Projeto de Sistemas Embutidos

Prof. José Raimundo de Oliveira
Profa. Wu Shin-Ting
DCA – FEEC – UNICAMP

03 de março de 2006

Um projeto de engenharia atravessa diversas etapas desde a sua concepção, passa por etapas de desenvolvimento, até a sua implementação física. Todas estas etapas formam o chamado **ciclo de vida do projeto**. O ciclo de vida de um projeto pode continuar além da sua implementação em suas novas versões ou nas manutenções do mesmo [8].

Cada etapa do desenvolvimento de um projeto pode ser entendida como sendo uma descrição mais detalhada da etapa anterior, até que se obtenha o próprio sistema que é, então, a concretização da idéia inicialmente concebida.

Durante a realização do projeto de um sistema, o projetista deve se preocupar, além da **aceitabilidade** do produto final, com o **tempo** e o **custo** do desenvolvimento de um projeto. Muitas vezes pode existir um compromisso entre estes dois fatores. Por exemplo, para reduzir o tempo de desenvolvimento de um projeto, eleva-se os gastos com a equipe e com recursos para o seu desenvolvimento. Devido à concorrência de mercado, o tempo de desenvolvimento, também chamado de **tempo para o mercado** *(time to market)*, é cada vez mais importante. Colaboram com a redução deste tempo o uso de ferramentas computacionais de apoio a projeto *(CAD Computer Aided Design, CAE Computer Aided Engineering, CAM Computer Aided Manufacturing)*.

O custo de um desenvolvimento é composto principalmente pela equipe envolvida no projeto e pelos recursos utilizados para a sua realização. O custo se eleva também se for considerado o tempo de desenvolvimento. Por isto, é muito importante que cada etapa seja muito bem estudada e avaliada antes de passar para a seguinte, pois quanto mais avançado no ciclo de vida de um projeto for detectado um erro de projeto, mais caro é o reparo e mais demorado irá ser atingida a implementação física. Um ciclo de vida de projeto que otimize os custos e minimize o tempo para o mercado é então determinante no sucesso, ou insucesso, de um novo projeto.

As equipes de desenvolvimento procuram definir uma metodologia de projeto própria que agregue um diferencial de seus concorrentes. Estas metodologias, em geral, baseiam-se nos recursos disponíveis, nas normas técnicas de projeto e nas experiências dos projetos anteriores. Uma atenção especial é dada a uma **rigorosa disciplina de documentação**. Como regra geral, pode-se dizer que quanto mais complexo o sistema a ser projetado, mais rigorosa deve ser esta disciplina em cada etapa de desenvolvimento de um projeto, a fim de facilitar a reavaliação das decisões tomadas anteirormente em cada etapa, e a posterior manutenção e atualização do produto final.

1 Introdução a Projetos Eletrônicos

Um sistema eletrônico pode ser bastante complexo e o seu projeto ter custo bem elevado. O detalhamento de todas as etapas possíveis do desenvolvimento deste tipo de projeto e dos custos envolvidos depende da tecnologia dos componentes a serem utilizados.

Mesmo para um mesmo tipo de tecnologia, não é possível generalizar o número de etapas de desenvolvimento de um projeto eletrônico, assim como, as tarefas nelas realizadas. Em geral, cada empresa define a disciplina de trabalho de sua equipe de engenharia. Formas diferentes de particionamento das tarefas de projeto podem definir o sucesso, ou não, dos produtos de uma empresa.

O que é possível fazer é classificar os tipos de atividades. Segundo Gajski-Kuhn [4], referenciado em [11], o desenvolvimento de um projeto eletrônico pode ser descrito por um modelo gráfico com três eixos de atividades, chamado "**Diagrama Y de Gajski-Kuhn**":

1. um eixo de atividades de descrição do comportamento do circuito, chamado **eixo comportamental**;

2. um eixo de atividades que descrevem a estrutura do circuito, ou seja, as partes ou componentes e as ligações entre eles, chamado **eixo estrutural**; e

3. um eixo de atividades que descrevem a forma física do circuito, chamado **eixo geométrico**.

Nos extremos externos destes eixos temos as descrições mais abstratas dos três tipos de atividades e na junção dos três eixos é representada a implementação final do circuito. Assim, na medida que o desenvolvimento avança, as atividades nestes três eixos ficam cada vez mais próximos, ou seja, têm maior interação. A figura 1 ilustra o diagrama de um projeto de circuito integrado.

