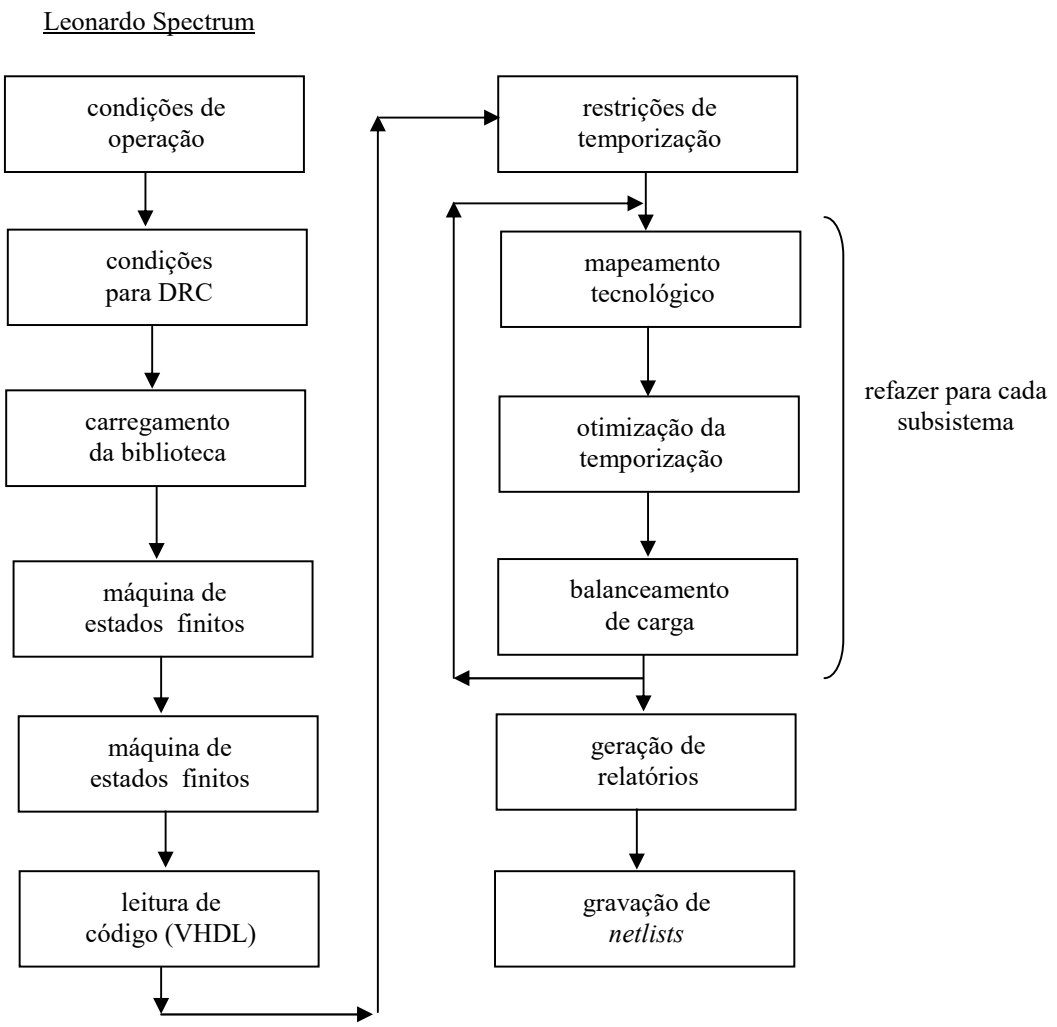


Figura 5. Fluxo de Projeto no Leonardo Spectrum

- Leitura de código original (função **Read**): O Leonardo permite a leitura de código da descrição RTL de um circuito ou em VHDL ou em Verilog. Aceita descrições hierárquicas dos circuitos, mas os módulos devem ser lidos *bottom-up*. A leitura significa uma análise e interpretação (elaboração) do código do projeto feitas em um único passo. Na leitura, a síntese da máquina de estados é realizada, e a lógica é mapeada para um conjunto de células do tipo PRIMITIVES enquanto que as operações são mapeadas para células do tipo OPERATORS. A realização desta etapa leva a uma descrição independente da tecnologia a ser usada e corresponde aproximadamente a chegar à descrição da figura 3(d).
- Definição de restrições gerais de *timing*: Restrições de temporização como frequência de operação, tempos de chegada nas entradas e requeridos nas saídas podem ser colocadas e serão interpretadas e utilizadas pelo do software na fase de mapeamento.
- Mapeamento Tecnológico e Otimização da temporização (função **Optimization**): As diversas arquiteturas hierárquicas podem ser otimizadas isoladamente. Ocorrem duas otimizações nesta etapa, realizadas automaticamente em sequência:
  - Primeira otimização: mapeamento tecnológico realizado por algoritmos globais (critérios de minimização de área ou tempo), baseado nos parâmetros das células da biblioteca de tecnologia (no caso, o ADK, carregado anteriormente).

