Especificações, modelagem e técnicas de projeto de Sistemas Embarcados e de SOC (System On Chip)

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 sistemas digitais: sistemas embarcados e SoC (*System on Chip*).

O SoC, que em tradução livre seria sistema-em-um-chip, é a junção de vários componentes de um sistema eletrônico em um único circuito integrado (CI) ou chip. As suas funções são várias, incluindo funções digitais, analógicas e mistas. Eles são componentes importantes de sistemas embarcados. Uma boa analogia para entender o SoC é, dado um computador completo, reduzir e comprimi-lo ao máximo até ser capaz de colocá-lo em um único chip. Por exemplo, o SoC pode conter ao que se assemelha uma placa mãe (barramentos, BIOS e sistemas de controle de fluxo de dados), processador, placa de vídeo, memória RAM e assim por diante, como é o caso do SoC usado em diversos smartphones, como por exemplo o Samsung Galaxy S4.

Os grandes avanços tecnológicos que vem acontecendo na área de eletrônica, principalmente nas últimas cinco décadas, proporcionaram que chips contenham bilhões de transistores em uma única pastilha ou encapsulamento. Isso só foi possível devido ao tamanho dos componentes que foram reduzidos a escala de nanômetros. De acordo com a lei de Moore (que atualmente já está chegando a um limite), aproximadamente a cada 18 meses, o número de transistores em uma placa vai dobrar.

A consequência dessa evolução é que componentes antes desenvolvidos separadamente e posteriormente conectados a uma placa de circuito impresso (PCB - *Printed Circuit Board*), agora podem ser integradas em um único chip. Desse cenário parte o desenvolvimento de sistemas SoCs (Figura 1).

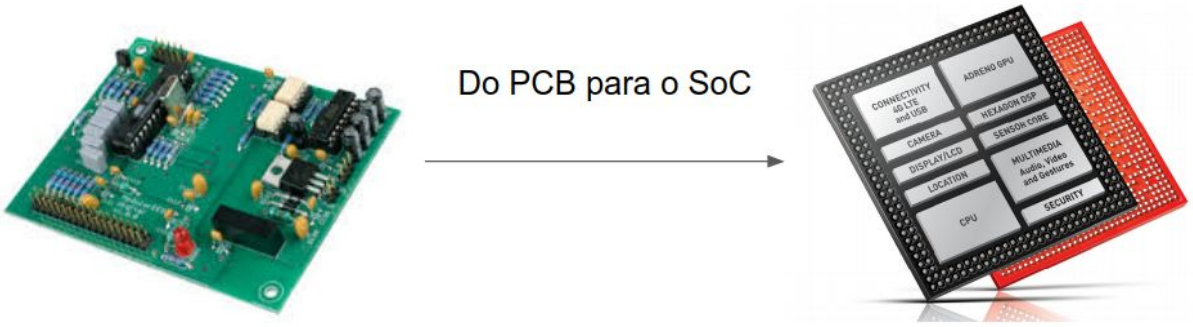


Figura 1: Evolução dos PCBs para os SoC

A comparação entre um microcontrolador e um SoC é comum, uma vez que ambos possuem características semelhantes, como serem sistemas de um único chip e serem sistemas projetados e focados para uma aplicação em específico. A diferença entre os dois está na complexidade envolvida em sua construção. Enquanto normalmente o microcontrolador possui menos de 100k de RAM e recursos limitados de processamento, os SoCs possuem processadores mais potentes e capazes de executarem sistemas mais complexos como o Windows e o Linux, os quais precisam de uma grande quantidade de memória RAM e ROM/Flash e de componentes periféricos acoplados para funcionarem.

Em sua essência, o SoC é um sistema que pode ter embutido vários microcontroladores e DSPs (Processadores de sinais digitais, do inglês *Digital Signal Process*) conectados a componentes de todos os tipos, os quais resultam em vários subsistemas interconectados para realizar uma função específica para um usuário (Figura 2). Devido a grande quantidade de processadores e outros componentes, os controladores de DMA (*Direct memory access*) e outras complexidades de barramentos podem ser incluídas. O seu uso atualmente é diverso, desde smartphones, tablets e videogames até carros.

