Dispositivos lógicos programáveis e desenvolvimento de sistemas FPGA/CPLD

Um sistema de computação, ou 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 computação 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). O enfoque deste texto é a descrição dos dispositivos lógicos programáveis.

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 do encapsulamento de um CI. 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 quantidade de instruções processadas por unidade de tempo (capacidade computacional), uma vez que a solução de hardware para o seu projeto de sistema digital sempre será mais rápido do que uma solução de software.

Os FPGAs e CPLDs se incluem dentro da subfamília dos PLDs (*Programmable Logic Device* - dispositivos lógicos programáveis), e estes por sua vez são parte da família ASIC. A partir dos PLDs é possível criar diversos circuitos digitais, desde simples portas lógicas até sistemas complexos. A sua grande vantagem em relação aos demais integrantes da

família ASIC (matrizes de portas, célula padrão e totalmente personalizado) está no fato do baixo custo para prototipar o software e o hardware. Os demais integrantes estão associados a altos custos e, geralmente, devem ser comprados em grandes quantidades juntos a uma empresa. Por vezes, pode-se referenciar aos PLDs como dispositivos lógicos programáveis em campo (*field-programmable logic devices* - FPLDs).

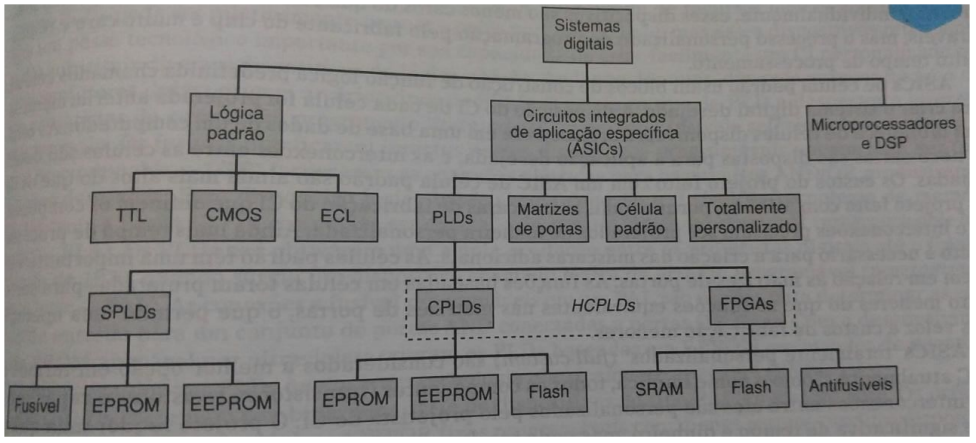

Figura 1: Famílias de sistemas digitais

Os primeiros PLDs surgiram há mais de 30 anos. Sua principal característica está na capacidade de ser configurado pós-fábrica pelo usuário (fácil modificação), o que reduz os custos e tempo de desenvolvimento necessários quando comparado com outras tecnologias. Inicialmente eles continham ao equivalente de algumas poucas centenas de portas lógicas, enquanto atualmente esse número ultrapassa facilmente alguns milhões. Os primeiros PLDs eram muito limitados, uma vez que podiam lidar com somente poucas entradas/saídas e podiam ser programados uma única vez. Se caso houvesse a necessidade de trocar a programação do PLD, seria necessário a substituição de todo PLD. Com a evolução da tecnologia, os PLDs começaram a ter a característica de serem reprogramáveis e a capacidade de lidarem com centenas de entradas e saídas. Inicialmente, os PLDs foram baseados nos vários tipos de memória de semicondutores, e, à medida que novos tipos de memória são desenvolvidas, novos tipos de PLDs surgem.

Os PLDs podem ser divididos em três principais tipos: (i) dispositivos lógicos programáveis simples (*simple programmable logic devices* - SPLDs), (ii) dispositivos lógicos programáveis complexos (*complex programmable logic devices* - CPLDs) e (iii) matrizes de portas programáveis em campo (*field programmable gate arrays* - FPGAs). Como existem diversos fabricantes de PLDs de diversas famílias, podem haver variações desses tipos citados. Além disso, com o passar dos anos a diferenciação entre esses tipos está ficando cada vez mais difíceis.

