Uma Abordagem do Problema de Particionamento *Hardware* e *Software* para *Design* de Sistemas Computacionais *Wearables*

Discente: Rodolfo Labiapari Mansur Guimarães
Orientador: Ricardo Augusto Rabelo Oliveira

rodolfolabiapari@decom.ufop.br – Lattes: `http://goo.gl/MZv4Dc`
Departamento de Computação – Instituto de Ciências Exatas e Biológicas (DECOM-ICEB)
Universidade Federal de Ouro Preto (UFOP)
Ouro Preto - MG – Brasil

Última Atualização: 3 de maio de 2018

Sumário

Sumário

Sumário

FPGA

- FPGAs eram utilizados unicamente na protitipação ASIC.
- Mas com a elevação do custo de engenharia não recorrente[1].
- Interessou-se na utilização de FPGAs para SE devido [1]:
  - Suas vantagens em termos de flexibilidade de projeto e;
  - Custo zero de engenharia não recorrente citada
- Entretanto, enquanto configurar um *hardware* reconfigurável é uma tarefa fácil graças às ferramentas disponíveis hoje, *criar um design de hardware inicial não é* [2].

[1] NRE, que refere-se ao custo de pesquisa, *design*, desenvolvimento e teste de um novo produto e exigências de mercado.

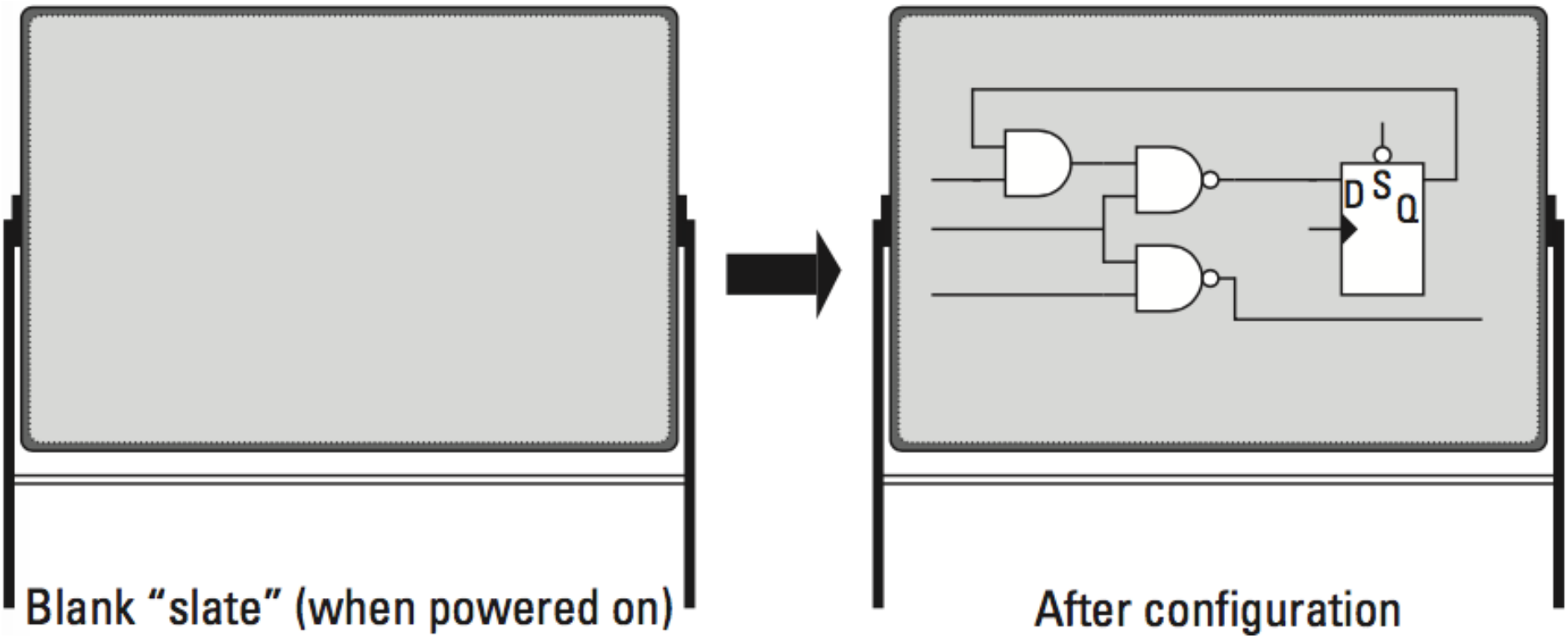

Figura 1: Ilustração em alto nível do funcionamento interno do FPGA. Fonte: [2].

Tecnologia dos FPGA

Vários módulos lógicos programáveis relativamente **pequenos** e independentes, interconectados.

- A maioria dos FPGAs utilizam *look-up table* (LUT) para criar as funções lógicas desejadas.
  - Uma LUT funciona como uma tabela-verdade, ou seja, registradores programáveis.
  - Cada módulo lida com até quatro ou cinco variáveis de entrada;
  - A saída é programada para criar a função combinacional armazenando valores verdadeiros e falsos adequado a cada combinação de entrada;
  - Eles não são associados a nenhum pino de entrada e saída (I/O, do inglês *Input and Output*).

Tecnologia dos FPGA

Vários módulos lógicos programáveis relativamente **pequenos** e **independentes**, **interconectados**.

- Os recursos de roteamento de sinal programável dentro do chip
  - Os atrasos de sinal em um projeto dependem do roteamento real de sinal selecionado pelo *software* de programação.
- Pinos de I/O são conectados ao bloco programável de entrada e saída que, por sua vez, é conectado aos módulos lógicos com linhas de roteamento selecionadas.
- Assim, FPGAs são chips que podem ser programados instantaneamente para funções de qualquer circuito digital [3].

