Ferramenta protótipo para síntese de software de sistemas embarcados utilizando modelamento em UML e implementação em SystemC

Fábio Henrique Rodrigues Cerqueira[1]

[1]Ciências da Computação – Universidade Presidente Antônio Carlos (UNIPAC)
Barbacena – MG – Brasil

Abstract. *This work aims to generate a development framework for embedded platforms, that allows a single tool in the description of new devices, system modeling embedded applying [UML](Unified Modeling Language) on [SystemC] and portability of the same model between different architectures, facilitating the reuse of flows and comparative performance across architectures.*

Resumo. *Este trabalho tem como objetivo gerar um framework de desenvolvimento para plataformas embarcadas, que possibilite em uma única ferramenta a descrição de novos dispositivos, modelagem do sistema embarcado aplicando [UML](Unified Modeling Linguage) sobre [SystemC] e portabilidade de um mesmo modelo entre diferentes arquiteturas, facilitando o reaproveitamento de fluxos e comparativos de desempenho entre arquiteturas.*

1. Introdução

Sistemas embarcados são arquiteturas microprocessadas em que o processador é completamente encapsulado e dedicado ao dispositivo ou sistema que ele controla. Diferente de computadores de propósito geral, um sistema embarcado realiza um conjunto de tarefas pré-definidas, geralmente com requisitos específicos ligados a aplicabilidade de seu modelo e o contexto onde está inserido. Tipicamente, são implementados a partir de diferentes tecnologias, como microprocessadores, microcontroladores, DSP, circuitos reconfiguráveis, circuitos analógicos e de micro-ondas e sistemas micro eletromecânicos (MEMS - MicroElectroMechanical Systems), e oferecem melhor desempenho do que sistemas programáveis por serem dedicados à implementação da aplicação-alvo.

Tais sistemas, cada vez mais comuns , estão ficando complexos e poderosos graças aos avanços da tecnologia, fazendo com que estes sistemas possuam uma maior capacidade de processamento, memória e adaptação à diferentes necessidades, podendo ser reconfigurados de acordo com a aplicação à qual se destinam, evolução que encaminha o desenvolvimento de sistemas embutidos para o conceito de SOC-system-on-chip- (Sistemas que independem de comunicação com outros módulos). A complexidade na geração do software para estes sistemas tem levado ao estudo e desenvolvimento de técnicas sofisticadas de modelamento e simulação para resolver os problemas de prototipagem rápida e de concepção conjunta hardware/software relacionados à implementação da aplicação embarcada. Isso exige o desenvolvimento de metodologias de concepção que permitam a especificação, validação, otimização e implementação de sistemas heterogêneos utilizando, de preferência, um modelo unificado ao longo de todo o ciclo de concepção.

A abordagem predominante para o projeto de um sistema digital no nível de arquitetura é descrêve -lo através de uma linguagem de descrição de hardware. Tal descrição, além de servir como documentação, permite a simulação da arquitetura do sistema e a síntese automática do circuito lógico que implementa a arquitetura descrita. As linguagens de descrição de hardware mais populares atualmente são [VHDL] e [VERILOG]. Neste projeto adotamos a linguagem [SystemC] (linguagem de descrição de hardware e software baseada em C++ que inclui uma série de mecanismos para modelagem de hardware) pela sua fácil extensibilidade através do C++ como também compilação e simulação do projeto para arquiteturas especificas.

1.1. A ferramenta proposta

As ferramentas de concepção de sistemas embarcados existentes são ainda muito limitadas e exigem, por exemplo, que as especificações de hardware e software sejam feitas separadamente, como também em alguns casos ha obrigatoriedade de utilização de IDE's proprietárias ligadas aos componentes, descartando qualquer modularidade e reaproveitamento de projetos entre diferentes arquiteturas. Este trabalho propõe uma ferramenta que forneça a especificação independente da implementação e síntese automática de sistemas reativos em tempo real.

Guiando-se pelas experiencias bem sucedidas na utilização da linguagem de descrição de hardware [SystemC] em sistemas embarcados, como listado na seção 3, foi adotada a mesma no projeto, tendo em vista que a partir da especificação hardware/software o caminho para migrar a mesma para o dispositivo alvo é de fácil acesso por inúmeras aplicações já contempladas pela comunidade [SystemC], ou mesmo com a compilação de tal arquivo através de um compilador gcc, possuindo as bibliotecas da linguagem.

Baseado no conhecimento de tais facilidades de utilização da linguagem [SystemC], e visando aproveitar ao máximo seus benefícios e implementar uma solução multi-arquitetura, a abstração de software embarcado foi elevado a níveis de modelagem de diagramas, partindo da proposta de [BITENCOURTE et al. 2004], que desenvolveu uma metodologia de geração de projetos em [SystemC] utilizando diagramas da UML para modelar tanto o hardware do componente quanto as funcionalidades e o paralelismo do mesmo como descrito na seção 4. A ferramenta implementada batizada de SystemEMB tem por finalidade uma interface a nível de diagramas implementando tal metodologia , que permita uma ágil modelagem de sistemas em silício e uma tradução automática de tais modelos para a linguagem de descrição [SystemC].