Os FPGAs e os CPLDs são chamados também de dispositivos lógicos programáveis de alta capacidade (*high-capacity programmable logic devices* - HCPLDs). A principal diferença entre os SPLDs e HCPLDs está relacionada a quantidade de recursos lógicos disponível. Enquanto os SPLDs costumam conter o equivalente a 600 ou menos portas lógicas, os HCPLDs ultrapassam as centenas de milhares. Devido a quantidade de portas lógicas que os SPLDs apresentam, os seus recursos tendem a ser muito mais limitados e apresentarem um custo e complexidade muito menores quando comparados os HCPLDs. A complexidade

envolvida na criação dos HCPLDs resultam na possibilidade da criação de circuitos para complexos sistemas digitais completos.

Os primeiros PLDs feitos com fusíveis também são incluídos dentro da classificação de SPLD. O primeiro PLD foi criado queimando fusíveis literalmente da matriz de programação. Os que restaram intactos resultam em conexões elétricas para os circuitos AND/OR, os quais resultam nas funções desejadas. Esse tipo de abordagem são chamados de dispositivos programáveis apenas uma vez (*one-time programmable* - OTP) e são baseados na tecnologia de memória PROM e também recebiam o nome de arranjo lógico programável (*programmable logic array* - PLA). Os PLAs foram criados no início dos anos 70 pela Phillips.

Os PLAs possibilitaram as primeiras abordagens eficiente para implementação de circuitos lógicos. Esses dispositivos possuem dois tipos de portas: AND (e) e OR (ou). Primeiro há um plano de portas AND, e estas, seguidas por portas OR, ambas programáveis. No caso de somente o uso da ROM, o plano de portas AND não era programável, somente o plano de portas OR. Eles foram criados de tal forma a saída de qualquer porta AND ser ligada a entrada de uma porta OR. A estrutura criada no PLA foi pensada para que cada saída do plano das portas AND correspondessem a qualquer termo produto da entrada. Seguindo a mesma abordagem, cada saída do plano das portas OR podem ser configuradas para produzir a soma lógica de quaisquer saídas do plano AND.

Funções lógicas com a forma soma de produtos é facilmente implementada usando PLAs. Isso por causa da grande quantidade nos termos de ambos os planos AND e OR. Entretanto, isso traz a desvantagem de haver dois níveis de lógica configurável (AND e OR), como mostra a Figura 2. O custo de fabricação desse tipo de sistema é alto e pode haver diversos atrasos de propagação dos pulsos elétricos.

Apesar dos avanços que esse tipo de tecnologia trazia, como a facilidade de criação de Circuitos Integrados (CI) personalizados, os PLDs só obtiveram uma aceitação maior na década de 1970, quando os dispositivos chamados de lógica de arranjo programável (*programmable array logic* - PAL) surgiram. Ele surgiu para sanar as deficiências encontradas nos PLAs. Nesse tipo de tecnologia há somente um único nível de programação, custos reduzidos e um desempenho melhorado. As conexões de entrada para um plano de portas AND continuam iguais as conexões de fusíveis programáveis em um PLA, a alteração é o plano de portas OR que possuem conexões fixas. Apesar de ser mais limitado do que o PLA (portas OR fixas), ele é mais fácil de programas, além de ser mais barato. Esse esquema é mostrado na Figura 3.

Devido a necessidade de alguns circuitos que necessitam da armazenagem de dados, como os circuitos sequenciais, os circuitos PALs possuem flip-flops conectados às saídas das portas OR. É comum encontrar esse tipo de estrutura. Os PALs possibilitaram a evolução dessa área, principalmente em projetos de hardware digital, além de serem a base de diversas novas e sofisticadas arquiteturas. Tanto os PLAs, os PALs e outros pequenos PLDs (variantes do PAL e PLA) são agrupados como SPLD. Conforme as memórias PROM foram evoluindo para EPROM (pode ser reescrita e apagada com luz ultravioleta) e EEPROM (pode ser reescrita e apaga eletricamente), os SPLDs passaram a ser reprogramáveis e uma nova subcategoria surgiu, a GAL (*Generic Array Logic*) que pode ser

reprogramada. As GALs são uma melhoria dos PALs, estes que são baseados em PROM e são escritos uma única vez.