Tecnologia dos FPGA

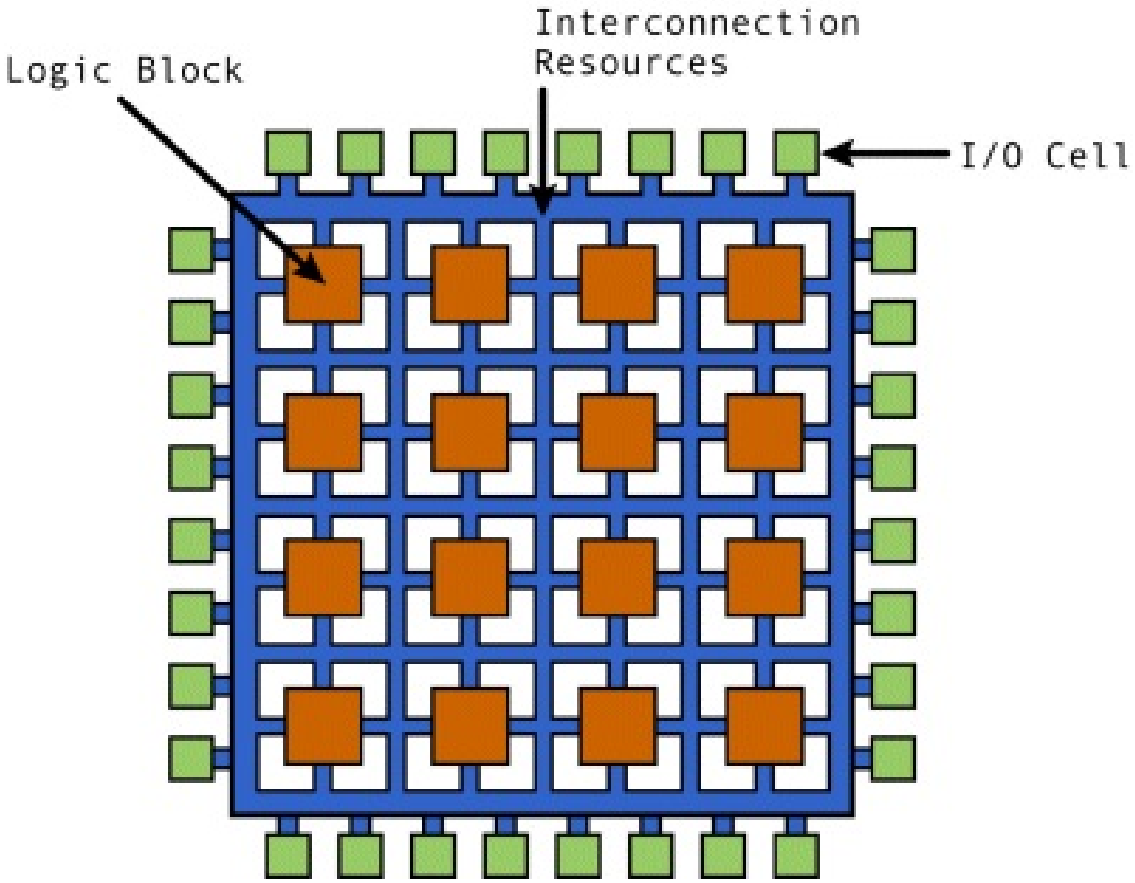

Figura 2: Exemplo da arquitetura internas de um FPGA. Fonte: `http://www.eetimes.com/document.asp?doc_id=1274496`. Acesso: 30/05/2017.

Hoje, esses dispositivos possuem [3]

- Mi de portas de lógica programável;
- Bi transistores;
- Blocos de *hardware* dedicados dedicados como
  - Memórias embarcadas;
  - Multiplicadores de ponto-fixo.

Tecnologia dos FPGA

Segundo [4]

- Podem fornecer uma série de opções de projeto sendo voltados para indústria e até mesmo educação.
- Ao utilizar tecnologia CMOS, o consumo de energia é relativamente baixo comparado com outras tecnologias
  - Pode-se confeccionar em nível de tensão elétrica, frequências e cargas para os sinais de I/O.
- O mercado fornece diferentes graus de velocidade de FPGA a fim de que o projetista utilize o mais adequado ao projeto.
- Entretanto, um dispositivo FPGA pode ser configurado para um número infinito de projetos.
  - Não é possível afirmar o montante de dissipação de energia para um dispositivo FPGA.

Tecnologia dos FPGA

- Ao utilizar FPGA, é possível obter inúmeras vantagens [4, 5]
  - Como são dispositivos programáveis, a mesma funcionalidade pode ser obtida com um CI ao invés de diversos circuitos individuais;
  - Maior confiabilidade;
  - Menor espaço ocupado na placa, consumo de energia, complexidade de desenvolvimento e, geralmente, menor custo de fabricação.

Hard e Software Cores

A CPU pode ser implementada em duas formas, sendo estas *hard* e *soft cores* [5].

- ***Hard Core*:** *core* dedicado;
- ***Soft Core*:** feito por meio da sintetização e mapeamento de um processador no FPGA com seus recursos lógicos.

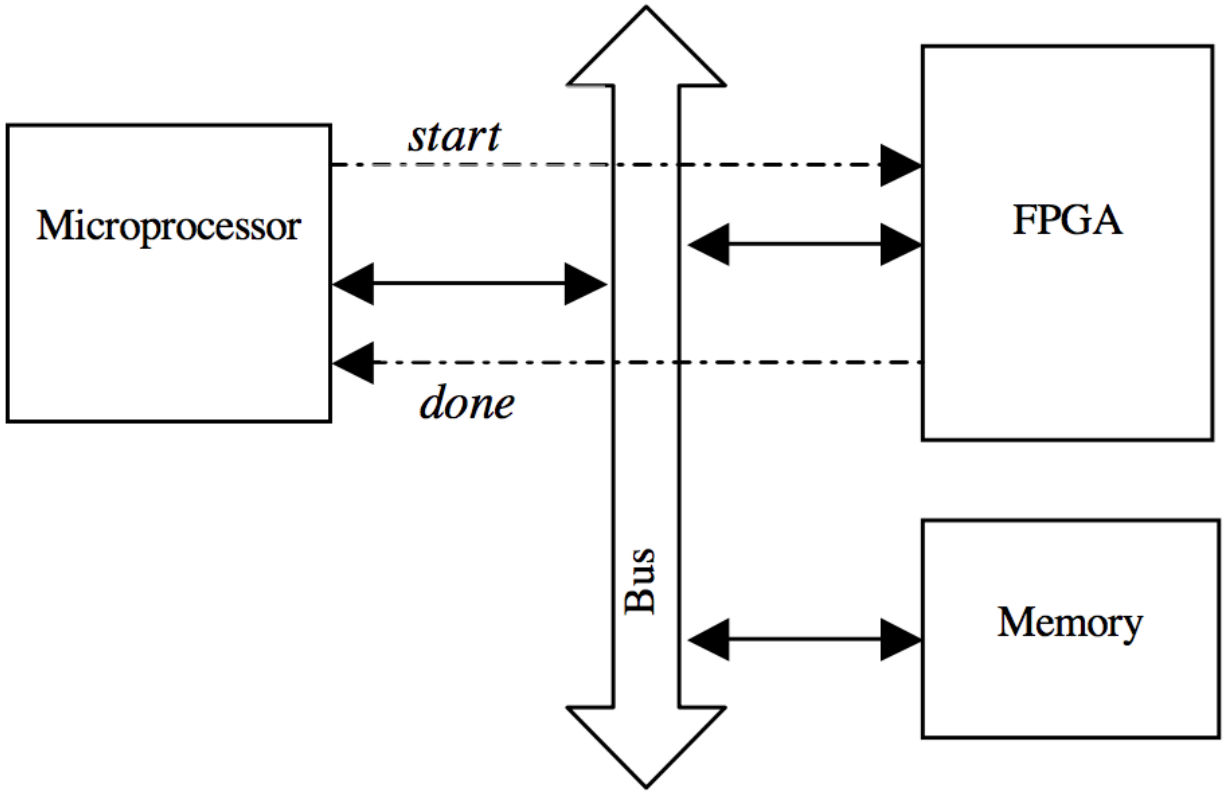

Figura 3: Visão geral de um SoC FPGA.

Hardware Description Language (HDL)

- Classes de linguagens de computação usados para descrever formalmente um circuito eletrônico.
- É capaz de descrever o comportamento temporal ou a estrutura de circuito (espacial) de um sistema eletrônico.
  - Origem: documentação do comportamento do *hardware* [2].
- Amplamente utilizada em *design* de *hardware*
  - Pois possui detalhação altíssima do *hardware*.
- A utilização de uma HDL é o primeiro passo no processo de síntese em FPGA [6].
  - A propriedade única deste é que, com esse processo de síntese, é possível alterar o código HDL, e sintetizar no mesmo dispositivo para testar;
  - Quantas vezes forem necessárias, sem custo adicional.
- Maioria de engenheiros utilizam-na.
  - Mas possuem um nível elevado de complexidade [3], existe outras linguagens disponíveis para uso [2].

Hardware Description Language (HDL)

- Sintetizador em Alto Nível (HLS) são procedimentos que sintetizam códigos em alto nível para HDLs [3].
  - Podem reduzir os longos ciclos do processo de *design* de *hardware* e ainda;
  - Trazer melhoria em performance e eficiência energética.
- Ferramentas como o *framework* LegUp e OpenCL [7].
  - Entregar um bom HLS;
  - Amenizar esses problemas de projetos

HLS

LegUp High-Level Synthesis.

- Programa padrão *C* como entrada e automaticamente compila o programa para dispositivos FPGA [8].
  - Totalmente em *hardware* e outros híbridos (*hard* e *soft cores*).

OpenCL.

