Um gerador de sistemas embarcados a partir de modelo independente de plataforma baseado no perfil MARTE

Roberto de Medeiros Farias Filho

Um gerador de sistemas embarcados a partir de modelo independente de plataforma baseado no perfil MARTE

Roberto de Medeiros Farias Filho

Orientador: **Prof. Dr. Vanderlei Bonato**

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências - Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

USP – São Carlos
Julho de 2013

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

d376g

de Medeiros Farias Filho, Roberto
Um gerador de sistemas embarcados a partir de modelo independente de plataforma baseado no perfil MARTE / Roberto de Medeiros Farias Filho; orientador Vanderlei Bonato. -- São Carlos, 2013.
77 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2013.

1. Sistemas Embarcados. 2. FPGA. 3. MARTE. 4. MDA. I. Bonato, Vanderlei, orient. II. Título.

Agradecimentos

À CAPES pelo apoio financeiro.
À FAPESP pelo apoio de viagens.
À Pós-graduação do ICMC-USP pelo apoio intelectual.
À Association for Computing Machinery, pelas menções honoráveis como coach do ICPC.
Ao Prof. Dr. Vanderlei Bonato, meu orientador, pelo apoio intelectual e dedicação.
À Microsoft, pela liderança nas células acadêmicas UNICAP.NET e WINFX-GT.
Ao Prof. Dr. Francisco Madeiro Bernardino Junior, pelo 1o lugar CCET/PIBIC/CNPq
Aos meus pais, Profa. Lauriane e Sr. Roberto.
A todos meus professores, parentes e amigos.

A Deus, por tudo.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AML	*Abstract Modeling Level*
ASIC	*Application-Specific Integrated Circuits*
BFM	*Bus Functional Model*
BSM	*Behavioral Synthesis Model*
CPLD	*Complex Programmable Logic Device*
CPU	*Central Process Unit*
DML	*Detailed Modeling Level*
EDA	*Electronic Design Automation*
EKF	*Extended Kalman Filter*
EML	*Executable Modeling Level*
ESL	*Eletronic System Level*
FM	*Functional Model*
FPGA	*Field-Programmable Gate Array*
FPD	*Field-Programmable Device*
FPL	*Field-Programmable Logic*
GLM	*Gate Level Modeling*
HDL	*Hardware Description Language*
IP	*Intellectual Property*
MARTE	*Modeling and Analysis of Real-Time and Embedded Systems*
MDA	*Model-Driven Architecture*
MoPCoM	*Modeling and specialization of Platform and Components MDA*
PIM	*Platform-Independent Model*
PIP	*Programmable Interconnect Point*
PSM	*Platform-Specific Models*
RTL	*Register Transfer Level*
SAM	*System Architectural Model*
SoC	*System-on-a-Chip*
SoPC	*System-on-a-Programmable Chip*
SPLD	*Simple Programmable Logic Device*
SPM	*System Performance Model*
VLSI	*Very-Large-Scale Integration*
TLM	*Transact Level Modeling*
UML	*Unified Modeling Language*

Resumo

O aumento da complexidade dos sistemas embarcados e a necessidade de um desenvolvimento cada vez mais acelerado têm motivado o uso de modelos abstratos que possibilitem maior flexibilidade e reusabilidade. Para isso, faz-se necessária a aceitação das linguagens e perfis mais abstratos, como o MARTE. Neste trabalho, foi desenvolvida uma ferramenta para conversão de sistemas embarcados independente de plataforma (PIM) em sistemas de uma plataforma específica (PSM), denominada I2S (*Independente to Specific*). O I2S é totalmente acoplável a novos desenvolvimentos e necessidades do projetista, capaz de modelar representações gráficas de sistemas embarcados, usando componentes do MARTE e permitindo uma implementação final em tecnologia reconfigurável. A partir de um modelo independente de plataforma faz-se a conversão para o padrão de projeto SOPC-Builder da Altera e XPS da Xilinx, possibilitando a exploração do espaço de projeto nessas duas tecnologias de modo automático. O trabalho faz análise de sistemas convertidos em diversas configurações e traz resultados relevantes para a área que validam o uso da proposta, atendendo aos requisitos de projeto.

Palavras chaves: **MARTE**, **MDA**, **Sistemas Embarcados**, and **FPGA**.

Abstract

The growing of embedded systems complexity and the want for a quicker development has motivated the use of abstract models that improves flexibility and reusability. To these objective, we searched for the most adequate languages and profiles, like MARTE. In this work we developed a tool for conversion from platform independent models (PIM) to platfom specific models (PSM), named I2S (Independent to Specific). The I2S is totally acceptable to new developments and necessities of the designer, to open up modelling graphic representations of embedded systems using MARTE components and doing implementation in reconfigurable technology. A platform independent model is converted to the pattern of Altera's SOPC-Builder and Xilinx's XPS, making possible the exploitation of the project space in theses two tecnologies automatically. The work does analysis of systems converted in different configurations and shows relevant results to the area that validate the use of the proposal, meeting the project requirements.

Keywords: ***MARTE***, ***MDA***, ***Embedded Systems***, and ***FPGA***.

CAPÍTULO

1

Introdução

A complexidade das tarefas executadas pelos sistemas embarcados está em aumento constante e requer sistemas cada vez mais sofisticados(Ghosal et al., 2010). Exemplos típicos de sistemas embarcados existentes no mercado incluem, entre outros, *tablets*, *smart phones*, câmeras digitais, *TV set-top boxes*, sistemas de segurança, robôs pessoais, dispositivos de controle automotivo e equipamentos médicos (Martins, 2002).

O desenvolvimento de tais sistemas tem sido modificado pelas novas exigências do mercado, que incluem: desenvolvimento cada vez mais seguro, tempo de projeto com curto prazo, a necessidade de adoção de tecnologias mais inteligentes e flexíveis, necessidade de aumento de desempenho da tecnologia, e a redução do consumo de energia e capacidade de integração de tarefas, sem comprometer o desempenho nem exceder o consumo de energia (Koudri et al., 2009). Sistemas embarcados são tecnologias que possuem unidades de processamento cada vez mais poderosas, além de serem capazes de suportar um ambiente seguro e dinâmico enquanto características de tempo real forem levadas em consideração. Acredita-se, que no futuro sejam acrescentados mais valores como **segurança**, **escalabilidade** e **usabilidade**, que deverão incrementar no "leque" de necessidades fundamentais do projeto. Alguns dos requisitos confrontam determinadas características com outras, como **tamanho versus flexibilidade**, **robustez versus funcionalidade**, ou **potência versus desempenho** (Dejan, 2000).

De acordo com Densmore et al. (2006), inicialmente esses sistemas eram projetados a nível de portas lógicas (*Gates Logicals Level*); posteriormente a nível RTL (*Register Transfer Level*), e, devido à necessidade de modelos mais abstratos, está tornando-se cada vez mais comum o desenvolvimento em nível ESL (*Eletronic System Level*).

A metodologia ESL apresenta como vantagens: uma plataforma virtual de hardware para desenvolvimento de software; o desenvolvimento de modelos de referência para verificação de hardware; o desenvolvimento de arquiteturas de sistemas; especificações e o mapeamento no *chip* das aplicações hardware/software (Bonato & Marques, 2009; Lin et al., 2011).

A modelagem dos sistemas embarcados tem gerado grandes questões para a comunidade científica, principalmente quanto ao desenvolvimento com base em **modelos de alto nível** (Shukla & Edwards, 2003; Brisolara et al., 2005; Wang et al., 2012; Herrera et al., 2012), que possuem mais flexibilidade, que tem como objetivo representar todo o hardware funcional e não funcional, além dos aspectos de software, por uma linguagem de modelagem única.

De acordo com Sangiovanni-Vincentelli et al. (2009), **reduzir o tempo de desenvolvimento** e a **complexidade do desenvolvimento multiplataforma** são as principais fronteiras para os projetistas de sistemas embarcados. O processo de projetar sistemas embarcados para atender a tais requisitos deve possuir as etapas: integração, refinamento e análise.

Refinamento cobre a especificação inicial do sistema até a sua aplicação final (produto final) com diferentes níveis de modelagem que aumentam o detalhamento e dependência da plataforma de destino. **Integração** é o arranjo de vários modelos, para explorar especificações de modelagem distintas. A fase de **análise** do processo consiste na inspeção de comportamento do sistema de acordo com a especificação do projeto.

Algumas regras adicionais são fundamentais para se estabelecer uma plataforma de modelagem de projetos de forma adequada a se obter a distribuição exata das camadas, tais quais: os processos de projeto estarem definidos precisamente em camadas de abstração; cada camada de abstração estar definida por uma plataforma de projeto, sendo esta uma representação de uma família de projetos que satisfazem um conjunto de restrições de plataformas específicas; e, projetos em cada plataforma estarem representados por projetos de plataformas específicas, de tal forma que um projeto completo pode ser obtido por uma plataforma para criação de

projetos de modelagem, onde esta faz uma instância dos componentes e utiliza-se de uma plataforma de mapeamento que tem por base seu fluxo em camadas de abstração subsequentes (Sangiovanni-Vincentelli et al., 2009).

Com a busca de empresas e desenvolvedores por sistemas embarcados que executem em mais de uma plataforma de hardware ou sistema operacional, surge-se o conceito de **multiplataforma**. O conceito de multiplataforma é dado como a capacidade de um software atuar independente de uma arquitetura, que em outras palavras significa deixar o projetista preocupado com os conceitos das funcionalidades necessárias e não das limitações de cada plataforma. Tal abordagem decrementa o tempo de desenvolvimento das consecutivas gerações de um sistema para outras plataformas, que ajuda a diminuir a duração e o tempo entre os ciclos de desenvolvimento.

Abstração é um conceito proveniente de alto nível de modelagem, que significa uma propriedade da aplicação em isolar detalhes da implementação de sistemas e os conceitos de "raiz" através de uma divisão em compartimentos (Stolper, 2002). O uso de tal conceito é considerado como uma forma criativa, elegante, simples e pragmática de resolver problemas de multiplataforma. Stolper (2002) sugere uma estratégia que envolva o trabalho com algumas bibliotecas intermediárias com a aplicação embarcada, chamadas também de camadas, mas estas são todas interligadas ao sistema embarcado de forma individual. São bibliotecas: framework de plataforma específica, abstração de sistema operacional, protocolos, serviços de sistema, e interfaces físicas. É fundamental a criação de uma boa plataforma de simulação de software através do reuso de componentes e conceitos de multiplataforma de forma efetiva e factível com ganhos de tempo e performance (Powell et al., 1995).

O MDA (Model-Driven Architecture) implementa uma metodologia onde Modelos Independentes de Plataforma (PIM) são projetados primeiro e depois convertidos em Modelos de Plataforma Específica (PSM). O emprego desta metodologia permite a exploração das suas várias configurações de arquitetura independentemente da plataforma, proporcionando assim uma maior velocidade de simulação e análises num nível arquitetural (Carcamo et al., 2005; OMG, 2011). Tais condições favorecem a redução do tempo de desenvolvimento e atualizações de sistemas embarcados e facilitam a exploração de novas plataformas.

O perfil MARTE (*Modeling and Analysis of Real-Time and Embedded Systems*) (OMG, 2009; Heaton, 2011) apresenta-se como uma solução sofisticada e bem planejada para representar os projetos de sistemas embarcados. Tal linguagem de modelagem baseia-se em UML (Alemfin & Alvarez, 2000; Object Management Group (OMG), 1999) para projeto de sistemas embarcados e de tempo real de forma independente de plataforma. Esta escolha traz robustez e diversas vantagens, como: padronização, reuso computacional, pacotes diferenciados para cada tipo de modelos e estrutura independente pronta para análise.

Este trabalho apresenta um conversor automático de PIM para modelos PSM chamado **I2S** (*Independente-to-Specific*), gerando modelos específicos a partir de modelos MARTE UML para as ferramentas das plataformas das empresas Altera e Xilinx. Nesta dissertação, o modelo final é uma representação que pode ser sintetizados em hardware usando ferramentas EDA (*Eletronic Design Automation*) dessas plataformas específicas.

O software do conversor I2S permite que um modelo seja explorado em diversas plataformas ao mesmo tempo a apartir de um único modelo inicial PIM. Ele fornece ao desenvolvedor a possibilidade de utilizar não somente configurações de modelos diferentes mas também o mesmo modelo em diferentes plataformas de computação, aumentando as soluções possíveis para satisfazer as exigências de um dado sistema embarcado. A ferramenta I2S é totalmente acoplável a novas necessidades do projetista, que modelará projetos de sistemas embarcados com possibilidade de total implementação em tecnologia reconfigurável do tipo FPGA (Field-Programmable Gate Array).

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo deste trabalho de pesquisa é desenvolver um gerador de sistemas embarcados, que faça a conversão de modelo independente de plataforma (PIM) em modelo de plataforma específica (PSM), sendo as plataformas SOPC Builder da Altera e XPS da Xilinx. Esse ambiente deverá prover uma infra estrutura que permita a exploração eficaz das arquiteturas de sistemas embarcados onde diversas soluções poderão ser geradas de modo automático para

permitir que o usuário escolha uma das soluções geradas em função do desempenho e recursos de hardware que seja adequada à aplicação.

1.1.2 Objetivos Específicos

São objetivos específicos do trabalho os tópicos a seguir:

1. Abordar um estudo de modelagem em plataformas de FPGA com mais de um fabricante;
2. Explorar a modelagem MARTE;
3. Utilizar a técnica do MDA para um projeto multiplataforma;
4. Analisar a representação de sistemas embarcados nas plataformas SOPC Builder da Altera e XPS da Xilinx;
5. Desenvolver sistema que facilite a modelagem de sistemas embarcados a partir de padrão universal, transformando-o em um padrão específico com possibilidade de manuseio pelo projetista ainda na fase final;
6. Criar regras de bibliotecas genéricas para modelagem de diferentes sistemas;
7. Projetar um software final que esteja disponível para uso da comunidade acadêmica (https://drive.google.com/folderview?id=0Bx3fIen60flLelo5SHZwWmhnanM&usp=sharing).

1.2 Justificativa

O trabalho explora a geração de sistemas embarcados a partir de modelos independentes de plataforma (Lu et al., 2005; Kang & Draper, 2007; Dekeyser et al., 2005; Phillips & Thullen, 1989; Kang et al., 2008; Kamaruddin et al., 2011; Soler et al., 2007). Tal proposta contribui para a redução da complexidade na modelagem de sistemas embarcados (Kopetz, 2008; Khan et al., 2006; Raghunathan et al., 2007; Konrad et al., 2004; Liggesmeyer & Trapp, 2009; Pimentel et al., 2001), pois torna o projetista livre da preocupação com o modelo do fabricante desejado, voltando atenção aos componentes e suas propriedades que serão inseridos e analisados no projeto e não à compatibilidade dos recursos que seriam utilizados.

O ambiente proposto traz um modelo visual e interativo para a exploração de sistemas embarcados em nível de modelagem arquitetural, com componentes, atributos, propriedades e conexões entre os atributos. A aplicação possibilita projetos em diversas plataformas, sendo necessário apenas a modelagem UML do MARTE para sistemas embarcados. Tal ambiente pode ser facilmente adaptado para novas plataformas com a inclusão de novas regras de conversão.

1.3 Organização

O trabalho é organizado por capítulos. O **capítulo 2** apresenta as principais linguagens e métodos para a modelagem de sistemas embarcados. O **capítulo 3** apresenta alguns detalhes da ferramenta I2S para converter sistemas independentes (MARTE) em específicos (Altera e Xilinx). O **capítulo 4** apresenta os resultados obtidos com a ferramenta I2S para a exploração do espaço de projeto e demonstra o ganho em relação ao tempo de processamento do algoritmo EKF. O **capítulo 5** apresenta a conclusão do trabalho com propostas de trabalhos futuros. As referências bibliográficas são apresentadas no fim desta dissertação.

CAPÍTULO

2

Modelagem de Sistemas Embarcados

O avanço contínuo da capacidade dos circuitos integrados e a necessidade de sistemas embarcados mais complexos para lidar com os problemas atuais de abstração estão direcionando o desenvolvimento desses sistemas para ambientes com alto nível arquitetural, tornando-os mais distantes dos detalhes de hardware. Esta abordagem tem sido bem sucedida, principalmente para representar e simular o comportamento de hardware funcional (Shukla et al., 2006; Coussy & Morawiec, 2008; Coussy et al., 2009).

Existe uma vasta gama de linguagens de alto nível para modelagem de sistemas embarcados. Muitas dessas linguagens usam como base o código C, ou incorporam novos recursos para este, como estruturas de dados, instruções que exploram paralelismo, mecanismos de sincronização e interfaces (Sangiovanni-Vincentelli, 2007; Ma & Zacharda, 2010). Algumas outras linguagens são extensões de códigos mais abstratos, como o UML e o Matlab.

Este capítulo aborda as técnicas de modelagem de sistemas embarcados, alguns conceitos fundamentais de cada metodologia e faz um planejamento de como serão empregadas as tecnologias no projeto final.

2.1 Linguagens de Modelagem

Na maioria dos casos, a modelagem de sistemas embarcados também exige uma linguagem que envolve, não só o hardware, mas também o software e a integração entre eles. Duas linguagens de modelagem recentes de alto nível de abstração são SysML (Friedenthal et al., 2008a) e MARTE (Heaton, 2011), que são linguagens gráficas da UML tradicional com perfis específicos para a modelagem e análise de sistemas embarcados.

Outra linguagem de modelagem de alto nível é o Meta-Model Metropolis (MMM) (Berkley, 2010). A flexibilidade dessa linguagem permite que diferentes modelos computacionais sejam especificados para o desenvolvimento de sistemas heterogêneos. Diferentemente da SysML e MARTE, MMM não é uma extensão da UML, mas uma linguagem textual. Desenvolvimento com MMM ocorre pelo mapeamento dos modelos funcionais em plataformas de execução para fornecer um sistema completo incorporado. Todas essas linguagens fornecem um caminho comum para a modelagem de aspectos de hardware e software que facilitam a comunicação entre os desenvolvedores favorecendo a operação conjunta entre ferramentas de desenvolvimento usadas para especificação, projeto, verificação e geração de código.

2.2 MARTE

O MARTE (*Modeling and Analysis of Real-Time and Embedded Systems*), formalmente publicado pela OMG em 2009, é uma linguagem de modelagem de sistemas embarcados e de tempo real adotado neste trabalho por ser considerada mais adequada para modelar as requisições do nosso modelo. SysML (Friedenthal et al., 2008a; IBM, 2009; Friedenthal et al., 2008b; Paredis et al., 2010; Hansen, 2010) é uma linguagem de modelagem que poderia também ter sido utilizada, no entanto, uma vez que não é específico para sistemas embarcados, não existe uma boa distinção entre modelos hardware/software. Além disso, MARTE tem modelos específicos para diversos aspectos de tempo real, que embora não tenham sido explorados neste trabalho, são essenciais em diversas aplicações de sistemas embarcados.

O modelo de tal especificação tem algumas propriedades comuns ao SysML, tais como os recursos de hardware de modelagem de software e alocação desses recursos. Nestes casos, em que existem alguns tipos de sobreposições, MARTE mantém terminologias e significados compatíveis com SysML. Espinoza et al. (2009) apresenta um estudo comparativo entre estas duas linguagens e sua integração. Segundo o autor, como existem aspectos multidisciplinares em sistemas embarcados, o uso de uma linguagem de modelagem única poderá não ser suficiente para cobrir todos os detalhes que uma determinada aplicação requer.