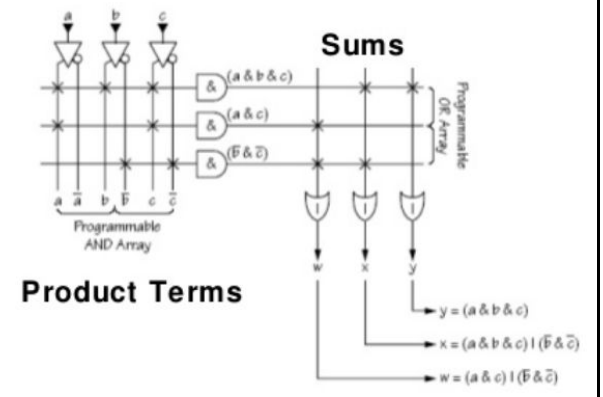

Figura 2: Exemplo de PLA

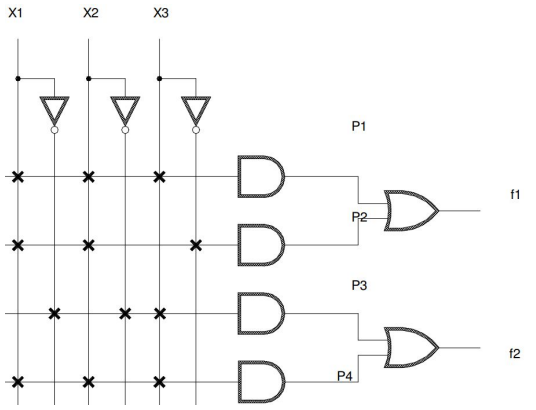

Figura 3: Exemplo de PAL

A união entre o plano de portas AND, seguido pelo plano de portas OR fixos mais o flip-flop é conhecida como macrocélulas, a qual pode ser vista na Figura 4. Para cada saída, está associada uma única macrocélula. Um SPLD pode ser associado a circuitos que utilizam no mínimo duas macrocélulas. As suas principais vantagens estão associadas a facilidade para o tempo de desenvolvimento, baixo custo, alto desempenho e a previsibilidade do tempo gasto pelo sistema. Suas desvantagens estão relacionadas ao baixo número de equações possíveis dentro de um SPLD (entre 10 a 20 equações). Atrelado a esse ponto, existe a ineficiência do uso em sistemas mais complexos. Isso ocorre pois há uma dificuldade em aumentar a capacidade da arquitetura dos SPLDs, uma vez que ao aumentar o número de entradas, a estrutura dos planos lógicos programáveis aumenta rapidamente.

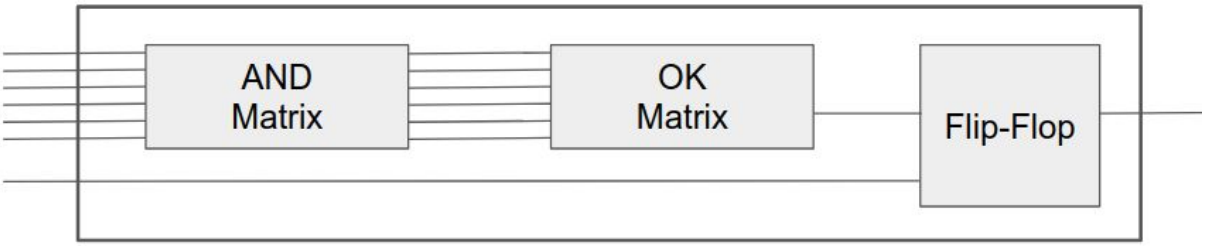

Figura 4: Esquema de uma Macrocélula

Aumentar os circuitos das macrocélulas se mostrou inviável. Uma abordagem para contornar esse problema e permitir a criação de sistemas com maior capacidade baseados na arquitetura do SPLD foi o sistema chamado de *Complex PLD* (CPLD), o qual pode ser visto na Figura 5. Essa abordagem consiste em integrar múltiplas macrocélulas em um único chip, geralmente cerca de 16 e entre 5 e 20 entradas para cada porta OR. As saídas das portas OR são conectadas, por sua vez, a flip-flops para armazenar o dado. Uma adaptação também é o uso de buffers ligados a cada pino do CPLD. Eles são usados para definir se um pino será usado como entrada ou saída. Para realizar o roteamento entre os blocos lógicos foi necessário a criação de uma matriz de interconexões programáveis (matriz switch). O CPLD é organizado em blocos lógicos, baseados em um SPLD, onde cada bloco lógico possui várias macrocélulas. A matriz switch é a responsável por receber

as entradas dos blocos I/O e direcioná-los às macrocélulas individuais. Da mesma forma, direcionar as saídas das macrocélulas para as destinos necessários. Ela pode ser baseada em interconexões de array ou multiplexador.