- API comum para execução de programas em sistemas heterogêneos.
  - Processadores *multicores*, GPUs ou outros aceleradores [9, 10].
- Nível de paralelismo em tarefas e dados.
- Define-se um *core Runtime Environment* e seus dispositivos.

Sumário

Profile

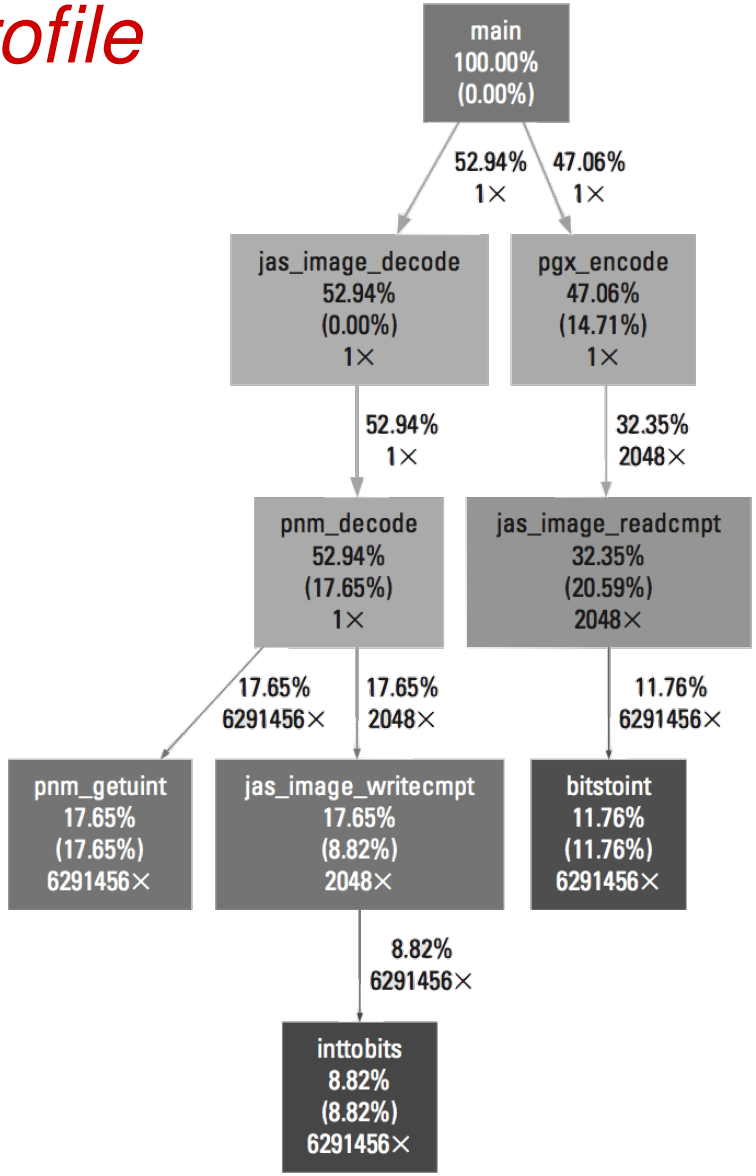

Figura 4: *Profile* da codificação de imagem em formato JPEG. Fonte: [2].

- Realiza-se interrupções periódica no programa e amostra o seu *program counter*
  - Utiliza-se de um histograma para contar o endereço particular;
  - E assim, calcular a fração aproximada do tempo total de execução gasto em suas partes.
- Distribuições GNU/Linux possuem a ferramenta `gprof` na qual realiza o cálculo de tais informações de *software* [11].

Sumário

Sistemas Computacionais *Wearables*

Definição [12]:

Com a possibilidade de ter um computador acoplado ao corpo, proporciona-se ao usuário um nível de informações contextualizadas dentro de um ambiente interativo.

Definição [13]:

Quando possui sua '*wearability*' sendo este definido como a interação entre o corpo humano e o objeto *wearable* estendendo ao corpo em movimento.

Sistemas Computacionais *Wearables*

- Introduzido em 1968 e novamente em 1996 e 1997 [14, 15, 16]
- Entretanto, foi depende diretamente da miniaturização dos módulos eletrônicos.
- Exemplo: *smartwatches*, *fitness trackers*, óculos, equipamentos de VR e AR, nas indústrias e nas atividades pessoais de usuários.
- Dados de teclados ou *display* LCD contradiz com o paradigma de operações *hands-free* e a propriedade de computação *wearable* discreta [5].
  - Tais componentes são *prestadores de auxílio à tarefas pessoais de usuários*.
  - Sistemas distribuídos construídos a partir desses equipamentos interativos são *ineficientes*.
    - Falta-se a especialização de *componentes individuais de propósito específico*.

Sistemas Computacionais *Wearables*

- Sendo miniaturizados, aumenta-se a sua atração devido à
  - Fácil disponibilidade dos dispositivos;
  - Preço baixo;
  - Ferramentas de desenvolvimento aplicações específicas incluindo compiladores e sistemas operacionais, mas não são otimizados para uso *wearable*.

Sistemas Computacionais *Wearables*

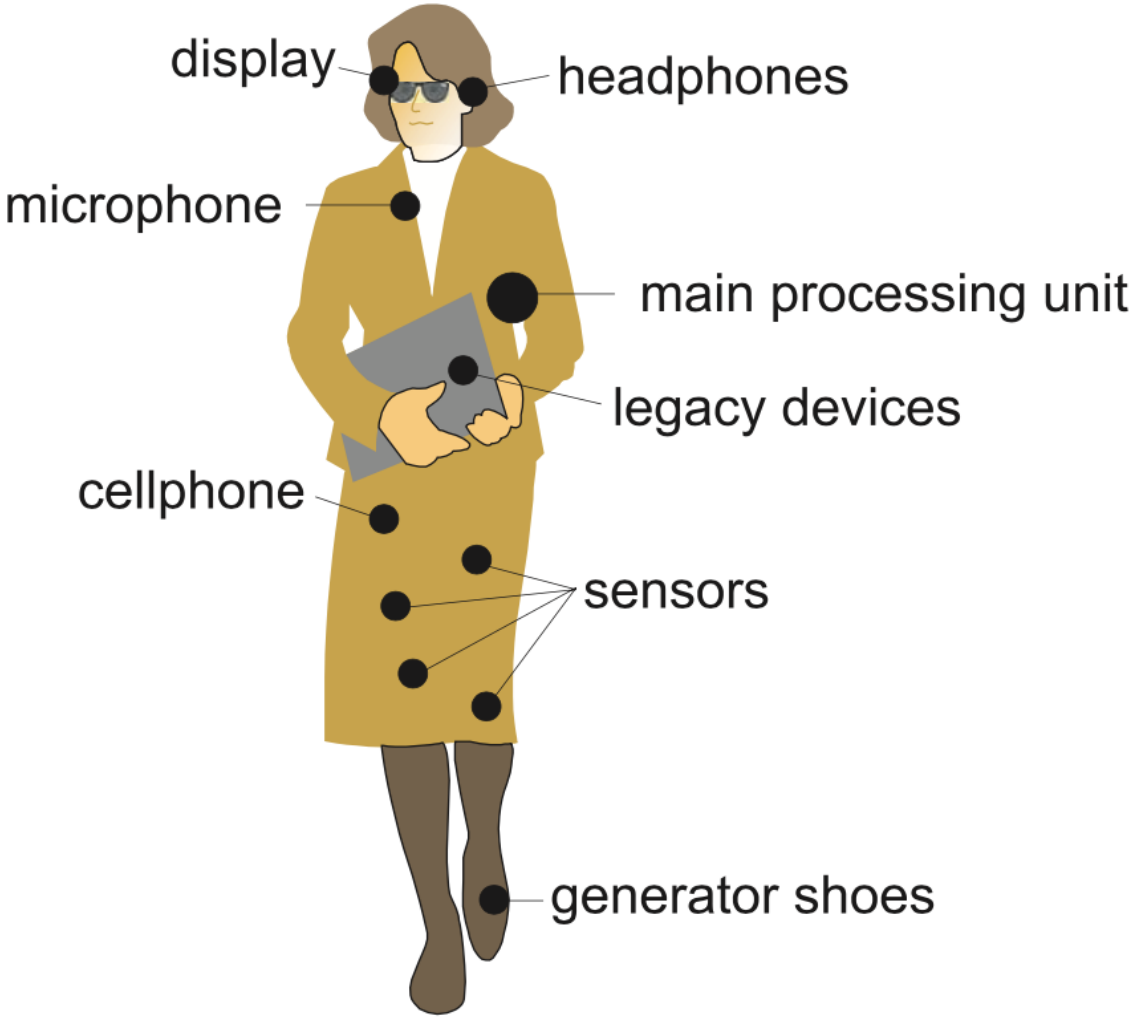

Figura 5: Exemplificação de alguns dispositivos *wearables*. Fonte: [5].