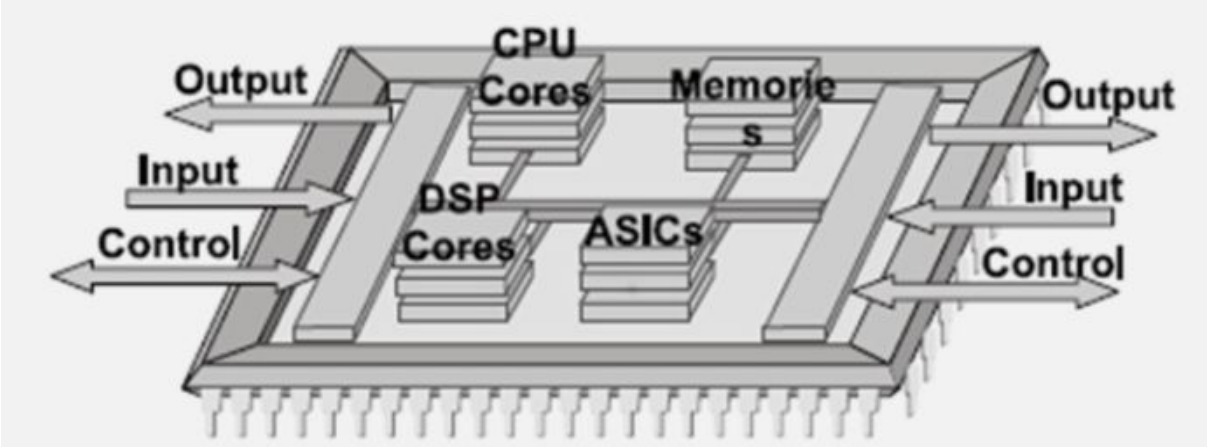


Figura 2: Componentes heterogêneos integrados

O design de um SoC envolve muita complexidade, uma vez que diversos componentes são colocados lado-a-lado dentro de um mesmo chip. Isso envolve a exploração do espaço dentro do chip, integração dos componentes e o tempo gasto durante o processamento, além da energia exigida para o funcionamento. Outro fator que deve ser levado em conta é o calor produzido pelo chip. Uma vez que os modelos são projetados, é necessário validar e verificar se o seu comportamento está de acordo com o esperado, além de testes com diversas complexidades em diversos cenários. O custo e tempo de design são peças importante dentro do contexto do produto final, desde a escolha dos componentes até o produto final.

O design de qualquer SoC começa pelas especificações da aplicação de destino. As especificações impactarão tanto nas escolhas de projeto do Software (à nível de aplicação e de Sistema Operacional) e Hardware. Essas especificações serão então convertidos em requisitos funcionais (requisitos de sistema que especifica uma necessidade funcional que

pode ser parte de uma funcionalidade que o sistema deve ser capaz de realizar, como adicionar um contato em um smartphone) e requisitos não funcionais (apesar de não ser uma funcionalidade, ainda é algo que precisa ser realizado para que a aplicação atenda seu propósito, como tempo máximo para que o processamento de um dado seja feito).

Após essa especificação, começa o design em si do SoC, o qual pode ser dividido em três etapas principais, como mostra a Figura 3:

1. Design à nível de sistema: Toda a aplicação de software é desenvolvida neste nível. Geralmente ela é executada sobre um sistema operacional e há uma abstração entre os programas desenvolvidos e o hardware.
2. Design à nível de software: Esse nível envolve a escolha/desenvolvimento do software que fará a interface entre os programas de alto-nível e o hardware. Nesse ponto envolve a escolha do sistema operacional a ser usado e os softwares de embarcados. A escolha de qual software usar nesse ponto é de extrema relevância, uma vez que gera impacto em toda a aplicação. Por exemplo, se o sistema for de tempo real, é necessário a escolha de um Sistema Operacional de Tempo Real (RTOS - *Real Time Operating System*), como RTLinux e RTAI, ambos de código aberto, ou os mais usados na indústria, uC/OS (micro COS) e vxWorks..
3. Design à nível de hardware: O design dessa etapa é a base do restante do sistema e é parte fundamental do design do SoC já que engloba diversos elementos de hardware em um único chip. Ele pode ser dividido em dois segmentos de design: baseado em propriedade intelectual (IP - *Intelectual Property* - componentes já prontos) e baseado em plataforma, sendo o primeiro o mais tradicional.