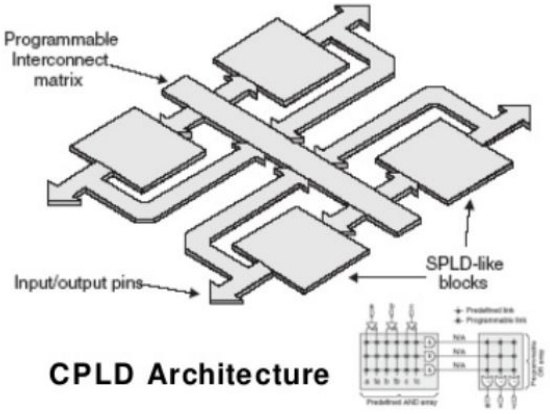

Figura 5: Arquitetura de um CPLD

As macrocélulas lidam, em geral, com muitas variáveis de entrada, e os recursos internos de roteamento de sinal lógico programável tendem a ser bastante uniformes em todo o chip, produzindo atrasos de sinal consistentes. Isso ocorre a simplicidade da estrutura de interconexões. Quando são necessários mais termos-produto, as vários macrocélulas podem compartilhar algumas portas, ou até mesmo podem ser combinadas para implementar uma expressão. Normalmente, a implementação dos registradores na macrocélula pode ser criado com flip-flops do tipo JK (incluindo D e o T) ou SR. Dentro da arquitetura do CPLD, existe uma relação entre os pinos de entrada/saída e macrocélulas específicas. Há macrocélulas ocultas adicionais dentro do CPLD que não são conectadas a nenhum pino. A tecnologia usada pelo CPLD são não-voláteis, ou seja, não perdem os dados quando a fonte de alimentação é cortada. O CPLD pode ser implementado com EPROM (EPLD - *Erasable PLD*), EEPROM e memória flash, sendo o mais comum o uso da EEPROM, a qual resulta no EEPLD (*Electrically Erasable PLD*).

As principais vantagens relacionadas ao CPLD é o seu tempo previsível de execução, além do uso eficiente dos recursos disponíveis. Em contrapartida, a principal desvantagem está relacionada ao roteamento complexo da matriz switch.

Diferente do CPLD, o FPGA pode utilizar outros tipos de memória como a flash, SRAM e antifusível, além de não usar as portas AND ou OR. Dentre as três tecnologias citadas, a SRAM é a mais utilizada. A SRAM é uma memória com tecnologia volátil, e por isso, os FPGAs que usam essa tecnologia devem ser reconfigurados (programados) toda vez que o dispositivo é energizado. Nesse tipo de sistema, faz-se a utilização de uma memória externa que armazena a configuração do circuito. Quando o dispositivo é energizado, os dados da memória externa são baixados para o FPGA. Os FPGAs que utilizam memória Flash são mais confiáveis dos que usam SRAM e também são reprogramáveis. Apesar de serem mais confiáveis, os FPGAs que utilizam Flash tendem a ser mais caros dos que o com SRAM. Os FPGAs que utilizam antifusíveis também são conhecidos por sua confiabilidade, contudo são programáveis uma única vez e utilizam tecnologia não-volátil. Ela é o oposto da tecnologia usada pelo OPT (ao invés de queimar os fusíveis que irão ser usados, queima-se os que não serão usados). A escolha por qual tipo de memória será

usada é importantíssimo, uma vez que elas impactam na performance e nas capacidades do FPGA.

Os FPGAs não possuem planos OR ou AND, consistem em um grande arranjo de células configuráveis que podem ser utilizadas para a implementação de funções lógicas. O FPGA é constituído de pequenos e independentes módulos lógicos programáveis (*Configurable Logic Blocks* - CLB), blocos de I/O configuráveis e interconexões programáveis. O seu esquema pode ser visto na Figura 6. Os CLB podem ser interconectados para criar funções maiores. Usualmente, cada módulo lógico possui até cinco variáveis de entrada. As interconexões programáveis ocupam os canais entre as linhas e colunas dos CLB e entre os CLB e os blocos de I/O e são os responsáveis por prover os caminhos de roteamento para conectá-los. As funções lógicas dos módulos lógicos programáveis são criadas usando a tabela LUT (*Look Up Table*, em tradução livre Tabela de pesquisa). A tabela LUT possui o comportamento idêntico à tabela verdade. Isso quer dizer que a saída pode ser programada para criar a função combinacional desejada armazenando o 0 ou 1 adequado a cada combinação de entrada. O CLB é capaz de implementar qualquer função booleana baseado nas variáveis de entrada. Os blocos lógicos são estruturados em forma de array.