MARTE vem como uma reposição ao perfil UML SPT (*Schedulability, Performance, and Time*) que pertence ao conjunto UML 2 e OCL 2. O perfil MARTE está organizado conforme a figura 1, onde uma sequência de pacotes são incluídos e classificados.

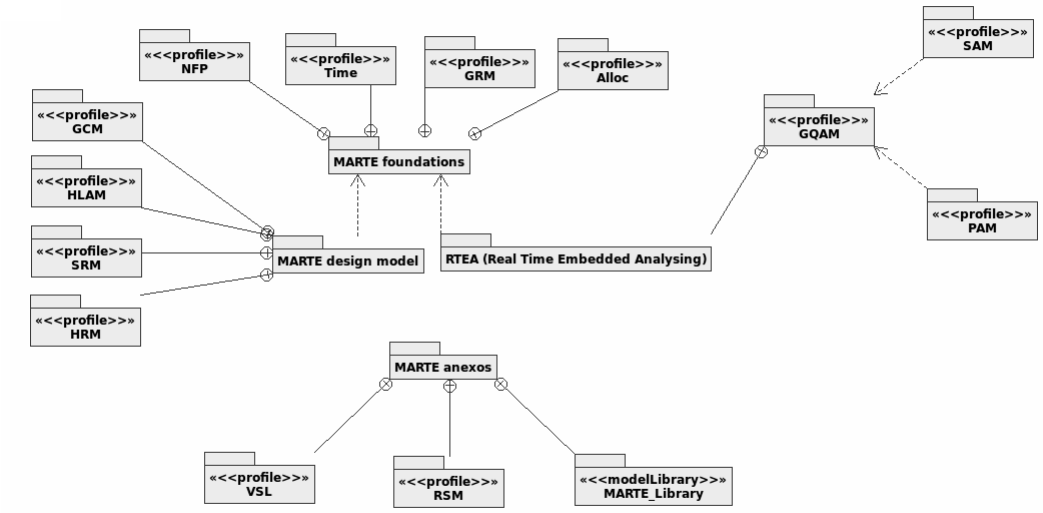

Figura 1: Organização dos pacotes do perfil MARTE. Adaptada de Heaton (2011).

Os pacotes incluem os recursos necessários para as seguintes especificações:

- **NFP**: propriedades não funcionais;
- **Time**: modelagem de tempo causal/temporal, clock/síncrono, e física/tempo-real;
- **GRM**: modelagem de estereótipos e notações que ajudam na modelagem;
- **Alloc**: alocação de elementos das plataformas de execução;

- **HLAM**: suporte a conceitos de modelagem em alto nível para alcançar características de tempo-real;
- **GCM**: conceitos adicionais (paradigmas de componentes não-usuais) que têm sido identificados como necessários para a esquematização;
- **SRM**: voltado a descrever programação de interfaces de aplicações de suporte a softwares de execução multitarefa;
- **HRM**: descrever suportes de execução de hardware através de diferentes visões e níveis de detalhes;
- **GQAM**: domínios especializados com análise baseada no comportamento do software, bem como performance, agendamento, energia, memória, custo, consumo, e segurança;
- **SAM**: cálculo e análise sensitiva do agendamento;
- **PAM**: descreve a análise de propriedades temporais de sistemas.

Os perfis da linguagem MARTE podem ser agrupados em dois pacotes principais: o dos modelos de análise e de *esquemas*. Em nosso trabalho foi adotado o modelo de *design* especificado pelo subpacote HRM::MARTE (Hardware Resource Modeling). A versão atual do gerador desenvolvido neste trabalho funciona com modelos que representam a arquitetura de processamento, que compõe apenas componentes de hardware. De acordo com os perfis MARTE, um modelo de hardware pode ser lógico (*Hw_Logical*) ou físico (*Hw_Physical*). A parte física é composta por componentes que não são utilizados na computação, enquanto a parte lógica é usada no armazenamento, comunicação, entrada, saída e processamento de dados de um sistema completo embarcado. O HRM modela a parte de hardware do sistema embarcado, que é composto pelo modelo *HW_General* para definir a estrutura típica das plataformas de execução, sendo esse subdividido em modelos *HW_Logical* e *HW_Physical* para representar os elementos lógicos e físicos de um sistema embarcado. O modelo HW_Logical é voltado para a classificação do tipo de funcionalidade das entidades de hardware, as quais podem ser: computação (*HW_Computing*), tempo (*HW_Timing*), comunicação (*HW_Communication*), armazenamento (*HW_Storage*) e recursos de expansão (*HW_Device*).

HW_Physical representa as propriedades físicas dos componentes, como informações de posicionamento (*HW_Layout*) e de consumo de energia (*HW_Power*). Um próximo passo para a expansão deste trabalho seria a inclusão do subpacote MARTE::SRM (*Software Resource Modeling*) para a modelagem do software que executa nos modelos HRM.

A figura 2 ilustra a árvore de especializações MARTE utilizadas neste trabalho, sendo o recurso DRM (*Detailed Resource Modeling*) um pacote de modelo de projeto MARTE, onde os recursos de hardware e software são apresentadas para o projetista. Nesta figura, a raiz é o pacote DRM, que se ramifica em SRM e HRM. O HRM é o foco de nosso trabalho e cuja tecnologia se subdivide em Lógica e Física. Física contém *Layout* e *Power*. Lógico, usado em nosso projeto, contém Computação (ASIC, PLD, Processador), Temporização (*Clock*,*Timer*,*Watchdog*, Recursos de Tempo), Armazenamento (Memória, Cache, RAM, ROM, *Driver*), Dispositivo (I/O, Suporte) e Comunicação (Barramento, *Bridge*, *EndPoint*, Recursos de Comunicação, Arbitro, Mídia).

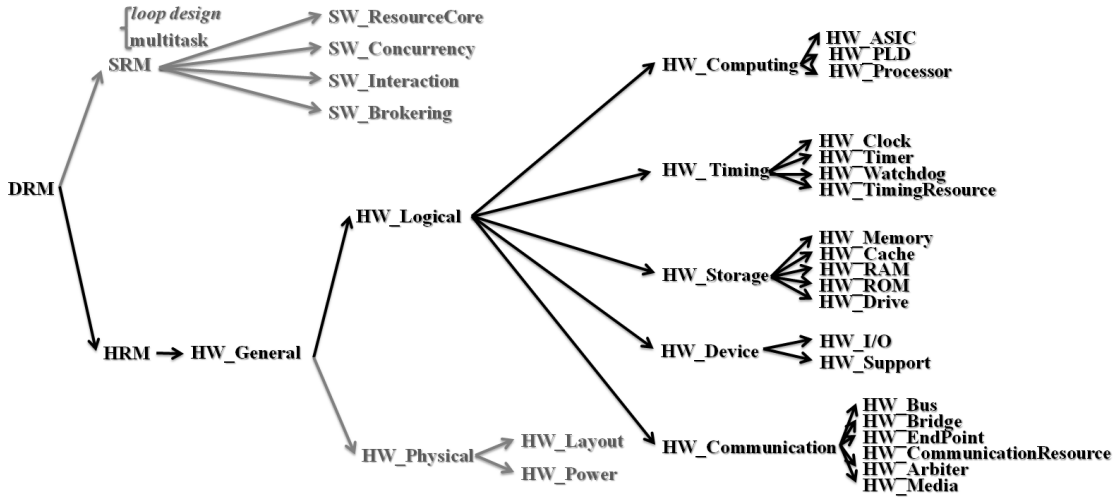

Figura 2: Composição do pacote utilizado na modelagem HRM-MARTE. A parte em negrito representa os recursos utilizados neste trabalho.

As subseções a seguir trazem explicação de cada “visão” e seus respectivos detalhes.

2.2.1 Visão Descritiva

A modelagem ilustrada na figura 3 é a visão/modelagem descritiva. Nesse nível é especificado como cada componente deve iteragir, sem seus atributos ou demais qualidades. Deve-se apenas expressar como são interligados cada um dos componentes, através de entidades com estereótipo nomeado *«Resource»*. O componente principal será o nome do projeto de sistemas eletrônicos. O número de componentes não é relatado nesse nível.

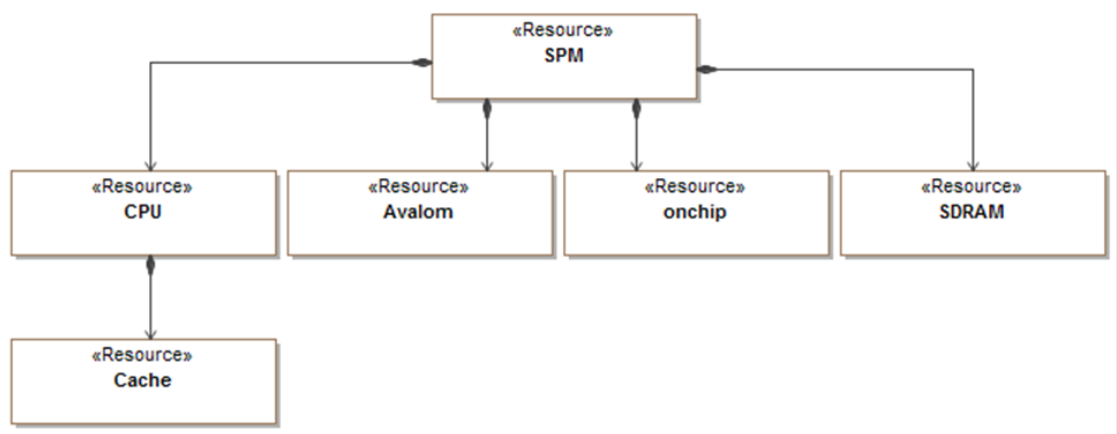

Figura 3: Visão descritiva.

2.2.2 Visão Lógica

Nesse nível de modelagem, figura 4, é descrito o comportamento lógico do componente e seus atributos são específicos a características lógicas, mas sem especificar sua capacidade ou detalhes menores. São exemplos de atributos lógicos: sincronização, número de núcleos, volatilidade, estaticidade, etc.

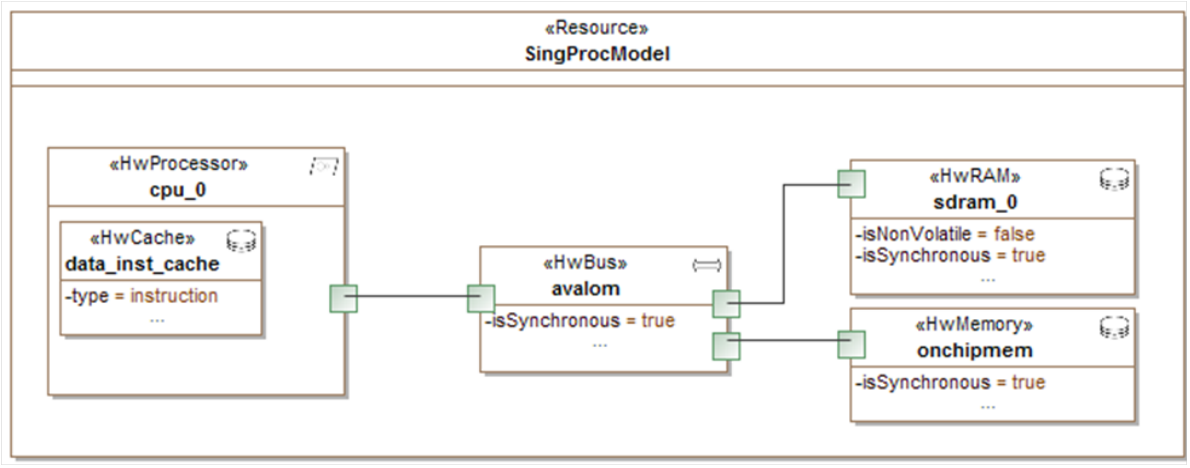

Figura 4: Visão lógica.

2.2.3 Visão Física

A preocupação nessa visão é relacionada com o tipo de hardware que será usado e quais estereótipos invocam. Ver figura 5.

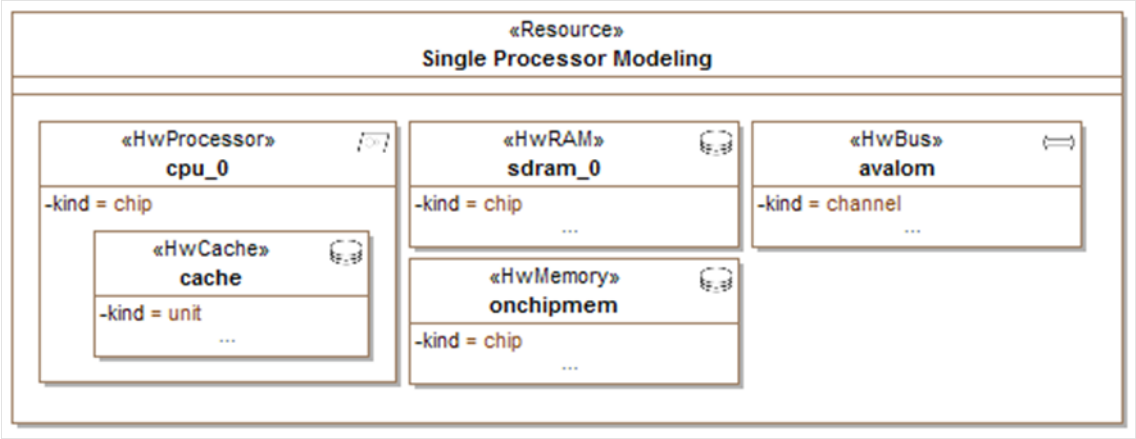

Figura 5: Visão física.

2.2.4 Visão de Fusão

Aqui, faz-se uma combinação (fusão) da modelagem física com a lógica.

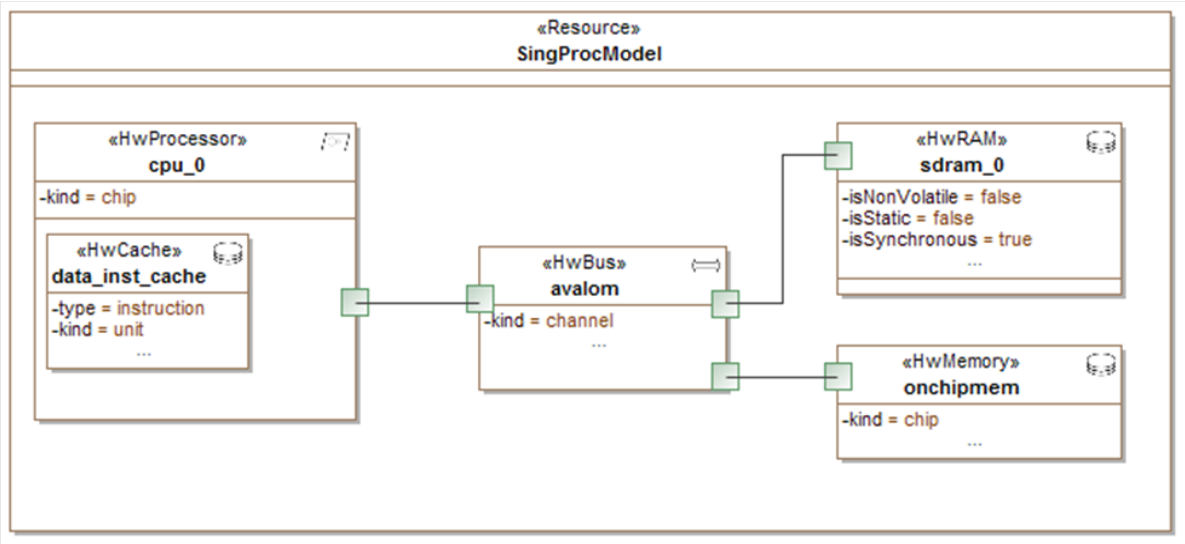

Figura 6: Visão da fusão (*merge*) da física com lógica.

2.2.5 Visão de Lógica Detalhada

Esta fase atribui as características lógicas restantes. O modelo inclui valores de capacidade, número de células de memória, etc. A figura 7 traz essa modelagem e o software proposto neste trabalho utiliza de tal modelo para sua especificação inicial do projeto de sistema embarcado.

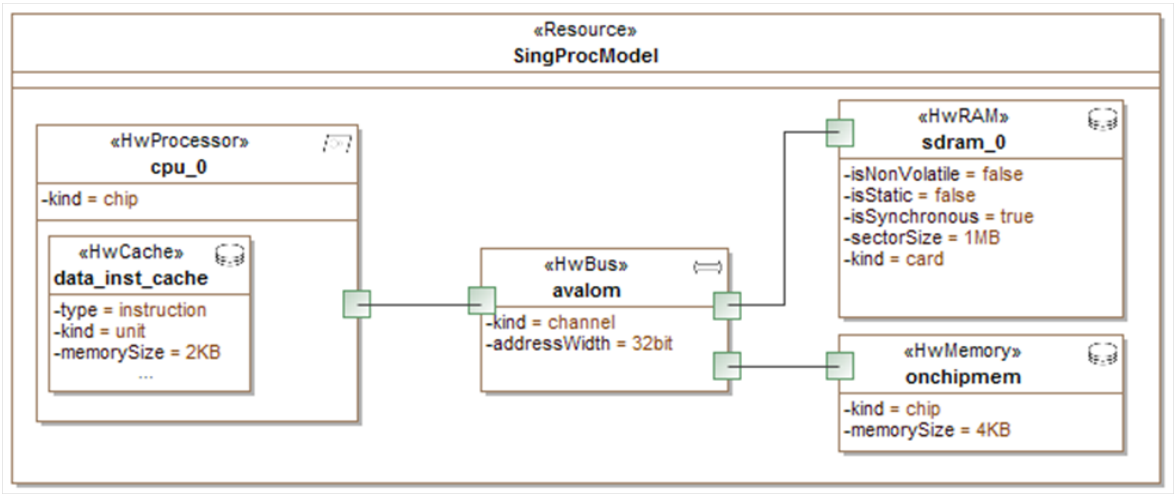

Figura 7: Visão lógica detalhada. Modelo empregado neste trabalho.

2.3 MDA

MDA (*Model Driven Architecture*) (OMG, 2011) é uma metodologia que foi desenvolvida para projeto de sistemas em geral com a finalidade de separar a aplicação dos negócios e permitir

o uso de modelos independentes (PIM, *Platform Independent Model*), que normalmente são desenvolvidos em UML ou outra linguagem de modelagem, para serem posteriormente integrados com plataformas específicas (PSM, *Platform Specific Model*) (Heaton, 2011). PIM é a visualização da plataforma de aplicativos independentes, escondendo detalhes de uma modelagem particular. Ele pode ser considerado uma descrição funcional com um certo grau de independência. PSM é o resultado de modelagem final pronto para ser mapeado para uma plataforma específica (Koudri et al., 2009; Miller & Mukerji, 2003; Duby, 2003). Neste trabalho, o PSM representa o SOPC da Altera (Altera, 2010a) e XPS da Xilinx (Xilinx, 2010c).