1.2. Organização deste documento

Este documento foi organizado para um menlhor entendimento do solução proposta a partir dos fundamentos apresentados.Na seção 2 são abordados os conceitos básicos e as tecnologias envolvidas no projeto implementado.A seção 3 apresenta algumas soluções utilizando as tecnologias do mesmo escopo deste projeto. A seção 4 demonstra a metodologia proposta por [BITENCOURTE et al. 2004]. A seção 6 apresenta a ferramenta implementada.A seção 7 é a conclusão do projeto.

2. Tecnologias envolvidas

Nesta seção é abordado alguns conhecimentos básicos para o entendimento da ferramenta desenvolvida.

2.1. Linguagem de modelagem unificada [UML]

A [UML] surgiu em meados da década de 90 com intuito de padronizar e unificar os diversos métodos de modelagem existentes até então, cada qual com diagramas e nomenclaturas próprias o que dificultava muito a comunicação entre equipes de desenvolvimento ou mesmo empresas que não utilizassem um mesmo método. Tendo fundamento na experiência já conceituada na modelagem orientada a objetos de seus criadores: Grady Booch, Ivar Jacobson e James Rumbaugh teve seu primeiro esboço(chamado de versão 0.8 do Método Unificado) lançado em outubro de 1995[BOOCH et al. 2000], desde então vem sofrendo vários aperfeiçoamentos e se tornou um padrão para modelagem de software.

A Linguagem de Modelagem Unificada (UML) é uma linguagem padrão para visualização, especificação e documentação de sistemas adotada pela OMG(Object Management Group), que atualmente dispõe de doze diagramas assim distribuídos[OMG]:

- Diagramas estruturados: de classe, de objetos, de componentes e de implantação;
- Diagramas comportamentais: de casos de uso, de sequência, de atividades, de colaboração e de estados;
- Diagramas de gerenciamento: diagramas de pacotes, subsistemas e modelos.

Em geral não há obrigatoriedade de nenhum dos diagramas em especifico, o uso de um diagrama ou outro esta ligado inteiramente com as necessidades aplicadas ao projeto e a equipe que irá fazer uso destes diagramas.

Na implementação da ferramenta foram utilizados os diagramas de classe, caso de uso e sequência utilizando o escopo natural dos mesmos moldado a perspectiva decrita na seção 4 , assim possibilitando uma melhor abstração orientada a objeto para as aplicações embarcadas em sistemas de silício.

2.2. Linguagem de descrição de hardware HDL

Uma linguagem de descrição de hardware ou HDL é qualquer linguagem de uma classe de linguagens de programação usada para a descrição formal de circuitos eletrônicos , mais especificamente, a lógica digital e operacional . Possibilitando descrever o funcionamento do circuito, em sua concepção, organização e testa-lo para verificar o seu funcionamento por meio de simulação .HDLs são expressões textuais que padronizam as estruturas espaciais e temporais do comportamento dos sistemas eletrônicos.Como uma linguagem de programação concorrente ,a sintaxe e semântica da HDL inclui anotações explicitas para exprimir a simultaneidade .No entanto, em contraste com a maioria das linguagens de programação de software, HDL também inclui uma noção explícita de tempo, que é um atributo primário do hardware.HDLs são usados para escrever especificações executáveis de alguma peça de hardware.

As linguagens de descrição de hardware mais populares atualmente são [VHDL](Verê High Speed Integrated Circuit Hardware Description Languag), originalmente desenvolvida sob o comando do Departamento de Defesa dos EUA a fim de docu-

mentar o comportamento de ASICs que empresas fornecedoras embarcavão nos equipamentos, foi desenvolvido originalmente como uma alternativa à enormes complexos manuais com detalhes específicos de implementação, e [VERILOG] com sintaxe semelhante à linguagem de programação C , que já era amplamente utilizado no desenvolvimento de software de engenharia , já possuia diretivas como: case-sensitive , base de pré-processamento (embora menos sofisticado do que a de ANSI C / C + +), equivalente controle de fluxo de palavras-chave (if / else, for, while, caso, etc) e compatiblidade com a precedência do operador.