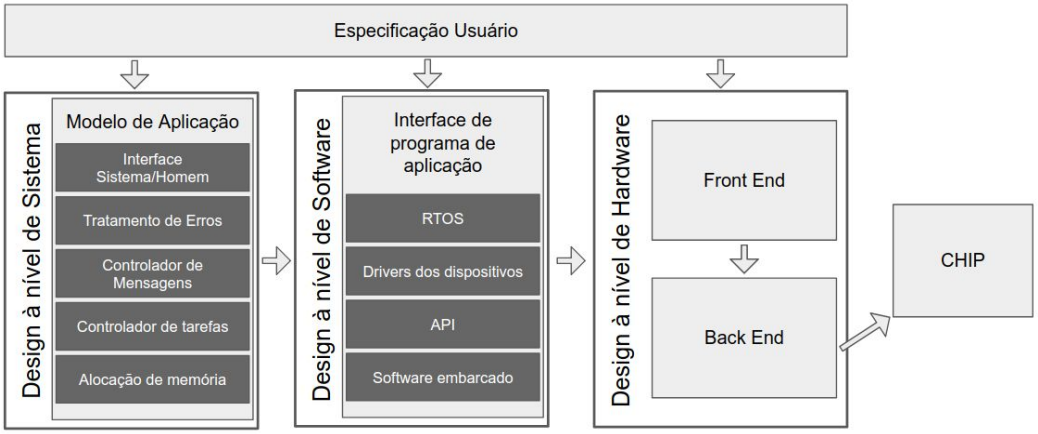


Figura 3: Processo de desenvolvimento de um SoC

O design baseado em IP busca criar os sistemas baseados em múltiplas fontes de IP. Esse tipo de abordagem traz alguns problemas. É necessário calcular e balancear o consumo de energia, além do tempo e custos envolvidos em licenciar as peças. A busca pela resolução destes percalços podem resultar em três a seis meses de atraso para o início do planejamento do SoC em si. Quando o design começa, os designers já devem ter conhecimento sobre cada protocolo de hardware e software específicos dos IPs. Uma vez com esse conhecimento, o projeto de como será integrado hardware e software começa. A integração deve sofrer pequenos ajustes para obter o melhor ou o desempenho desejado.

Como diferentes componentes são usados, é necessário realizar uma extensa verificação de hardware e software, a qual é difícil e não significa ausência de problemas. O resultado é uma grande variedade de possibilidades para a construção do sistema.

Normalmente, os IPs podem ser disponibilizados de três formas: (i) *Hard* - é uma caixa-preta com o layout interno o mais otimizado e modelo criptografado, por exemplo microprocessadores; (ii) *Firm* - uma arquitetura do circuito (netlist) pronta, a qual pode ser simulada e alterada de acordo com a necessidade do designer; e por fim, (iii) *Soft* - nesse caso, a responsabilidade pela sintetização e do layout do circuito fica todo em cargo do usuário (FPGAs). A decisão por qual tipo escolher, pode acarretar em mais ou menos complexidade, uma vez que a flexibilização de um componente pode facilitar a integração a um sistema, contudo traz mais complexidade para a sua configuração. Atrelada a escolha de qual forma utilizar o IP, está a escolha entre reusabilidade, portabilidade e flexibilidade do uso (*soft*) e o desempenho e tempo de comercialização (*hard*). Os sistemas *soft* tendem a diminuir o tempo para confecção de um sistema, contudo a performance é comprometida quando comparado ao *hard*. Esse cenário é mostrado na Figura 4. Os IPs *soft* oferecem a criação de PSoCs, que são SoCs programáveis. Em uma visão macro, é um misto de SoC com FPGA, os quais são muito versáteis. Exemplos de PSoCs são os PSoC® 5 da CyPress.

Diferente do design baseado em IP, o design baseado em plataforma visa oferecer aos designers até 90% dos sistema de hardware e software prontos. Os designers ficam focados em desenvolver recursos diferenciados do produto final. Este tipo de design busca usar blocos comuns em arquiteturas (processador, memória, I/O e estrutura de barramento) para sanar os requisitos funcionais e não funcionais. Os componentes específicos de uma aplicação especializada (requisitos de produtos) ficam encargo dos projetistas. Os projetos baseados em tal plataforma possuem alguns benefícios, ressaltando a redução do custo e tempo necessários para criar o produto final, além de problemas relacionados a energia que são reduzidos (a preocupação maior está nos novos recursos criados). O tempo de verificação do sistema como um todo também é menor, uma vez que grande parte do sistema já foi testado em outros projetos e é, às vezes, uma plataforma/arquitetura consolidada no mercado. O fluxo de design baseado em plataforma é visível na Figura 5.