MDA é vista como uma metodologia que possui as distinções: reagir rapidamente à troca de requisitos funcionais e tecnológicos das plataformas; estender longevidade do sistema; aumentar a produtividade no desenvolvimento; habilitar reuso em larga-escala dos PIMs; ter baixo custo de manutenção; facilitar documentação; reduzir custos e aumentar a qualidade dos sistemas embarcados (Duby, 2003).

No MDA, o processo de modelagem trabalha com níveis respectivos às necessidades particulares (Aulagnier et al., 2009; Koudri et al., 2009):

- AML (*Abstract Modeling Level*): descreve a execução virtual da plataforma, especifica níveis de concorrência trabalhando com conceitos de *RTUnits*, *RTeConnectors* e o pacote *HLAM*, em que a figura 8 traz uma diagramação do processo de utilização da abstração em comparação com a funcionalidade;

- EML (*Executable Modeling Level*): a plataforma é composta por componentes genéricos como processador e unidades de memória, a alocação descreve o modelo previamente gerado na plataforma, o resultado permite análise de agendamento e obter performance e faz-se uso do subprofile *SAM*;

- DML (*Detailed Modeling Level*): contém as informações necessárias para mapeamento de PSM AML e geração de código HDL (*Hardware Description Language*) do RTL, utilizando-se *HRM* e *SRM*.

A figura 8 descreve o processo de alocação de modelos em MDA. São entrados dois esquemas: o funcional e uma modelagem AML. Uma plataforma captura os modelos e escolhe

a alocação através de recebimento de certa configuração de análise. Sendo fornecido um **modelo para transformação** (alocador) que será trabalhado por um processo de conversão menor e convertido em um **modelo alocado final** para a plataforma específica. Este processo é padrão para se trabalhar com modelos abstratos para um formato alocado em qualquer plataforma e foi feito aprimoramentos desse método no presente trabalho para o produto final.

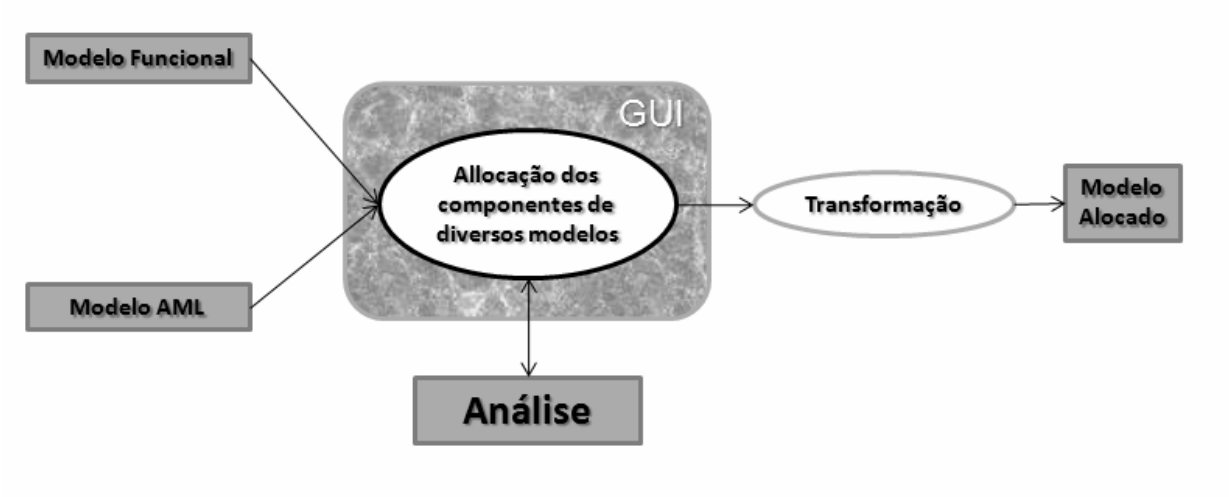

Figura 8: Processo de alocação de modelos AML (*Abstract Modeling Level*) e funcional. Tal processo utiliza-se de combinação de dois modelos, uma análise estratégica da comunicação e concorrência, e uma transformação num modelo já alocado (Aulagnier et al., 2009).

Algumas terminologias são importantes de se tratar a respeito do MDA para que que se possa compreender melhor os conceitos da técnica. São eles: **Sistema** como sendo um programa, simples sistema de computação, uma combinação de partes de diferentes sistemas, um sistema da federação, um controle isolado, pessoa, empresa, ou federação de empresas; **Modelo** como sendo uma especificação do sistema para algum certo propósito, ao mesmo tempo que uma combinação do desenho e do texto projetado, e que o texto pode ser linguagem de modelagem ou natural; **MDA** é uma abordagem para desenvolvimento de sistema que incrementa a força dos modelos no trabalho; Descreve **arquitetura** como sendo uma especificação de partes e conectores e sua regra; **Panoramas** é uma técnica para abstração usando um conjunto selecionado de conceitos arquiteturais e regras estruturais; já **Visão** é uma representação de um sistema em um dado panorama; **Plataforma** é um conjunto de subsistemas e tecnologias que suporta uma coerente grupo de funcionalidades através de interfaces ou padrões, assim como Object, Batch, Dataflow, Java 2 Enterprise Edition (J2EE), Microsoft

.NET, dentre outras; **Aplicação** define-se de uma funcionalidade inicialmente desenvolvida que em conjunto compõe um sistema; **Independente de Plataforma** é a qualidade que um modelo pode exibir; **Panorama Independente de Computação** foca no ambiente e os requisitos para o sistema; **Panorama Independente de Plataforma** foca na operação do sistema e oculta detalhes necessários de uma plataforma em particular; **Panorama Específico à Plataforma** combina o panorama anterior com um foco adicional no detalhe do uso de uma plataforma específica pelo sistema (Miller & Mukerji, 2003).

É importante se conhecer as características de cada nível do MDA, que segue nos tópicos a seguir (Miller & Mukerji, 2003; Koudri et al., 2008):

- **CIM**: comumente chamado de Modelo de Domínio, é uma visão do sistema do panorama independente de computação, onde a estrutura não exibe detalhes do sistema, fazendo ainda uma ponte entre especialistas do domínio, do design e da construção;

- **PIM**: é uma visão do Panorama Independente de Plataforma que exibe um certo grau de independência;

- **PM**: suporta um conjunto de conceitos técnicos para diferentes tipos de partes que compõem uma plataforma e seus serviços, e conceitos representando variedade de tipos de elementos para especificar o uso da plataforma na aplicação para usar no PSM;

- **PSM**: é um modelo do mesmo sistema especificado pelo PIM e como o sistema usará a plataforma escolhida.

2.4 Modelos RTL, ESL e Ferramentas EDA

Com a alta complexidade dos sistemas atuais, tanto de desenvolvimento como de depuração, e a forte pressão do mercado por um curto prazo de entrega do produto, tem-se nutrido cada vez mais a necessidade de modelos mais abstratos em relação ao tradicional desenvolvimento em RTL (*Register Transfer Level*). Sendo o nível de modelagem mais adequado para projetar tais sistemas, adota-se a modelagem em ESL (*Eletronic System Level*). As ferramentas EDA (*Elec-*

tronic Design Automation) são fundamentais para alcançar desenvolvimento com flexibilidade em sistemas e circuitos eletrônicos ESL (Sangiovanni-Vincentelli, 2007).

Para aumentar o grau de flexibilidade do projeto os desenvolvedores de tais sistemas buscam por uma abordagem mais ampla e direcionada que contenha as características chaves para a construção de uma plataforma de modelagem de tal forma que alguns processos sigam um padrão preestabelecido, composto por: **refinamento**, **integração** e **análise**. São princípios de uma abordagem unificada de projetos de modelos de sistemas embarcados (Sangiovanni-Vincentelli, 2007):

- Incluir ambos projeto de hardware e software embarcado;
- Usar altos níveis de abstração para a descrição de projetos iniciais;
- Oferecer efetiva exploração do projeto arquitetural;
- Armazenar implementação detalhada para síntese ou refinamento manual;
- Conceito correto de plataforma, ou seja, de que é uma "biblioteca de componentes que será montada para gerar um projeto no nível de abstração";
- Mapeamento, automatizado de modo a combinar a plataforma com a funcionalidade. Ver figura 9, percebe-se uma distinção entre Funcionalidade e Plataformas que só é corrigida pelo mapeamento com seus métodos para adaptação em cada plataforma.

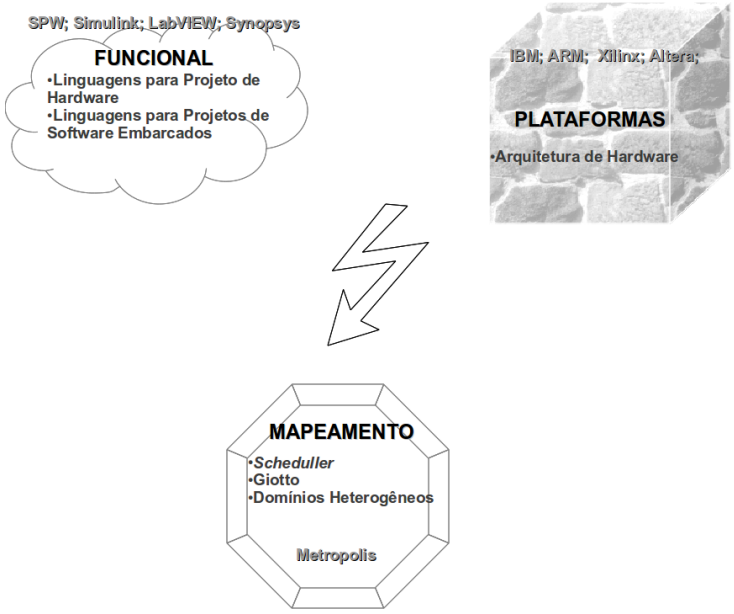

Figura 9: Modelo Funcional e Modelo Arquitetural. Mapeamento, que realizada combinação de ambos os modelos (Sangiovanni-Vincentelli, 2007).

Para refinamento no fluxo de modelagem, o processo dá-se com a seleção dos IPs, *Intellectual Property*, ou seja, as regras de cada componente. Uma vez selecionado, deve ser analisada a validade da placa e da arquitetura. Para criação do IP, Devem ser levadas em consideração (Berger, 2002):

1. A compatibilidade do IP com o compilador;
2. Análise dos requisitos do cliente;
3. Separação entre projeto de hardware ou projeto de software;
4. No projeto de hardware, será simulada a lógica (no caso de ASICs) e a mesma analisada a diferença de potencial e tensão, trabalhando com hardware inicial sem ativar sistema de memória;
5. O projetista de hardware deve estar atento ao desenvolvimento da placa e componentes de hardware do sistema, deve ser analisada o circuito como um todo, em particular e as demais instruções que suportar;
6. Na fase de integração, são compreendidas algumas subfases como: análise da performance do software, emulação no circuito, processador ou analisador lógico,

emulação JTAG, emulação ROM, e a aplicação do sistema no MiniMON29K (no caso da ilustração do autor).

Os ciclos de um projeto de sistema embarcado iniciam na especificação do produto, onde características comerciais são implantadas e outras peculiaridades especificadas em documento de especificação de requisitos e componentes, e tem sua conclusão integração do software com o hardware, sendo o produto validado e lançado. Compreendem fases intermediárias: o particionamento do hardware com o software, e esboço do hardware ou software detalhado (fase principal para o projetista), uma vez que o projeto de ambos está dividido, o software e o hardware são projetados separadamente, embora exista uma fase que se interliguem (Berger, 2002).

2.5 Ferramentas EDA de Plataformas Específicas

As ferramentas EDA de plataformas específicas são apresentadas para se tratar da modelagem dos sistemas específicos a um fabricante. O objetivo principal de uma ferramenta EDA é a automação no processo de esquematização dos sistemas eletrônicos.

2.5.1 Plataforma de Modelagem SOPC Builder

A Altera é fabricante de dispositivos para lógicas programáveis (Altera, 2012). Dentre seus produtos estão os hardwares FPGA, SoC FPGA (*System-o-a-Chip FPGA*), CPLD (*Complex Programmable Logic Device*) e ASIC em combinação com ferramentas de software, propriedades intelectuais e processadores embarcados soluções programáveis de altos valores. A FPGA SoC da Altera contém processador *hard* ARM (dual-core ARM Cortex-A9 MPCore), cujo sistema consiste de: processador, periféricos e interface de memória com conexão com a FPGA de fábrica usando alta banda larga interconectada por *backbone*. Ela combina performance e potência de IP hard com flexibilidade. São características da FPGA Altera: reduzir potência do sistema, redução de custo, redução de tamanho da placa, integração de processadores discretos, funções de processamento de sinal digital, hardware e software padronizado, suporte à virtualização, interface padrão, vida do produto estendida, famílias

diversas, TSMC's 28-nm Low-Power (28LP) process, conduzidos em baixa energia e custo quando habilitados (Altera, 2012).

O Quartus II (Altera, 2012) é o software da Altera para modelagem de sistemas CPLD, ASIC, FPGA e Sistemas em Chip FPGA (SoC FPGA) número um em performance e produtividade. A figura 10 apresenta alguns de seus softwares integrados, incluindo o SOPC Builder. Dentre outros, os principais apresentados são: ModelSim, TimeQuest, TimeAnalyser, SOPC Builder, QSYS, PowerPlay e Power Analyser. ModelSim (Altera, 2011) é uma ferramenta projetada para verificação e simulação das linguagens VHDL, Verilog, SystemVerilog e outras mixtas. *TimeQuest* e *Timer Analyser* são ferramentas para controlar restrições de tempo e gerar relatório. *PowerPlay* e *Power Analysis* é responsável por calcular energia através de condições ambientais, dispositivos diferentes, recursos usados em cada dispositivo e monitoramento de sinais.

SOPC Builder é a ferramenta principal de nosso trabalho, cujo objetivo é definir e gerar sistemas a partir de componentes da biblioteca. O SOPC Builder é capaz de criar mapeamento para memória além de trabalhar com modelos para base de testes. QSYS é uma extensão do SOPC Builder que possui funções de: interconexão de alta performance, voltado a hierarquia, gerenciamento de propriedades intelectuais novas, interface padrão do novo Quartus II e depuração em tempo real.

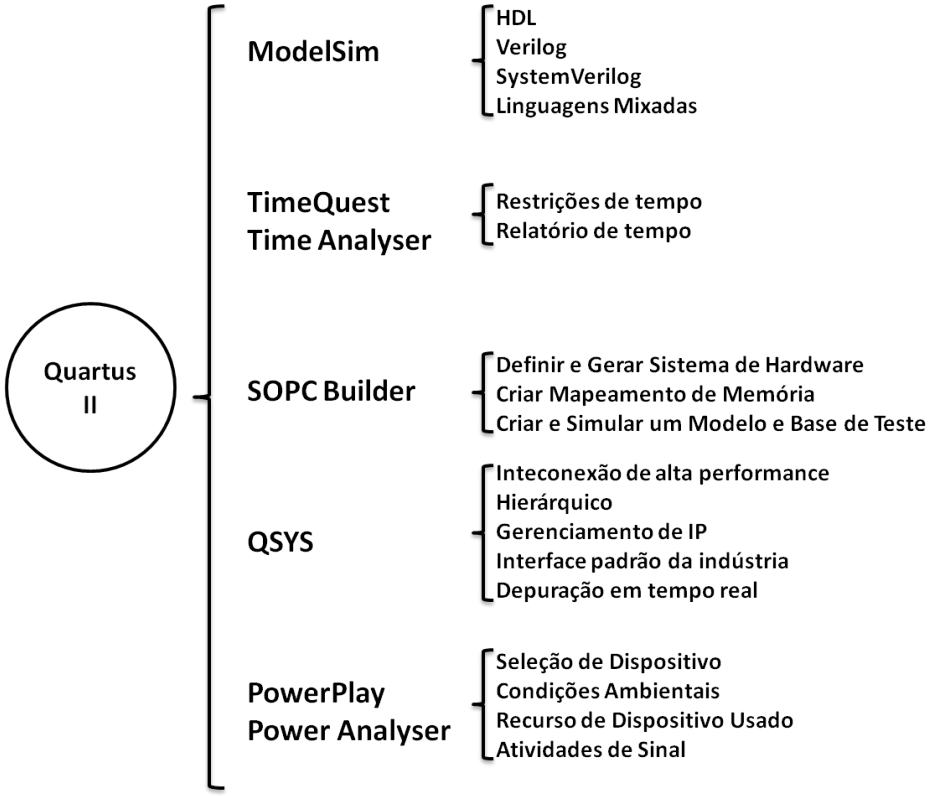

Figura 10: Quartus II e softwares componentes.

Um componente SOPC Builder (Altera, 2010) é um objeto de hardware, módulo do esquema, que o SOPC Builder reconhece e integra no sistema. Múltiplos módulos podem ser interconectados para criar um arquivo HDL *top-level* chamado sistema SOPC, gerando interconectividade. O resultado final do sistema criado pela ferramenta é um arquivo XML do tipo ".sopc", que possui sua estrutura como na figura 11. Um sistema principal contém dois grupos menores: um de parâmetros e um outro de módulos. Os parâmetros do primeiro grupo são configurações do sistema, onde se tem destaque: *deviceFamily* (família do dispositivo), *hdlLanguage* (HDL usado) e *projectName* (nome do projeto). O segundo grupo contém outra estrutura, são *module* subsequentes que representam os componentes de hardware e *parameter* a configuração dos atributos de cada componente.

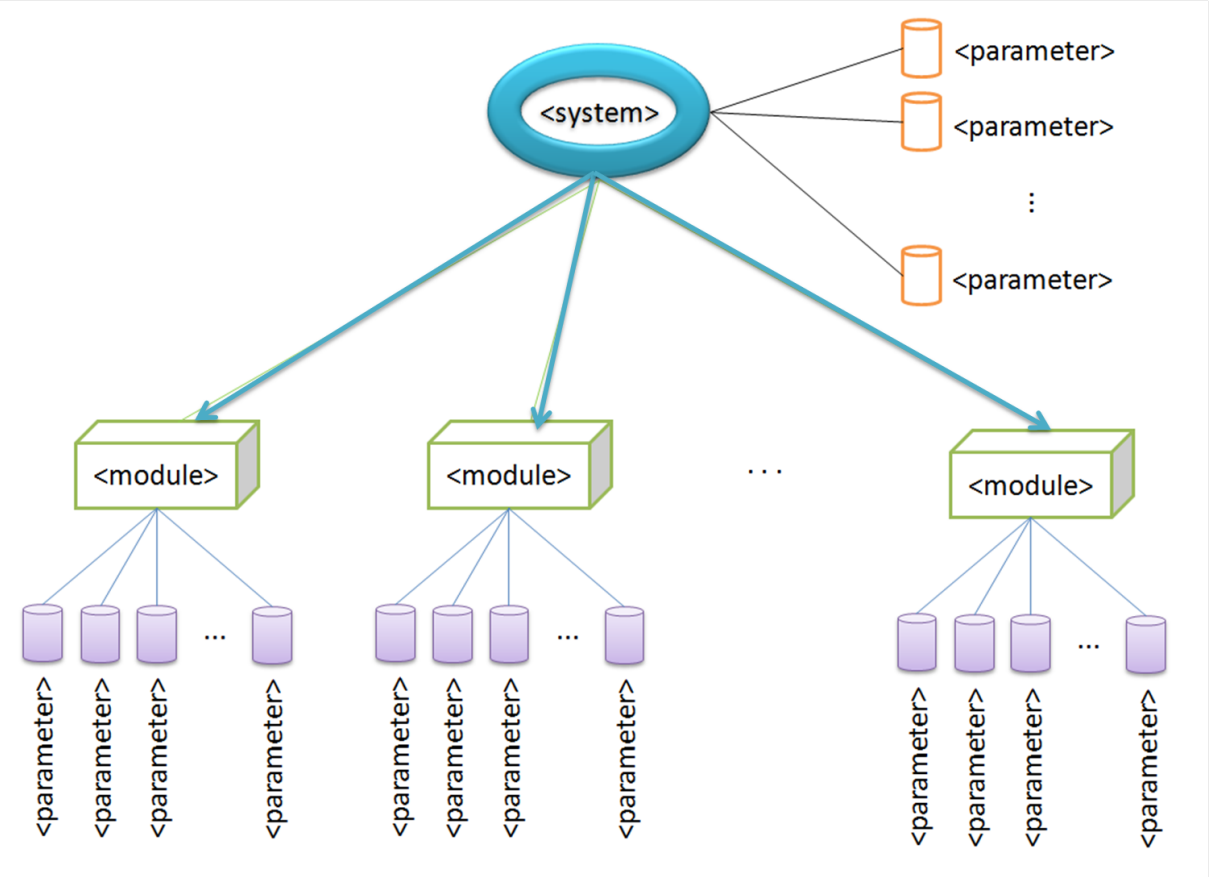

Figura 11: Estrutura de um arquivo SOPC.