2.2.1. A Linguagem [SystemC]

[SystemC] é classificada como uma linguagem de descrição de hardware como VHDL e Verilog, mas sua descrição mais apropriada é como uma linguagem de descrição de sistemas, desde que exibe seu poder real durante a modelagem RTL (Register Transfer Level) e no modelamento comportamental. [SystemC] é um conjunto de rotinas e macros encapsulados em bibliotecas C++, que possibilitam a simulação de processos simultâneos, cada um descrito pela sintaxe C++. Uma vez instanciado na estrutura do [SystemC], os objetos descritos podem comunicar-se em um simulador de ambiente real-time, usando todos os sinais oferecidos pelo C++, inclusive alguns adicionais oferecidos pela sua biblioteca.Os comportamentos (processos) definidos podem ser instanciados qualquer número de vezes, e as provisões são feitas para que os processos possam ser definidos hierarquicamente de outros processos.A linguagem oferecida tem similaridades semânticas a VHDL e Verilog, mas pode-se dizer que possui uma sobrecarga sintática em relação a estes. Por outro lado, uma maior liberdade de expressividade é oferecido em troca, como objeto de particionamento orientado ao desenvolvimento de classes de modelo .

[SystemC] é uma linguagem descrição e um kernel de simulação. O código escrito irá compilar o kernel, juntamente com a biblioteca de simulação para gerar um executável que se comporta como o modelo descrito quando ele é executado.

Neste projeto foi adotado a utilização de [SystemC] pela fácil integração do mesmo a linguagem C++ e ao compilador gcc e pela baixa complexidade em se modelar um sistema de silício no mesmo.

2.3. Ferramenta multiplataforma para desenvolvimento em C++ [QT]

A ferramenta [QT] é um framework de desenvolvimento em C++ multiplataforma (MS WIndows, Mac OS X, Linux, e disposivos embarcados), de alto nível que permite o desenvolvimento desde a criação de interfaces através de mecanismos WYSWYG(What You See Is What You Get”), até implementações de baixo nível com alocação de recursos e reserva de espaços de memória. A representação de [QT] pertence ao grupo Nokia, o qual utiliza extensivamente sua tecnologia em seus dispositivos, no entanto opera de modo independente o que possibilita o trabalho da mesma com demais companhias externas ao espaço corporativo da Nokia.

[QT] é disponibilizado sobre três licenças:

- licença de desenvolvedor comercial: apropriada para o desenvolvimentos de softwares proprietarios e/ou comercial que não pretendam compartilhas suas aplicações

- licença [LGPL 2007] v.2.1: apropriada para desenvolvedores de aplicações (proprietarias ou Open Source) que estejam de acordo com os termos contidos em [LGPL 2007] v2.1
- licença [GPL 2007] v.3.0 : apropriada para desenvolvedores de aplicações que estejam de acordo com os termos da [GPL 2007] v.3.0.

A utilização do framework [QT] foi essencial no projeto por possibilitar a utilização de uma ferramenta de alto nível, acessível sobre licença [GPL 2007] e compatível com o sistema linux, plataforma escolhida para desenvolvimento do projeto pelo fácil interfaceamento entre o compilador gcc e a linguagem de descrição de hardware [SystemC] .

3. Estado da Arte

Esforços para integrar linguagens de descrição de hardware a ferramentas de apoio ao desenvolvimento de softwares para sistemas embarcados, tanto com foco no desenvolvimento do software embarcado, como para simulação e testes dos mesmos, tem aparecido cada vez mais em propostas acadêmicas.Soluções como :

- [HARTKE 2004] que utiliza o [SystemC] para descrição do hardware e geração do software através da especificação do projeto embarcado incorporando otimizações através de grafos de fluxos
- modelagens como [VASILEVSKI et al. 2007] que utiliza uma abordagem em [SystemC] para modelar e simular uma rede sensor wireless com dois nós emulando a rede real sem nem mesmo criá-la fisicamente
- [CANTANHEDENCE 2005] que aplica o mesmo [SystemC] para simulação de uma aplicação distribuída em um SoC para redes de sensores

"É necessário uma única linguagem comum e fundações de modelagem para criar um mercado de interoperabilidade de ferramentas de desenvolvimento, serviços e fazer IP (Intellectual Property) realidade."[HARTKE 2004]

4. Revisão bibliografica

Nesta seção sera abordado os fundamentos utilizados da pesquisa ciêntifica inicial, que derão origem ao projeto implementado.

4.1. A modelagem de alto nível em [SystemC]

Um dos conceitos reaproveitados no projeto foi a metodologia proposta por [BITENCOURTE et al. 2004], visando explorar as capacidades do [SystemC] em modelagem de alto nível, em uma concepção mais aproximada da resolução do problema de software todo o projeto de hardware/ software foi modelado usando diretivas da [UML], segundo os métodos listados.