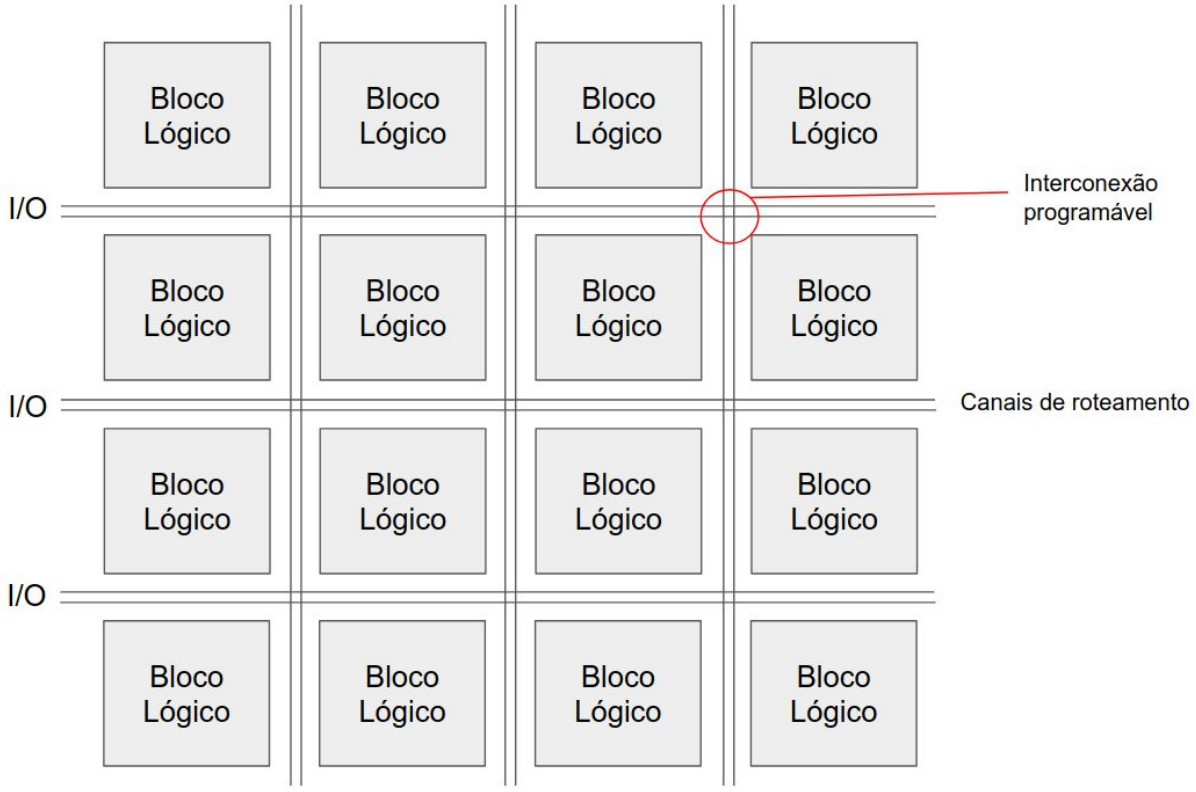

Figura 6: Esquema de um FPGA

Diferente do CPLD que possui alguns módulos lógicos ligados aos pinos de I/O, o FPGA possui cada pino de I/O conectado a um bloco de I/O. Esses módulos são depois conectados aos módulos lógicos com linhas e colunas das interconexões programáveis. Os blocos de I/O podem ser configurados tanto como saída, como entrada ou bidirecional.