A grande vantagem relacionado ao design baseado em plataforma em relação ao baseado em IP é a redução significativa no tempo e custo de desenvolvimento do projeto, além da redução do risco do projeto. Esse contexto, favorecido pela redução do número de fornecedores de Hardware e Software necessários, contribui com a diminuição da necessidade de treinar engenheiros em protocolos especializados. A desvantagem está no ponto em que o primeiro tende a ser mais lento que o segundo. Essa economia de tempo no desenvolvimento pode reduzir o ciclo de design até a produção em seis meses ou menos.

Em ambos os casos (tanto o baseado em plataforma ou em IP) está envolvido, em alguma escala, o design mais específico do hardware. Há duas abordagens mais populares para projetar o hardware de um SoC: *top down* e *bottom up*. Ambas podem ser extendidas para software e Sistema Operacional. A abordagem top down envolve a decomposição de todo o produto final em partes. Essa decomposição chega até certo nível, uma vez que faz-se uma

abstração dos componentes mais baixo nível. Isso facilita no ponto em que a modelagem do sistema baseia-se em uma abordagem de estímulo-resposta, onde o estímulo é a entrada que deve produzir uma resposta, a qual é dirigida a atuadores. Já os sistemas bottom up iniciam o processo dos componentes mais baixo nível relacionados ao hardware e vão construíndo cada parte do sistema final. Esse tipo de processo facilita bastante a reutilização de módulos e acaba por adaptar-se melhor a pequenos projetos. Em ambos os casos, top down e bottom up, as decisões de baixo nível relacionadas ao hardware precisam ser tomadas no início do projeto. Esse tipo de ação, diminui muito a flexibilidade do projeto, uma vez que o hardware é pensado para uma aplicação em específico. Em muitos casos, quando o processo top-down é escolhido, as definições de baixo nível do hardware que precisam ser tomadas não são práticas, uma vez que elas precisam se adaptar ao que foi planejado.

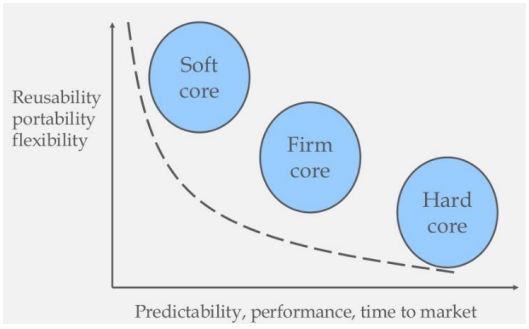


Figura 4: Trade-off entre sistemas *soft*, *firm* e *hard*.

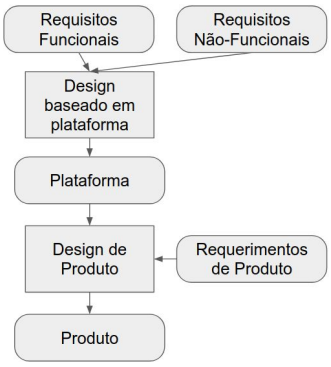


Figura 5: Fluxo de design baseado em plataforma

Existem diferentes técnicas para a criação do diagrama para o que foi projetado para o hardware, as quais se enquadram entre as abordagens *top down* e *bottom up*. Elas possuem diferentes objetivos, contudo, ajudam na visualização da distribuição dos componentes e no seu comportamento dentro do circuito. As mais comuns:

- Diagrama de blocos: Abstrai todos os detalhes de implementação. Esse cenário considera a maioria dos componentes da placa do Sistema Embarcado, ou SoC, como blocos. Isso fornece uma representação em alto-nível e uma representação física do layout da placa com todos (ou a maioria) dos componentes. Essa representação possibilita a visualização de como diferentes componentes funcionarão em nível de arquitetura do sistema. A principal vantagem é a facilidade em descrever, demonstrar e entender um overview básico do Hardware. A principal desvantagem está no ponto em que não é suficientemente detalhado ao ponto de um designer de software escrever o código baixo nível capaz de controlar o hardware da forma mais otimizada.
- Diagrama esquemático: Detalha mais o hardware quando comparado ao anterior, e seu enfoque não é mostrar o layout físico dos componentes na placa, mas sim fornecer informações do fluxo de informação no sistema. Fornece informações detalhados do processador até os resistores. É o diagrama mais útil tanto para o

designer de hardware quanto para o de software quando se trata de como o sistema funciona como um todo, além de ser usado para debugar o hardware/software.