4.1.1. Utilizando Caso de Uso para entendimento do contexto

"É necessario analisar o contexto no qual o processador está inserido e, para tanto, o diagrama de casos de uso é muito apropriado, uma vez que além de permitir a indentificação

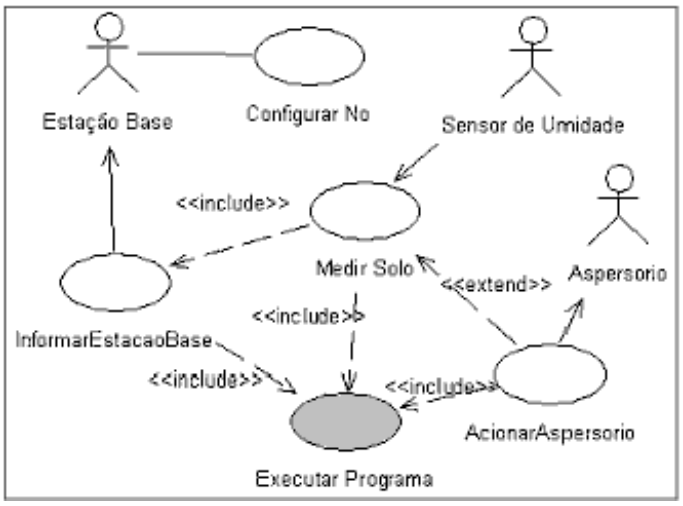

Figura 1. Casos de Uso do exemplo considerado[BITENCOURTE et al. 2004]

clara das funcionalidades desejadas, permite ainda identificar quem são os usuários de cada módulo."[BITENCOURTE et al. 2004]

Como meio de descrever o contexto do software embarcado, analisar principais processos executados pelo mesmo e abstrair os objetos externos que podem vir a interagir com o sistema, foi utilizado a abstração fornecida pelo diagrama de caso de uso.

No contexto do software embarcado cada nó existente no sistema traduz uma necessidade(recurso a ser implementado) que o sistema inclui, com exceção de um caso de uso especial,sempre presente no diagrama que representa o loop principal do sistema embarcado, nomeado como Executar Programa o qual deve por obrigatoriedade estar associado de alguma forma aos casos de uso que serão executados pelo sistema.

Os atores neste contexto são representações de objetos externos ao sistema modelado, como: outros sistemas embarcados, sinais analógicos, sinais digitais e qualquer outro hardware que possa ser interligado ao sistema embarcado. Frisando a necessidade de em alguma parte da modelagem existir um vínculo entre este ator especificado e uma porta ou canal físico do sistema.

4.1.2. Modelando o processador através de diagrama de classe

Nesta modelagem fora adotado a modelagem do processador em uma classe descrita como cpu que se relaciona por uma agregação com todos os demais módulos agregados do sistema embarcado, como: módulo de registradores, módulo de controle, módulo ULA e quaisquer outros módulos que comporem o sistema de silício. É uma relação de associação com os módulos interligados ao sistema que não estão emcapsulados na pastilha, como: módulos externos de mémoria, extensões de endereços etc. Um detalhe desta modelagem é a existência do método StartExecution() na classe CPU , na qual remete ao caso de uso Executar Programa e identifica loop principal do programa.

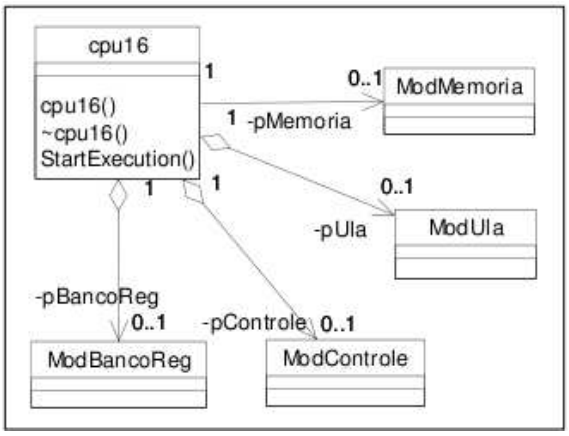

Figura 2. Hierarquia de classes do processador[BITENCOURTE et al. 2004]

4.1.3. Representando sensibilidades e paralelismo através de diagramas de classe

Ao se apronfudar nas descrições do modelo acima foi encontrado impecilios quanto a especificação do comportamento dos módulos modelados, e para passar tal barreira foi implantado um modelo computacional híbrido emulado em [SystemC] que aproxime a sintaxe e semântica da linguagem [UML] com o funcionamento real do sistema que esta sendo modelado.

No contexto do [SystemC], o modelo computacional mais abstrato previsto é o RTL(Register Transfer Level), que representa a conexão temporal e estrutural entre os componentes na forma de uma chamada de função, fazendo uma analogia para a chamada de métodos da orientação a objeto, pode-se-ia modelar o comportamento e ligação de um módulo funcional a outro como uma mera transação em que o módulo chamador solicitar ao módulo servidor. Dessa forma temos que:

"O sincronismo é feito por operações de envio e recebimento de mensagens em modo blocante. As mensagens transportam pacotes sem estrutura rígida, cuja forma é conhecida apenas pelo emissor e pelo receptor. É possível, dessa forma, representar o comportamento temporal do sistema, baseado exclusivamente na troca de tokens entre os módulos. Para que esse modelo funcionasse adequadamente em [SystemC] foi definido um conjunto de interfaces, canais e portas, que extendem os canais e interfaces primitivos, permitindo a correta implementação do modelo proposto."[BITENCOURTE et al. 2004]

4.1.4. Descrevendo operações de controle com diagramas de sequência

Ao se adentrar no detalhamento do módulo de controle, ouve a necessidade de um melhor refinamento na descrição da sequencia das operações processadas e no controle do acessos do modulo de controle aos demais módulos do sistema. Para abstrair tais especificações para o nosso módelo foi adotado o diagrama de sequência, frequêntimente utilizado na engenharia de software para o aprofundamento dos casos gerais.