As interconexões programáveis podem ser divididas em três tipos:

1. Interconexão de propósito geral: As interconexões programáveis podem ser estabelecidas por meio de roteamento automático ou por meio de seleção dos pares desejados da matriz de pinos para serem conectados ou desconectados. Os buffers de interconexão podem seguir qualquer direção. O sistema de desenvolvimento define automaticamente a direção do buffer com base na localização da origem da rede de interconexão. Ela pode conectar um CLB a outro qualquer dentro do circuito, como é mostrado na Figura 7.
2. Conexão direta: Oferece implementações mais eficientes para conectar blocos CLB adjacentes ou I/O. Esse tipo de conexão exige o mínimo de propagação de interconexão e não usam os recursos gerais de interconexão para o roteamento de bloco para bloco. A saída de uma CLB pode estar diretamente conectada a entrada de um CLB a sua direita, esquerda, acima e abaixo. Ela é usada para maximizar a velocidade das partes de alta performance. Exemplo na Figura 7.
3. Linhas longas: Elas ultrapassam as matrizes de comutação e destinam-se principalmente a sinais que devem percorrer longas distâncias, ou devem ter um mínimo de desvio entre vários destinos. Elas estão posicionadas verticalmente ou horizontalmente como mostra a Figura 7.

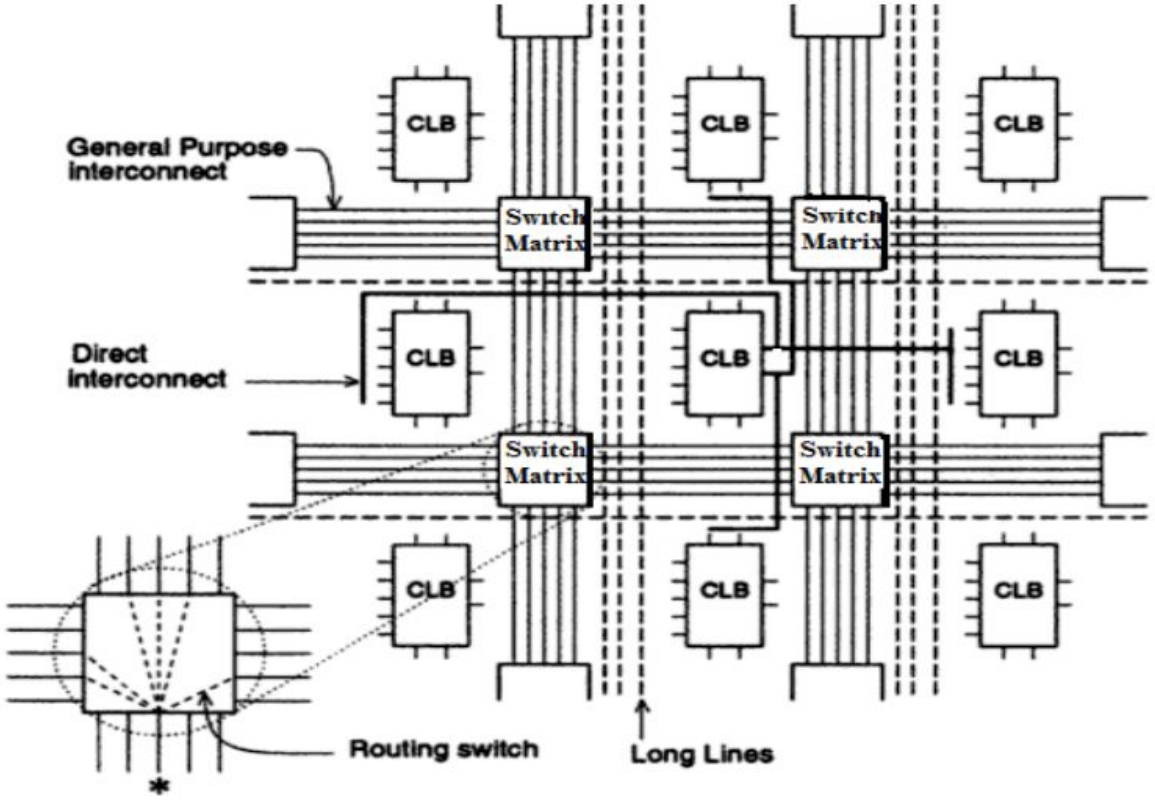

Figura 7: Exemplos dos tipos de interconexões.