- Diagrama de fios: O seu principal objetivo é representar todas as conexões de barramento entre todos os componentes (dos principais ao mais simples) dentro de um chip. Ele pode apresentar uma descrição próxima do layout físico dos componentes ou da placa.
- Diagramas lógicos impressos: Representação próxima ao diagrama esquemático, uma vez que mostra uma todos os componentes se relacionando no circuito. Entretanto, ele se difere no ponto em que simplifica certos tipos de circuitos com o intuito de facilitar o funcionamento do chip.
- Diagrama de Tempo: Mostra o gráfico dos sinais de I/O de um circuito ao longo do tempo, além da relação entre eles. Depois do diagrama de blocos, é o mais usado em manuais e *datasheets*.

O uso desses diagramas é bastante útil no ponto em que oferecem um overview de todo o sistema para o programador. Além de especificar algumas escolhas de projeto, como qual arquitetura do sistema será usada (von Neumann ou Harvard) e consequentemente arquitetura do processador (CISC ou RISC). E uma vez pronto, pode ser usado como fonte de consulta posteriormente em *datasheets*.

Normalmente, a escolha pela abordagem *top down* se dá por causa da abstração a modelagem que ela oferece. Com o uso da modelagem em um nível abstrato o projetista ganha a flexibilidade de explorar a divisão entre hardware/software e as várias tomadas de decisão sobre a arquitetura, sem a necessidade da especificação de detalhes de baixo nível, como qual será o protocolo de comunicação.

O design *top down* de hardware, usualmente, pode ser dividido em duas etapas : Front end e Back end. Essa divisão é feita baseado no processo de RTL (Register Transfer Level), mais especificamente na parte estrutural (estrutura do sistema). RTL descreve a funcionalidade interna em termos de registradores e funções lógicas combinacionais necessárias, de tal forma a descrever o sistema em termos do fluxo de sinais (ou transferência de dados entre os registradores presentes no hardware e as operações lógicas conduzidas com estes sinais).

O processo de planejamento e modelagem do *front end* é feito em termos de requisitos alto-nível, como funções, taxa de transferência e consumo de energia. Ele pode ser descrito como o seguinte fluxo:

Especificação + Exploração + Design + Captura + Síntese => RTL estrutural

Cada fase possui um enfoque:

- Especificação: Vem do usuário e parte da especificação de todo o sistema.
- Exploração: É o processo que envolve o uso de diferentes combinações de processadores, memórias e estruturas de barramento para encontrar a melhor combinação que casa com as especificações desejadas, como bom consumo de energia e o balanceamento de carga entre os componentes. Um modelo de alto-nível é suficiente para se calcular o desempenho de uma arquitetura. A

extensão da exploração é definida de acordo com a natureza do design: baseado em IP (mais exploratório) ou plataforma (exige uma exploração menor).

- Design: É o momento onde componentes vendidos no mercado, como processadores, DSPs, memória, entre outros (IP) são empregados nos blocos funcionais. Há a opção também de realizar o uso de de componentes com tecnologia interna da empresa. A diferença entre o IP e o de tecnologia interna é a necessidade de pagamento de licença do IP, e a construção (tempo de planejamento) do interno.
- Captura: É o processo de converter as especificações do sistema para um documento mais técnico e próximo da realidade de implementação. Esse documento técnico pode ser, na maioria das vezes, um código em linguagem de descrição de hardware (HDL - do inglês *Hardware Description Language*). Pode ser também a descrição da construção de um sistema artesanal (circuitos muito mais simples).
- Síntese: Consiste em converter todo o planejamento do passo anterior (RTL comportamental) para o RTL estrutural (netlist). No caso quando HDL é usado, consiste no processo de geração de uma descrição tipo *netlist* (lista de componentes e suas interligações) a partir de uma descrição feita com VHDL ou Verilog (tipos de linguagem HDL). Além de converter para o netlist, ainda é feita uma otimização do circuito nesse ponto.

Todos os componentes dentro de um SoC precisam se comunicar, e para isso, é necessário a definição da interface (mapas de registradores e outras APIs) entre os componentes. Essa definição deve ser salva. O padrão mais comum de ser usado é o IP-XACT que é escrito em formato XML. Ele é usado para permitir que seja realizada a substituição a biblioteca de componentes complexos entre as ferramentas de automação de design eletrônico para design de SoC.