Distribuição espacial dos módulos pelo corpo, a comunicação:

- Torna-se importante no consumo de energia [17].
- Pode ser mista, entre conexões cabeadas e sem-fios, mas a predominância sem-fio por causa da mobilidade [5].

Características de um *Wearable*

- Caracteriza-se um *wearable* acordando às classificações pré-estabelecidas em relação à suas funcionalidades e requisitos de *hardware*.
  - Mesmo existindo inúmeros modelos e características diferentes, muitas soluções em *hardware* compartilham uma arq. e org. interna de recursos comum.
  - Também podem ser expandidos às características de OS [18, 12].

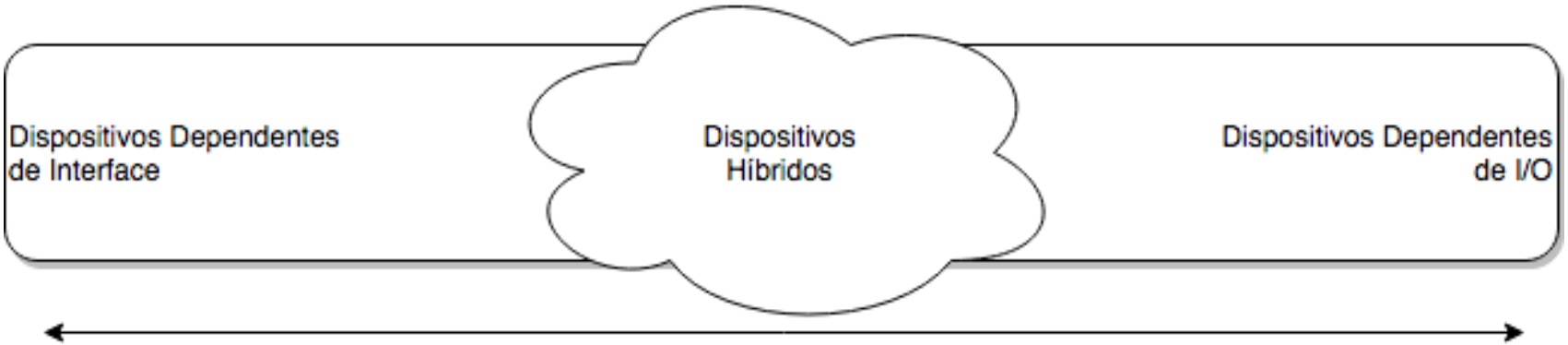

Figura 6: Classificação de *wearables*. Fonte: Adaptado de [12].

Características de um *Wearable*

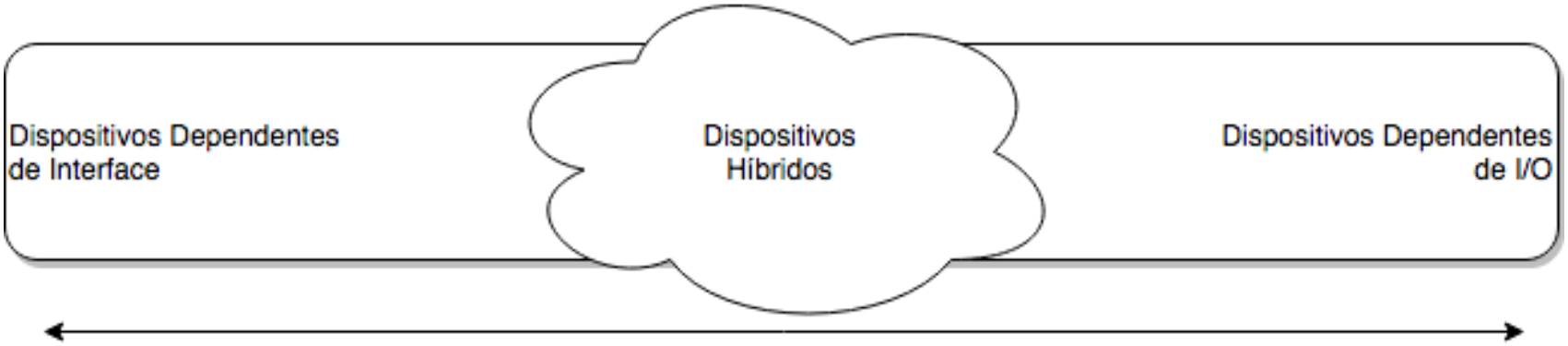

Figura 7: Classificação de *wearables*. Fonte: Adaptado de [12].

- A separação dos conceitos computação *wearable* e IoT ainda não estão claros segundo a bibliografia.
- OS para *wearable* são comumente focado em um único tipo de seguimento de produto como os *smartwatches*, proporcionando aos desenvolvedores um meio para sua aplicação final além de um produto de alta qualidade.
- Também, atualmente, não existe nenhum sistema que satisfaça todos os requisitos de todas as classificações [12].

Características de um *Wearable*

- Já [19] caracteriza como um sistema *cyber*-físico[2] móvel autônomo.
- Podem ser utilizados para aplicações de consumidores, extensões ou reposição de capacidades humanas, ou industriais
- Representam uma grande parte da heterogeneidade de sistemas embarcados,
  - Desde um dispositivo inteligente integrado à roupa, focado no campo de computação *mobile* pessoal, até indústrias como dispositivos de segurança.
  - Podem trabalhar de forma colaborativa com *smartphones*, redes e outros sistemas criando um sistema mais complexo.

[2] Sendo *cyber-* uma combinação dos termos 'computador', 'rede de computadores' ou 'realidade virtual' com um segundo termo, no caso o 'físico' oriundo de circuitos.

Características de um *Wearable*

Segundo [20], a distribuição de elementos computacionais, sensores, em objetos mundanos em nosso cotidiano se adéqua à computação ubíqua.

- Na qual computação *wearable* também não foge deste ramo
  - Superfícies de roupas são uma plataforma de suporte ideal para uma grande quantidade de sensores.

- A restrição de tamanho relaciona-se com a própria qualidade do equipamento
  - Levando muitos atuadores e sensores a serem simplistas.

Sendo *wearables* uma subclasse de SE, há restrições de *design* [5]:

- **Performance de multi-nós:**
  - Requer uma performance base fixa para tarefas que não mostram altas demandas computacionais nem restrições de tempo rigorosa.
  - E também *wearables* executam rajadas de tarefas de computação intensiva que consideram restrições de tempo-real. Não realizando as tarefas, o sistema torna-se inaplicável.
- **Gasto energético consciente:**
  - É essencial num sistema se manter ativo e funcional num certo período de tempo.
  - O alto gasto de energia conduz problemas como gerenciamento energético eficiente e energização dinâmica.
- **Flexibilidade:**
  - Utilizado em situações altamente dinâmicas.
  - Os requisitos de aplicação variam de acordo com as escolhas do usuário, mas também com o contexto e local utilizado;
  - Trocas de roupa;
  - Critérios: confiabilidade, disponibilidade e itens dependentes de sua forma como volume e peso.