No **eixo estrutural** encontram-se as tarefas que progressivamente descrevem a estrutura do circuito. Diagramas de blocos, descrição do circuito em termos do

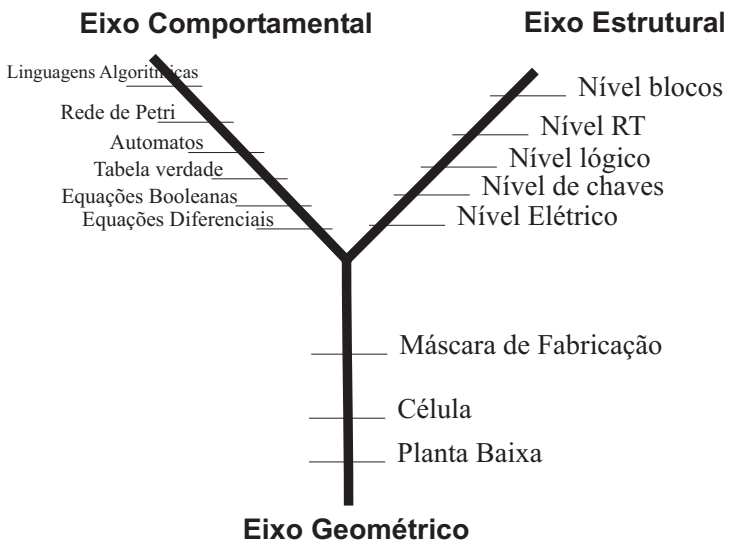

Figura 1: Diagrama Y para um projeto de circuito integrado

fluxo de dados entre os registradores numa base de tempo (*RT level* - Register Transfer Level), linguagem de descrição de circuitos *HDL - Hardware Description Language* e diagrama esquemáticos são exemplos de recursos que descrevem a estrutura de um circuito. Neste eixo destacamos os seguintes pontos notáveis, detalhando de uma concepção a uma implementação com uso de componentes elementares:

Nível de Sistema: interconexão de subsistemas mais complexos, processadores, controladores e periféricos.

Nível de Transferência entre Registradores (RT): registradores, somadores, memórias, etc.

Nível Lógico: interconexão de portas lógicas elementares, buffers, tri-state etc.

Niveis de Chaves (Gate-Level): transistores = chaves digitais (MOS).

Nível Elétrico: transistores, capacitores, resistores.

No **eixo comportamental** estão as tarefas que descrevem sucessivamente a funcionalidade do dispositivo projetado. Linguagem algorítmica, rede de Petri, diagramas de estado, e equações diferenciais são exemplos de recursos utilizados nas descrições comportamentais de um circuito. Destacamos neste eixos os seguintes recursos, na ordem que usualmente são empregados para uma descrição do nível de abstração maior para o menor:

Linguagens Algorítmicas: permitem a descrição tanto do fluxo de controle como das ações do bloco operacional associado nos diversos estados deste fluxo.

Rede de Petri: representa o fluxo de controle.

Autômatos: representam a lógica de controle em diversos níveis estruturais.

Tabela-Verdade: descrevem a função a nível lógico , tabelas de estado

Equações Booleanas: descrevem a função de blocos da lógica combinacional nos níveis estruturais lógico ou RT.

Equações Diferenciais: usadas para descrever comportamento a nível elétrico.

No **eixo geométrico** (muito dependente da tecnologia de implementação) estão as atividades que dão forma ao sistema projetado. Planta-baixa, *lay-out* e máscara de fabricação são exemplos de recursos que descrevem a configuração geométrica do circuito projetado. Na figura 1 destacamos sobre este eixo:

Planta Baixa: geometria dos diversos blocos e distribuição destes.

Células: geometria das células dos blocos (localização).

Máscaras de Fabricação: geometria exata em função da tecnologia utilizada.

Recomendações para um Projeto Complexo

As ferramentas baseadas em computador permitem desenvolver projetos de extrema complexidade. Mesmo como o uso de ferramentas adequadas é muito difícil entender um projeto complexo em sua totalidade mesmo para o seu próprio autor depois de um certo tempo. Isso pode conduzir a erros graves de interpretação que, em última instância, pode levar a erros na implementação. Diversas táticas têm sido adotadas para reduzir esta ambigüidade na interpretação e facilitar o entendimento do projeto. Alguns princípios comumente adotados pelos bons projetistas são:

Projeto é igual à documentação: Um projeto bem documentado reduz o custo da manutenção e da sua atualização. Além disso, devemos lembrar que grandes projetos envolvem grande equipe cujos integrantes podem mudar ao longo do tempo.

Uso de diferentes níveis de abstração: O uso de hierarquia e abstração no desenvolvimento do projeto ajuda a tratar sistemas complexos. A decomposição hierárquica é feita de forma recursiva, evitando-se os detalhes desnecessários em certos níveis de análise.

Descrição precisa: Descrições precisas, procurando utilizar notações padronizadas e modelos convencionais, asseguram maior legibilidade do projeto.

Descrição concisa: Descrições concisas, procurando usar uma linguagem objetiva e direta, aumentam a legibilidade do projeto o que poderia facilitar a sua compreensão.

Uso de métodos estruturados: Muito usado em programação de sistemas. A divisão de um projeto em pequenos sub-projetos é um exemplo de método estruturado. Componentes são decompostos em termos de fluxo de dados e de controle, cada qual desemepnha uma função específica na "transformação dos sinais". O paradigma de orientação a objetos é outro exemplo de método estruturado.

