3 Metodologia de Projeto de Sistema Embarcado

Autores: José Raimundo de Oliveira e Wu Shin-Ting
DCA - FEEC - Unicamp
Agosto de 2019

Sumário

3.1 Introdução

O projeto de sistemas embarcados é envolvido por diversas questões que vão desde aspectos de mercados até as ações tecnológicas de implementação do produto. Este tópico inicia com uma breve introdução aos conceitos relacionados ao ciclo de vida de um produto desde sua concepção até a sua extinção. Depois são apresentados os conceitos relativos ao projeto de sistema eletrônico, com ênfase às ferramentas de apoio a projetos eletrônicos. Aspectos específicos de projetos de sistemas embarcados são apresentados em seguida. Por último, é proposto uma metodologia de projeto a ser usada nesta disciplina. Ao final, além da bibliografia referenciada, são listados um conjunto de páginas da *internet* (Webliografia) e um conjunto de *twitters* que podem ser seguidos, relacionados ao presente tópico e ao conteúdo da disciplina.

3.2 O Ciclo de Vida de Projeto

Um projeto de engenharia atravessa diversas etapas desde a sua concepção, passa por etapas de desenvolvimento, sua implementação física, sua entrada no mercado, sua participação no mercado até o seu desaparecimento ou extinção. Todas estas etapas formam o chamado **Ciclo de Vida do Projeto**, ou **Ciclo de Vida do Produto**. Ações de engenharia duram todo o ciclo de vida de um projeto, pode continuar além da sua implementação em novas versões do implementado, suas atualizações ou nas manutenções dele.

A Figura 3.1 mostra a relação financeira no tempo do ciclo de vida de um produto inovador. Este ciclo inicia com o surgimento da oportunidade, a ideia de inovar. Inicia-se então o chamado Ciclo de Inovação, que passa pela **Percepção do bom negócio**, pela etapa de **Investigação**, pelo **Congelamento do Conceito**, pelo **Lançamento do produto** até os **Primeiros Clientes Satisfeitos**. O Ciclo de Inovação é caracterizado pelo desembolso de investimentos. Após o Lançamento e a obtenção dos Primeiros Clientes, a entrada de receitas é usada para cobrir as despesas e, principalmente, os investimentos efetuados. Quando os custos dos investimentos forem totalmente cobertos, é atingido o **Ponto de Equilíbrio Investimento/Retorno** (*BEP – Break Even Point*). A partir daí, a receita obtida com o produto é dividida em despesas operacionais e lucro. Isto dura até o produto atingir a **Extinção**, quando o produto se torna obsoleto e sai do mercado.

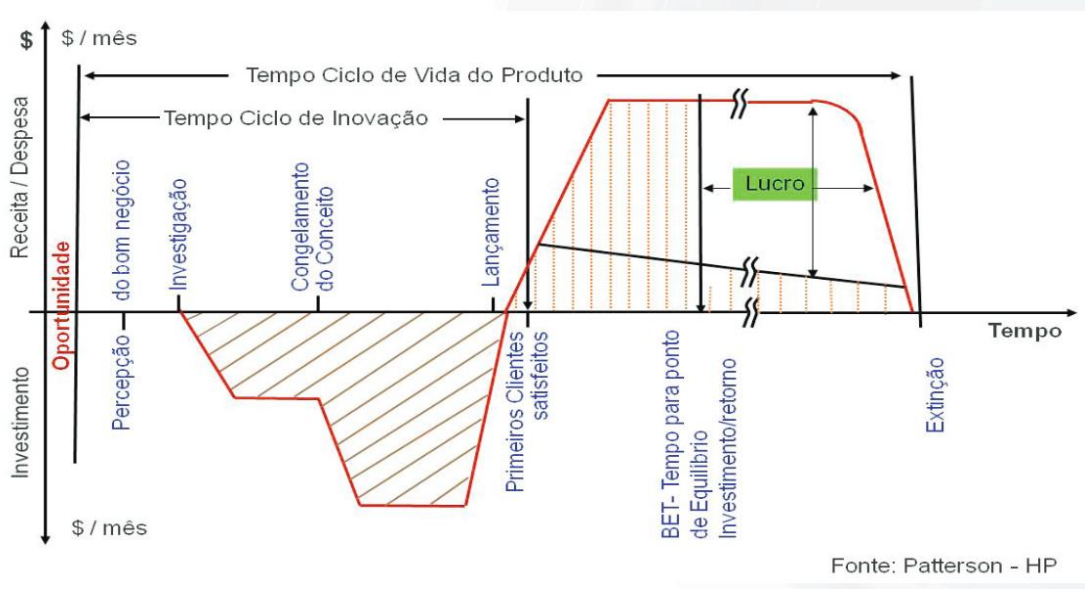

Figura 3.1- O ciclo de vida do desenvolvimento de produtos (Schneider & Souza, 2011)