Wearable e FPGA

- Dessa forma, pode-se estabelecer que [5]:
  - Os requisitos flexíveis em um *wearable* demanda um sistema de computação programável de propósito geral;
  - Enquanto os requisitos de alta performance e consumo de energia consciente demandam um sistema computacional especializado.
- Então, utilizaram-se de um *hardware* reconfigurável para incorporar ao sistema.
  - O trabalho exibe um sistema *wearables* compreendendo de um processador de até médio porte em termos de processamento e módulos reconfiguráveis.
- A utilização de *hardware* reconfigurável nos permite alcançar alto processamento com maior eficiência energética em relação à processadores para tarefas de computação intensiva em tempo real.

Sumário

Particionamento para Sistemas Gerais

Definição Introdutória Particionamento [21]

Problema de otimização na qual utiliza-se métodos para auxiliar na decisão de cada componente do *design* referencial.

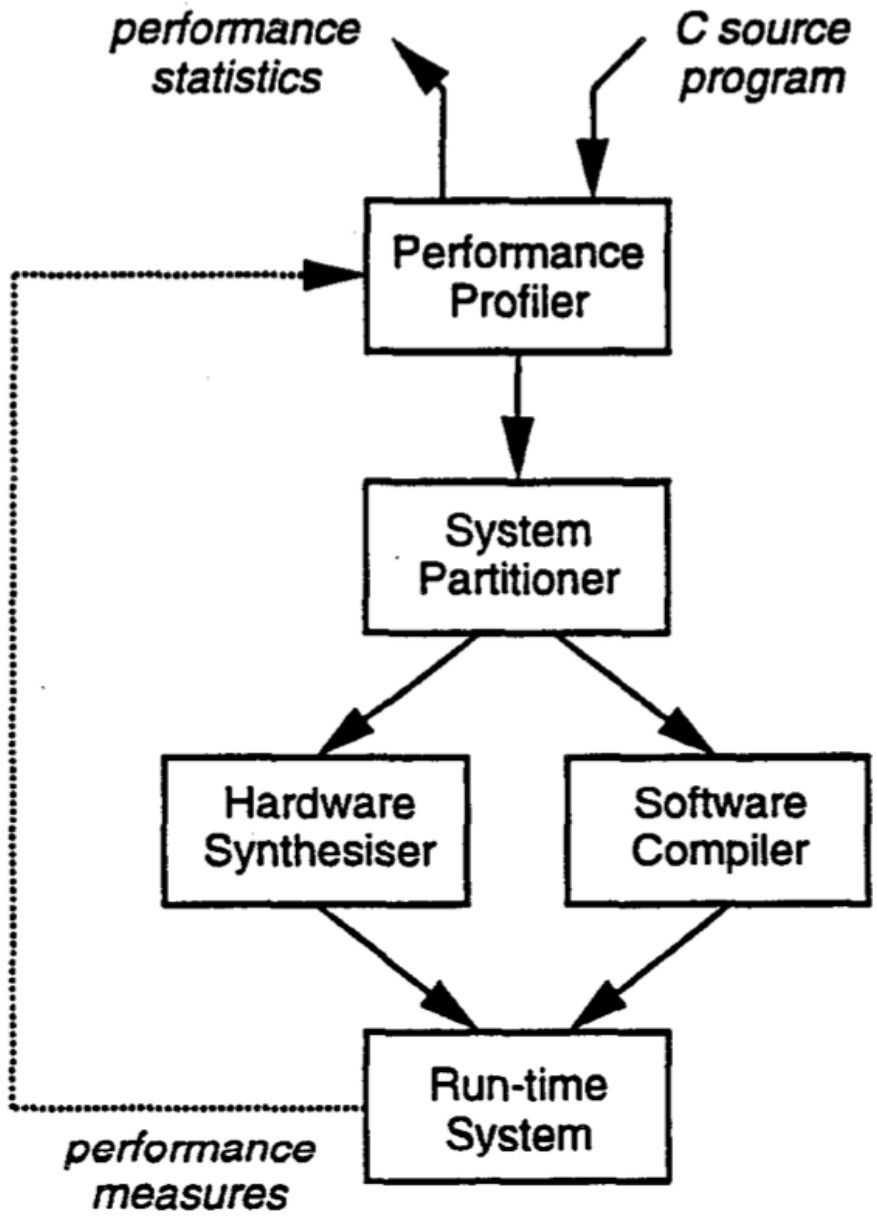

Figura 8: Metodologia de *codesign*. Fonte: [22].

[22]

- Utilizam do particionamento para aprimorar a performance do seu *software*.

- Realiza-se a tentativa de tratar regiões críticas da aplicação
  - A solução em *software* não pode chegar à restrições de performance requeridas;

- Apresenta-se uma metodologia para desenvolvimento *codesign*.
  - Consiste na construção do código da aplicação em *C* e sua análise;

  - Regiões críticas são identificadas e particionadas.

- Feito isso, realizar-se mensurações
  - Um novo particionamento é feito.

- Utiliza-se de um FPGA.

Otimizações

- [23] pioneiro no método de otimização **dinâmico**.
- Processos:
  1. Detecta a região de mais executada e;
  2. Reimplementa-a em *hardware* de FPGA.
- Afirmam isso vantagens comparado com abordagens tradicionais manuais.
- Como Stitt, outros propuseram algoritmos **exatos** baseados em estratégias:
  - *Branch-and-Bound* [24, 25, 26];
  - Programação dinâmica [27, 28] e;
  - Programação Linear inteira [29].

Otimizações

- [30] exame da relação entre o *footprint* do FPGA e o *speedup* do *software*.
  - FPGA utilizado na implementação de *loops* e sub-rotinas críticas.
- Abordagem direta utilizando ferramentas comerciais como *Synopys' Nimble Compiler* e *Proceler* para a facilitação do processo.
- [31] parte da otimização *position disturbed particle swarm* com otimização invasiva de *weed* como o método de particionamento *hardware* e *software*.

[32]

- “Particionamento depende da exploração de caracterização, estimação e *design* espacial das métricas de custo e performance sistêmica.”
  - Citam o *codesign* é tão complexo que a simplificação do particionamento só para duas partes não é suficiente para a representação do problema como um todo.

- Incluem parâmetros chave de *design* e uso de recursos que deveriam ser incorporados à modelagem do sistema
  - Considera a modelagem de incerteza para particionamento de sistemas com um conjunto melhorado de parâmetros para compartilhamento de recursos *hardware* e *software*.

[32]

- Esse terceiro item a ser considerado ao problema de particionamento podem ser definidos em três tipos, sendo esses:
  1. **Conjunto de recursos necessários para particionar uma dada tarefa:** sendo esses RAM, ROM, DPS, blocos IP do inglês, *intelectual propriety*, entre outros;
  2. **Parâmetros de configurações que provê diferentes *trade-off* entre recursos e performances:** exemplo circuitos de multiplicações sintetizados ou não;
  3. **Dificuldade da mensura de desempenho e impacto no uso de recursos em várias configurações de partição com precisão precisando ter em mente a partição com a natureza incerta[3] dessas estimativas não precisas.**

[3] Teoria da incerteza: Matemática axiomática para modelagem de graus de crença.

Embarcados

- SE é amplamente pesquisado desde 90 [33, 34, 35, 36, 37].
- [1] Descreve um particionamento além de uma abordagem de escalonamento para DRESs[4]
  - Para o escalonamento, [1] Algoritmo Genético.
- Contribuições
  - Análise de tempo de configuração e;
  - Análise do tempo de reconfiguração parcial do FPGA.