Modularidade: Divisão em pequenos módulos funcionais, de preferência em componentes básicos existentes, é uma prática comum hoje em dia na implementação de um sistema. Ela não só facilita o reuso das soluções existentes (por exemplo, uso de bibilotecas de componentes) como também facilita a manutenção e a reposição dos componentes de um circuito. No entanto, cuidados na integração devem ser tomados.

Realização dos testes por parte: O projeto e o teste andam em paralelo. A cada passo é feito um teste, procurando **dividir** um problema complexo em problemas menores **e conquistar** soluções mais simples.

Uso adequado de ferramentas e metodologia de projeto: O uso correto das ferramentas é fundamental para assegurar o sucesso de um projeto. Como já várias metodologias do projeto, cada equipe de projetistas deve escolher aquela que tenha maior familiaridade e que seja mais aproriada para o seu projeto. E em função da metodologia a ser utilizada, selecionar a ferramenta a ser utilizada.

2 Automação do Projeto Eletrônico – EDA (do inglês: *Electronic Design Automation*)

Automação de Projeto Eletrônico é ainda um conceito ambicioso. Idealmente seria um sistema automático de geração de projetos a partir de uma descrição abstrata/concepção/idealização de um circuito. No atual estágio, algoritmos de síntese só existem para pequenas etapas do problema de projeto, como por exemplo, para módulos integrados do tipo FPGA. Exemplos destes algoritmos são

os implementados no sistema Altera [1] e na ambiente de desenvolvimento Xilink [10].

Sistemas de EDA são, na verdade, coleções de ferramentas que envolvem mais a participação do projetista. Algumas destas ferramentas podem ser geradores automáticos de partes do objeto desejado. Outras são dirigidas explicitamente pelo projetista.

Um sistema de EDA consiste de 3 classes de ferramentas: Síntese, Análise e Gerência de Informação.

Síntese: ferramentas que assistem o projetista na criação do objeto.

Análise: ferramentas que assistem na verificação da correção do projeto.

Gerência de Informação: ferramentas que organizam a estrutura dos dados de projeto.

2.1 Ferramentas de Apoio à Síntese:

Captura de Projeto: As mais simples ferramentas de apoio à síntese são aquelas usadas para a captura de projeto. Estas ferramentas permitem ao projetista comunicar o seu projeto ao computador através de uma linguagem de descrição, podendo ser gráfica. Um exemplo são os programas de captura de esquemático para apoio a projetos digitais.

Geradores de Módulos: Mapeam a descrição funcional de um subsistema em descrições mais detalhadas do objeto a ser projetado.

Alocação e Roteamento: Os subsistemas do objeto em projeto são posicionados (alocados) no espaço (2D para projetos de circuitos ou 3D para projetos mecânicos). Cada subsistema tem interface para interconexão (rota de ligação) com outros subsistemas (trilhas em circuitos impressos, conduits entre gabinetes de um projeto civil). Existe uma forte interação entre a alocação dos componentes e o roteamento de suas interconexões, uma vez que uma alocação errada pode levar a um roteamento difícil (ou mesmo, impossível).

2.2 Ferramentas de Apoio à Análise

Um importante conjunto de ferramentas é aquele que auxilia na verificação e na validação do projeto.

Simuladores: São programas que modelam o comportamento do sistema em projeto. Vale ressaltar aqui que o modelo é sempre de alguma forma abstrato. Para viabilizar uma simulação por computador, é muito comum utilizar modelos matemáticos, muitas vezes com simplificações. Portanto, eles aproximam do real mas nunca o reproduzem fielmente. Os simuladores foram as primeiras ferramentas de EDA disponíveis.

Ferramentas de Análise Topológica: São responsáveis pela verificação da exatidão do *lay-out* e da interconexão dos subsistemas.

Analisadores de "*Timing*": Auxiliam na análise da compatibilidade temporal entre os componentes, assegurando que os atrasos inerentes em cada um sejam contemplados de forma a não comprometer os resultados desejados.

2.3 Ferramentas de Gerência de Informação

São responsáveis pela criação, manutenção e visualização de uma base de dados consistente da descrição de um projeto. Uma base de dados integrada é aquela pela qual dados de projetos podem ser compartilhados entre as diversas ferramentas de um ambiente de projeto [6].

Uma base de dados de projeto de engenharia tem uma estrutura particularmente rica e complexa. Ele precisa organizar os dados do projeto através de diversas representações do projeto, suas versões e implementações alternativas. Como exemplo, considere o projeto de um circuito integrado. O circuito é descrito ao mesmo tempo pela geometria das máscaras, pela interligação dos transistores, pela interligação dos *gates* (*netlist*), pela planta baixa (alocação física dos *chips*), descrição funcional-comportamental, descrição em diagrama de blocos, etc.

São ferramentas de gerência de informação de projeto:

Base de dados de projeto – Organiza a descrição do projeto dentro de cada representação, correlaciona descrições equivalentes através das representações e tenta manter essas correspondências ao longo da evolução do projeto.