Segundo Schneider & Souza (2011), um estudo feito nos USA, nos anos 90, apresentou resultados bastante instrutivos quanto ao insucesso no desenvolvimento e lançamento de produtos inovadores. Este estudo mostrou que a maioria dos projetos de produtos nunca chega ao mercado. Outros dados deste estudo mostram que (Robert Cooper, 2001):

- A taxa de sucesso de negócio é de 65%, relativo aos produtos lançados;
- De cada 7 novas ideias de produtos, 4 são desenvolvidos, 1,5 lançados e 1 tem sucesso;
- 46% dos recursos de P&D das empresas são alocados em projetos cancelados ou sem êxito.

Estes indicadores não são mostrados para desestimular as iniciativas de inovação tecnológica e sim, para destacar a importância do gerenciamento do processo competente e integrado com as várias áreas da empresa. Entende-se que os riscos podem ser reduzidos, expressivamente, se estes cuidados forem tomados (Schneider & Souza, 2011).

O que leva a estes dados tão preocupantes? Ainda segundo Schneider & Souza (2011), é muito instrutivo observar o que o referido estudo aponta como causas do insucesso de 80 produtos

inovadores lançados no mercado naquela ocasião. Como fica evidenciado na Figura 3.2 a maior causa de fracasso no desenvolvimento de um novo produto está numa **inadequada análise de mercado**. Portanto, realizar um criterioso estudo de viabilidade técnica, econômica e comercial é fundamental para minimizar este tipo de risco. Ocorre que nem sempre as empresas fazem isto, na ilusão de que estão reduzindo custos no desenvolvimento. De todas as causas de fracasso, somente uma, a **força do competidor**, não está sob o controle da própria empresa, o que demonstra que o gerenciamento do desenvolvimento do novo produto é fator-chave para a minimização do risco.

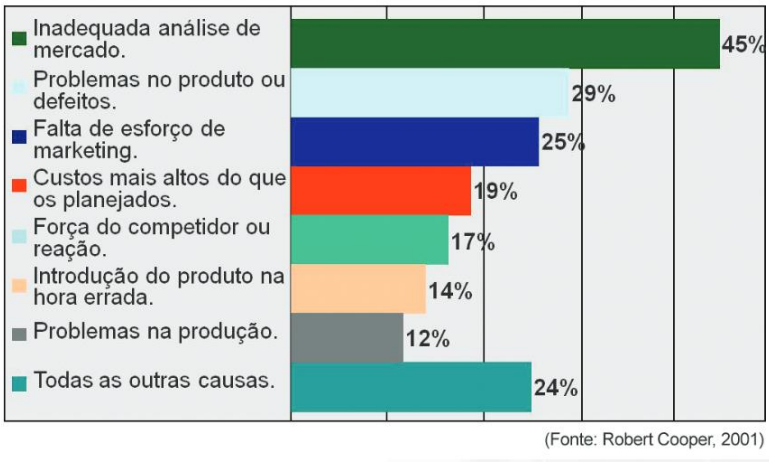

Figura 3.2- Causas de fracasso no desenvolvimento de produtos (Schneider & Souza, 2011).

3.3 Tempo para o Mercado (*time to market*)

Durante a realização do projeto de um sistema, o projetista deve se preocupar com o **tempo** e o **custo** do desenvolvimento de um projeto. Muitas vezes pode existir um compromisso entre estes dois fatores. 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. Além disso, devido à concorrência de mercado, o tempo que se toma para projetar e fabricar um produto antes dele estar disponível para compra, o chamado de **tempo para o mercado** (*time to market*), é cada vez mais importante.

Por isso 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 será 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 não de um novo projeto. As equipes de desenvolvimento procuram definir metodologia de projeto própria que agregue um diferencial de seus concorrentes. Estas metodologias em geral baseiam-se nos recursos disponíveis, utilizam normas técnicas de projeto e partes ou experiências de seus projetos anteriores. Colaboram com a redução deste tempo o uso de ferramentas computacionais de apoio a projeto.