[4] Do inglês *dynamically reconfigurable embedded systems*.

a

- [38] descreve versões diferentes do particionamento
  - Sistemas de tempo-real e custo restringido respectivamente;
  - Fornece análise matemática formal do problema, provando que é $\mathcal{NP}$-difícil.
- Utiliza
  - Programação linear inteira resolvendo de forma otimizada, e;
  - Outra utilizando GA com soluções próximas ao ótimo global.

title

- [25] descreveram uma primeira tentativa para um algoritmo exato, não heurístico, para o problema de particionamento.

- Implementa-se a estratégia *branch-and-bound* como um *framework*, permitindo o incremento de outros algoritmos.

- Realizaram investigações para incrementar a eficiência do algoritmo com as técnicas:
  - *Lower bounds based on LP-relaxation*, uma mecânica de inferência customizada, condições não-triviais necessárias baseadas num algoritmo *minimum-cut*, e diferentes heurísticas com passos pré-otimizados.

- Demonstram que podem resolver problemas complexos em tempo razoável.
  - O resultado é em entorno de dez minutos mais rápido que algoritmos exatos anteriores baseados em programação linear inteira para os testes realizados.

title

- [39] aplicou otimizações sobre o tempo de execução e gasto energético.
- O algoritmo proposto destina-se a alcançar um particionamento de grafos à procurar o melhor conjunto da relação energia e tempo de execução.
- Comparou-se com *Simulated Annealing*, Busca Tabu e Genético
  - Mostra-se mais adequado para aplicações em *cores* de sistemas embarcados que necessitam do equilíbrio no *tradeoff* de sistemas embarcados.

title

- [40] utiliza do GA para solucionar o problema.
- Propõe novas abordagens usando técnicas de verificação baseadas nas teorias de módulo de satisfação (SMT, do inglês *satisfiability modulo theories*).
- Apresentam um exemplo de particionamento, modelam e solucionam-o usando três diferentes técnicas
  - A ideia principal é aplicar o método de verificação SMT ao particionamento, e;
  - Comparar os resultados com otimizações tradicionais como ILP e GA.

- O *survey* [19] considera os aspectos de uma aplicação embutida
  - Suas tecnologias de *design* com foco sistemas móveis modernos e *wearables*.

- Cita-se dois paradigmas de desen. para SE sobre sistema de multi-processadores heterogêneos. O paradigma de sistemas
  - ***Life-inspired*:**
    - Especifica princípios, características e organização funcional e estrutural de SE com analogia à vida de um organismo inteligente;
    - Soluções de mecanismos e arquiteturas de sistemas para sua implementação.
  - ***Quality-driven*:**
    - Solução para o *design* de disp. que necessitam de exigências de *real-time*, baixo consumo de energia, entre outros;
    - Dessa forma, especifica-se qual a nova qualidade do sistema a ser requeria e como esta meta é obtida.
    - Assim, "define-se *qualidade* de uma solução sistêmica proposta como o *total de sua eficácia*[5] **e** *eficiência*[6] *na resolução do problema real*".

[5] Grau em que uma solução atinge seus objetivos.
[6] Grau em que uma solução usa recursos para realizar seus objetivos.

title

Entretanto, descreveram ao final que [19]

"Enquanto *designsers* aprenderam bastante na **construção** de plataformas de *hardware* heterogêneos altamente paralelos, métodos e ferramentas automatizadas para a sua programação e o paralelismo do algoritmo, bem como o *codesign* coerente da arquitetura *hardware* e *software* **ainda estão atrasados perante à tecnologia**".

Wearablee FPGA

- [5, 41, 42, 43, 44] que relacionam FPGAs com aplicação *wearable*.
- Entretanto, nenhum menciona análise metodológica do problema de particionamento *hardware* e *software* para *design* de sistemas computacionais *wearables*.

Sumário

- Metodologia baseada em [2, 38, 21, 25, 39].

Sumário

Grafo de Controle de Fluxo (GCF)

- Descrever um sistema, tanto em *software* ou *hardware*, livre de uma especificação de forma
  - Desenvolvendo rápidos protótipos, referenciado como *design* de referência de *software*.

- Generalização de uma especificação bastante completa, eliminando qualquer tipo de incertezas sobre o comportamento do sistema com o simples fato de analisar o *design* referencial do *software*.

- O sistema inicial é dado como grafo de tarefas/rotinas, define-se $G = (B, F)$ onde B são vértices que representam os blocos básicos e F são arestas que indicam a todas as possibilidades de caminhos entre os blocos.

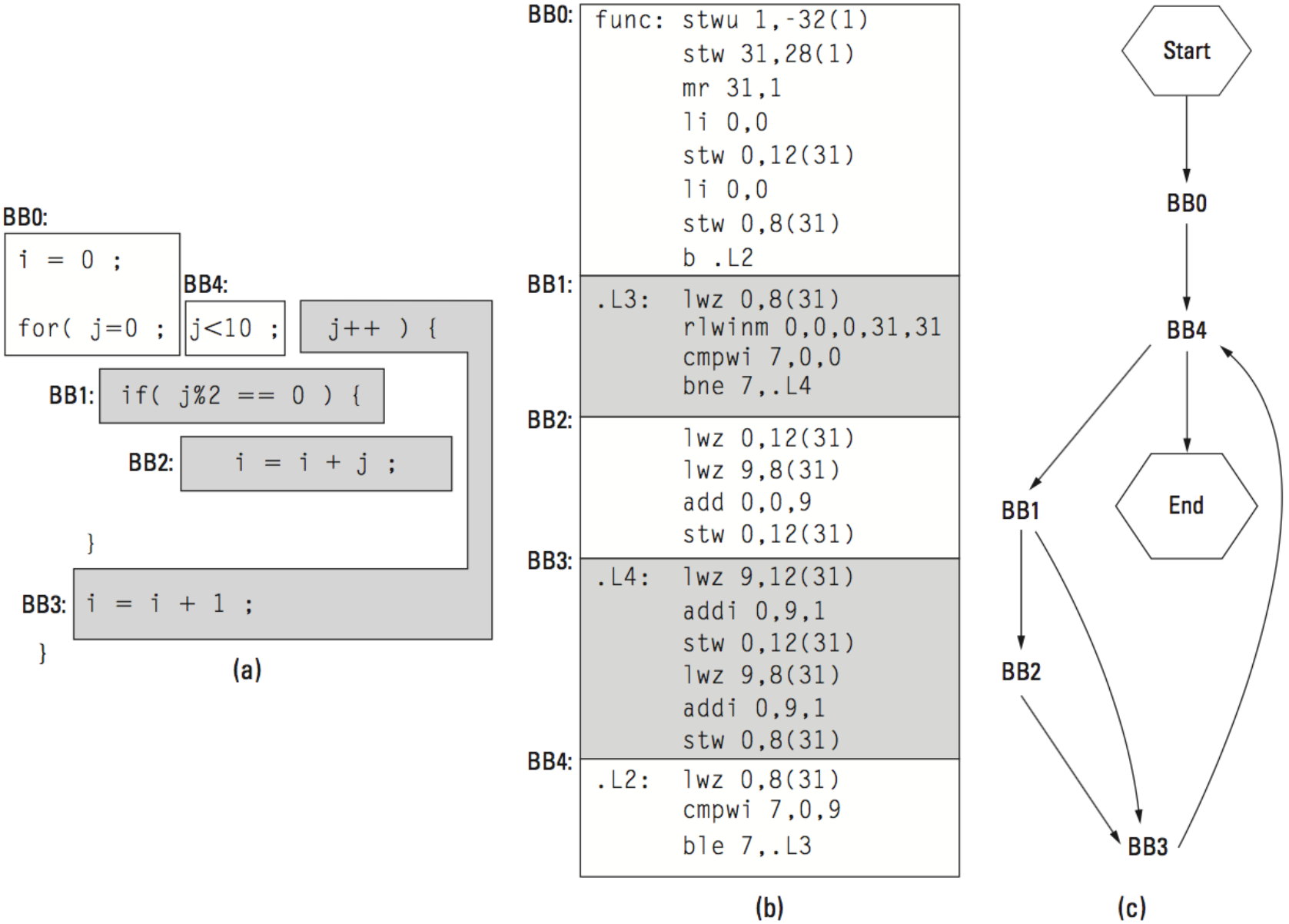

Figura 9: Identificação de blocos básicos e a representação de cada modelo de descrição de *software*.