2.5.2 Plataforma de Modelagem *Xilinx Platform Studio*

Xilinx atende a um largo alcance de requisições para integração dos sistemas programáveis com FPGAs e SoC. Dentre algumas de suas vantagens estão redução de custos de projeto, baixos riscos, além de incrementar dramaticamente flexibilidade. Alguns de seus mercados encontram-se: Aeroespacial/Defesa; Automotivo; *Broadcast*; Computação de Alt Performance; Industrial/Científica/Medica (ISM); Rede cabeada; *Wireless* (Xilinx, 2012). O fabricante Xilinx Corporation oferece uma gama grande de soluções de FPGA e CPLD, bem como a única família de Plataformas com Processamento Extensível, oferecendo flexibilidade e escalabilidade combinando uma performance próxima da ASIC com potência e facilidade do ASSP.

O *Xilinx Platform Studio* (XPS) (Xilinx, 2012, 2004) é uma ferramenta de esquematização de hardware para construção, conexão e configuração de processadores embarcados através de integração de *plug and play IP cores* e cálculos para estimar consumo e potência de energia.

Possui uma interface gráfica com visões e auxiliares (*wizards*) com alguns necessários passos de configuração. O XPS será abordado por nossa ferramenta, que gerará dois arquivos principais ".XMP" e ".MHS". O arquivo MHS possui uma estrutura semelhante a uma linguagem de programação com componentes representados por funções com *BEGIN* e *END* e variáveis (*TIPO identificador = valor*) são portas e parâmetros, ver figura 12. O arquivo XMP possui estrutura sequencial simples monolítica com nomes e valores discriminados, onde tem-se como importante citar: *Architecture* (arquitetura), *MHS File* (arquivo MHS), *Processor* (nome do componente de processamento), *HdlLang* (HDL) e *Device* (dispositivo).

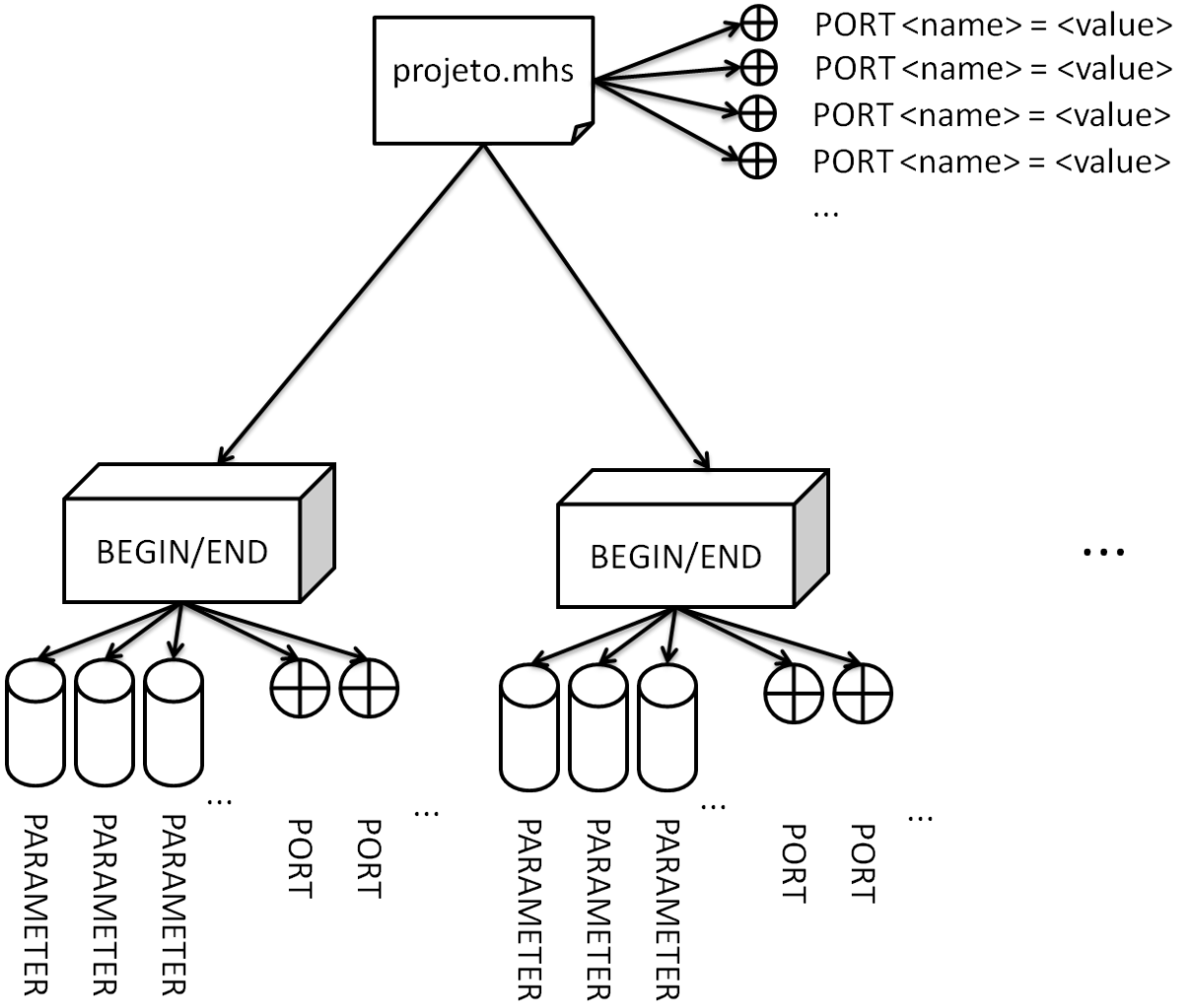

Figura 12: Arquitetura do arquivo XMP.

2.6 Conversão de Componentes

A tabela 1 exibe alguns exemplos de nomes de componentes MARTEe suas equivalências em SOPC Builder e XPS. A tabela descreve os componentes de barramento (HW_Bus), processadores (HW_Processor), clock (HW_Clock), timer (HW_Timer), memory (HW_Memory), RAM (HW_RAM) e I/O (HW_IO). Barramento é denotado como Avalom e PLB na conversão. Além desses componentes, MARTE também inclui muitos outros elementos, tais quais memória cache, que poderá ser aplicado de acordo com a necessidade do projeto. Entretanto, o compilador precisa sempre verificar se o recurso/componente da plataforma alvo está disponível. Na versão atual, se o recurso não existe, uma menssagem de erro é exibida. O modelo MARTE contém os estereótipos necessários para boa implementação porém, um novo modelo proposto poderá vir a conter novas peculiaridades.

Tabela 1: Equivalência de componente de Hw usados no SOPC builder e XPS.

MARTE	**SOPC**	**XPS**
HW_Bus	Avalom	PLB
HW_Processor	nios2_cpu	microblaze
HW_Clock	clk	clock_generator
HW_Timer	interval_timer	proc_sys_reset
HW_Memory	onchip_memory	bram_block
HW_RAM	sdram,ssram,flash	DDR3_SDRAM
HW_I/O	jtag_uart	RS232_Uart
HW_Cache	nios2_cpu− Cache	microblaze− cache
HW_Arbiter	PLL	pll
HW_Media	AUDIO,VGA	audio_if,video_if

2.7 Considerações Finais

O capítulo apresentou a linguagem de modelagem MARTE e os pacotes adotados na execução do projeto. Além disso, apresentou-se o método MDA e sua relação com MARTE e as plataformas de processamento específicas empregadas neste trabalho. Por fim, as ferramentas SOPC Builder e XPS para modelagem nas plataformas da ALtera e Xilinx foram apresentadas.

CAPÍTULO

3

Conversor I2S

A figura 13 ilustra o fluxo de dados do I2S. O conversor I2S recebe como entrada os modelos em formato XMI (XML Metadata Interchange), modelado a partir de uma ferramenta que modele e exporte o MARTE, como o MagicDraw UML (No Magic, Inc, 2012; Object Management Group, Inc, 2013), que habilita a criação de modelos por qualquer ferramenta que suporte o perfil MARTE e que tem exportação para o formato XMI. Esta flexibilidade reduz a dependência por uma particular ferramenta de modelagem, permitindo, por exemplo, o uso de soluções não comerciais, como a ferramenta Papyrus (Dubois et al., 2010).

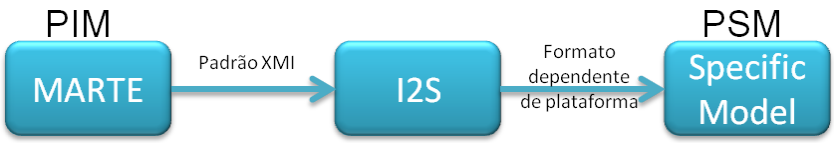

Figura 13: Fluxo do I2S. PIM -> Transmutação -> PSM.

A saída gerada pelo I2S é um arquivo com o formato suportado pela plataforma alvo (psm). Por exemplo, para gerar um modelo para a plataforma Altera, o arquivo gerado é um XML, que contém "tags" compatível com a ferramenta SOPC Builder.

Os passos poderão serem agrupados em dois estágios de conversão: interpretar o modelo de entrada e fazer sua transformação. Os tópicos 1 e 2 descrevem as atividades de cada respectivo estágio.

1. Captura: capture componentes, atributos, valores e conexões: A captura tem desenvolvimento para interpretar os modelos descritos na linguagem MARTE representada em um arquivo no formato XMI. O XMI é representado por *tags* XML com regras fixas em relação a seus nós, elementos e atributos. Um algoritmo foi usado para buscar cada componente, cada atributo dos componentes e cada valor respectivo a cada atributo. Cada conexão entre cada componente é representado por um objeto na memória do computador que contém uma lista de atributo com seus valores. Conexões são processadas como qualquer outro componente, entretanto seus objetos são ponteiros para o componente que é conectado.

2. Transformação PSM: depois de carregar os componentes na memória do computador o sistema ajusta todos objetos de acordo com a configuração desejada. Com objetivo de ajustar os valores dos atributos dos componentes, nós analisamos os atributos de cada componente possível nas plataformas SOPC-Bulder ou XPS, iniciando esta informação armazenada em bibliotecas de arquivos XML. Cada plataforma tem seus próprios arquivos XML, que podem ser facilmente atualizados em caso de qualquer modificação/aprimoramento dessas plataformas.

A figura 14 exibe os passos do algoritmo para conversão de sistemas PIM em PSM, como detalhado anteriormente: o primeiro é responsável por interpretar o modelo de entrada e o segundo responsável por garantir performance em suas transformações. Percebe-se podem existir vários componentes, atributos e conexões. As conexões são carregadas após todos os componentes para que se obtenha o registro ID de cada elemento, sendo estas ponteiros com ligações de N para N componentes. O módulo de captura foi desenvolvido para interpretar os modelos descritos na linguagem MARTE representada em um formato de arquivo XMI. O XMI é representado por tags usando formato XML com regras fixas para seus nós, elementos e atributos. O algoritmo exibido adiante foi projetado para buscar cada componente, cada atributo do componente e cada valor do atributo.

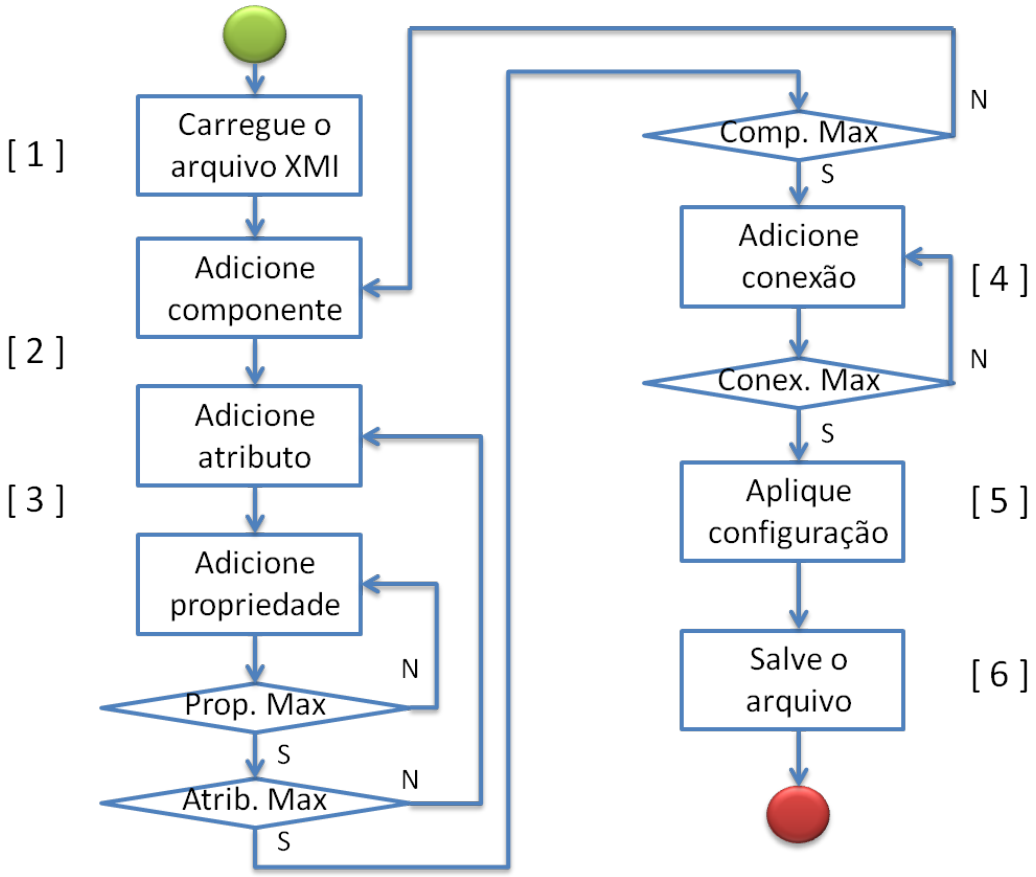

Figura 14: Gráfico de fluxo do conversor I2S.

Outra plataforma poderá ser incluída. Entretanto, neste caso, a apropriada regra de conversão deve ser bem desenvolvida. O conversor I2S proposto ajuda a modularizar a tal ponto de suportar tais expansões. Os arquivos da plataforma de especificação XML armazenam não somente os nomes dos componentes e atributos, mas também valores padrões a serem usados quando o valor apropriado não puder ser inferido no modelo de entrada. Cada plataforma PSM tem uma estrutura distinta diferente. SOPC Builder trabalha com a leitura e interpretação de dos arquivo “.sopc” do tipo XML, enquanto o XPS trabalha com os arquivos similares à programação procedural “.mhs”. Ambos “.sopc” e “.mhs” são sintetizáveis e podem gerar configuração (bitstream) para ter o sistema embarcado final sintetizável em FPGA.

3.1 Implementação da Ferramenta

A figura 15 exibe a inteface do usuário da ferramenta, cujas funcionalidades principais é converter um XML em arquivo SOPC ou MHS. Pode-se escolher modelo de placas e configurações.

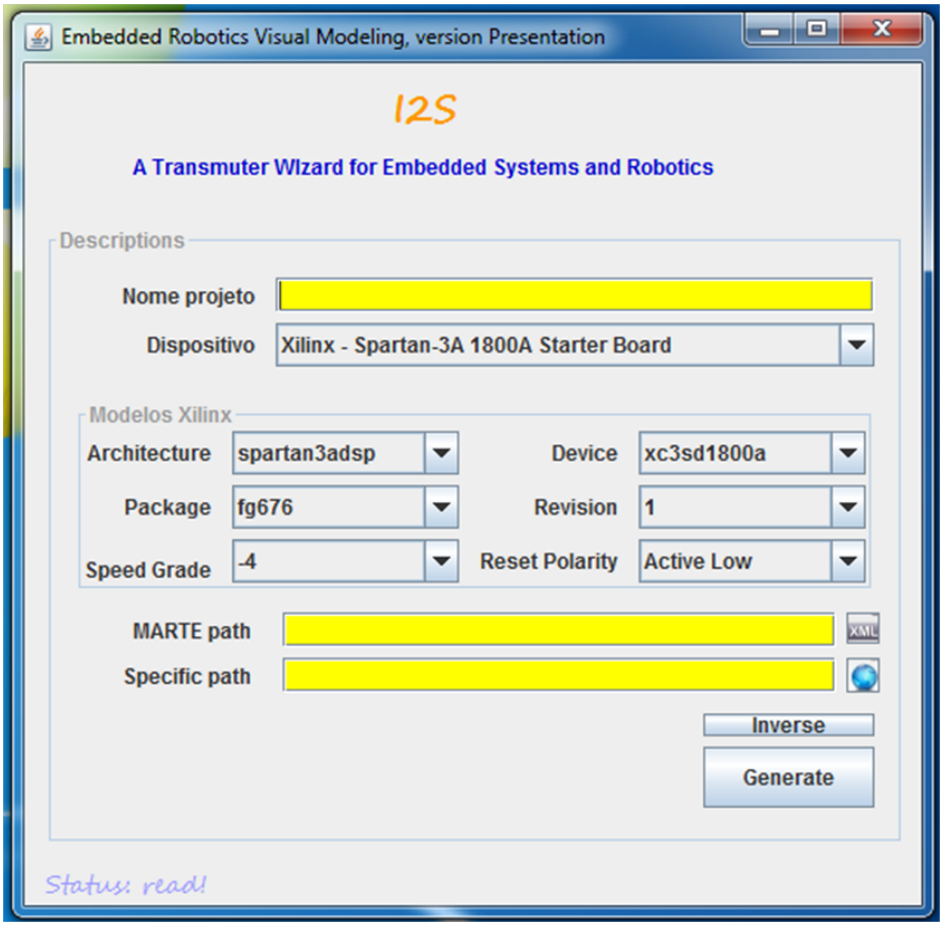

Figura 15: A interface da ferramenta I2S: auxiliar de transmutação PIM para PSM.