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 a disciplina de implementação das etapas de desenvolvimento de um projeto.

3.4 NRE

NRE Non-recurring engineering, segundo Wikipedia (2019), refere-se ao custo único para pesquisar, projetar, desenvolver e testar um novo produto ou aprimoramento de produto. Ao orçar um novo produto, o NRE deve ser considerado para analisar se um novo produto será

lucrativo. Mesmo que uma empresa pague pelo NRE em um projeto apenas uma vez, os custos do NRE podem ser proibitivamente altos e o produto precisará vender bem o suficiente para produzir um retorno sobre o investimento inicial. O NRE é diferente dos **custos de produção**, que devem ser pagos constantemente para manter a produção de um produto. É uma forma de custo fixo em termos econômicos. Uma vez que um sistema é projetado, qualquer número de unidades pode ser fabricado sem aumentar o custo do NRE. O NRE também pode ser formulado e pago através de outro termo comercial denominado **taxa de *royalty***, em inglês *Royalty Fee*. A taxa de *royalty* poderia ser uma porcentagem da receita de vendas ou lucro ou combinação desses dois, que devem ser incorporados em um contrato de médio e longo prazo entre o fornecedor de tecnologia e o fabricante original de equipamento (em inglês, *Original Equipment Manufacturer - OEM*).

O Custo NRE deve ser recuperado na venda do produto somado ao custo por unidade para compor o custo de cada unidade da seguinte forma (Vahid, 2002):

$$Custo\ total = custo\ NRE + custo\ da\ unidade\ \times quantas\ unidades$$

$$Custo\ por\ unidade\ do\ produto = \frac{Custo\ total}{quantas\ unidades}$$

$$Custo\ por\ unidade\ do\ produto = \frac{custo\ NRE}{quantas\ unidades} + custo\ da\ unidade$$

Ao iniciar um novo projeto, o engenheiro precisa definir qual tecnologia irá usar, por exemplo, componentes discretos, processador de proposito-geral, processador de aplicação-específica processador dedicado, FPGA, ASIC, IC *Semi* ou *full-custom*, etc. Dentre outros fatores que podem influenciar nesta definição, os custos de NRE e da unidade têm um peso muito importante. Pois estes custos podem ser significativamente diferentes entre diferentes tecnologias. A Tabela 3.1 faz uma comparação dos custos NRE de algumas tecnologias. Assim, antes de definir qual tecnologia adotar, estes custos devem ser ponderados. Por exemplo, considere três tecnologias hipotéticas que têm custos diferentes:

Tecnologia A: NRE=$2,000, custo unitário=$100

Tecnologia B: NRE=$30,000, custo unitário=$30

Tecnologia C: NRE=$100,000, custo unitário=$2

Dependendo do volume de produção, a escolha pode ser mudada. Do exemplo acima, obtêm-se os gráficos da Figura 3.3.

Figura 3.3 - Comparação dos custos Total (a) e por produto (b) entre os três exemplos de tecnologias A,B e C.

Tabela 3.1 - NRE de algumas tecnologias

Tecnologia	Custo NRE
PLD / FPGA	Baixo
Semi-custom	*Médio*
Full-custom	*Alto*
Processadores de Propósito-geral	+ Baixo

3.5 Introdução a Projeto Eletrônico

Tendo uma visão geral do ciclo de vida de um projeto, dos custos e dos tempos envolvidos no seu desenvolvimento e do impacto destes fatores no custo final do produto, fica mais claro para um projetista entender a importância de aplicar uma metodologia apropriada na fase de desenvolvimento de uma ideia inovadora (o período entre a Investigação e o Lançamento) como uma tentativa de reduzir o risco de insucesso. Vamos começar com os tradicionais 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 (1983), referenciado em Wagner (1988), o desenvolvimento de um projeto eletrônico pode ser descrito por um modelo gráfico com três eixos de atividades, chamado "**Diagrama Y de Gajski-Kuhn**". Um eixo de atividades de descrição do comportamento do circuito, chamado **eixo comportamental**, 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 um eixo de atividades que descrevem a forma física do circuito, chamado **eixo geométrico**. Nos extremos externos desses eixos tem-se 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 3.4 exemplifica este diagrama para projeto de circuito integrado.