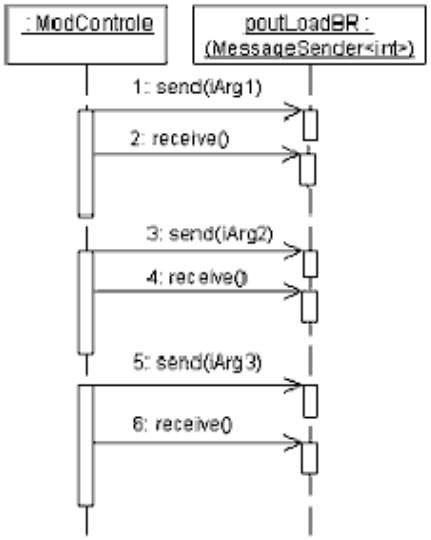

Figura 3. Diagrama de sequência da etapa Load[BITENCOURTE et al. 2004]

5. Metodologia

Inicialmente foi executado uma pesquisa exaustiva para que se pudesse ter um grande número de fontes que comprovassem o funcionamento das ferramentas que contemplariam o escopo da ferramenta proposta. Com base neste estudo foi confirmado inumeras situações em que o uso de [SystemC] para a modelagem e implementação de aplicações embarcadas em sistemas de silício resultarão em bons resultados comprobando assim a forte base em que o [SystemC] tem sua concepção. Alguns destes projetos foram listados na seção 3.

Para dar início a implementação da ferramenta fora necessario um estudo aprofundado das caracteristicas que envolvem a linguagem C++, tanto voltado para o escopo de programação de sistemas embarcados quanto na geração de aplicações desktop visto a falta de experiência do executor deste projeto na linguagem em questão. Durante este período também foi executado uma pesquisa forte nas minucias na ferramenta [QT] pois a mesma que seria o ambiemte de desenvolvimento final, em tais estudos foram consultados referências como:[EZUST and EZUST 2006] que aborda de maneira simples e direta os conceitos basicos de c++ em união com a ferramenta [QT],[BLACK and DONOVAM 2004] que demonstra todos as caractéristicas da linguagem [SystemC], abordando alguns exemplos e conceitos de modelagem de hardwares em [SystemC] e mais alguma ajuda de tutorias online do grupo [QT] e da comunidade [SystemC].

Como um teste da linguagem [SystemC] foi executado uma modelagem simples, conténdo algumas portas e um sistema de clock simples para obter a algum resultado da linguagem e comprovar a que a compilação das bibliotecas [SystemC] junto ao compilador gcc foi bem sucedida. Os resultados obtidos foram os esperados, os principios temporais e as sensibilidades de portas funcionaram de maneira coesa e condizente com a teoria estudada.

Dado início a implementação da ferramenta os esforços foram voltados para o desenvolvimento das interfaces de criação de diagramas para a modelagem [UML], o

qual ocupou muito tempo da implementação devido a alta complexidade de se desenvolver interfaces personalizadas de interação utilizando a linguagem c++.

Após o fechamento previo dos testes de interface da ferramenta, foi implementado as classes de persistênia que possibilitariam a exportação dos modelos criados pelo sistema e o reaproveitamento de código através do reuso de modelos já criados pela ferramenta. Como também fora implementados os mecanismos internos da ferramenta que possibilitariam tal reuso de modelo.

Tendo toda a faze de interface superada foi implementado as classes que fariam a tradução automatica dos modelos de [UML] em classes descritas em [SystemC] utilizando os métodos descritos por [BITENCOURTE et al. 2004], está também gerou alguns problemas durante a implementação pelo alto volume de iterações entre as classes conversoras e as classes de entidade que representavam a modelagem descrita.

6. A Ferramenta SystemEMB

Nesta seção é apresentada a ferramenta SystemEMB, desenvolvida para permitir uma tradução automática dos diagramas de UML em arquivos de compilação SystemC seguindo a metodologia proposta em [BITENCOURTE et al. 2004] abordado na seção 4, e seguindo os padrões de modelagem naturalmente estabelecidos na UML [BOOCH et al. 2000]. A ferramenta apresenta uma área de interação amigável e trata todos os componentes do sistema de forma individualizada, assim possibilitando o uso de diagramas de descrição em diferentes projetos, sobre a utilização de importação e exportação de arquivos compatíveis com o sistema.O sistema desenvolvido tem como principais áreas de modelagem, os módulos relacionados a seguir.