Grafo de Controle de Fluxo (GCF)

- A decomposição de um DRS pode gerar dois componentes:
  - Porção a ser realizada em *hardware* e outra executada em *software*;
- Essa decisão de divisão é chamada de problema de particionamento.
- Para sistemas baseados em Plataformas FPGA, particionamento é um sub-problema de um problema mais geral localizado no *codesign*, onde refere-se ao *design* cooperativo.

Grafo de Chamada

Modelado uma sub-rotina de um DRS utilizando o GCF, define-se o Grafo de Chamada (GC).

- Consiste num conjunto de GCFs, um por sub-rotinas, ou seja, $\mathcal{C} = C_0, C_1, \ldots C_{n-1}$ onde $C_i = (B_i, F_i)$ representa o GCF de uma sub-rotina i.
- Sendo assim, o grafo estático de chamada da aplicação é escrito por $\mathcal{A} = (\mathcal{C}, \mathcal{L})$ onde
  - $\mathcal{A}$ representa uma aplicação específica e;
  - $\mathcal{L} \subseteq \mathcal{C} \times \mathcal{C}$.

Duas sub-rotinas são relacionadas

Se podem ser determinadas, no tempo de compilação, que a sub-rotina i tem potencial de invocar a sub-rotina j, ou seja, $(C_i, C_j) \in \mathcal{L}$.

Sumário

OpenCL

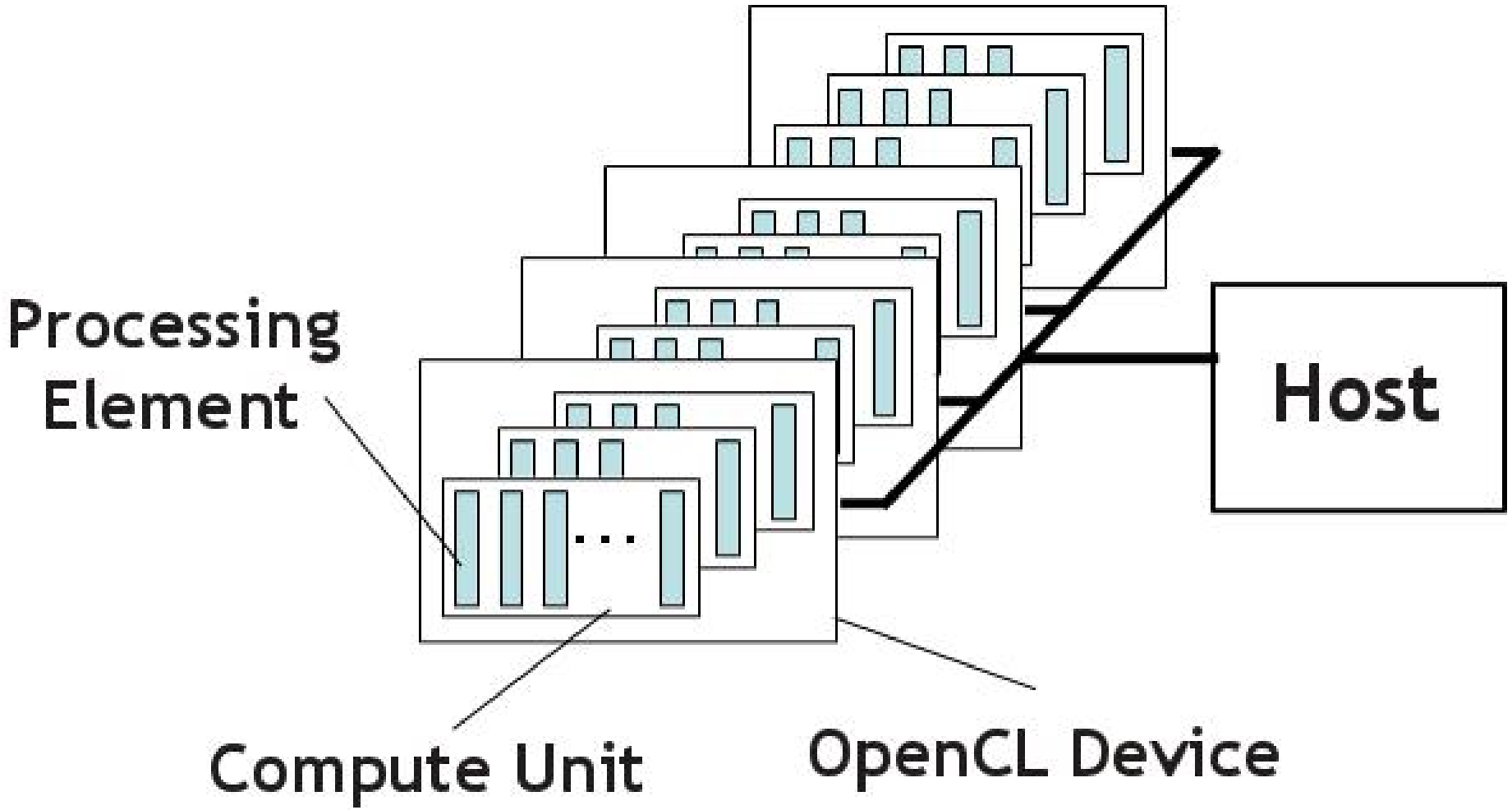

Figura 10: Sistema hierárquico de um projeto em OpenCL. Fonte: `https://handsonopencl.github.io/` Acessado em: 07/08/2017.

Dúvidas, Sugestões ou Reclamações?

- rodolfolabiapari@decom.ufop.br
- https://www.guerrillamail.com/

Uma Abordagem do Problema de Particionamento *Hardware* e *Software* para *Design* de Sistemas Computacionais *Wearables*

Discente: Rodolfo Labiapari Mansur Guimarães
Orientador: Ricardo Augusto Rabelo Oliveira

rodolfolabiapari@decom.ufop.br – Lattes: `http://goo.gl/MZv4Dc`
Departamento de Computação – Instituto de Ciências Exatas e Biológicas (DECOM-ICEB)
Universidade Federal de Ouro Preto (UFOP)
Ouro Preto - MG – Brasil

Última Atualização: 3 de maio de 2018

Sumário

MEI, B.; SCHAUMONT, P.; VERNALDE, S.
A hardware-software partitioning and scheduling algorithm for dynamically reconfigurable embedded systems.
Proceedings of ProRISC, 2000.

SASS, R.; SCHMIDT, A.
Embedded Systems Design with Platform FPGAs Principles and Practices.
1. ed. Morgan Kaufmann, 2010.

CHOI, J.
From Software Threads to Parallel Hardware with LegUp High-Level Synthesis.
2016.

TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L.
Sistemas digitais: princípios e aplicações.
Prentice Hall, 2011.
v. 11.


PLESSL, C.; ENZLER, R.; WALDER, H.; BEUTEL, J.; PLATZNER, M.; THIELE, L.; TRöSTER, G.
The case for reconfigurable hardware in wearable computing.
Personal and Ubiquitous Computing, v. 7, n. 5,
p. 299–308, 2003.


SMITH, D.; BY-ZAMFIRESCU, A. F.
HDL Chip Design: A practical guide for designing, synthesizing and simulating ASICs and FPGAs using VHDL or Verilog.
1998.


TREVETT, N.
OpenCL: The open standard for heterogeneous parallel programming.
Group, , n. November, 2008.