A principal vantagem do FPGA é o seu uso eficiente de recursos e que estruturas complexas podem ser implementadas. Contudo, ele possui a desvantagem em relação a previsibilidade do tempo de processamento. É bem variado o tempo necessário de roteamento de sinal programável dentro do chip, isso porque os FPGAs precisam cascatear

diversos estágios de lógicas de elementos, uma vez que os CLB possuem limitações de entrada. Alguns FPGAs podem ou não incluir blocos de memória RAM.

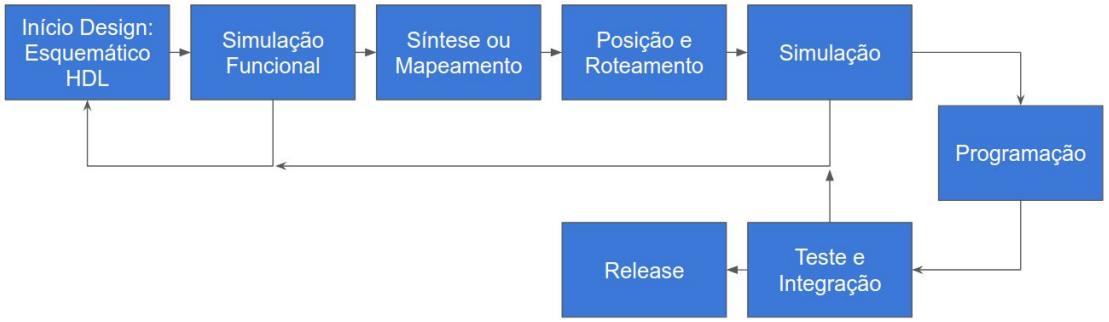

Figura 8: Fluxo de desenvolvimento de sistema FPGA

O processo de desenvolvimento de um FPGA passa pelo fluxo apresentado na Figura 8. Primeiro, o processo é iniciado com o design da aplicação, a qual envolve a escolha dos componentes do circuito, além das tecnologias escolhidas de desenvolvimento. O passo seguinte é a simulação do que foi feito no design para verificar se a lógica foi implementada como esperado. O próximo passo é o mapeamento da lógica planejada para a arquitetura do dispositivo FPGA. O quarto passo envolve o posicionamento e o roteamento dentro da estrutura do FPGA. Como uma boa prática, é aconselhável rodar novamente uma simulação após o processo de posição/roteamento. Se tudo estiver correto, o próximo passo é criar o arquivo de programa, o qual será salvo no dispositivo do FPGA para testar. Se caso o teste falhar, volta-se o processo de design e repete-se todo o processo.As formas mais comuns na qual o processo de design pode ocorrer são por meio de captura esquemática, importação de blocos propriedade intelectual (IP - componentes prontos do mercado, do inglês, *Intellectual Property*) e HDL, incluindo VHDL e Verilog. Todos esses processos podem usados no mesmo projeto, misturados ou combinados.

O passo de simulação funcional e síntese são responsáveis por checar se os arquivos de design contém erros e por construir uma base de dados a qual coloca os arquivos de design em hierarquia, sintetiza e mapeia a lógica do design para os recursos do dispositivo. É nesse passo que ocorre a otimização da arquitetura planejada, a qual visa ao máximo reduzir ao máximo o número de CLB e minimizar ao máximo delays, tanto dos CLB quanto do roteamento.

O processo de posicionar e rotear o design da melhor forma possível ainda é um dos principais desafios no design de FPGAs. A simulação após esse passo é importante, uma vez que atestar se há violações dos requisitos de tempos com análise de tempo é super importante, principalmente em Sistemas de Tempo Real, onde a entrega de uma resposta depois do tempo esperado é tão ruim quanto uma falha na lógica. A sincronização dos sinais dentro do FPGA é fundamental para que o design seja confiável.

Ferramentas como Quartus Prime da Altera/Intel e ISE Design suite da Xilinx são de extrema relevância no cenário de design, uma vez que auxiliam em todo o fluxo de design apresentado na Figura 8. Eles são úteis desde a criação do projeto de design, até a análise de tempo.

Há cenários onde a escolha por CPLDs é mais aconselhável do que pelo FPGA, isso porque em sistemas onde a lógica é menor não é necessário toda a complexidade do FPGA e o custo extra que é acarretado. Mas para sistemas mais complexos, o FPGA é melhor, uma vez que ele é mais escalável que o CPLD. Já que a arquitetura do FPGA é mais refinada do que os CPLDs, o roteamento tem mais impacto no desempenho.