Gerência de Configurações – Com o tempo, um subsistema de um projeto pode evoluir. Estas mudanças representam novas versões de parte do projeto. Reunindo uma escolha particular de versões de subsistemas produz-se uma configuração do sistema. Se várias configurações são mantidas ao mesmo tempo, estas são chamadas de alternativas.

Mecanismo de liberação – Para ajudar a manter um projeto consistente, as ferramentas de gerenciamento de informação são também responsáveis pelos

procedimentos de *Check-in* e *Check-out* (mecanismos de liberação = *Release Mechanisms*). Estes mecanismos garantem que uma nova versão de um subsistema não pode ser incorporado, até que seja comprovada a sua consistência através de ferramentas de análise.

3 Validação de Projeto

Testar um sistema é, de fato, um experimento prático no qual o sistema é executado sob distintas condições e as suas respostas são analisadas para verificar a corretude do seu comportamento. Se algum comportamento inesperado ou incorreto for detectado, é necessário passar para a segunda fase de teste que consiste a **diagnose**, ou a localização, da causa do problema.

Diversas são as técnicas de validação de projeto, dentre as quais destacamos a elaboração de um conjunto de testes que são realizados. A **testabilidade** de circuitos eletrônicos é motivo de diversos trabalhos de pesquisa e pode definir a ementa de um curso completo de pós-graduação. A definição do termo ”testabilidade” não é muito precisa, mas pode-se dizer que está relacionada a quão facilmente um circuito pode ser adequadamente testado. Como regra geral, tem-se que a testabilidade cresce quando o custo da geração do teste ou da aplicação do mesmo decresce.

Um **projeto visando testabilidade** (DFT – *Design for Testability*) é um projeto que integra nas técnicas de projeto algumas características de testabilidade, de forma a aumentar a sua testabilidade na fase da sua produção/manufatura. Isso visa a reduzir os custos de teste e também facilitar a diagnosabilidade. Estas técnicas são especialmente importantes em projetos de circuitos integrados, nos quais é impossível o acesso aos pontos internos do circuito [7].

3.1 Tipos de Testes

Paramétricos: avaliação de parametros físicos e elétricos;

Exaustivos: todas as combinações de estados que o circuito pode assumir são testados. Inviável para circuitos sequenciais;

Estrutural: procura-se identificar padrões de teste capazes de detectar falhas dentro do universo de padrões de testes possíveis (p.e. pontos preso num valor ’0’ ou ’1’ *stuck-at*). Exige conhecimento da estrutura interna do circuito.

Funcional: ignora a estrutura interna e identifica como um circuito bom aquele que executa as funções esperadas (caixa preta). **Teste funcional implícito**,

concorrente ou *on-line* ou chamado de *checking*, se refere aos testes durante a operação do sistema para detecção de erros., tais como técnicas de paridade e CRC.

Teste Explícito: executado fora do circuito. Inclui os teste nas pastilhas de circuito integrado, os testes de produção, testes de aceitação, manutenção e de reparos.

Técnicas *Ad-hoc*: técnicas de melhoramento da testabilidade dirigidas pelo projetista. Consistem, tipicamente, de uma lista de características do projeto que criam problemas de teste junto com sugestões de implementação.

3.2 Parâmetros de Testabilidade

Observabilidade: se refere a quão fácil o estado de um sinal interno pode ser determinado nos pontos de saída de um circuito.

Visibilidade: se refere ao grau em que os efeitos dos vários estados dos nós internos podem ser identificdos nos pinos de saída do circuito.

Controlabilidade: se refere a facilidade de se produzir um valor específico para um sinal interno aplicando-se sinais aos pinos de entrada.

4 Projeto de Sistemas Embutidos

Um **Sistema embutido**, do inglês *Embedded System* segundo Tech-Encyclopedia (2005) é, literalmente: ”um tipo de sistema computacional dedicado a um propósito específico que é usado dentro de um dispositivo. Por exemplo, um forno de microondas contém um sistema embutido que recebe comandos de um painel, controla o display, liga e desliga o elemento de aquecimento que cozinha a comida. Sistemas embutidos geralmente usam microcontroladores que implementam várias funções de um computador num único componente. Motorola e Intel fabricam os mais populares microcontroladores”. Por ser a aplicação de sistemas embutidos em veículos (automóveis, barcos e aeronaves) o seu maior mercado, diversos autores preferem chamá-los de **sistemas embarcados**. Para uma descrição mais detalhada sobre sistemas embutidos recomendamos consultar a referência [3].

O ciclo de vida de um projeto que envolve unidades processadoras pode ser dividido em duas principais linhas, uma relacionada com o circuito eletrônico (*hardware*) e outra relacionada com a programação (*software*). A interdependência entre estas linhas é muito forte e técnicas de co-projeto (*co-design*) podem ser aplicadas para o desenvolvimento integrado delas.