6.1. Modelagem de hardware

O modelo de hardware é uma dependência obrigatória para que um projeto seja considerado funcional , se caracteriza por representar o hardware que esta sendo modelado. No formato de um diagrama de classes como descrito na seção 4.1.2.O modelo prevê uma primeira classe obrigatória declarada como classe ”CPU, que representa a unidade de processamento central do dispositivo modelado. A classe CPU por sua vez poderá possuir associações com módulos externos do sistema, e possuir uma relação de composição para com os componentes internos da pastilha de silício, como banco de registradores, modulo de controle e módulos ULA. Sempre frisando a obrigatoriedade de a classe ”CPU”estar do lado do todo na agregação.

Os métodos contidos na classe CPU caracterialzão as operações de nível mais básicos desempenhados pelo sistema, inicialmente quando gerada a classe CPU, é automaticamente criado, os métodos construtores e um método especial ”StartExecution()”, que representa o loop infinito no qual o sistema embutido executa as suas funções.

Uma vez modelado e checado a consistência do modelo criado, o modelo é salvo sobre a extensão de ”.dclh”pode ser facilmente exportado para futuros usos em novos projetos, como também importado de um arquivo compatível anteriormente criado pela mesma ferramenta. Possuindo a única restrição de somente um arquivo classificado como Modelo de hardware poder ser carregado de cada vez.

6.2. Modelagem das sensibilidades do sistema

O modelo de sensibilidade embasado nos conceitos descritos na seção 4.1.3,tem como principal função descrever as portas/pinos que o dispositivo terá para comunicação externa, sendo discretizando cada pino como entrada ou saída ou ainda como canal(USART) para envio e recebimento de mensagens para outros sistemas. O diagrama de sensibilidades é formado por um diagrama de classe, sendo cada classe caracterizada como um meio de comunicação estas classes por sua vez tem a obrigatoriedade de serem especializações de interfaces inicialmente disponibilizadas pelo sistemas, estas que classificarão portas e canais para configurar cada pino do dispositivo alvo de acordo com a aplicação desejada.

6.3. Modelagem funcional

O modelo funcional já abstrai a ideia de funcionalidades do sistema como um todo, fazendo uso de diagramas de caso de uso como descrito na seção 4.1.1, tal modelagem demonstra as principais iterações entre o sistema modelado e os meios externos ao sistema. Este modelo possui dependência exclusiva da existência do um modelo de hardware devidamente validado no projeto, e mantém a ele um elo de aplicação quando instanciado para manter a consistência do sistema.

Como obrigatoriedade o modelo funcional possuí como principal caso de uso uma elipse denominada ”Executar Programa”que esta diretamente relacionada com método ”StartExecution()”default da classe CPU no diagrama de hardware. Este caso de uso representa o loop principal da aplicação e necessita estar conectado diretamente ou indiretamente a todas as outras funções requeridas pelo sistema, assim caracterizando uma associação de dependência de todas as demais funções do sistema para com o caso de uso ”Executar Programa”sobre o esteriótipo include , definindo que cada função será tratada em arquivos individuais e que todos os arquivos terão visão na classe principal ”CPU”e no método raiz ”StartExecution()”.

Outro componente importante neste diagrama é o ator, aqui representando algum componente externo ao sistema modelado que interagi com o mesmo, disparando algum evento ou recebendo algum dado específico, para cada ator existente no modelo funcional é necessário uma ligação com uma porta ou canal de comunicação do sistema modelado no diagrama de sensibilidades.

6.4. Tradução para [SystemC]

A tradução é executada após a descrição dos três diagramas obrigatórios, tendo confirmação da consistência dos mesmos, seguindo as diretrizes estabelecidas na metodologia proposta por [BITENCOURTE et al. 2004] é gerado as classes relativas a cada função estabelecida pelo módulo de hardware, tendo como métodos as chamadas executadas pelo módulo em questão, e todos estes, visíveis no escopo da classe ”CPU”. Os arquivos gerados formão o esqueleto do sistema modelado, mantendo compatibilidade com o padrão estabelecido pela comunicada Open Source [SystemC].

Após a tradução, os arquivos estarão disponíveis no diretório ”SystemC”do projeto atual, para que os mesmo possam ser utilizados em conjunto com demais ferramentas da comunidade [SystemC], para finalmente poder ser simulado e/ou gerado o arquivo binário para o dispositivo alvo.

6.5. Modelagem Sequencial

Para uma definição mais refinada do módulo de controle seguindo os conceitos presentes na seção 4.1.4.Descrevendo tanto o comportamento das execuções do sistema, quanto a ordem de execução dos métodos. Descreve-se a ordem na troca de mensagens entre as classes geradas nos demais diagramas, através de um diagrama de sequência. A utilização deste diagrama não é essencial no contexto do projeto, até porque a ferramenta proposta não especifica as operações internas dos métodos, mas tem sua utilidade quanto ao aproveitamento da engenharia de software na aplicação embarcada.