CANIS, A.; CHOI, J.; ALDHAM, M.; ZHANG, V.; KAMMOONA, A.; ANDERSON, J. H.; BROWN, S.; CZAJKOWSKI, T.
LegUp: High-Level Synthesis for FPGA-Based Processor/Accelerator Systems.

Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays - FPGA '11, p. 33, 2011.

SHAGRITHAYA, K.; KEPA, K.; ATHANAS, P.
Enabling development of OpenCL applications on FPGA platforms.

In: .
c2013.
p. 26–30.

CZAJKOWSKI, T. S.; AYDONAT, U.; DENISENKO, D.; FREEMAN, J.; KINSNER, M.; NETO, D.; WONG, J.; YIANNACOURAS, P.; SINGH, D. P.
From OpenCL to high-performance hardware on FPGAs.
In: .
c2012.
p. 531–534.

GRAHAM, S. L.; KESSLER, P. B.; MCKUSICK, M. K.
Gprof: A call graph execution profiler.

In: .
SIGPLAN '82. New York, NY, USA: ACM, c1982.
p. 120–126.


AMORIM, V. J. P.; DELABRIDA, S.; OLIVEIRA, R. A. R.
A constraint-driven assessment of operating systems for wearable devices.
In: .
c2017.
p. 150–155.


GEMPERLE, F.; KASABACH, C.; STIVORIC, J.; BAUER, M.; MARTIN, R.
Design for wearability.
In: .
IEEE Comput. Soc, c1998.
p. 116–122.


SUTHERLAND, I.
A head-mounted three dimensional display.

Proceedings of the December 9-11, 1968, fall joint, 1968.


MANN, S.
Smart clothing: the shift to wearable computing.
Communications of the ACM, v. 39, n. 8,
p. 23–24, 1996.


MANN, S.
Wearable computing: A first step toward personal imaging.
Computer, 1997.


KYMISSIS, J.; KENDALL, C.; PARADISO, J.
Parasitic power harvesting in shoes.
, 1998. Digest of . . ., 1998.


DELABRIDA, S.; D'ANGELO, T.; OLIVEIRA, R.; LOUREIRO, A.
Building wearables for geology.
2016.


JÓŹWIAK, L.
Advanced mobile and wearable systems.
Microprocessors and Microsystems, 2017.


Van Laerhoven, K.; SCHMIDT, A.; GELLERSEN, H. W.
Multi-sensor context aware clothing.
In: .
c2002.
v. 2002-January.
p. 49–56.


ARATO, P.; MANN, Z. A.; ORBAN, A.
Algorithmic aspects of hardware/software partitioning.
Acm Transactions on Design Automation of Electronic Systems, v. 10, n. 1,
p. 136–156, 2005.


EDWARDS, M.; FORREST, J.
A development environment for the cosynthesis of embedded software/hardware systems.
and Test Conference, 1994. EDAC, The . . ., 1994.


STITT, G.; LYSECKY, R.; VAHID, F.
Dynamic hardware/software partitioning: a first approach.

Proceedings 2003. Design Automation Conference (IEEE Cat. No.03CH37451), p. 250–255, 2003.


JIGANG, W.; THAMBIPILLAI, S.
A branch-and-bound algorithm for hardware/software partitioning.
Signal Processing and Information, 2004.


MANN, Z.; ORBÁN, A.; ARATÓ, P.
Finding optimal hardware/software partitions.
Formal Methods in System Design, 2007.


STRACHACKI, M.
Speedup of branch and bound method for hardware/software partitioning.
Information Technology, 2008. IT 2008. 1st, 2008.


MADSEN, J.; GRODE, J.; KNUDSEN, P. V.; PETERSEN, M. E.; HAXTHAUSEN, A.
LYCOS: the Lyngby Co-Synthesis System.
Design Automation for Embedded Systems, v. 2, n. 2, p. 195–235, 1997.


WU, J.; SRIKANTHAN, T.
Low-complex dynamic programming algorithm for hardware/software partitioning.
Information Processing Letters, v. 98, n. 2, p. 41–46, 2006.


NIEMANN, R.; MARWEDEL, P.
An algorithm for hardware/software partitioning using mixed integer linear programming.
Design Automation for Embedded Systems, 1997.


NEMATBAKHSH, S.; STITT, G.; VAHID, F.; NEMATBAKHSH, S.; STITT, G.; VAHID, F.
The effect of fpga size on software speedup from hardware/software partitioning.


YAN, X.-H.; HE, F.-Z.; CHEN, Y.-L.
A Novel Hardware/Software Partitioning Method Based on Position Disturbed Particle Swarm Optimization with Invasive Weed Optimization.

Journal of Computer Science and Technology, v. 32, n. 2, p. 340–355, mar 2017.


WANG, R.; HUNG, W. N. N.; YANG, G.; SONG, X.
Uncertainty model for configurable hardware/software and resource partitioning.
IEEE Transactions on Computers, v. 65, n. 10, p. 3217–3223, oct 2016.


ERNST, R.; HENKEL, J.; BENNER, T.
Hardware-Software Cosynthesis for Microcontrollers.
IEEE Design and Test of Computers, v. 10, n. 4, p. 64–75, 1993.


GUPTA, R. K.; BY-MICHELI, R. K.; DE, G.
Co-synthesis of hardware and software for digital embedded systems.
Kluwer Academic Publishers, 1995.


HARDT, W.
An automated approach to HW/SW-codesign.

In: .
Number figure 1. c1995.
p. 4.


GAJSKI, D.; VAHID, F.; NARAYAN, S.; GONG, J.
Specification and design of embedded systems.
1994.


BOLSENS, I.; De Man, H. J.; LIN, B.; Van Rompaey, K.; VERCAUTEREN, S.; VERKEST, D.
Hardware/software co-design of digital telecommunication systems.
Proceedings of the IEEE, v. 85, n. 3,
p. 391–417, 1997.


ARATÓ, P.; JUHÁSZ, S.; MANN, Z. Á.; ORBÁN, A.; PAPP, D.
Hardware-software partitioning in embedded system design.
In: .
c2003.
p. 197–202.


HASSINE, S. B. H.; JEMAI, M.; OUNI, B.
Power and Execution Time Optimization through Hardware Software Partitioning Algorithm for Core Based Embedded System.
Journal of Optimization, 2017.


TRINDADE, A. B.; CORDEIRO, L. C.
Applying SMT-based verification to hardware/software partitioning in embedded systems.
Design Automation for Embedded Systems, v. 20, n. 1, p. 1–19, mar 2016.


AHOLA, T.; KORPINEN, P.; RAKKOLA, J.; TEEMU, R.
Wearable FPGA Based Wireless Sensor Platform.
Proceedings of the 29th Annual International Conference of the IEEE EMBS, p. 2288–2291, aug 2007.


ABDELHEDI, S.; BOURGUIBA, R.; MOUINE, J.; YOUSSEF, A.
Fall Detection FPGA-Based Systems : A Survey.

International Journal of Automation and Smart Technology, v. 6, n. 4,
p. 191–202, dec 2016.

NARUMI, T.
An FPGA-Based Tiled Display System for a Wearable Display.
The Fifth International Conference on Informatics and, 2016.

LEE, B.
Design and Implementation of a Face Recognition System-on-a-Chip for Wearable / Mobile Applications.
Journal of Korea Multimedia Society, v. 18, n. 2,
p. 244–252, 2015.

Uma Abordagem do Problema de Particionamento *Hardware* e *Software* para *Design* de Sistemas Computacionais *Wearables*

Discente: Rodolfo Labiapari Mansur Guimarães
Orientador: Ricardo Augusto Rabelo Oliveira

rodolfolabiapari@decom.ufop.br – Lattes: `http://goo.gl/MZv4Dc`
Departamento de Computação – Instituto de Ciências Exatas e Biológicas (DECOM-ICEB)
Universidade Federal de Ouro Preto (UFOP)
Ouro Preto - MG – Brasil

Última Atualização: 3 de maio de 2018