O conversor I2S recebe como entrada os modelos em formato XMI (XML Metadata Interchange), que habilita a criação de modelos por qualquer ferramenta que suporte o perfil MARTE. Essa flexibilidade reduz a dependência na correta ferramenta de modelagem particular, permitindo, por exemplo, o uso de soluções não comerciais, por exemplo a ferramenta open source Papyrus. O conversor gera arquivos de saída de acordo com o formato suportado pela plataforma alvo. Por exemplo, para gerar um modelo para a plataforma Altera, o arquivo gerado é um XML, que contém “tags” com a ferramenta SOPC Builder. A figura 16, a seguir, ilustra o fluxo dos dados do I2S.

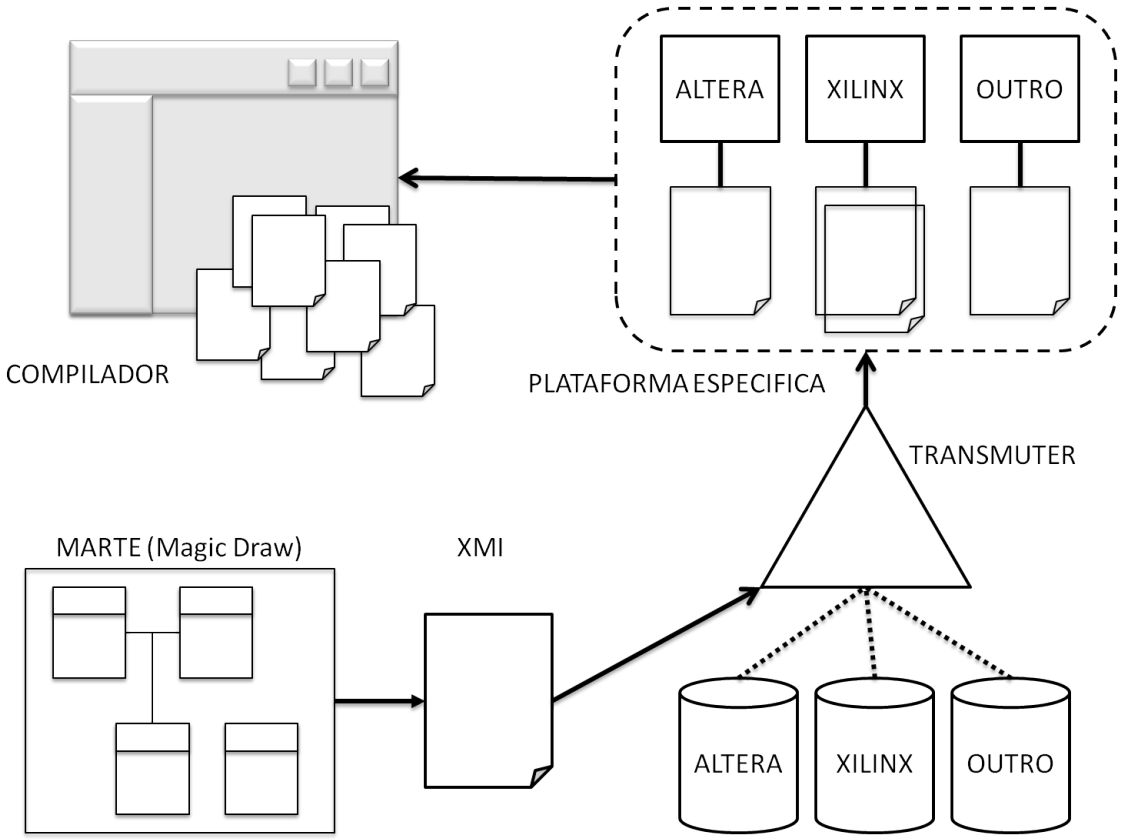

Figura 16: Fluxo do I2S. PIM -> Transmutação -> PSM. Parte de uma modelagem MARTE e exportação para XMI. Transmutador carrega o XMI e sua respectiva configuração para a plataforma específica. Exportação para uma PSM (Altera, Xilinx ou Outra). Carga do código de alto nível pelo compilador.

A figura 17 contém a transmutação inversa, cujo processo não modelará para a forma exata original e sim uma forma bem próxima" do planejado. Tal processo justifica o nome **Transmutador** porque a transformação ocorre com perdas e a volta ao modelo original não necessariamente trará a um modelo igual ao primeiro. A transmutação inversa é baseada na carga dos arquivos finais e de suas respectivas bibliotecas de configuração. Serão lidos cada componente PSM e buscado sua representação no MARTE. A ligação, que é dada por conexões do barramento a outro componente através de um endereço de hardware será transformada em ligação UML. Um arquivo XMI final será gerado e poderá ser carregado por uma ferramenta modeladora de MARTE UML.

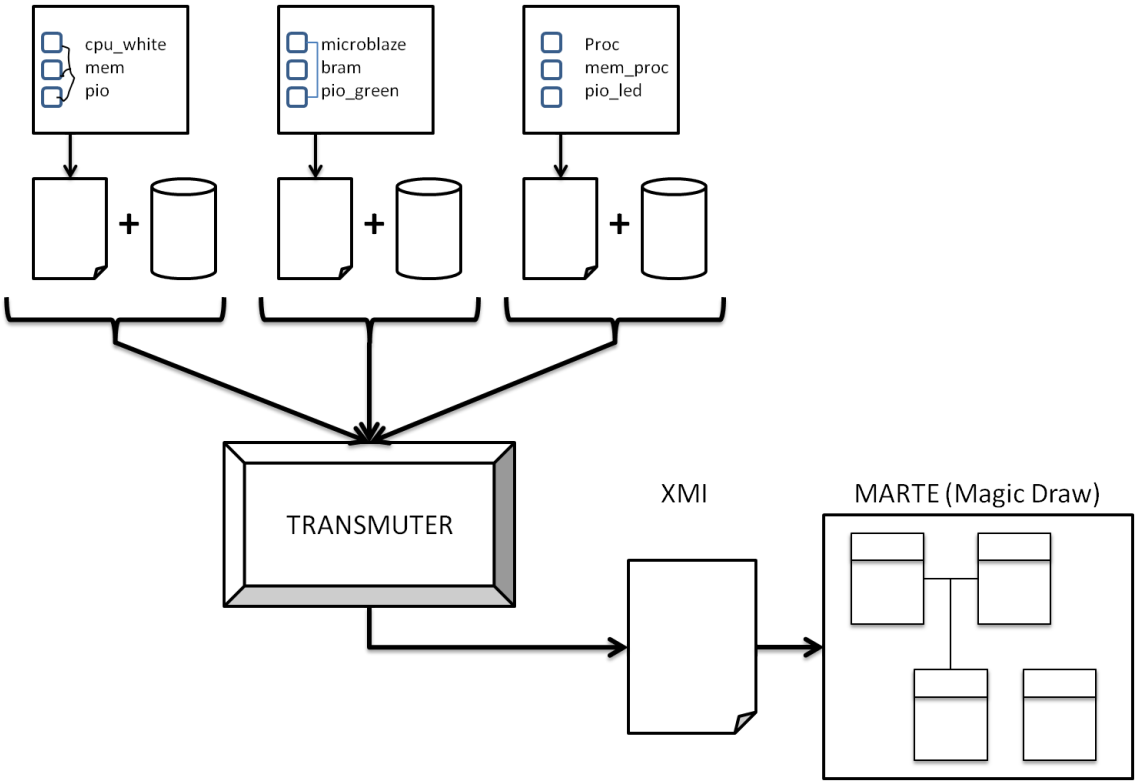

Figura 17: A partir de modelagem em plataforma específica e exportação dos modelos dos esquemas, carrega-se configuração e transforma-se (sem retorno) para um arquivo de modelo MARTE em XMI. O XMI pdoe ser carregado por uma ferramenta que use modelagem UML MARTE.

Cada plataforma PSM tem uma estrutura distinta. SOPC Builder trabalha com a leitura e interpretação dos arquivo ".sopc" do tipo XML, enquanto o XPS trabalha com os arquivos similares à programação procedural ".mhs". Ambos ".sopc" e ".mhs" são sintetizáveis e podem gerar configuração (bitstream) para ter o sistema embarcado final sintetizável em FPGA.

3.2 Conversão para Plataformas Específicas

O trecho de código do Algoritmo 1 descreve o elemento principal que especifica o processador (representado na figura 19 como $<< HwProcessor >>$) para a plataforma SOPC Builder da Altera (PSM), com um exemplo de associação de atributo. Neste caso, o processador a ser definido é o Nios II da Altera, que permite 3 diferentes configurações de estágios de pipeline: 1 (*Tiny*), 5 (*Small*) e 6 (*Fast*). Dessa forma, se no MARTE o pipeline é definido como 5, então o processador terá a configuração "Small".

Algoritmo 1: Exemplo de um código SOPC para o Nios II de um processador (cpu_0), onde o núcleo configurado é “Small” (5 estágios de pipeline).

```
1 <module name=“cpu_0” enabled=“1” version=“10.1” kind=“altera_nios2”>
2 ...
3 <parameter name=“impl” value=“Small”/>
4 ...
```

O algoritmo 2 mantém a ideia anterior, entretanto ele especifica o processador para a plataforma XPS da Xilinx (PSM). Esta plataforma suporta 3 e 5 estágios de pipeline (Xilinx, 2012). Assim, para configurar 5 estágios de pipeline, o parâmetro “C_AREA_OPTIMIZED” precisará ser configurado para 0 - seria configurado em 1 caso desejasse obter 3 estágios de pipeline.

Algoritmo 2: File of Xilinx hardware specification (“system.mhs”).

```
1 BEGIN microblaze
2 ...
3 PARAMETER C_AREA_OPTIMIZED = 1
4 ...
5 END
```

O modelo final montado é armazenado em arquivos “.sopc”para SOPC Builder e em “.mhs” para XPS. O “.sopc” emprega representação baseada em XML, na qual os componentes são funções e parâmetros são variáveis do tipo “parameter”.

3.3 Exemplo para Compreensão da Conversão

Como um exemplo bem abstrato do trabalho a ser desenvolvido, a figura 18 traz uma ilustração dos passos de conversão de modelos PIM em PSM. Desta forma, a conversão compõe-se de fazer reconhecimento dos padrões fundamentais que deverão assumir os

componentes, modelados nas regras de UML, e de convertê-los a um modelo PSM. A figura ilustra como o desenho de um quadrado com uma haste em sua parte inferior se converte em signos ordenados que lembram a forma geométrica inicial, descrita anteriormente. A tradução do modelo UML para o formato específico a partir de regras de bibliotecas de configuração requer que seja especificado "O Quê" deverá ser transformado e "Como" transformar a partir das configurações compatíveis selecionadas na especificação do primeiro modelo. Os dados serão convertidos, sem retorno, para um modelo final. Um percurso inverso, indicando da modelagem específica à genérica poderá também ser realizado mas não necessariamente será igual a forma original. Nesta mesma figura, é mostrado como uma composição de estrelas, círculos e triângulos, lembrando uma forma de um retângulo e uma haste, que é a "forma inicial", ou o modelo UML, e as bibliotecas de configuração especificando quais tipos de formas e que tipo de ligação deve possuir no final da conversão.

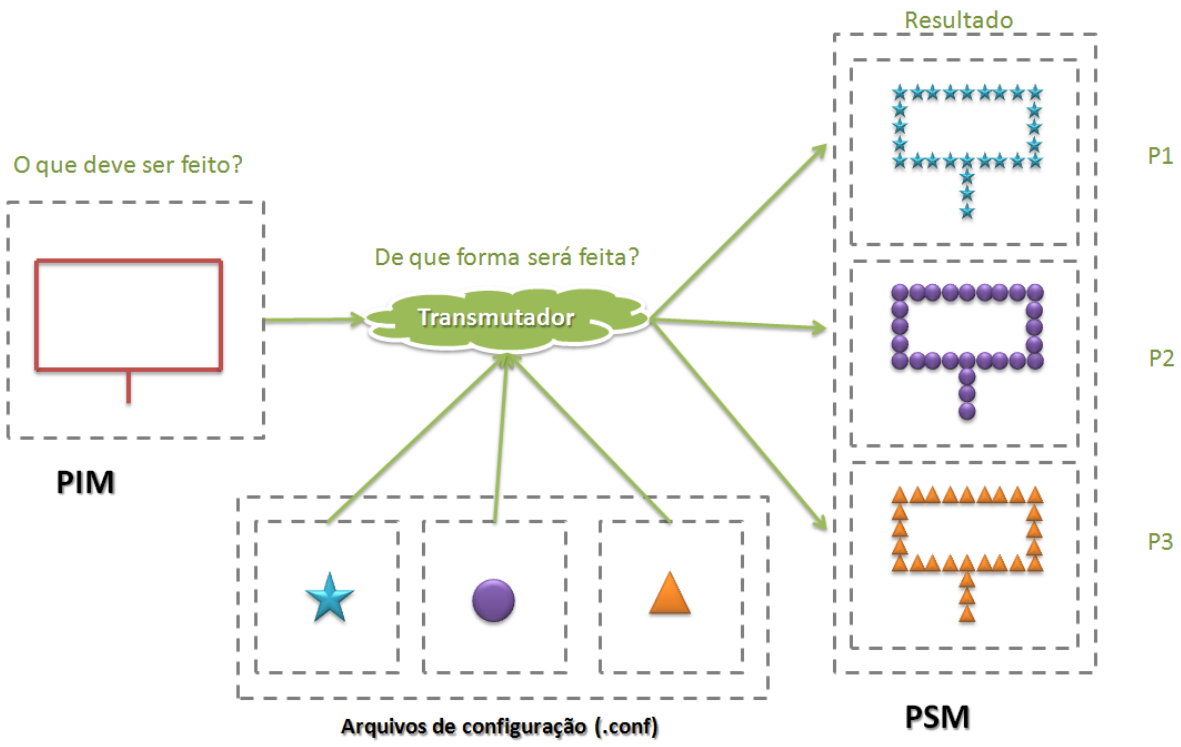

Figura 18: Exemplo da lógica de transmutação de modelo PIM em PSM. No exemplo, um quadrado foi desenhado e convertido em combinação de componentes especificados pelas respectivas biliotecas.

3.4 Biblioteca de Configuração para Outras Plataformas

Baseado na necessidade por um conjunto de regras que permita a inserção de novas plataformas, de tal forma que se possa ter o mínimo de esforço possível para se implementar essa conversão.

A lógica da carga da nova plataforma se baseará em uma biblioteca de configurações em XML, onde um arquivo com as configurações da nova plataforma é criado com o formato ".config". A estrutura será responsável por conter os dados do componente MARTE para a nova plataforma, bem como atributos e valores na configuração MARTE e na configuração nova. Para que se trabalhe com mais de um arquivo ou regras especiais, deve-se optar por modificar a classe java para aceitação da nova plataforma.

Tal estrutura do XML de configuração obedece ao padrão, com 1 ou mais de cada nó em respectiva alocação:

1. Nome
2. Versão
    (a) Componente MARTE
    (b) Componente PSM
        i. Atributo MARTE
        ii. Atributo PSM
            A. Valor MARTE
            B. Valor PSM

3.5 Considerações Finais

Nesta seção obteve-se uma visão mais funcional da ferramenta I2S. Foi apresentado o algoritmo do software desenvolvido, um exemplo de como funciona a transmutação, um modelo

genérico de biblioteca e uma tabela de conversão do PIM MARTE em PSM. O software foi projetado a partir da lógica MDA, onde um transmutador converterá um AML em modelo final mapeado.

CAPÍTULO

4

Resultados

Este capítulo apresenta resultados experimentais obtidos com o gerador I2S aplicado na geração de sistemas embarcados voltados à execução do algoritmo EKF (Extended Kalman Filter). O objetivo desse experimento é realizar a exploração de modo que um conjunto de variações de arquiteturas de hardware sejam geradas por aritmética através do software I2S.

Para desenvolvimento da pesquisa, foi utilizado as ferramentas Quartus e ISE exploradas com os módulos SOPC e XPS em versões 10 e 8.1, respectivamente, para os modelos finais. O compilador Nios II (Altera) e XPS EDK (Xilinx) são usados para testes com os algoritmos em linguagem de programação C/C++. Para validar os experimentos foram utilizadas as placas Altera DE2-70 e Xilinx Spartan 3. Foi utilizado o Compilador Netbeans, a princípio, com a linguagem de programação Java e padrões (Pires et al., 2010; Liu & Chen, 2009) para desenvolvimento do software.

4.1 Especificação do Modelo

A figura 19 exibe um modelo (MARTE) de uma arquitetura tradicional de sistema embarcado contendo um processador, uma memória, um barramento de comunicação, I/O e dispositivos de monitoramento. Cada um contém uma série de parâmetros para configurar o sistema de acordo com os requisitos do projeto. Nesse exemplo, o processador é composto de

1 core e tem 5 estágios de pipeline; a memória tem parâmetros indicando que é síncrona, não estática e tem capacidade de 8KB.

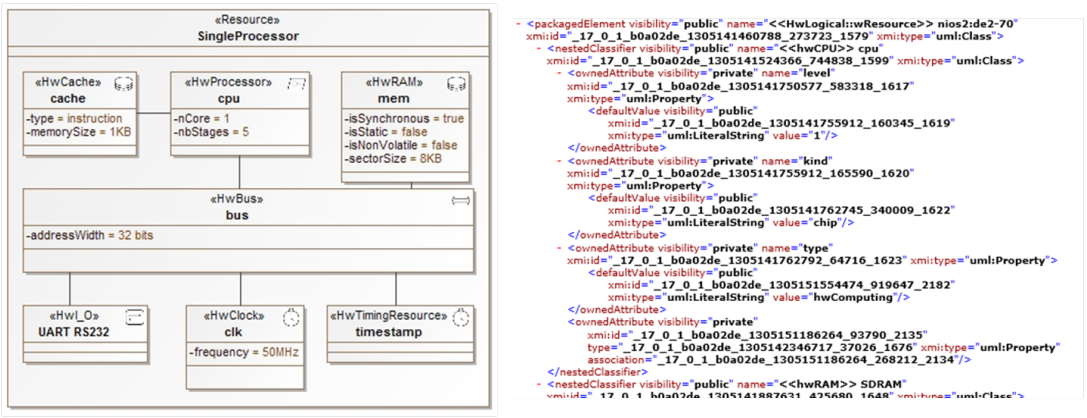

Figura 19: Exemplo de um modelo básico de sistema embarcado usando o perfil MARTE.

No caso dos parâmetros não especificados no modelo MARTE, o conversor I2S automaticamente utiliza um valor padrão, assegurando a consistência e a funcionalidade do projeto a ser gerado. Por exemplo, se a frequência de clock não é especificada, o valor padrão predefinido dos componentes das bibliotecas é assumido. A complexidade do conversor é diretamente relacionada à complexidade dos sistemas embarcados projetados. Em um sistema é necessário garantir a inexistência de conflitos entre componentes para ter um sistema trabalhando corretamente. Neste sentido, é necessário, por exemplo, gerenciar o endereço de cada elemento conectado ao barramento, para conectar os componentes corretamente, evitando conflito entre os dispositivos e alocar recursos compatíveis com as funcionalidades especificadas no modelo de entrada.