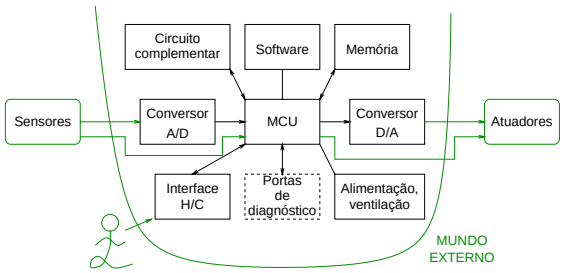

Figura 2: Uma organização típica de um sistema embutido

Além de uma unidade microcontroladora (MCU), uma memória e acessórios complementares como base de tempo, alimentação e refrigeração, um **sistema embutido**, tipicamente, contém:

- interfaces com o mundo externo, através de atuadores e sensores, ou com os operadores humanos (interface H/C). Estas interfaces são dependentes das aplicações;
- *software* que coordena as funções que a unidade microcontroladora deve desempenhar para assegurar corretas interações com o mundo externo;
- circuitaria dedicada (*hardware*) para complementar a estrutura do sistema tanto para execução do programa como para interfaceamento com o mundo externo. Esta circuitaria pode ser uma FPGA dedicada (*field programmable gate array*), ASIC (*application specific integrated circuit*), portas lógicas ou componentes eletrônicos discretos (transistores, resistores, capacitores, etc).
- opcionalmente, porta de acessos para diagnosticar problemas do sistema.

Figura 2 ilustra a organização típica de um sistema embutido.

4.1 Uma Metodologia para Projetos de Sistemas Embutidos

Diversos trabalhos de pesquisa e desenvolvimento são realizados visando a criação, a análise, a definição de metodologias e de ferramentas para projeto de sistemas embutidos. Uma simples busca na Internet por "*embedded system*" lista uma enorme relação de páginas acadêmicas e comerciais oferecendo recursos e apresentando exemplos de projetos nesta área. Listamos alguns exemplos em nossa web-grafia.

No contexto do nosso curso, propomos utilizar o modelo de metodologia de projeto apresentado na Figura 3.

Nas seções seguintes descrevemos as principais etapas deste modelo.

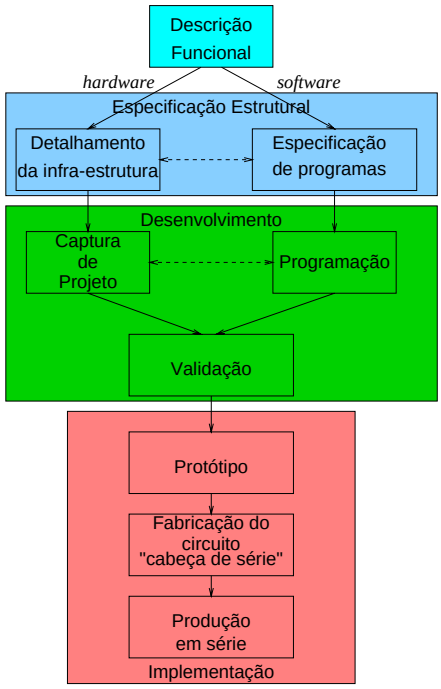

Figura 3: Um modelo de desenvolvimento

4.2 Descrição Funcional

Diferentemente do projeto de sistemas computationais tradicionais, como *mainframe*, *desktops* ou até *laptop*, que consiste tipicamente no aperfeiçoamento do seu desempenho (tipo de CPU), no aumento da sua robustez (sistemas redundantes) e da sua capacidade de armazenamento (quantidade de memória), o projeto de um sistema embutido deve ser essencialmente orientado às tarefas que ele realizará. No projeto, deve-se levar em conta as interações que o sistema deve possuir com o mundo externo e os algoritmos que viabilizem estas interações para satisfazer as funcionalidades descritas. A unidade microcontroladora e a memória constituem meros mecanismos para concretizá-las.

Sob esta óptica, é importante definir em primeiro lugar os tipos de interações que o sistema terá com o mundo externo. A seguir, o tipo de monitoramento e/ou controle o sistema deverá prover, para garantir uma resposta correta. Só depois, com base nesta análise, será avaliada a estrutura necessária para implementar o sistema desejado e integrá-lo no mundo real. Por exemplo, um sistema embutido de acionamento de motores requer usualmente interfaces constituídas pelos componentes eletrônicos (não-digitais) para converter sinais do nível digital para sinais analógicos com potência suficiente.

Nesta seção apresentamos os principais itens que devem constar na descrição

funcional de um sistema embutido.

4.2.1 Descrição do Comportamento

Com base na análise das funcionalidades e das características desejadas para o sistema em projeto, deve-se descrever:

funcionalidades: ou seja, detalhar todas as tarefas que o sistema será capaz de executar.

configurabilidade: isto é, todas as possíveis configurações do circuito, e todos os pontos de alteração da configuração.

eventos: que o sistema deve tratar. Estes eventos são classificados em eventos periódicos e não-periódicos. Caso forem eventos periódicos, deve-se especificar ainda a periodicidade de ocorrência destes eventos e se forem não-periódicos deve-se especificar o tempo mínimo entre dois eventos sucessivos.

tratamento de eventos: ou seja o comportamento que o sistema deverá ter para tratar corretamente **cada** evento.