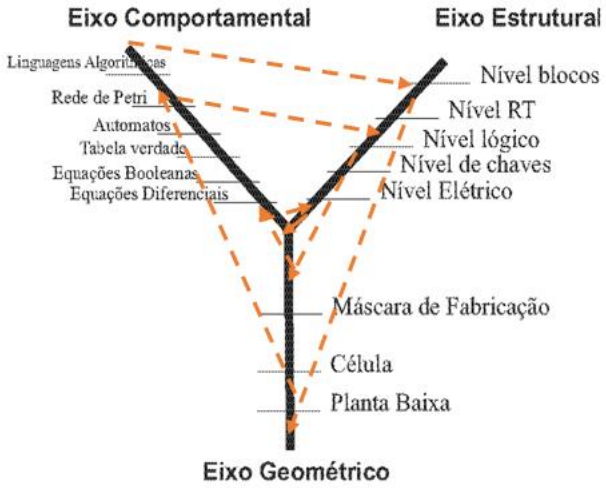

Figura 3.4 - Diagrama Y de Gajski-Kuhn

No **Eixo Comportamental** estão as tarefas que descrevem sucessivamente a funcionalidade do dispositivo projetado. Linguagem algorítmica, rede de petri, diagramas de estados finitos, equações diferenciais são exemplos de recursos utilizados nas descrições comportamentais de um circuito. Na Figura 3.4 destacam-se neste eixo os seguintes pontos:

Equações Diferenciais: são usadas para descrever comportamento a nível elétrico.

Equações Booleanas: descrevem a função de blocos da lógica combinacional nos níveis estruturais lógicos ou transferência entre registradores, em inglês register transfer (RT).

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

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

Redes de Petri: representam o fluxo de controle.

Diagramas de estados finitos: representam os estados e as transições entre eles.

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

No **Eixo Estrutural** importam as tarefas que progressivamente descrevem a estrutura do circuito. Diagramas de blocos, RT level, linguagem de descrição de circuitos, em inglês (HDL - Hardware Description Language), diagramas esquemáticos, são exemplos de recursos que descrevem a estrutura de um circuito. Na Figura 3.4, neste eixo destacam-se os seguintes pontos:

Nível Elétrico: onde estrutura pode ser descrita com transistores, capacitores, resistors, etc;

Nível de Chaves (*Gate-Level*): onde a descrição da estrutura se dá por transistores (TTL ou CMOS) atuando como chaves digitais;

Nível Lógico: onde a estrutura é descrita por portas lógicas elementares, buffers, tri-state etc;

Nível de Transferência entre Registradores (*Register Transfer*): onde a estrutura é descrita no nível de dispositivos mais complexos, como por exemplo, Registradores, somadores, memórias, etc;

Nível de Sistema: onde a descrição estrutural se dá por interconexão de subsistemas mais complexos, como por exemplo, processadores, controladores e periféricos.

No **Eixo Geométrico** (o mais 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 forma do circuito projetado. Na Figura 3.4 destacam-se sobre este eixo os pontos relacionados a tecnologia de circuitos integrados:

Planta Baixa: descreve a geometria dos diversos blocos e distribuição destes;

Células: que descreve a geometria das células dos blocos (localização);

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

Um projeto não segue somente um destes eixos. É comum, muitas vezes necessário, que o projeto avance migrando de um eixo para o outro, numa espiral até o centro, quando o projeto atinge o comportamento esperado, a estrutura definida e a geometria obtida, como na linha tracejada da Figura 3.4.

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

Automação de Projeto Eletrônico é ainda um conceito ambicioso. Idealmente seria que, a partir de uma descrição abstrata de um circuito a ser construído, um sistema automático de projetos o geraria. No atual estágio, algoritmos de síntese só existem para pequenas etapas do problema de projeto, como para módulos integrados do tipo FPGA (Altera, Xilink, etc). Sistemas de EDA são, na verdade, coleções de ferramentas que envolvem 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 em 3 classes de ferramentas: Síntese, Análise e Gerência de Informação.

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

Análise: Ferramentas que assistem a verificação da correção do projeto.

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

Entre as ferramentas de apoio à síntese, tem-se:

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 a sua descrição do projeto para o computador. Programas de captura de esquemático para apoio a projetos digitais é um exemplo.

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

Alocação e Roteamento: Os subsistemas do objeto que está sendo projetado são colocados (alocados) no espaço físico (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).

Assumindo que o projeto não tenha sido gerado automaticamente, um importante conjunto de ferramentas é aquele que auxilia na análise e na verificação do projeto.