4.2 Exploração do Espaço de Projeto

A principal contribuição deste trabalho é otimizar o processo de exploração do espaço de projeto, permitindo ao usuário gerar um conjunto de configurações de sistema embarcados cada um com necessidades particulares para recursos de hardware e desempenho. A tabela 1 demonstra um conjunto de configurações geradas de variações de estágios de pipeline e tamanho da memória cache cujo projeto de exemplo é suportado pela figura 19. A tabela exibe o resultado de síntese para duas FPGAs: Stratix, da Altera; e Spartan da Xilinx. Nestes exemplos, o sistema

é compilado usando nenhuma cache ou variações de 1, 2 e 4 KBytes de cache de instrução. Quanto ao número de estágios de pipeline, as configurações variam entre 1, 3, 5 e 6 níveis, os quais dependem do nível permitido pelo processador alvo. Neste experimento, foi usado processadores Nios II da Altera e Microblaze da Xilinx, que permitem níveis 1, 3 e 6, e níveis 3 e 5, respectivamente. A tabela somente apresenta uma fração de possíveis configurações, uma vez que elas podem ser combinadas com outros parâmetros, como cache de dados, número de cores/processadores (multi-cores) e aceleradores de hardware customizados (i.e., FPU). Baseado neste conjunto de configuração e na performance de cada opção, o projetista pode escolher a configuração que melhor satisfaz os requisitos de especificação. Além da performance, a potência e energia de consumo é também outra característica importante de sistemas embarcados que poderá ser incluída na análise de comparação. Como exibido na tabela 1, o incremento nos estágios do pipeline resulta em mais registradores e CL (Células Lógicas), incluindo *Look-Up Tables* (Tabelas de Consulta) e controle de elemento de *data-path*, enquanto as modificações no cache têm pouco impacto nos elementos necessário para implementar o controlador de memória. Baseado neste resultado de síntese, projetistas poderão escolher aquelas configurações que se encaixarem melhor nos requisitos do sistema.

Tabela 1: Recursos FPGA para cada configuração de sistema embarcado gerada automaticamente pelo conversor I2S.

FPGA	Pipeline Stages	Cache (KB)	EL (%)	FF	On-chip Memory (Bits)
	1	no	1,354(13%)	700	76,800(8%)
	5	1	1,824(17%)	994	85,376(9%)
	5	2	1,823(17%)	995	93,888(10%)
Altera Stratix	5	4	1,825(17%)	996	110,848(12%)
EP1S10B672C6	6	1	2,017(19%)	1,120	85,888(9%)
	6	2	2,018(19%)	1,121	94,400(10%)
	6	4	2,021(19%)	1,122	111,360(12%)
	3	no	2,553(54%)	2,475	165,888(45%)
Xilinx Spartan	5	1	3,901(83%)	3,279	184,320(50%)
XC3S500E	5	2	3,198(68%)	3,187	221,184(60%)
	5	4	3,150(67%)	3,187	239,616(65%)

4.2.1 Análise de FPGAs

Para se compreender melhor as FPGAs trabalhadas e quais recursos são vantagens de cada família, tem-se uma análise comparando: velocidade de processamento (MHz), quantidade de elementos lógicos (x1000), memória de instruções (Mbits), quantidade de canais transcepção, e quantidade de instruções de entrada/saída. São analisados os dois maiores fabricantes de FPGA (Altera e Xilinx), cuja descrição do estudo segue adiante. Foram comparados os melhores e piores resultados de cada uma e incorporados nas tabelas a seguir. São as famílias da Altera que foram analisadas: Stratix IV, Arria II, Cyclone IV (Altera, 2011, 2010a; First Silicon, 2006; Altera, 2010b). Das famílias Xilinx, foram estudadas: Virtex, Kintex e Artix na versão 7, todas (Xilinx, 2010a,e; Baidu, 2011; Xilinx, 2010b,d, 2009). Os dados analisados são expostos a seguir, primeiro nas tabelas de menor índice 2 e maior índice 3, depois em gráficos característicos de menores taxas, ver figura 20, e maiores taxas, ver figura 21, para cada família, onde se destaca a Stratix em frequência de processamento, canais de transcepção e pilha de acesso às instruções, a Virtex se destaca em número de elementos lógicos e em número de entrada/saída. Vale iterar que foram considerados os processadores por ano e proximidade.

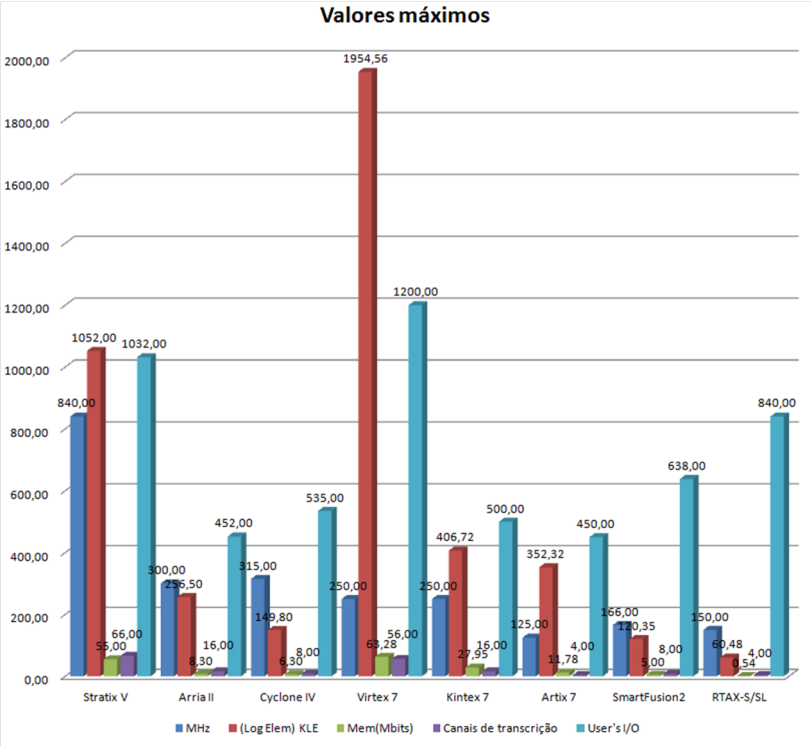

Figura 20: Gráfico resultante das características de cada FPGA. Análise dos maiores valores. Análise feita em fevereiro de 2011.

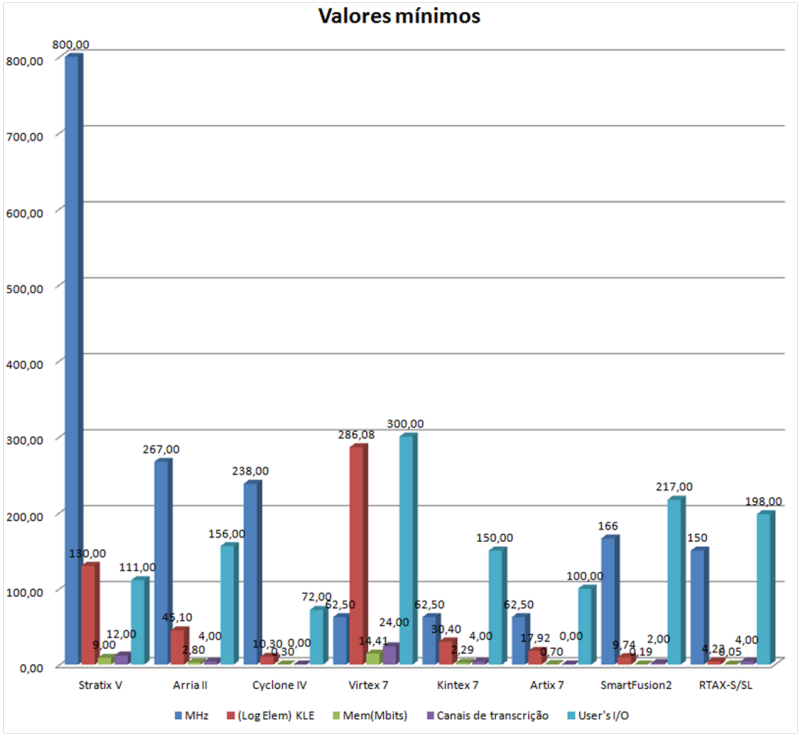

Figura 21: Gráfico resultante das características de cada FPGA. Análise dos menores valores. Análise feita em fevereiro de 2011.

Tabela 2: Tabela de comparativo entre elementos de processamento das familias mais populares em 2011. Maiores valores. Famílias: Stratix V, Arria II, Cyclone IV, Virtex 7, Kintex 7 e Artix 7.

Processador	MHz	KLE	Mem (Mbits)	Transc.	User's I/O
Stratix V	840,00	1052,00	55,00	66,00	1032,00
Arria II	300,00	256,50	8,30	16,00	452,00
Cyclone IV	315,00	149,80	6,30	8,00	535,00
Virtex 7	250,00	1954,56	63,28	56,00	1200,00
Kintex 7	250,00	406,72	27,95	16,00	500,00
Artix 7	125,00	352,32	11,78	4,00	450,00

Tabela 3: Tabela de comparativo entre elementos de processamento das familias mais populares em 2011. Menores valores. Famílias: Stratix V, Arria II, Cyclone IV, Virtex 7, Kintex 7 e Artix 7.

Processador	MHz	KLE	Mem (Mbits)	Transc.	User's I/O
Stratix V	800,00	130,00	9,00	12,00	111,00
Arria II	267,00	45,00	2,80	4,00	156,00
Cyclone IV	238,00	10,30	0,30	0,00	72,00
Virtex 7	62,50	286,08	14,41	24,00	300,00
Kintex 7	62,50	30,40	2,29	4,00	150,00
Artix 7	62,50	17,92	0,70	0,00	100,00

4.3 Análise da Aplicação do EKF

A partir das configurações apresentadas na tabela 1, que foram geradas automaticamente pelo conversor I2S a partir do modelo MARTE da figura 19, realizou-se uma análise do tempo de processamento do algoritmo EKF (Kalman, 1960) em função dos recursos de hardware de cada configuração gerada para as FPGA da Altera e da Xilinx. Cabe lembrar que, no caso da a FPGA da Altera, o sistema foi gerado no SOPC Builder e o FPGA Xilinx, através da ferramenta XPS.

As figuras 22 e 23 apresentam os respectivos modelos gerados. Na figura 22, são exibidos os componentes finais da plataforma SOPC Builder: clocks, JTAG, CPU, memória AltPll e Timestamp. Na figura 23, a modelagem representa os componentes na plataforma XPS: CPU, Memória de dados e instrução, PLB, RAM Blocks, RS232 e clock.

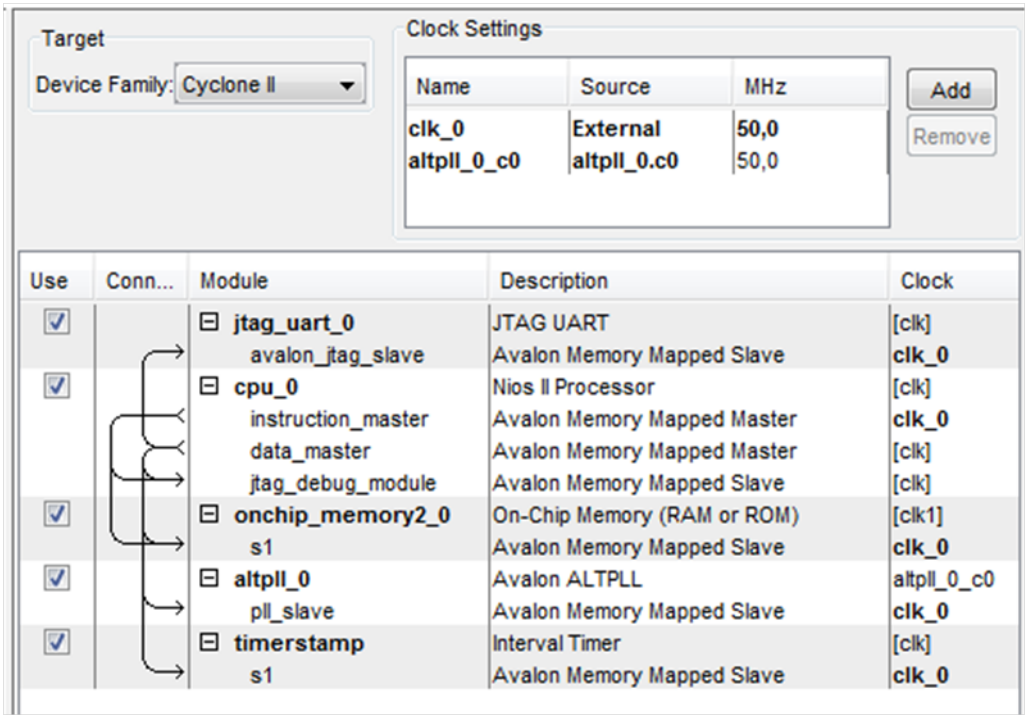

Figura 22: Modelo específico da modelagem Altera, no SOPC Builder, com recursos mínimos.

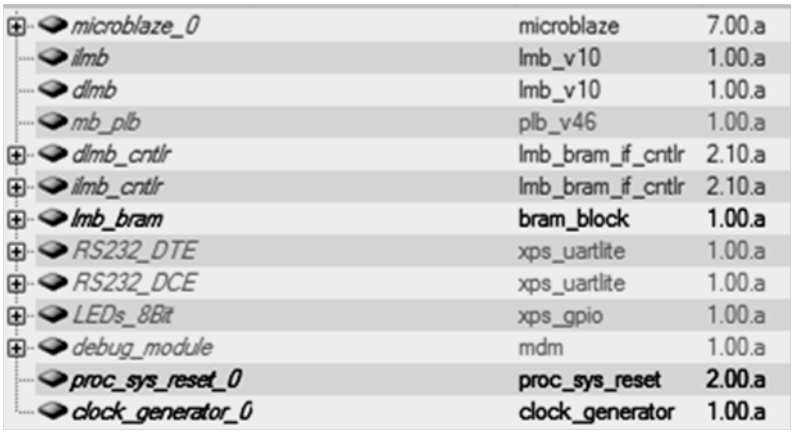

Figura 23: Modelo específico da modelagem Xilinx, através do XPS, com recursos mínimos.

O tempo de processamento pode ser analisado nas figuras 24, para Altera, e 25, para Xilinx, onde o tempo é apresentado em milissegundos (ms). Para Altera, apenas para a configuração de 1 estágio de pipeline que se tem uma grande diferença de tempo, as demais apresentaram uma curva quase linear com uma taxa de crescimento padrão. Pode-se concluir que qualquer configuração diferente de 1 estágio de pipeline pode ser usada com máxima eficiência, uma vez que a diferença pode ser considerada pequena. Os gráficos são Fronteiras de Pareto. Na figura 24, os gráficos com a fronteira de pareto mostram dados importantes quanto ao ganho de tempo versus recursos que representam-se nas barras paralelas.

Pode-se perceber uma padronização na linha quando no trecho de Pipe 5 e Cache 1i a Pipe 6 e Cache 2d2i com uma porcentagem de ganho de tempo de 5,044% e 17.987% com todas configurações. Para demais recursos da fronteira de pareto, uma sequência de número de elementos lógicos com padrão 34,712%,-0,055%, 0,11%, 10,521%, 0,05% e 14,866%. Ganho médio de EL de 3,437%. Para número de registradores, obteve-se uma média de crescimento de 9.31% e para blocos de memória1 2.929%.

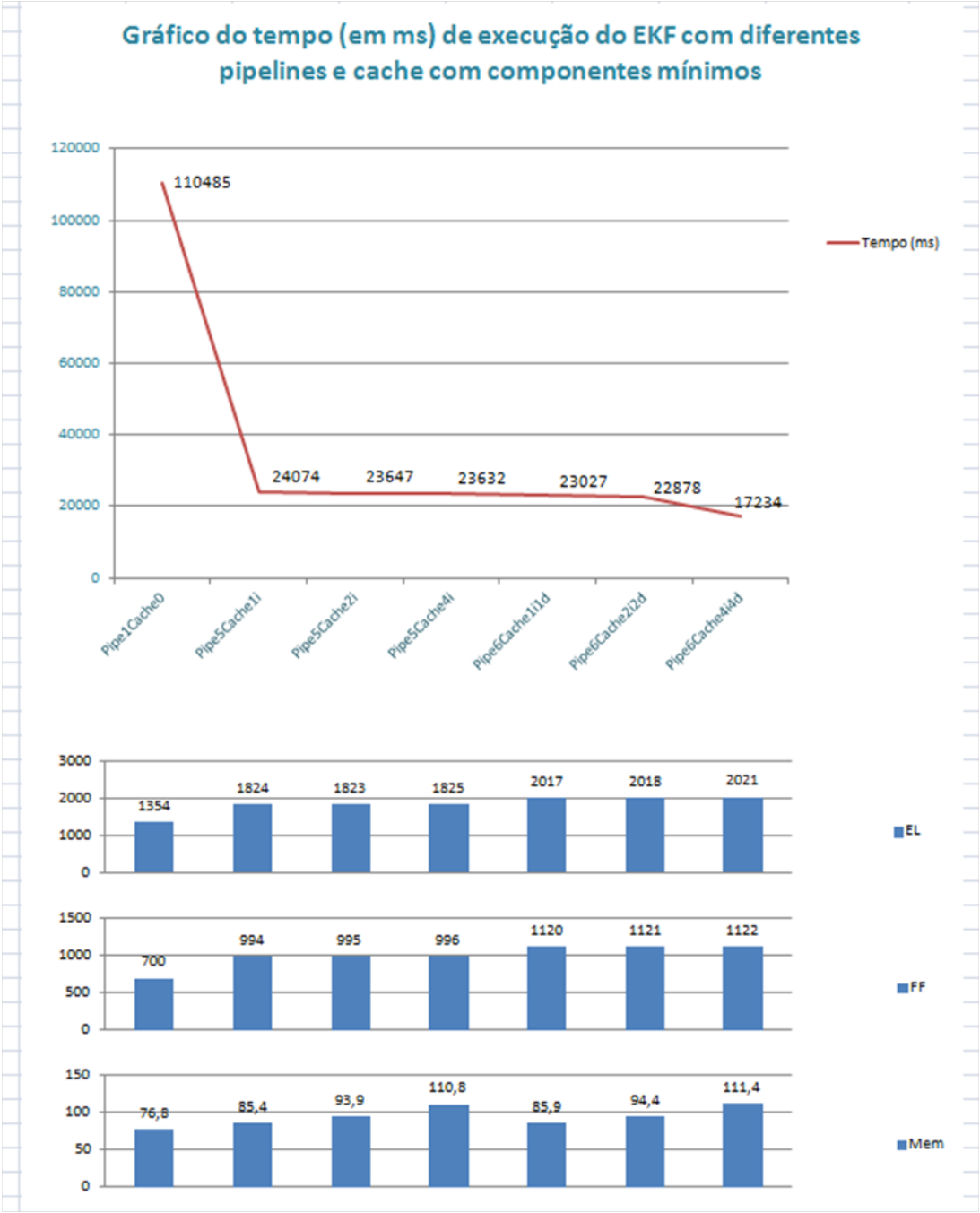

Figura 24: Gráfico de tempo do algoritmo EKF para Altera com FPGA Cyclone II.