Os System-on-chip (SoC) são um tipo de aplicação que são favorecidas pelos PLDs em geral, principalmente os FPGAs. Uma de suas implementações são o FPGA mais componentes IP. Os SoCs são a junção de vários componente de um sistema eletrônico em um único circuito integrado (chip) que resultam em um circuitos complexos miniaturizados. Estes que favorecem o desenvolvimento e servem de base de outros sistemas, como por exemplo os sistemas ciber-físicos (Cyber Physical Systems - CPS). Os CPSs são a convergência entre computação, comunicação e controle, sendo a integração da computação com processos físicos. Acaba-se por utilizar SoC para interagir com processos físicos, os quais adicionam novas propriedades ao sistema.

Os SoCs realizam todo o processamento e interface entre o mundo e o sistema digital de um CPS. São eles que oferecem a base necessária para o CPS desenvolver e também a Internet-das-coisas. Isso ocorre pois eles estão presentes em praticamente todos os dispositivos eletrônicos digitais que são usados diariamente, além de permitir a evolução de equipamentos que há anos não evoluíam (motores a combustão que atualmente podem funcionar com sistema bicombustível com um menor nível de poluição devido ao controle eletrônico empregado). O CPS está um nível abaixo da Internet-das-coisas.

No contexto de Internet-das-coisas, os CPSs possuem um papel de destaque, uma vez que colaboram para construir sistemas inteligentes, como por exemplo as cidades inteligentes, as quais podem ser vistas como sistemas ciber-físicos em ampla escala. As cidades inteligentes possuem sensores monitorando indicadores virtuais e físicos e por meio de atuadores, muda dinamicamente o ambiente urbano complexo de alguma maneira. Os CPSs podem ser usados na produção a âmbito industrial para construir arquiteturas baseadas na internet que facilitam o controle remoto de sistemas de produção *stand-alone*. Os CPSs vêm se destacando em vários campos, desde governos, organizações até indústrias de tecnologia, os quais direcionam seus esforços para integrar diversos sistemas em redes inteligentes, a fim de, por exemplo, controlar uma rede de energia ou o tráfego de trânsito para torná-lo mais seguro.

A cada ano que passa, os dispositivos lógicos programáveis ficam cada vez mais em evidência devido a sua aplicabilidade e o custo baixo em relação a outros sistemas ASIC. A evolução dos PLDs ocorreram de acordo com a evolução de tecnologia, iniciando com a memória PROM (fusíveis) que eram programáveis, depois os CPLDs que fizeram uso da união de vários SPLDs e de novas tecnologias, como a EPROM e possibilitaram circuitos mais complexos de serem criados, até os FPGAs que hoje permeiam a indústria. Além disso, o processo do design e as escolha por qual tipo usar é de suma importância para a aplicação final, uma vez que são grandes os impactos na performance do PLD.

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

https://pt.slideshare.net/yayavaram/cpld-fpga-cpld-fpga-architectures-applications

http://www.electronics-tutorial.net/programmable-logic-devices/complex-programmable-logic-device/index.html

http://zipcpu.com/blog/2017/10/13/fpga-v-asic.html

http://www.ni.com/white-paper/6984/pt/

https://pt.slideshare.net/abhilash128/lec-22-16642421

https://www.eetimes.com/document.asp?doc_id=1274593

https://www.coursera.org/learn/intro-fpga-design-embedded-systems/home/week/4

file:///home/pedro/Desktop/cpld-fpgaarchitecturesapplications-140101002310-phpapp02.pdf

https://www.tce.edu/sites/default/files/PDF/DSPwithFPGA_FlippedMaterial-K.Kalyani.pdf

http://coral.ufsm.br/gepoc/renes/Templates/arquivos/diversos/IntroducaoPLD%5B1%5D.pdf

http://www.demic.fee.unicamp.br/~elnatan/ee610/24a%20Aula.pdf

http://www.optotech.net.br/msx/cpld/11-55-1-PB.pdf

http://www.ece.ubc.ca/~edc/464/lectures/lec6.pdf

http://www.feg.unesp.br/Home/PaginasPessoais/ProfMarceloWendling/logica-programavel.pdf

http://siva.bgk.uni-obuda.hu/jegyzetek/Mechatronikai_alapismeretek/English_Mechatr/PLDs/CPLD&FPGA.pdf