4.2.2 Descrição Estrutural do Sistema

Junto com a descrição do comportamento do sistema, deve-se especificar, em nível de bloco ou sistema, a estrutura necessária para captar os eventos do mundo externo, para tratá-los, e para atuar sobre o mundo externo, ou seja, deve-se destacar

blocos funcionais: que compõem o sistema, incluindo uma síntese das funcionalidades de cada bloco; e

relacionamentos entre estes blocos: ou seja, incluir os principais sinais de comunicação entre os blocos de forma a assegurar a execução de todas as tarefas que o sistema deve realizar.

O fluxo dos sinais pode ser representado graficamente por um **diagrama de blocos**, como a representação mostrada na Figura 2. Um diagrama de blocos contém duas primitivas gráficas, devidamente identificadas:

1. caixas retangulares para representar os blocos funcionais
2. setas entre as caixas para indicar o(s) sentido(s) dos fluxos dos sinais disparados pelos eventos.

4.2.3 Descrição da Infra-estrutura para o projeto

O sucesso de um projeto depende da factibilidade da sua concretização em distintas etapas. Portanto, é importante ter sempre claro, desde o início do seu desenvolvimento, as restrições para o seu desenvolvimento e os recursos disponíveis para o mesmo.

4.3 Especificação Estrutural

Nesta etapa, com base na análise dos recursos disponíveis e das restrições do projeto, são detalhadas as características de *hardware* e *software* que serão desenvolvidos. Em comparação com os sistemas computacionais tradicionais, as limitações impostas aos sistemas embutidos são muito maiores com, porém, menos soluções alternativas. Os sistemas embutidos serão instalados dentro de um sistema maior, o que pode impor fortes restrições sobre a sua geometria e o seu peso. Outro fator que se deve levar em consideração é a diversidade das condições ambientais sob as quais um sistema embutido pode operar. Limites de dissipação térmica, de interferência eletromagnética e de compatibilidade eletromagnética dos ambientes típicos de operação devem ser contemplados rigorosamente para evitar funcionamentos intermitentes ou danos do equipamento. Outros dois aspectos importantes que poderão assegurar a competitividade do produto no mercado são a robustez e o custo final.

Nas subseções a seguir veremos o detalhamento dos algoritmos de processamento de eventos e o detalhamento da estrutura do projeto a partir de uma descrição funcional. Deve-se incluir ainda a especificação do procedimento de testes a ser conduzido para verificar a correção de cada módulo e do sistema como todo.

4.3.1 Especificação de Algoritmos

No nosso projeto consideraremos que todos os eventos serão tratados por *software*. Deve ser, então, elaborado para **cada** evento um algoritmo para o seu tratamento. Com base no tamanho de cada algoritmo, estima-se o tamanho de memória necessária para armazenar todos os programas e os dados associados. Isso permitirá especificar a memória a ser utilizada e o espaço onde serão armazenados os programas.

O algoritmo de tratamento de evento pode ser representado graficamente por um **fluxograma**. Recomenda-se usar símbolos gráficos consistentes com a norma internacional ISO 1028-1973 e IS0 2972-1979 . Alguns exemplos de símbolos gráficos mais usuais num fluxograma são:

caixas retangulares: para representar uma ação, podendo ser uma função ou uma instrução.

losango: para representar uma decisão binária.

caixa elíptica achatada: para representar o fim do procedimento.

círculo rotulado: para estabelecer a conexão com o ponto do fluxograma rotulado com o mesmo rótulo.

setas: para indicar direção do fluxo de dados.

Acompanhado a cada algoritmo deve ser incluído ainda um procedimento de testes para verificar a sua corretude.

4.3.2 Detalhamento Estrutural

Entende-se por detalhamento estrutural a descrição tanto das caracterísitcas elétricas e temporais como das restrições físicas de cada bloco funcional mencionado na Seção 4.2.2.

Como o projeto de um sistema embutido é centralizado nas tarefas, recomenda-se iniciar com a **definição dos periféricos de entrada e saída** (atuadores e/ou sensores) apropriados para o sistema e definir um endereço distinto para cada um deles. Este endereço será utilizado pela unidade microcontroladora para acessá-los tanto para leitura como para escrita.

Tendo definidos os periféricos e a memória, é possível projetar um **decodificador de endereços** que converte o endereço referenciado no programa em sinal *Chip Select – CS* do dispositivo correspondente, habilitando-o para realizar um ciclo de leitura ou de escrita.

A seguir, deve-se projetar um **circuito de interface** para **cada** periférico com base na análise de

níveis de sinais elétricos requeridos pelo periférico. Quando o nível é distinto do nível digital, conversores A/D (analógico para digital) ou D/A (digital para analógico) podem ser necessários.

diagrama de tempo do ciclo de leitura e/ou do ciclo de escrita do periférico para avaliar a compatibilidade com o diagrama de tempo do ciclo de leitura e do ciclo de escrita da unidade microcontroladora. Circuitos de espera podem ser necessários para sincronizar os sinais. Vale comentar aqui que o alinhamento é sempre feito com base no sinal mais lento.

padrão de comunicação do periférico e avaliar a sua compatibilidade com o padrão suportado pela unidade microcontroladora. Circuitos adicionais, como adaptadores para interfaces de comunicação, podem ser necessários.