- Segunda otimização: só realizada caso a primeira otimização não atinja a meta de frequência desejada. Trata-se de otimização local, dos caminhos críticos que apresentaram atrasos excessivos. As outras partes do circuito que não estejam no caminho crítico, resultantes da otimização anterior, permanecem inalteradas.

- DRC (Balanceamento de carga): verificação de violação de regras de projeto no circuito e este é modificado através de replicação de lógica, inserção de buffer e dimensionamento de portas.
- Geração de relatórios: O Leonardo propõe um paradigma em que o usuário interage com o sistema. Relatórios são gerados dando informações ao projetista de como proceder. Relatórios referente à área e atraso são proporcionados.
- Gravação de *netlists*: O Leonardo permite vários formatos de gravação objetivando ferramentas diferentes. Os mais importantes são VHDL ou Verilog para simulação pós-síntese e EDIF para posicionamento e roteamento (leiaute) posterior.

3. Síntese de Leiaute

Células padrão são componentes pré-caracterizados, armazenados em bibliotecas. Cada célula é representada por descrições em diferentes dimensões, por exemplo, seu comportamento, seu leiaute, sua caixa preta (retângulo que envolve o leiaute, suas dimensões e as posições e camadas dos terminais de acesso), etc. Durante a etapa de síntese lógica, os parâmetros tempos de atraso, capacitâncias internas e ocupação de área (extraídos do leiaute) são passados ao algoritmo de mapeamento tecnológico juntamente com a sua descrição comportamental. O esquema lógico (*netlist*) do CI resultante da síntese lógica é o ponto de partida da tarefa de síntese do leiaute. Esta síntese é realizada através de 3 etapas consecutivas: floorplanning (planejamento global do leiaute do CI), posicionamento das células (*placement*) e roteamento dos sinais (*routing*). Todas as células da biblioteca apresentam comumente as seguintes características: altura fixa, saídas e entradas duplicadas e roteadas até as bordas inferior e superior, linhas de alimentação (V_{DD} e GND) dispostas horizontalmente do estremo esquerdo ao direito, respectivamente na parte superior e inferior da célula, sempre na mesma cota. Um exemplo de leiaute de uma célula é dada na figura 6.

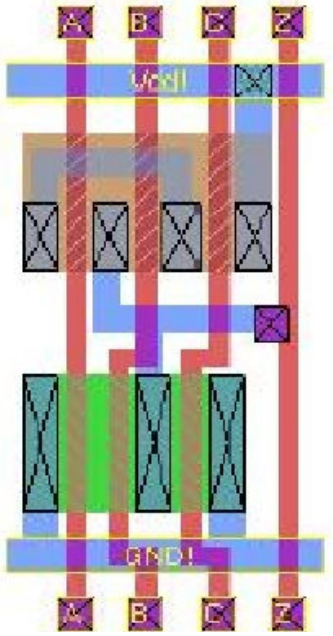


Figura 6. Leiaute de ^(A.B+C)

A caixa preta é o retângulo (virtual) que envolve este leiaute. A razão para que a altura das células seja fixa (o termo padrão, de célula padrão, decorre desta característica) é que a estratégia de leiaute consiste em dispor todas as células de forma a ficarem justapostas, lado a lado em uma fileira horizontal. Caso o número de células seja grande, diversas fileiras de células devem ser construídas para que o módulo final tenha um aspecto geométrico razoável (muitas vezes, tendendo a um quadrado). A figura 7 apresenta um exemplo de módulo implementado com 4 fileiras de células padrão. A conexão entre as entradas e saídas das células (de acordo com o esquema lógico) pode obedecer a diferentes estilos dependendo do número de camadas de metal disponíveis no processo de fabricação utilizado.