Para Xilinx, figura 25, ao todo, foi ganho, em média, 4.3% e com Pipe 5 e Cache 1d1i a Pipe5 Cache 4d4i, um ganho de 4.3% por teste (x_i), EL com média de crescimento de 11,093%, Registradores com taxa de crescimento de 9,895% e Taxa de crescimento médio de memória de 13,148%

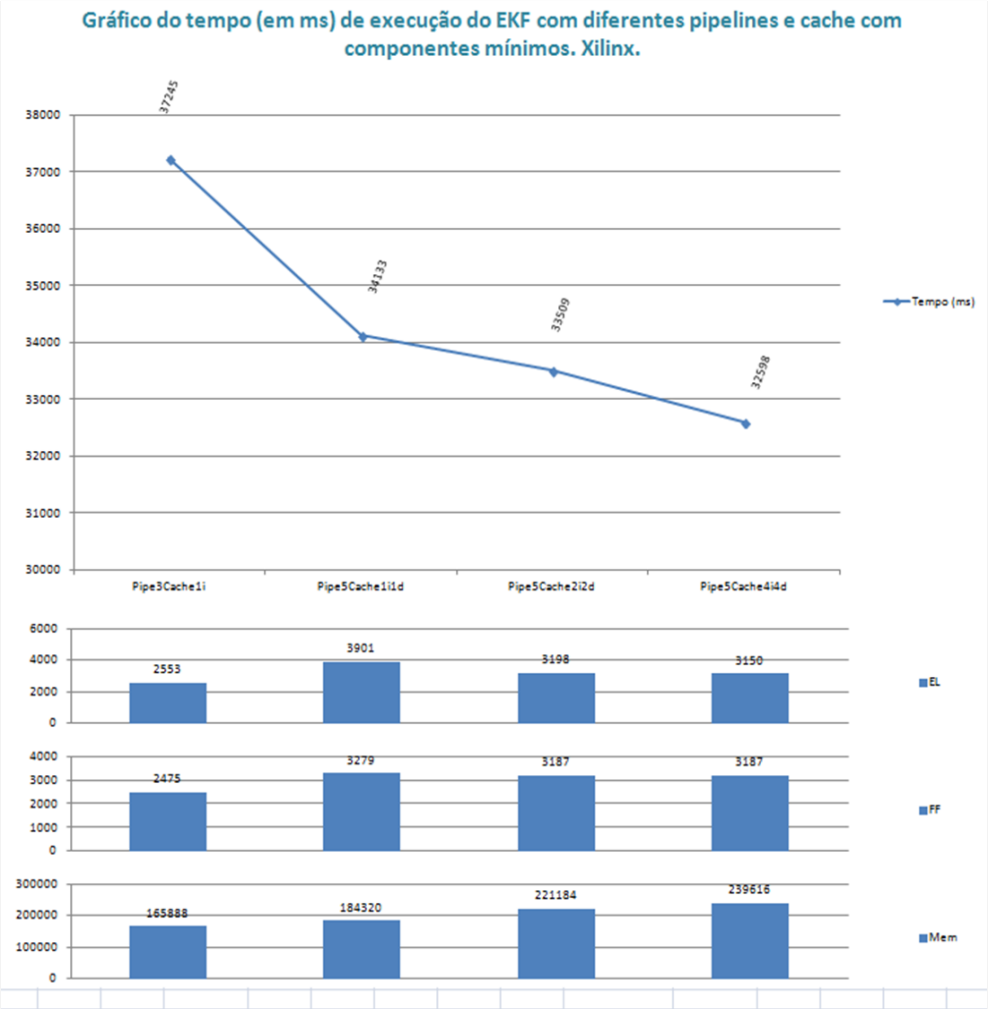

Figura 25: Gráfico de tempo do algoritmo EKF para FPGA da Xilinx com FPGA Spartan 3.

4.4 Análise do EKF com Sistema Utilizando Memória Externa

Foram feitas análises do algoritmo de EKF em sistema de hardware FPGA com recurso de acréscimo de memória externa, como na figura 26, onde 3 memórias diferentes (DRAM, SRAM, FLASH) foram inseridas no sistema tradicional, anterior, justificando se obter mais recursos de tal forma que se teste sua eficiência de tempo quanto a diferentes números de pipeline e cache, além de ter uma visão mais completa. O sistema possui todos recursos, incluindo memórias externas, cuja modelagem específica Altera e Xilinx é exibida pelas figuras respectivas 27 e 28.

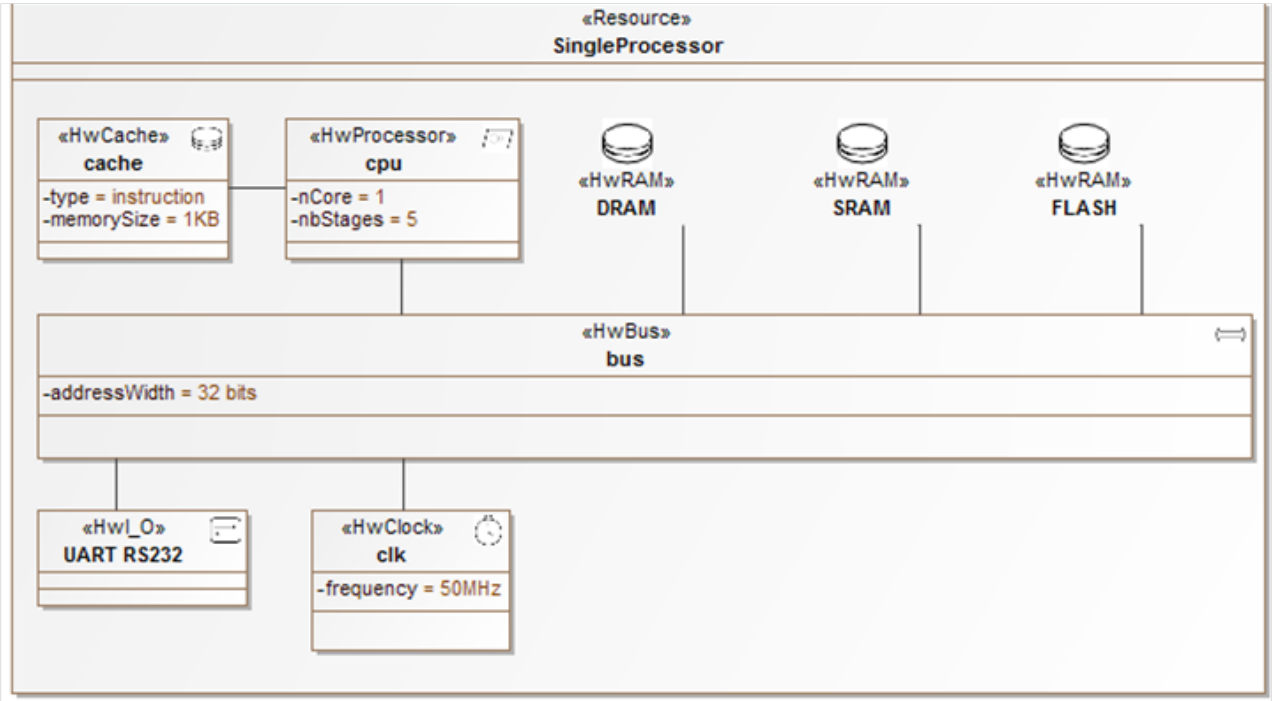

Figura 26: Modelagem de sistema com memória (DRAM, SRAM, FLASH) para se obter mais recursos externos.

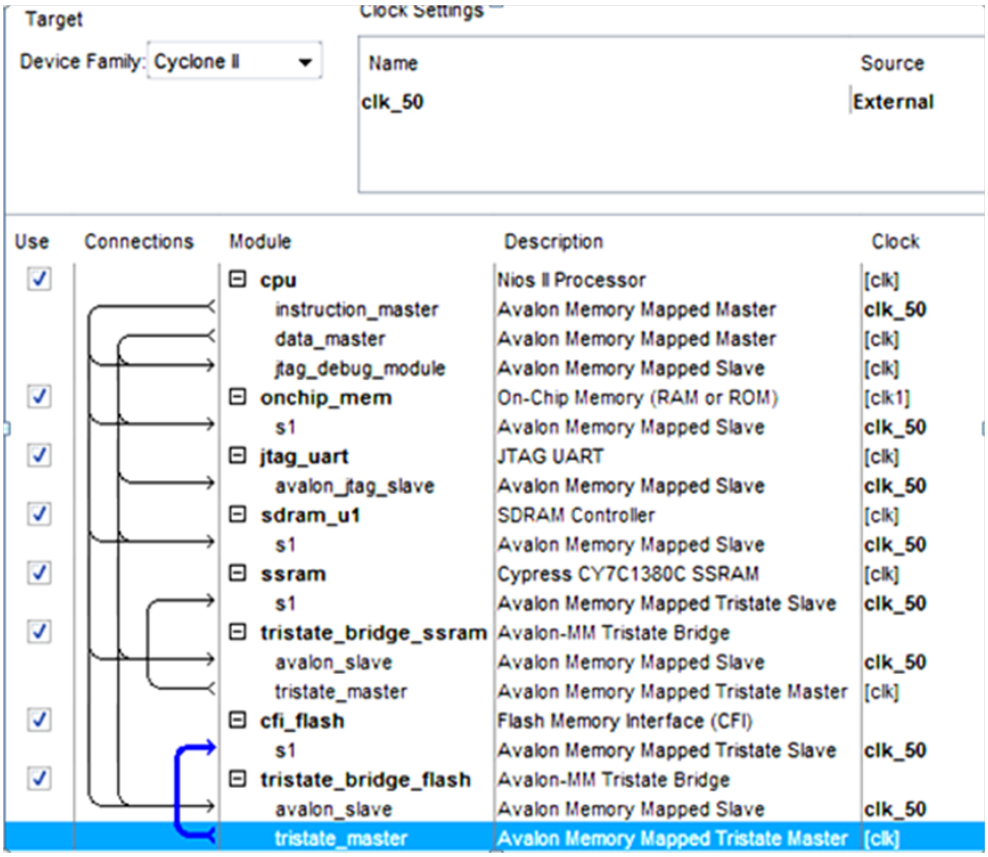

Figura 27: Modelo específico da modelagem Altera, no SOPC Builder, com memória externa.

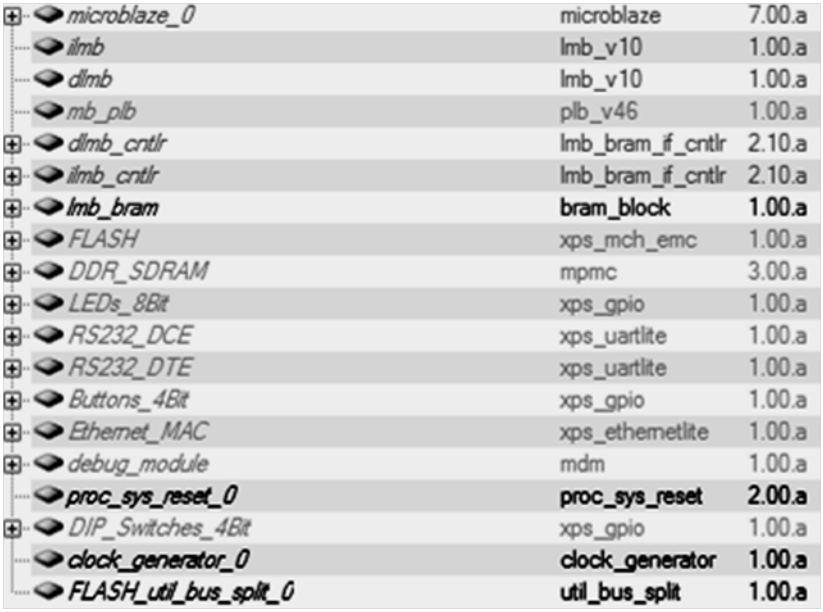

Figura 28: Modelo específico da modelagem Xilinx, através do XPS, com memória externa.

O resultado pode ser analisado na tabela 4 para Altera, e tabela 5 para Xilinx, onde o tempo de simulação é apresentado em milissegundos (ms). Percebe-se um decremento linear com relação ao incremento de estágios de pipeline e cache, ver figura 29 (Altera) e 30 (Xilinx). Os gráficos são Fronteiras de Pareto com os recursos de memória externa (1000 bits), número de registradores (FF) e elementos lógicos (EL).

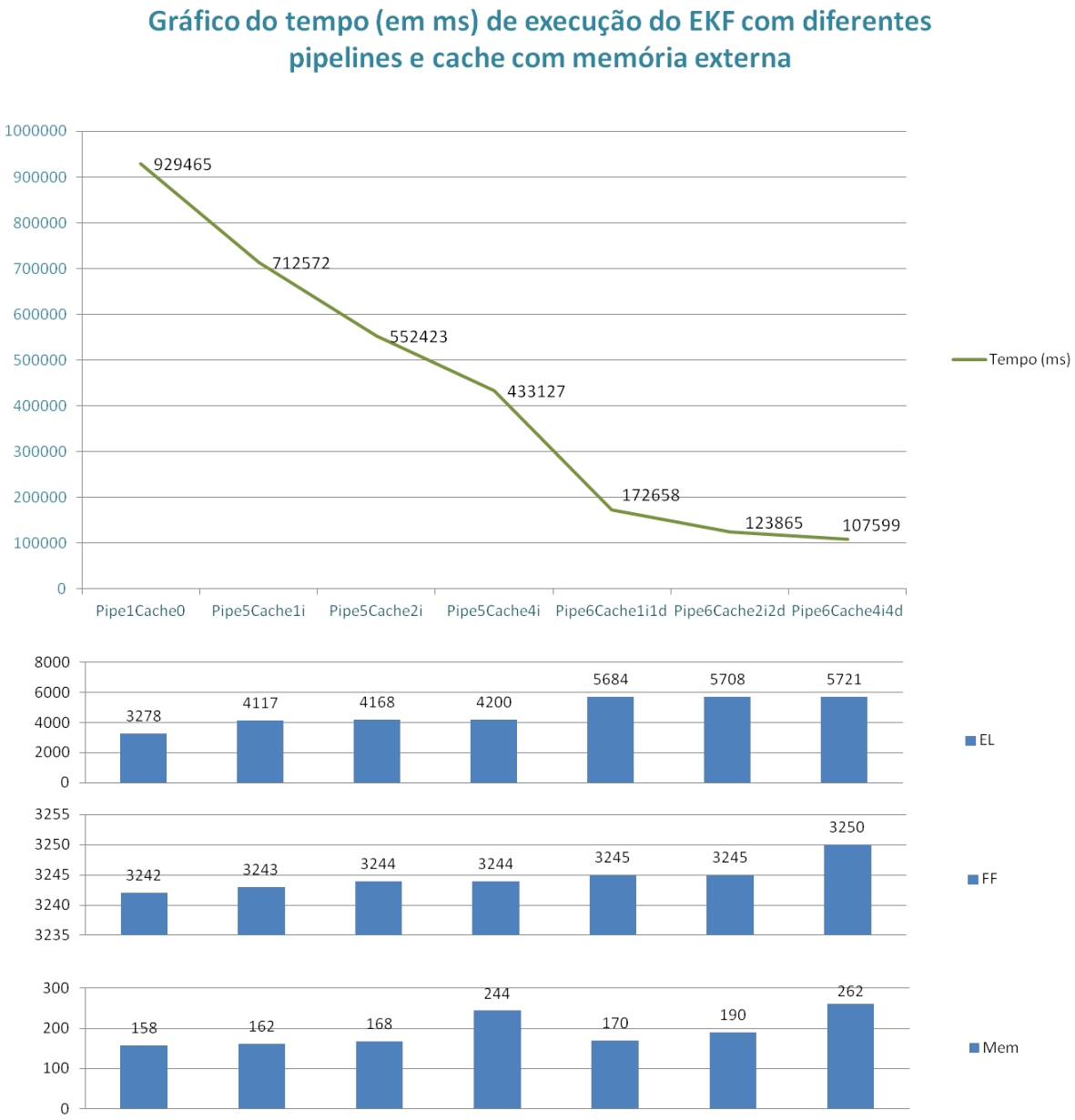

Figura 29: Gráfico de tempo do algoritmo EKF para sistema com memória externa para Altera Cyclone II.

Tabela 4: Análise do algoritmo EKF em sistema com memória externa para Altera.

Pipeline	Cache	Tempo (ms)	EL	FF	Mem
1	0	929465	3278	3242	158
5	1i	712572	4117	3303	162
5	2i	552423	4168	3311	168
5	4i	433127	4200	3321	244
6	1i1d	172658	5684	3345	170
6	2i2d	123865	5708	3369	190
6	4i4d	107599	5721	3391	262

Nestes resultados, percebe-se ganho de tempo quase linear em: 13.132%, 28.26%, 60.137%, 21.595%, 22.475%, 23.33%; com média de crescimento em 28.155%. Elementos Lógicos: 34.712%, -0.054%, 0.11%, 10.52%, 0.05% e 0.149%; e média 7,581%. Registradores: 42%, 0.1%, 0.1%, 12.45%, 0.089%, 0.089%; média 9,138%. Blocos de Memória: 11.198%, 9.953%, 17.998%, 22.473%, 9.895%, 18%; 7,428% de média.

Tabela 5: Análise do algoritmo EKF em sistema com modelagem de memória para Xilinx.