Deve-se especificar ainda as restrições físicas e ambientais, como limites mecânicos (altura, largura, profundidade), limites de dissipação térmica, e limites de interferência eletromagnética e de compatibilidade eletromagnética.

Por último, deve-se especificar o procedimento de testes a ser realizados, durante o desenvolvimento, para cada item definido no detalhamento estrutural.

4.4 Desenvolvimento

Esta etapa é caracterizada pelo desenvolvimento tanto do circuito projetado como dos programas elaborados. Hoje em dia, há muitas ferramentas computacionais de apoio à análise dos circuitos antes da sua implementação. Para isso, o projeto precisa ser **capturado** pelo computador. Quanto ao *software*, há também distintos recursos de apoio ao desenvolvimento, desde a geração de códigos de máquina até testes.

4.4.1 Captura de Projeto

Esta "captura" consiste na elaboração do desenho esquemático do circuito ou na descrição dos componentes do circuito, da listagem de ligações (*netlist*) e de materiais (*bill of material*), que serviriam como entrada a um ambiente de apoio à síntese e análise do projeto por computador.

O **diagrama esquemático** é uma representação visual de um circuito eletrônico. Este desenho deve indicar com detalhe as ligações entre os pinos dos componentes. Ele consiste essencialmente de seguintes elementos:

1. **símbolos esquemáticos** dos componentes eletrônicos, usualmente providos de mais de um terminal ou ponto de conexão. A cada ponto de conexão é atribuído um número que corresponde ao número de pinagem do componente. Recomenda-se utilizar o padrão de símbolos de circuitos sugerido pelo IEC-IEEE [9]. Todos os elementos devem ser devidamente identificados. Recomenda-se utilizar nomes que lembrem o significados de cada elemento, composta de até duas letras que lembram o seu tipo seguida de um número único para cada tipo de componente, As letras que lembram os tipos de componentes podem ser como a seguir:

    - Circuito integrado = U
    - Capacitor = C

- Resistor = R
- Transistor = T
- Transformador = TR
- Diodo = D
- Cristal = X

Para os elementos passivos, como capacitores, resistores e indutores, deve-se ainda indicar os seus valores. As unidades utilizadas para a indicação de valores são:

- Resistor = Ohms e W
- Capacitor = mF e VDC
- Transistor = Nome
- Diodo = nome
- Cristal = Hz

2. **linhas** que conectam os terminais dos componentes eletrônicos, podendo haver sobreposições.
3. **junções** são os pontos que mostram a conectividade entre as linhas e/ou terminais que se intersectam.
4. **rótulos de conexão** para indicar ligações entre os terminais dos componentes que não aparecem explicitamente no diagrama.

Exemplo 1 *Ver a Figura 6.1 e o esquemático da evolution board no Capítulo 6 da referência [2].*

Deve-se ainda elaborar uma **lista de ligações** entre os componentes do circuito. Esta lista é útil para a etapa de montagem do circuito, sendo normalmente organizada em ordem dos sinais. Cada sinal é descrito em termos dos pinos dos componentes que estão ligados a ele.

Exemplo 2 *A seguinte lista descreve as ligações de dois sinais, SMEMR e SMEMW. O sinal SMEMR está ligado ao pino 1 do circuito U09, ao pino 25 do circuito U02 e assim sucessivamente. Já o sinal SMEMW está ligado ao pino 3 do U09, ao pino 2 do U02 e ao pino 27 do U50.*

Sinal	*Descrição*
SMEMR	*U09.1, U02.25, U03.9, U04.3, U08.7, U15.8*
SMEMW	*U09.3, U02.2, U50.27*

Uma **lista de materiais** contendo todos os componentes necessários para a implementação do projeto do sistema embutido deve ser confeccionado. No mínimo duas informações são fornecidas para cada componente: (1) sua identificação no diagrama esquemático; e (2) sua descrição.

Exemplo 3 *A seguinte lista de materiais contém 4 componentes:*

Referência	*Descrição*
C1, C6, C12	*Capacitor, 10pF @ 50Vdc, ± 20%*
U09	*C.I., TTL74LS04, 6 inversores*

Exemplo 4 *Ver tabela 6-5 da referência [2].*

4.4.2 Programação

As instruções que uma unidade microcontroladora processam são, de fato, *firmware* – códigos de operação da máquina. O mnemônico destes códigos é tipicamente composto de 3 letras. Para amenizar a tarefa árdua de escrever os programas em mnemônicos da máquina, existe uma diversidade de ferramentas de suporte que possibilitam a confecção dos programas em linguagem de alto nível. Adicionalmente, os desenvolvedores podem usufruir os ambientes sofisticados de depuração de programas de alto nível que se dispõem no mercado para depurar e testar o seu programa.