6.6. Fluxo de Projeto

Para dividir melhor o desenvolvimento do projeto embarcado foi proposto um fluxo que guie o desenvolvedor por todas as etapas necessarias. Sobre o contexto da ferramenta SystemEMB foram relacionados três etapas essenciais para criação da aplicação a ser embarcada, são elas:

- Especificação: Consiste na modelagem do Hardware a ser utilizado, esta etapa é composta pela modelage de hardware, onde são especificados todas as unidades internas e externas que compões a unidade de silício, tendo enfase na correta associação entre as classes que representem os componentes.Nesta etapa é importante checar a consistência do diagrama de hardware pois o mesmo será de extrema importância nas demais etapas do projeto.
- Projeto: Esta etapa basicamente se foca nas características da aplicação implantada, descrição das funcionalidades do sistema, estes modelados através do diagrama funcional (através de casos de uso), frisando que cada caso de uso modelado irá gerar o cabeçalho de uma função na tradução para [SystemC], dessa forma é necessario evitar ao maximo casos de uso redundantes que possam gerar uma implementação distribuida de funções dependentes.E o interfaceamento das portas de comunicação, este modelado pelo modulo de sensibilidades estritamente necessário para a linkagem dos atores no diagrama funcional e uma coesão entre as entradas e saída do sistema. Dentro destes modulos é importante frisar a necessidade de um controle sobre aa ambiguidade e a falta de referência de componentes do projeto, pois a compilação de um modelo não coeso gerará um projeto [SystemC] não executavel ou não confiavel.
- Codificação: Etapa de tradução dos modelos criados para a descrição em [SystemC] criando o esqueleto do sistema. Até então o mesmo é tratado como um recurso automatizado, mas levando em conta o nível de detalhamento que um projeto em [SystemC] necessita é relacionado para trabalhos que reaproveitem este projeto incluir um filtro maior sobre as opções de conversão que contemplem de maneira mais ampla todas as ferramentas que a linguagem [SystemC] permite.

6.7. Dependências relacionadas ao Projeto

Retomando alguns conceitos de hardware e do funcionamento de sistemas microperocessados, ha algumas dependências que o projeto deve controlar para que o aplicativo seja coeso e coerente com o meio fisico aonde será executado. Como tais dependências estão relacionadas as necessidades do nucleo(kernel) para seu perfeito funcionamento então as mesmas serão relacionadas no modelo de hardware, são elas:

- CPU: O primeiro requsito para um modelo coerente é a exsitencia de uma classe que represente a CPU(Unidade de processamento central), a mesma deve possuir associações com outras classes onde serão especificados a sua composição interna. Os metodos nela compostos estão relacionados as funções executadas pelo sistema embarcado, a principal delas é a função StartExecution que representa o loop infinito do programa do dispositivo demais funções representam as associações ao caso de uso Executar Programa no diagrama funcional.
- ULA: Um requisito essencial, e que pode ser incluido automaticamente junto a inclusão da cpu, é a classe que representa a unidade logica aritmetica da CPU um componente primitivo da computação que executa as operações do sistema, a mesma é mantida como uma classe externa a CPU para manter compatibilidade com futuras implementações que prevejão CPUs com estruras diferenciadas que permitam mais de uma ULA ou uma unidade ULA com operandos personalizados. Contudo a existência da mesma é obrigatória para a modelagem ser considerada consistente.
- Controle: A classe que abstrai o controle interno da CPU sobre a ordem de execução das funções do sistema. Esta classe é uma depêndencia para que o sistema permita a criação do diagrama de sequência que ordena a aplicação por ordem de execução das funções do sistema.
- Memória: A classe que demonstra o total de meória de execução e memória fisica que o dispositivo possui, inicialmente não fora utilizado como relação para qualquer outra modelagem mas, considera-se cabivel para futuras implementações a utilzação dos dados fornecidos por esta para calculo de espaço fisico no dispositivo para delimitar limites para a modelagem, ou em futuro modulo de edição do arquivo [SystemC], implementar checagem de espaço para o código implementado. Contudo a existência da mesma é obrigatória para a modelagem ser considerada conssitente.
- Registradores: A classe que descreve os pinos de saída do microcontrolador, apesar de seu identificador lembrar espaço em memória esta classe foi descrita no projeto como a modelagem dos pinos do sistema, assim possibilitando uma abstração melhor dos pinos a serem utilizados. Esta classe é uma depêndencia para que o sistema permita a criação do diagrama de sensibilidades que descreve as características de cada pino do componente.