Pipeline	Cache	Tempo (ms)	EL	FF	Mem
3	1i	1646041	4211	3219	387651
5	1i1d	347210	5974	4301	398772
5	2i2d	255161	5978	4333	430022
5	4i4d	216273	6023	4342	440001

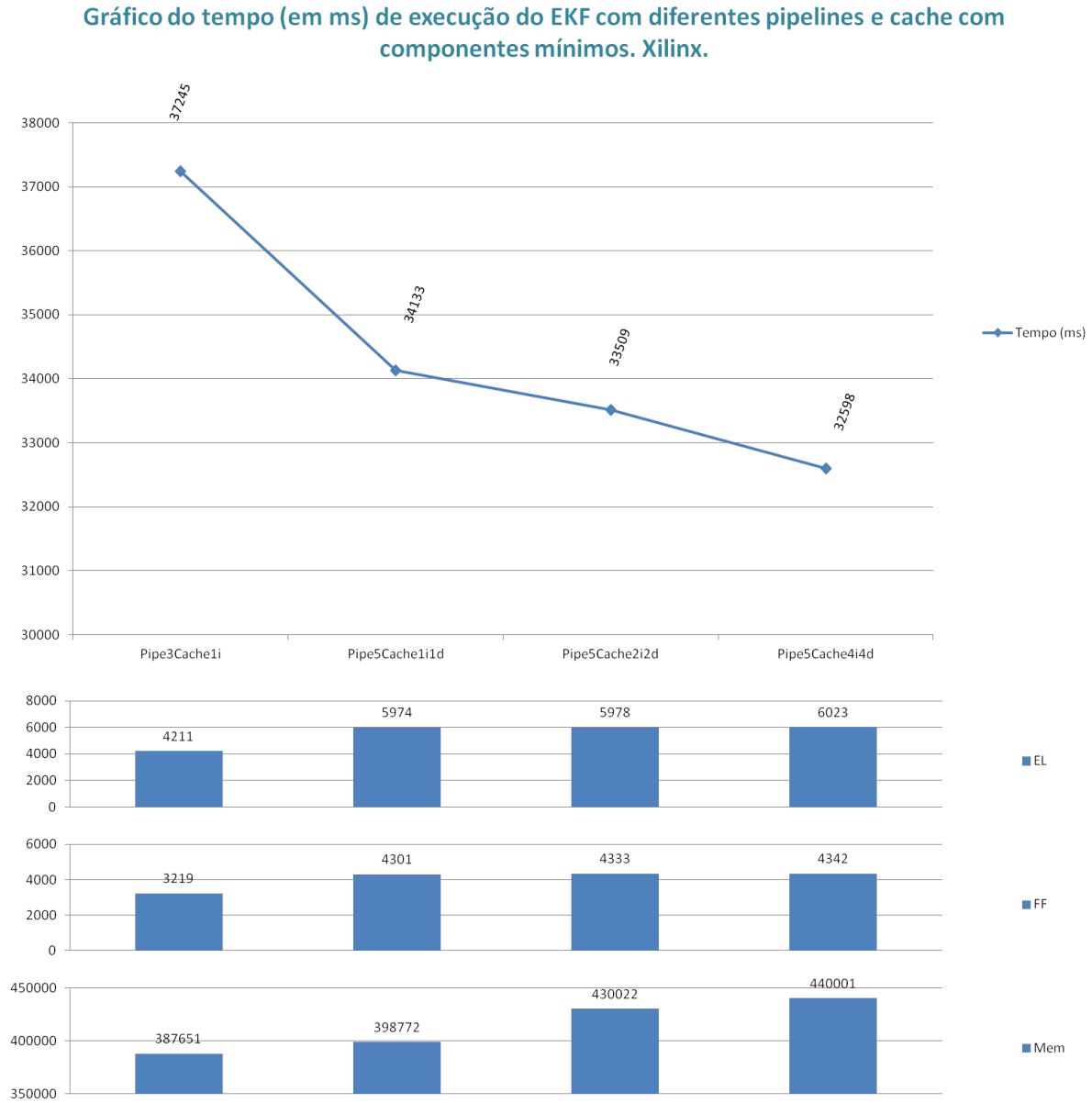

Figura 30: Gráfico de tempo do algoritmo EKF para sistema com memória externa para Xilinx Spartan 3.

Na Xilinx, com tais recursos de memória, houve de ganho de tempo uma sequencia em: 78%; 26,511%; 15,241%. O ganho médio de tempo em cada iteração é de 40,219%. EL: 52,8%, -18,021%, -1,5%; com média de crescimento de 11,093%. FF: 32,485%, -2,806%, 0%; média 9,893%. Mem: 11,111%, 20%, 8,333%; média e 13,148%.

4.5 Considerações Finais

Este capítulo apresentou a geração automática de diversas configurações de um sistema embarcado para o processamento do algoritmo EKF nas plataformas da Altera e da Xilinx. Para cada configuração foram extraídos o tempo de processamento do algoritmo e os recursos de hardware necessários à implementação dos sistemas em FPGAs.

CAPÍTULO

5

Conclusão

O trabalho atual apresentou um conversor que transforma Modelos Independente de Plataforma (PIM), desenvolvido no MARTE, para um Modelo de Plataforma Específica (PSM), que pode ser Altera ou Xilinx. Ele demonstrou que MARTE pode ser aplicado para ajudar no processo de desenvolvimento de sistemas embarcados baseados em FPGA. A principal vantagem da ferramenta proposta é que ela permite modelos serem especificados independente da plataforma alvo, fazendo possível o reuso e automatizar a exploração do espaço de projeto em mais que uma plataforma. Tal estrutura invoca como benefício redução de tempo de desenvolvimento e possibilidade de gerar sistemas com nível de customização mais próximo da especificação dos requisitos. Neste trabalho, foi explorada a modelagem de Hardware do MARTE (*MARTE::DRM::HRM*) e pode ser estendida para incluir modelagem de software (*MARTE::DRM::SRM*) como trabalho futuro, que poderá conter informações auxiliares à análise de performance, de tal forma a explorar aplicações de tempo real.

O uso da ferramenta I2S facilita a automatização da exploração do espaço de projeto em plataformas de FPGA da Altera e Xilinx. Com isso, o projetista poderá escolher de modo mais rápido o hardware do sistema que possuir a melhor relação custo x benefício para a sua aplicação. Neste trabalho, foi explorada a execução do EKF e pode-se notar que em algumas configurações o aumento dos recursos de hardware apresenta pouco impacto no tempo de processamento da aplicação. A ferramenta proposta possibilita que um modelo em alto nível

especificado com MARTE, que é um perfil da UML voltado a modelar sistemas embarcados, possa ser sintetizado em hardware. Esse tipo de recurso contribui para o aumento da abstração do desenvolvimento de sistemas embarcados, sendo essa uma tendência das ferramentas atuais para atender às demandas do mercado atual, onde os sistemas estão se tornando cada vez mais complexos.

Como resultado deste trabalho, foram publicados 2 artigos científicos em congressos, com qualificação B1, são eles:

- Em De Medeiros et al. (2012b). Publicação: 7th IEEE International Symposium on Industrial Embedded Systems (SIES'12). Ver em referências bibliográfica para endereço eletrônico da publicação;

- Em De Medeiros et al. (2012a). Publicação: 38th Annual Conference of the IEEE Industrial Electronics Society (IECON 2012). Ver em referências bibliográfica para endereço eletrônico da publicação;

Referências Bibliográficas

ALEMFIN, J. L. F.; ALVAREZ, A. T. **Can Intuition Become Rigorous? Foundations for UML Model Verification Tools**. *11th International Symposium on Software Reliability Engineering, 2000. ISSRE 2000*, 2000.

ALTERA **Altera Website**. 2010a.
Disponível em <<http://www.altera.com>>

ALTERA ***Nios II Processor Reference Handbook***. Relatório Técnico, Altera Corporation, 2010b.

ALTERA ***SOPC Builder: User Guide***. Relatório Técnico, Altera Corporation, 2010.

ALTERA **Altera Products Selector**. 2011.
Disponível em <<http://www.altera.com/products/selector/psg-index.html>>

ALTERA ***ModelSim Tutorial: Software Version 10.0d***. Relatório Técnico, Altera Corporation, 2011.

ALTERA **Altera Corporate Social Responsibility**. site, 2012.
Disponível em <<http://www.altera.com/corporate/social-responsibility/csr-index.html>>

ALTERA ***Quartus II Design Software***. Relatório Técnico, Altera Corporation, 2012.

ALTERA **SoC FPGA Overview**. site, 2012.
Disponível em <<http://www.altera.com/devices/processor/soc-fpga/proc-soc-fpga.html>>

AULAGNIER, D.; KOUDRI, A.; LECOMTE, S.; SOULARD, P.; CHAMPEAU, J.; VIDAL, J.; PERROUIN, G.; LERAY, P. **SoC/SoPC development using MDD and MARTE profile**. L objet, 2009.

BAIDU **Xilinx FPGA 7Series PCIExpress**. Baidu Council paperback site, 2011.
Disponível em <<http://wenku.baidu.com/view/721f8343336c1eb91a375d68.html>>

BERGER, A. S. ***Embedded Systems Design: An Introduction to Processes, Tools, and Techniques***. San Francisco, CA, USA: CMP Books, 2002.

BERKLEY **Metropolis Design Guidelines**. 2010.
Disponível em <<http://embedded.eecs.berkeley.edu/metropolis/metamodel.html>>

BONATO, V.; MARQUES, E. **RoboArch: A component-based tool proposal for developing hardware architecture for mobile robots**. *IEEE International Symposium on Industrial Embedded Systems. SIES '09*, v. ., p. 249–252, 2009.

BRISOLARA, L.; BECKE, L.; CARRO, L.; WAGNER, F.; PEREIRA, C. E.; REIS, R. **Comparing High-level Modeling Approaches for Embedded System Design**. *IEEE*, 2005.

CARCAMO, A. C.; VIDALES, M. A. S.; AGUILAR, L. J. **MDA model transformation using UML collaborations**. *International Computer Engineering Conference. ICENCO 2004*, 2005.

COUSSY, P.; GAJSKI, D.; MEREDITH, M.; TAKACH, A. **An Introduction to High-Level Synthesis**. *Design Test of Computers, IEEE*, v. 26, n. 4, p. 8 –17, 2009.

COUSSY, P.; MORAWIEC, A. **High-Level Synthesis: from Algorithm to Digital Circuit**, p. 300. 2008.

DE MEDEIROS, R.; BONATO, V.; GOIS, M. M. **Designing FPGA-based embedded systems with MARTE: a PIM to PSM converter**. *IECON 2012*, 2012a.
Disponível em <<http://www.iecon2012.org/final_program/program-dsr.html>>

DE MEDEIROS, R.; BONATO, V.; GOIS, M. M.; ROSSI, D. L. **Designing Embedded Systems with MARTE: A PIM to PSM Converter**. *SIES 2012*, 2012b.
Disponível em <<http://sies2012.ira.uka.de/program-flyer.pdf>>

DEJAN, M. **Trend Wars ñ Embedded Systems**. *IEEE Concurrency*, 2000.

DEKEYSER, J.-L.; BOULET, P.; MARQUET, P.; MEFTALI, S. **ModelIEEE Transaction on Computers Driven Engineering for SoC co-design**. *IEEE*, 2005.

DENSMORE, D.; PASSERONE, R.; VINCENTELLI, A. S. **A platform-based taxonomy for ESL design**. *IEEE Journal on Design and Test of Computers*, v. 23, n. 5, 2006.

DUBOIS, H.; LAKHAL, F.; GÉRARD, S. **The Papyrus tool as an Eclipse UML2-modeling environment for requirements**. *200IEEE Transaction on Computers9 Second International Workshop on Managing Requirements Knowledge (MaRK'09)*, 2010.

DUBY, C. K. ***Accelerating Embedded Software Development with a Model Driven Architecture***. Relatório Técnico, Pathfinder Solutions, 2003.

ESPINOZA, H.; CANCILA, D.; SELIC, B.; GERARD, S. **Challenges in combining SysML and MARTE for model-based design of embedded systems**. *Springer-Verlag*, p. 98–113, 2009.

FIRST SILICON ***NiosII Hardware Reference***. Relatório Técnico, First Silicon Solutions, Inc., 2006.

FRIEDENTHAL, S.; MOORE, A.; STEINER, R. **OMG Systems Modeling Language (OMG SysML) Tutorial**. 2008a.

FRIEDENTHAL, S.; MOORE, A.; STEINER, R. **OMG Systems Modeling Language (OMG SysML) Tutorial**. 2008b.

GHOSAL, A.; GIUSTO, P.; SANGIOVANNI-VINCENTELLI, A. L.; D'AMBROSIO, J.; NUCKOLLS, E.; WILHELM, H.; TUNG, J.; KUHL, M.; VAN STAA, P. **Education panel: designing the always connected car of the future**, p. 617–618. 2010.

HANSEN, F. O. **SysML a modeling language for Systems Engineering**. 2010.

HEATON, L. **Catalog of UML Profile Specifications**. 2011.
Disponível em <<http://www.omg.org/technology/documents/profile_catalog.htm>>

HERRERA, F.; PENIL, P.; POSADAS, H.; VILLAR, E. **A Model-Driven Methodology for The Development of SystemC Executable Enviroments**. *Specification and Design Languages (FDL), 2012 Forum on*, 2012.

IBM **Model Driven Development with SysML**. 2009.

KALMAN, R. E. **A New Approach to Linear Filtering and Prediction Problems**. *Transactions of the Journal of Basic Engineering*, v. 82 (Series D), p. 35–45, 1960.

KAMARUDDIN, A.; DIX, A.; RAZAK, F. H. A. **Using Diary to Uncover Users Personal Information Management (PIM) Behaviours**. *IEEE*, 2011.

KANG, J.-Y.; GUPTA, S.; GAUDIOT, J.-L. **An Efficient Data-Distribution Mechanism in a Processor-In-Memory (PIM) Architecture Applied to Motion Estimation**. *Published by the IEEE Computer Society*, 2008.

KANG, Y. H.; DRAPER, J. **Design Trade-offs for instituctionLoad/Store Buffers in Embedded Processing Environments**. *IEEE*, 2007.

KHAN, M. U.; GEIHS, K.; GUTBRODT, F.; GÖHNER, P.; TRAUTER, R.; AG, D. **Model-Driven Development of Real-Time Systems with UML 2.0 and C**. *Proceedings of the Fourth Workshop on Model-Based Development of Computer-Based Systems and Third International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MBD/MOMPES06)*, 2006.

KONRAD, S.; CHENG, B. H.; CAMPBELL, L. A. **Object Analysis Patterns for Embedded Systems**. *IEEE Transactions on Software Engineering*, 2004.

KOPETZ, H. **The Complexity Challenge in Embedded System Design**. *IEEE*, 2008.

KOUDRI, A.; AULAGNIER, D.; VOJTISEK, D.; SOULARD, P. **Using MARTE in a Co-Design Methodology**. *MARTE Workshop DATE'08*, 2008.

KOUDRI, A.; CHAMPEAU, J.; AULAGNIER, D.; SOULARD, P. **MoPCoM MDD process applied to a cognitive radio system design and analysis**. 2009.

LIGGESMEYER, P.; TRAPP, M. **Trends in Embedded Software Engineering**. *IEEE Software*, 2009.

LIN, C.-H.; HSIEH, W.-T.; HSIEH, H.-C.; LIU, C.-N.; YEH, J.-C. **System-Level Design Exploration for 3-D Stacked Memory Architectures**. *ACM*, 2011.

LIU, S.; CHEN, P. Developing java ee applications based on utilizing design patterns. v. 2, p. 398 –401, 2009.

LU, S.; HALANG, W. A.; ZHANG, L. **A Component-based UML Profile to Model Embedded Real-Time Systems Designed by the MDA Approach**. *IEEE*, 2005.

MA, A.; ZACHARDA, A. **Utilizing SystemC for design and verification**. 2010.

MARTINS, C. A. P. DA S.; ORDONEZ, E. D. M. C. J. B. T. ***Computacao Reconfiguravel: conceitos, tendencias e aplicacoes***. Relatório Técnico, Programa de Pos-Graduacao em Engenharia Eletrica (PPGEE). Pontifica Universidade Catolica de Minas Gerais (PUC-Minas), 2002.

MILLER, J.; MUKERJI, J. ***MDA Guide Version 1.0.1***. Relatório Técnico, OMG, 2003.

NO MAGIC, INC **"Magic Draw UML"**. 2012.
Disponível em <<http://www.nomagic.com/>>

OBJECT MANAGEMENT GROUP, INC **"MARTE Related Tools"**. 2013.
Disponível em <<http://www.omg.org/omgmarte/Tools.htm>>

OBJECT MANAGEMENT GROUP (OMG) **"UML Notation guide version 1.3"**. 1999.
Disponível em <<http://www.rational.com/uml>>

OMG ***UML Profile for MARTE: Modeling and Analysis of Real-Time Embedded Systems***. Relatório Técnico, 2009.

OMG **OMG Model Driven Architecture**. 2011.
Disponível em <<http://www.omg.org/mda/>>

PAREDIS, C. J.; BERNARD, Y.; BURKHART, R. M.; DE KONING, H.-P.; FRIEDENTHAL, S.; FRITZSON, P.; ROUQUETTE, N. F.; SCHAMAI, W. **An Overview of the SysML-Modelica Transformation Specification**. *INCOSE*, 2010.

PHILLIPS, R. E.; THULLEN, H. J. **Embedded Computer System Integration Support**. *IEEE*, 1989.

PIMENTEL, A. D.; HERTZBERGER, L. O.; LIEVERSE, P.; VAN DER WOLF, P.; DEPRETTERE, E. F. **Exploring Embedded-Systems Architectures with Artemis**. *IEEE Transaction on Computers*, 2001.

PIRES, W.; RAMALHO, F.; LEDO, A.; SEREY, D. Checking uml design patterns in java implementations, p. 120 –129. 2010.

POWELL, D.; MARTINS, E.; ARLAT, J.; CROUZET, Y. **Estimators for Fault Tolerance Converage Evaluation**. *IEEE Transaction on Computers*, 1995.

RAGHUNATHAN, A.; RAVI, S.; MANGARD, S. **Tutorial T1: Designing Secure SoCs**. *IEEE*, 2007.

SANGIOVANNI-VINCENTELLI, A. **Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design**. *Proceedings of the IEEE*, v. 95, n. 3, p. 467–506, 2007.

SANGIOVANNI-VINCENTELLI, A.; SHUKLA, S. K.; SZTIPANOVITS, J.; YANG, G.; MATHAIKUTTY, D. A. **Metamodeling: An Emerging Representation Paradigm for System-Level Design**. *IEEE Design & Test of Computers*, p. 54–69, 2009.

SHUKLA, S. K.; EDWARDS, S. A. **High Level Modeling and Validation Methodologies for Embedded Systems: Bridging the Productivity Gap**. *IEEE*, 2003.

SHUKLA, S. K.; PIXLEY, C.; SMITH, G. **Guest Editors' Introduction: The True State of the Art of ESL Design**. *Design Test of Computers, IEEE*, v. 23, n. 5, p. 335 – 337, 2006.

SOLER, E.; TRUJILLO, J.; FERNÁNDEZ-MEDINA, E.; PIATTINI, M. **A set of QVT relations to transform PIM to PSM in the Design of Secure Data Warehouses**. *Second International Conference on Availability, Reliability and Security (ARES'07)*, 2007.

STOLPER, S. A. **Software that travels**, p. 1–3. 2002.

WANG, Z.; WANG, X.; CHATTOPADHYAY, A.; RAKOSI, Z. E. **ASIC Synthesis using Architecture Description Language**. *IEEE*, 2012.

XILINX ***Platform Studio User Guide: Embedded Development Kit EDK 6.2i***. Relatório Técnico, Xilinx Corporation, 2004.

XILINX ***Virtex-5 FP GA Product Table***. Relatório Técnico, Xilinx Corporation, 2009.

XILINX ***7 Series FPGAs Product Brief***. Relatório Técnico, Xilinx, Inc., San Jose, CA 95124, U.S.A., 2010a.

XILINX ***ARTIX-7 FPGAS***. Relatório Técnico, Xilinx Corp., 2010b.

XILINX ***ISE Design Suite 12: Installation, Licensing, and Installation, Release Notes***. Relatório Técnico, Xilinx Corporation, 2010c.

XILINX ***KINTEX-7 FPGAS***. Relatório Técnico, Xilinx Corp., 2010d.

XILINX ***MicroBlaze Processor Reference Guide Embedded Development Kit EDK 12.2***. Relatório Técnico, Xilinx Corporation, 2010e.

XILINX **All Programmable Technologies and Devices**. 2012. Disponível em <<http://www.xilinx.com/about/company-overview/index.htm>>

XILINX **Xilinx Platform Studio**. 2012. Disponível em <<http://www.xilinx.com/tools/xps.htm>>