A função da síntese dentro do contexto de design RTL é, por exemplo, a de mapear o design para uma implementação de porta lógica digital. Essa síntese pode ser feita por meio de diversas tecnologias. O FPGA (*field-programmable gate array*) foi por muito tempo a tecnologia usada para realizar essa implementação e mapeamento, todavia, atualmente, o design de SoCs é uma semi customização de silício.

Quanto maior for o uso de tecnologias como o FPGA, maior é o custo e dificuldade em verificar o circuito. Maior é o tempo necessário de design também. Daí surgem as maiores dificuldades dos designs baseados em IP. No caso do design baseado em plataforma, o uso é reduzido.

No caso do planejamento do *back end*, o fluxo de funcionamento parte do RTL estrutural:

RTL estrutural => posicionamento + rotas + fabricação de máscara + fabricação

Diferente do processo do *front end*, o objetivo do *back end* é implementar o que foi planejado na etapa anterior. Nesse caso, utiliza-se o resultado da síntese feita no *front end,* a netlist, desconsiderando qualquer atraso das portas. O posicionamento e as rotas da estrutura da rede mostram as coordenadas 2D de todos os componentes, além de mostrar os blocos de entrada e saída externos, além da fiação entre os componentes (RTL estrutural). Quando a RTL é usada com os atrasos reais do circuito, ela fornece um modelo

preciso de potência e desempenho. É nesse momento que se verifica se o circuito criado casa com as especificações e necessidades da aplicação final. Caso estiver algo fora do esperado, deve-se realizar o processo de *front end* novamente.

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.

A parte de fabricação da máscara do sistema (com todas as trilhas e conexões do sistema - *netlist*) é a parte mais cara de todo processo de design, por esse motivo, é necessário que o circuito esteja correto logo na primeira vez que for fabricado. Uma vez que o sistema passou por todas as validações, o circuito planejado é enviado para empresas para que sejam produzidos. Normalmente, há a produção de milhares, até milhões de unidades do Sistema Embarcado ou SoC. Um erro no planejamento pode ocasionar prejuízos de milhões de dólares, como aconteceu com a empresa Samsung e o smartphone Galaxy Note 7 que ocasionou um prejuízo de mais de 15 bilhões de dólares. A bateria do smartphone explodia, devido a problemas com sua bateria que aqueciam, aliados a falta de dissipação de calor. Várias unidades foram vendidas, e todas que não foram vendidas deixaram de ser comercializadas.

A cada estágio, tanto de *front* quanto de *back end*, faz-se uso de técnicas e ferramentas de validação desses sistemas digitais. Uma vez que o posicionamento e rotas do design foram validados em uma especificação alto-nível, e o design foi avaliado de acordo com as operações e especificações iniciais, o design está pronto para ser fabricado em grande escala. Depois de fabricados, o chip deve ser novamente validado, normalmente, com o mesmos testes funcionais usados no estágio de especificação do sistema. Normalmente, no decorrer dos dias durante o processo de design de uma aplicação (sistema digital), os circuitos devem ser verificados, e qualquer mudança que acarrete alguma falha dos testes serão reportadas para que a aplicação seja reavaliada e as medidas cabíveis sejam tomadas.

O uso de SoC trás algumas vantagens. Como o tamanho físico dos SoC é bastante pequeno, os componentes internos tendem a ficar mais próximos o que resulta em um circuito mais rápido, além de reduzir o tempo de fabricação, dinheiro e recursos necessários para ligar partes significativas importantes. Atrelado ao tamanho, também há as vantagens do baixo custo de fabricação de um único chip e do baixo consumo de energia. Como os circuitos são resultados da integração de vários componentes em um único circuito, o planejamento é maior. Isso ocasiona uma maior segurança do design e a implementação tende a ser mais confiável.

Apesar dos vários benefícios dos SoC, algumas desvantagens também surgem, estas mais relacionadas ao processo de design. O tempo de design e planejamento não devem ser muito longos para não haver perda do timming de mercado (tecnologia pode ficar obsoleta antes mesmo de entrar no mercado). O custo de planejamento e fabricação podem ser

muito altos devido a complexidade de planejamento e a necessidade da verificação do circuito criado, principalmente se o design baseado em IP for usado. Vale ressaltar que o processo de verificação não exime o circuito de conter erros. Outra grande desvantagem dos SoCs é que eles não são adaptáveis. Isso quer dizer que os SoCs são integrados fortemente e tão pequenos a ponto que não possam ter sua estrutura adaptada e/ou melhorada para outros contextos. Além disso, se alguma coisa quebra dentro de um SoC, ele fica inutilizado, uma vez que não é possível repará-lo ou alterar somente a parte danificada. A única solução é trocar todo o SoC.