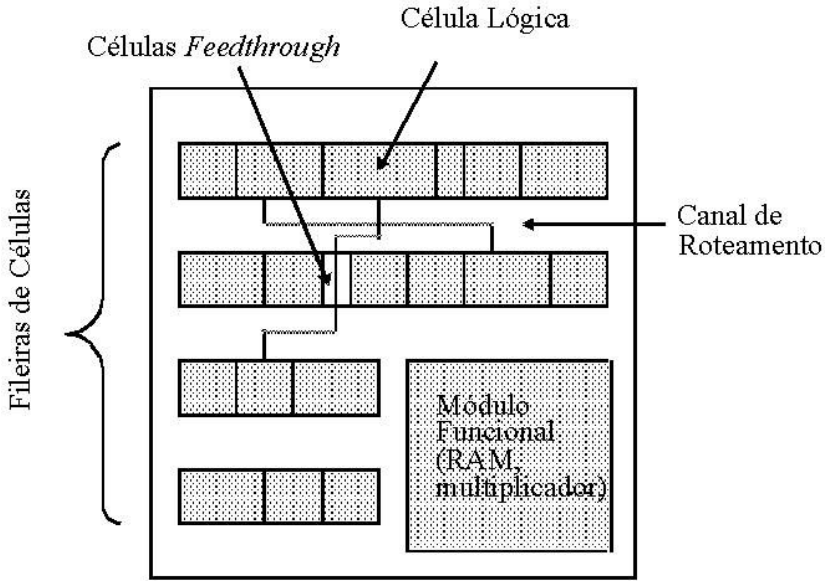


Figura 7. Módulo composto de células-padrão e módulos funcionais

As células padrão correspondem a portas lógicas básicas (NAND, NOR, etc.) ou a portas complexas (composições de AND, OR e inversores). Em geral tais portas apresentam um número reduzido de transistores, portanto é comum que apenas as duas ou três primeiras camadas de metal sejam adotadas na sua construção. Em tecnologias com poucas camadas de metal, como regra geral, as conexões entre as células não passam sobre as células, sendo implementadas em regiões específicas para este fim conhecidas como **canais**, como apresentado na figura 7. Existem ambientes de projeto que permitem outras opções: ainda nesta figura 7, pode-se observar que, para realizar conexões entre células posicionadas em fileiras não adjacentes, células especiais conhecidas como *feedthrough* podem ser utilizadas, assim como linhas de *feedthrough*, que passam verticalmente sobre células de lógica usuais, desde que exista espaço para a passagem da linha de metal. Este recurso reduz o número de conexões que teriam que contornar as fileiras (ficariam muito longas) caso dependessem apenas de canais.

No caso da tecnologia disponibilizar muitas camadas de metal (geralmente 3 ou mais), o roteamento das interconexões entre células podem ser realizadas sobre as próprias células, reduzindo consideravelmente a área ocupada. Tal técnica é denominado de *full-over-the-cell* (*FOTC*). Dependendo da tecnologia disponível, diferentes algoritmos com complexidades variadas são utilizados para minimizar a área de roteamento.

3.1. Síntese de Leiaute com Células-Padrão

A síntese do leiaute parte do esquema gerado pelo processo de mapeamento tecnológico de uma síntese lógica (o que significa que o esquema contém apenas células da biblioteca). São executadas as seguintes duas tarefas:

- Posicionamento das células- o objetivo é dispor as células em fileiras de forma a otimizar algum critério, como por exemplo, minimização de área, minimização do comprimento máximo das conexões, tempo de propagação, etc.
- Roteamento dos sinais- o objetivo é realizar as conexões entre as entradas e saídas das células (definidas no esquema lógico) através dos canais. As conexões entre células de fileiras vizinhas são efetuadas no canal que as separa. As demais utilizam os recursos (já mencionados) de *feedthroughs* internos ou externos, assim como a técnica de roteamento FOTC.

3.3. Vantagens e Desvantagens

Existe uma série de vantagens e desvantagens associadas ao uso de células-padrão. Algumas variam no tempo, podendo se tornar irrelevantes no futuro. As principais vantagens são:

1) Maleabilidade - a geometria (decorrente do *floorplanning*) do módulo baseado em células padrão pode ser definida sem grandes restrições. Por esta razão, a geometria deste tipo de módulo é a última a ser definida quando se faz o *floorplanning* de um CI contendo diversos módulos (combinando células padrão e macro-células).
2) Rapidez da síntese de leiaute – síntese automática do leiaute por posicionamento e roteamento é rápido comparado com leiautes dedicados.
3) Previsibilidade do comportamento elétrico – uma vez que as células já estão pré-caracterizadas, assim como os modelos para os fios, o desempenho, o consumo de potência, e a ocupação de área podem ser previstos com boa precisão.