6.8. Formatação das Classes

Visando melhorias na modelagem proposta por [BITENCOURTE et al. 2004] proponho uma melhor utilização do espaço de descrição de uma classe para utilização nos diagramas de hardware e de sensibilidade. O primeiro campo é caracterizado como o título, contendo o nome, idendificador único no sistema, seguido pela classificação da mesma. O segundo campo que seria em uma classificação comum em UML o campo de atributos, não se vê necessario neste contexto, pois a finalidade deste gerador é formar o esqueleto do arquivo final e não definir atributos de controle interno, prevendo uma melhor utilização deste espaço foi previsto nesta etapa a definição do endereço de uma porta do sistema embarcado, somente utilizando esta propriedade nos diagramas que referenciam uma porta diretamente, como as classes de registradores e a maioria das classes classificadas no diagrama de sensibilidades. O terceiro campo a ser descrito é o campo de métodos da classe, no qual é relacionado os metodos que cada classe irá executar.

Figura 4. Campos da represenação de uma classe

7. Conclusão

O uso de [SystemC] para modelagem de sistemas embarcados tem sua função muito bem desempenhada, e a união da modelagem [SystemC] com os conceitos de orientação a objeto da [UML] formam uma poderosa ferramenta no apoio ao desenvolvimento de software embarcado em sistemas de silício.

A ferramenta desenvolvida foi concebida até a etapa de geração das classe modulares de [SystemC] pelo determinado prazo de entrega.Embora o prazo de criação da ferramenta não foi suficiente para uma completa incorporação do poder das linguagens de hardware no projeto, este trabalho vem provar que a forte tendência de se utilizar linguagens de descrição de hardware para se criar aplicações embarcadas, vem se forteficando cada dia mais pelas facilidades de se moldar projetos modelados em [SystemC] a qualquer arquitetura de hardware desejada, contando com algumas facilidades até então pouco abordadas pelas ferramentas comerciais, como: a simulação previa do sistema antes de embarcado na pastilha de silício.

7.1. Trabahos Futuros

Como o tempo de projeto não foi o suficiente para uma implementação que contemplase todas as etapas na geração de uma aplicação embarcada, propõe-se a implementação de módulos que possibilitem editar de código, com checagem de sintaxe da linguagem [SystemC], que possa ser acoplado a ferramenta implementada assim possibilitando o detalhamento dos métodos na mesma, também é idealizado um módulo que conceda os recursos de simulação do [SystemC] para um completo ciclo de desenvolvimento em uma mesma ferramenta. Alguns recursos idealizados são a checagem da consistência entre os diagramas prevendo uma apurada conversão para [SystemC], e a importação e exportação depurada de diagramas como o diagrama de hardware e o diagrama funcional, assim possibilitando um melhor reaproveitamento de fluxos gerados pelo programa.

Referências

BITENCOURTE, J. J., de OLIVEIRA, J. C., and JACOBI, R. P. (2004). Utilizando uml para modelamento de sistemas em systemc. *X Workshop Iberchip*.

BLACK, D. C. and DONOVAM, J. (2004). *SystemC From The Ground Up*, volume 1. KLUWER ACADEMIC PUBLISHERS.

BOOCH, G., RUMBAUGH, J., and JACOBSON, I. (2000). *UML: Guia do Usuário*. Campos, 2 edition.

CANTANHEDENCE, R. S. (2005). Suporte a simulação distribuída em systemc. Master's thesis, Universidade de Brasília, Departamento da Ciências da Computação, Brasilia DF.

EZUST, A. and EZUST, P. (2006). *An Introduction to Design Patterns in C++ with Qt 4*, volume 1. Prentice Hall.

GPL (2007). Gnu general public license version 3. Disponível em `http://www.gnu.org/licenses/gpl.html` acessado em junho de 2010.

HARTKE, R. N. (2004). Uma ferramenta protótipo para síntese de software para sistemas embutidos a partir de systemc. Artigo de Graduação, Disponivel em `http://pet.inf.ufsc.br/node/75` acessado em abril 2010.

LGPL (2007). Gnu lesser general public license version 3. Disponível em `http://www.gnu.org/licenses/lgpl.html` acessado em junho de 2010.

OMG. Object management group. website `http://www.omg.org/` acessado em junho de 2010.

QT. Cross-platform aplication and ui framework. `http://qt.nokia.com/products/developer-tools/` acessado em Junho de 2010.

SystemC. Open systemc iniciative. website `http://www.systemc.org` acessado em 15 de abril de 2010.

UML. Unified modeling language. website `http://www.uml.org/` acessado em junho de 2010.

VASILEVSKI, M., ABOUSHADY, H., PÊCHEUX, F., and DE LAMARRE, L. (2007). Modeling wireless sensor network nodes using systemc-ams. *Microelectronics, 2007. ICM 2007. Internatonal Conference onx*, pages 53–56.

VERILOG. Verilog dot con. website `http://www.verilog.com` acessado em abril de 2010.

VHDL. Eda insdustry working groups. website `http://www.vhdl.org` acessado em abril de 2010.