O uso dos SoCs incluem smartphones, comunicação de dados (WiFi, 3G, 4G) e atualmente, em aplicações de *Smart Home* e carros. As empresas que mais se destacam na fabricação desse tipo de sistema são Qualcomm, Samsung, MediaTek, Huawei, NVIDIA e Broadcom. A Qualcomm, NVIDIA e MediaTek fabricam SoCs e vendem para outras companhias. A Broadcom foca a sua fabricação em dispositivos de rede, como roteadores. A Samsung e Huawei além de fabricar SoCs para outras empresas, também criam smartphones a partir dos SoCs.

Atualmente, outros tipos de aplicação que vêm se favorecendo com a miniaturização e complexidade dos SoC são 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 (carros que estão com sistemas cada vez mais inteligentes). O CPS está um nível abaixo da Internet-das-coisas.

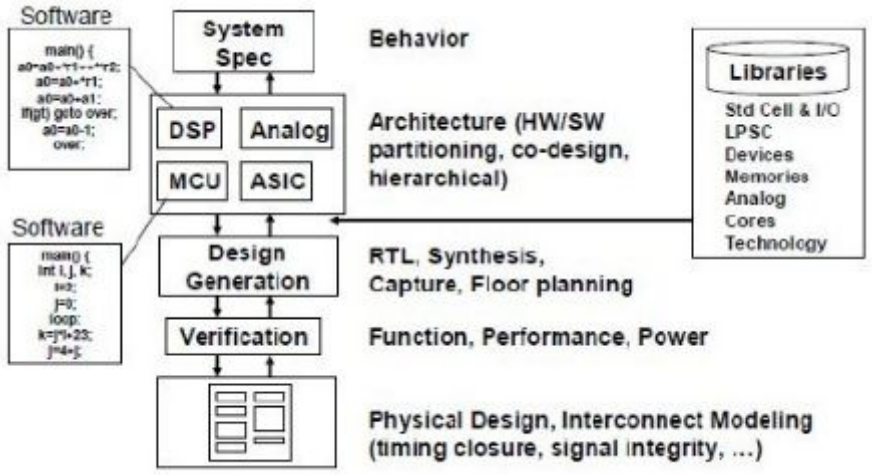
No contexto de Internet-das-coisas, os CPSs possuem um papel de destaque, uma vez que colaboram para construir sistemas inteligentes, como por exemplo as cidades inteligentes, as quais podem ser vistas como sistemas ciber-físicos em ampla escala. As cidades inteligentes possuem sensores monitorando indicadores 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 sistemas SoC ficam cada vez mais complexos e em evidência. Isso só é possível devido aos avanços da tecnologia que possibilitaram que sistemas completos pudessem ser implementados em um único chip. O processo do design é de suma importância, uma vez que o SoC foi fabricado, não há volta e diversos problemas podem surgir, desde transtornos ao usuário até perda de dinheiro. A evolução da tecnologia

e das técnicas de design estão proporcionando a criação de componentes mais especializados e adequados a certos problemas usando os PSoCs.

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

https://www.embarcados.com.br/hardware-business-model-canvas/
http://apt.cs.manchester.ac.uk/ftp/pub/apt/papers/LEMB_ToE09_O.pdf
http://www.cl.cam.ac.uk/teaching/1011/SysOnChip/socdam-notes1011.pdf
https://www.cs.ccu.edu.tw/~pahsiung/courses/soc/notes/soc01.pdf
https://pt.slideshare.net/ruliandi/system-on-chip-soc-44502780
https://www.digitalcitizen.life/soc-system-on-chip
https://www.design-reuse.com/articles/6481/why-platform-based-design-works-better-than-a-discrete-ip-approach-by-david-fritz-arc-international.html
file:///home/pedro/Desktop/Platform-Based_Design_and_Software_Design_Methodol.pdf
http://twins.ee.nctu.edu.tw/courses/ip_core_01/lab_hw_pdf/Overview.pdf



Esquema design