Exemplo 5 *Para os microcontroladores MC68HC11 existe o compilador ICC11 [5] que traduz programas desenvolvidos em linguagem C para os mnemônicos do seu repertório de instruções.*

4.4.3 Validação

Nesta fase é validado o projeto através das simulações, emulações e testes, utilizando os componentes mais próximos possíveis do produto final. Para esta etapa de validação são disponíveis no mercado uma série de *kits* de validação para distintas famílias de microcontroladores.

Exemplo 6 *Para o desenvolvimento de um projeto com uso dos microcontroladores MC68HC11 é disponível o ambiente de desenvolvimento EVB (Evolution Board) [2]. Este ambiente contém um conjunto mínimo de instruções denominado BUFFALO (Bit User Fast Friendly Aid to Logical Operations) que permite testar e diagnosticar um projeto a baixo custo.*

4.5 Prototipação

Nesta fase é desenvolvido um protótipo do produto.

5 WEB-Grafia

Apresentamos nesta seção alguns endereços da *web* de interesse para os tópicos abordados.

http://www.iso.org – site oficial da *International Standard Organization*, onde é possível adquirir diversas normas – acessado em agosto de 2005

http://www.tech-encyclopedia.com – site com glossário tecnológico – acessado em agosto de 2005

EDA

http://www.eda.org/ – Página com diversas informações sobre EDA – acessado em agosto de 2005

http://www.edacafe.com/ – Página com diversas informações sobre EDA – acessado em agosto de 2005

http://www.expresspcb.com/ – Ferramentas de CAD grátis – acessado em agosto de 2005

http://www.mentor.com/ – Fabricante de ferramentas de EDA – acessado em agosto de 2005

Sistemas embutidos (*embedded systems*)

http://www.microcontroller.com/ – Importante site sobre microcontroladores – acessado em agosto de 2005

http://www.msebilbao.com/tienda/default.php – Site sobre aplicação de microcontroladores em espanhol. – acessado em agosto de 2005

http://www.embedded.com/europe – site de revista europeia especializada em sistemas embutidos. – acessado em agosto de 2005

http://www.microcontrolador.com.br – Site brasileiro sobre aplicação de microcontroladores – acessado em agosto de 2005

http://www.esconline.com/ – Divulga os eventos acadêmicos (congressos, etc) relacionados com sistemas embutidos. – acessado em agosto de 2005

http://msdn.microsoft.com/embedded/ – site da MicroSoft com informações para desenvolvedores de sistemas embutidos usando o Windows CE. – acessado em agosto de 2005

http://www.atmel.com/ – site de fabricante de microcontroladores – acessado em agosto de 2005

http://www.intel.com/design/mcs51/ – site da Intel sobre o popular 8051 – acessado em agosto de 2005

http://www.microchip.com/ – site da Microchip, fabricante dos microcontroladores PIC – acessado em maio de 2005

HC11

http://www.mcu.motsps.com/ – Site oficial da Motorola – acessado em agosto de 2005

http://www.gmvhdl.com/hc11core.html – Descrição em VHDL de um core HC11. – acessado em agosto de 2005

http://www.msoe.edu/eecs/ce/ceb/resources/ – conjunto de ferramentas para desenvolvimento com o HC11. – acessado em agosto de 2005

http://www.realitydiluted.com/projects/hc11/ – Site pessoal com várias dicas e exemplos de projetos com o HC11. – acessado em agosto de 2005

Referências

[1] – , ALTERA, `http://www.altera.com/`

[2] -. M68HC11EVB Evaluation Board User's Manual. Motorola.

[3] Hennessy, John L.e David A. Patterson. Computer Architecture - A Quantitative Approach 3rd Edition, Morgan Kaufmann, 2003.

[4] Gajski, D e Kuhn, R. Guest's editors introduction, Computer, pag. 11-14, Dezembro 1983.

[5] Bernardes, Murillo F. Programação em linguagem C/EA870 – FEEC – Unicamp. `ftp://ftp.dca.fee.unicamp.br/pub/docs/ea079/manuais/tutorial.pdf`.

[6] Katz, R.H. Information Management for Engineering Design, SPRINGER-VERLAG, 1985.

[7] McCluskey, E.J. Logic Design Principles – with Emphasis on Testable Semicustom Circuits, Prentice Hall, 1986.

[8] Oliveira, J.R. Notas de Aulas – Curso EA060: Projeto e Análise de Circutos Digitais, FEEC-UNICAMP, 1993.

[9] –, 1981 Suplemet to the TTL Data Book for Design Engineers, Texas Instruments Incorported Semiconductor Group, 1988.

[10] – , Xilink: The Programmable Logic Company, `http://www.xilink.com/`

[11] Wagner, F.R., I.Jansch-Pôrto, R.F.Weber e T.S.Weber. Métodos de Validação de Sistema Digitais, VI Escola de Computação, Campinas, SP, 1988.