As principais desvantagens são:

1) Opção limitada de células para cada função lógica - devido à impossibilidade de se manter uma grande quantidade de opções para cada função lógica numa biblioteca, cada tipo de célula apresenta poucas versões, visando melhor área, desempenho ou consumo de potência.
2) Custo de migração para novas tecnologias – a cada nova tecnologia (redução de dimensões mínimas) toda a biblioteca deve ser novamente caracterizada. Em geral, as dimensões das tecnologias não evoluem de forma linear fazendo com que os leiautes das células sejam totalmente refeitas.
3) Superdimensionamento – como as células são construídas para comandar um carga (*fan-out*) de um determinado valor, pode ocorrer com uma certa freqüência que células sejam superdimensionadas (para a necessidade de um determinado nó do circuito) causando excesso de área e de capacitância e, portanto, afetando o desempenho.

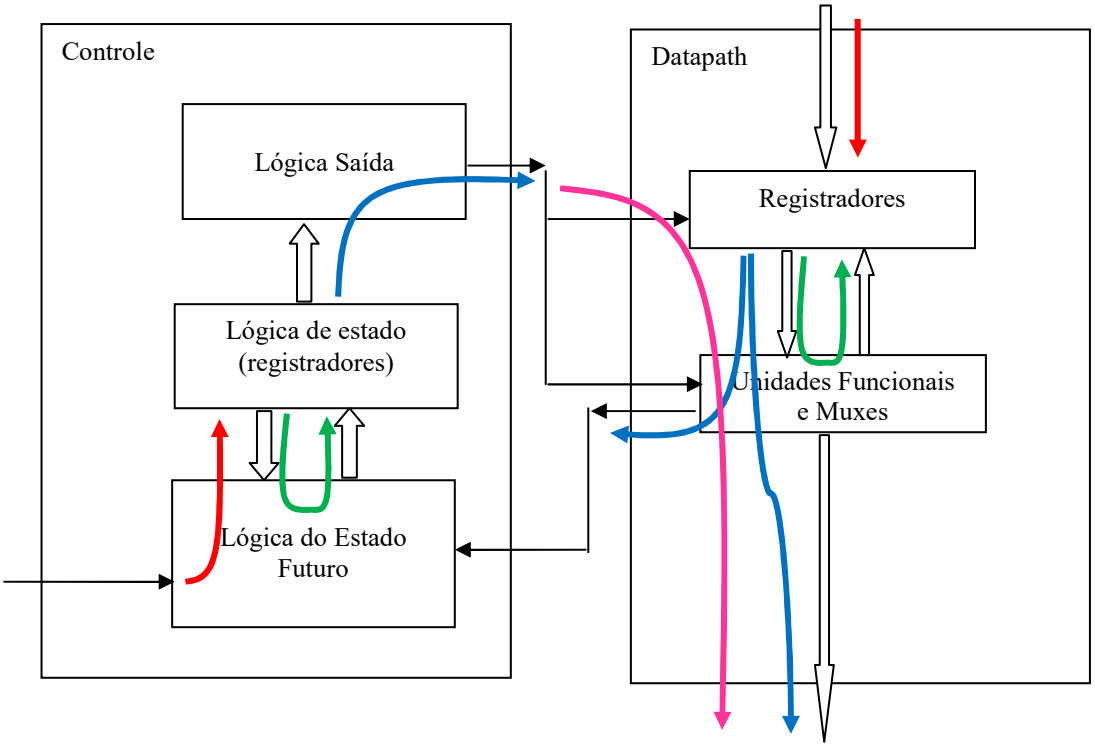
--> O que é o caminho crítico em lógica sequencial?

Em lógica sequencial, os registradores servem como sincronizadores de forma a permitir a transferência de dados e sinais de controle na ativação do sinal de relógio (na subida do clock, por exemplo). Para atender adequadamente a este sincronismo, deve-se monitorar os tempos de atraso referentes a todos os caminhos pelas lógicas combinacionais controlados pelo sinal de relógio. Para o controle ou datapath do circuito digital síncrono, como na figura a seguir, existem três tipos de caminho :

- da alguma entrada do circuito até um registrador (em vermelho)
- de um registrador até um outro registrador (em verde)
- de um registrador até uma saída do circuito (em azul)

Se considerarmos o circuito todo, o caminho em púrpura do datapath deve ser adicionado ao de cor azul que sai do controle.

Dentre todos os caminhos, o **caminho crítico** é aquele de tempo de atraso maior e que limitará o valor inferior do período do clock, portanto, o valor máximo de frequência.


Controle
Lógica Saída
Lógica de estado
(registradores)
Lógica do Estado
Futuro
Datapath
Registradores
Unidades Funcionais
e Muxes