Simuladores: São programas que modelam o comportamento E/S de um sistema. O modelo é sempre de alguma forma abstrato, desta forma ele aproxima do real mas nunca o reproduz fielmente. A modelagem pode ser feita matematicamente e assim é tratada

com mais facilidade por computador. Estas foram as primeiras ferramentas de EDA disponíveis.

Ferramentas de Análise Topológica: São responsáveis pela verificação da exatidão do layout e da interconexão dos subsistemas.

Analisadores de "*Timing*": Auxiliam na análise de tempos críticos, por exemplo, em circuitos eletrônicos e em análise de programas de computadores.

As 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. 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 mesmo com a evolução do projeto.

Gerência de Configurações: Com o tempo, um subsistema de um projeto pode evoluir. Estas mudanças representam novas versões da 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 manter o 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 à descrição do mesmo até que seja comprovado a sua consistência através de ferramentas de análise.

3.5.2 Validação de Projeto

Diversas são as técnicas de validação de projeto. Dentre estas destacam-se testes que são realizados. A Testabilidade de circuitos eletrônico é motivo de diversos trabalhos 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 dele decresce. As técnicas de Projeto Visando Testabilidade, em inglês Design for Testability, aumentam a testabilidade, decrescendo os custos de teste e aumentando a cobertura e a Diagnosabilidade. Estas técnicas são especialmente importantes em projetos de circuitos integrados, nos quais é impossível o acesso aos pontos internos do circuito. A seguir são apresentados somente alguns conceitos básicos para este estudo.

Distinguem-se os seguintes tipos de testes:

Paramétricos: são baseados na avaliação de parâmetros físicos e elétricos;

Exaustivos: são testadas todas as combinações de estados que o circuito. É Inviável para circuitos sequenciais;

Estrutural: procura-se identificar padrões de teste capazes de detectar falhas dentro do universo de padrões de testes possíveis (p.ex., pontos presos 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", também chamado de "checking", se refere aos testes durante a operação do sistema para detecção de erros. São exemplos Técnicas PARIDADE, CRC, ...etc;

Teste Explícito: é 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*" de melhoramento da testabilidade são fornecidas pelo projetista. Consiste, tipicamente, de uma lista de características do projeto que criam problemas de teste, junto com sugestões de implementação.

Para a medida de testabilidade são usados parâmetros como:

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

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

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

3.5.3 Estratégias para um Projeto Complexo.

As ferramentas baseadas em computador permitem conseguir desenvolver projetos de extrema complexidade. Mesmo como o uso de ferramentas adequadas é muito difícil entender um projeto complexo em sua totalidade. Tentativas para isto conduzem a erros graves de interpretação que, em última instância, podem levar a erros na implementação. Diversas táticas têm sido adotadas para reduzir esta complexidade e facilitar o entendimento do projeto. Algumas são descritas a seguir:

Projeto é igual à Documentação: - Manutenção e evolução têm custo muito elevado. - Grandes projetos envolvem grande equipe, que pode mudar seus elementos. 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 expressão definitiva da ideia inicialmente concebida.

Uso 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;

Descrição em Alto-nível: - Descrições concisas tornam o projeto fácil de ser entendido;

Partição: Divisão em pequenos módulos funcionais. Cuidados na integração devem ser tomados, quando for necessário compartilhar partes;

Métodos Estruturados: Muito usado em programação de sistemas. A partição é um método estruturado. Componentes são vistos como E/S, decompostos em termos de fluxo de dados e de controle;

Restrição a um conjunto limitado de construções: O uso de biblioteca de componentes básicos

Testar as partes: O projeto e o teste andam em paralelo. A cada passo é feito um teste. “Dividir e conquistar”;

Ferramentas e metodologia de projeto: O uso correto das ferramentas corretas. As ferramentas devem refletir a metodologia do projeto.

3.6 Projeto de sistemas embarcados

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 (s*oftware*). A interdependência entre estas linhas é muito forte e técnicas de co-projeto (*co-design*) podem ser aplicadas para o desenvolvimento integrado delas. Este documento apresenta as etapas do desenvolvimento de que envolvem microcontroladores. Além de uma unidade microcontroladora (MCU), uma memória e acessórios complementares como alimentação e refrigeração, um sistema embarcado, 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.);

Outras interfaces incluindo portas para diagnosticar problemas do sistema.

A Figura 3.5 ilustra um sistema ciber-físico (CPS) com destaque os detalhes que compõem a parte computacional deste sistema, ou seja, o sistema embarcado.

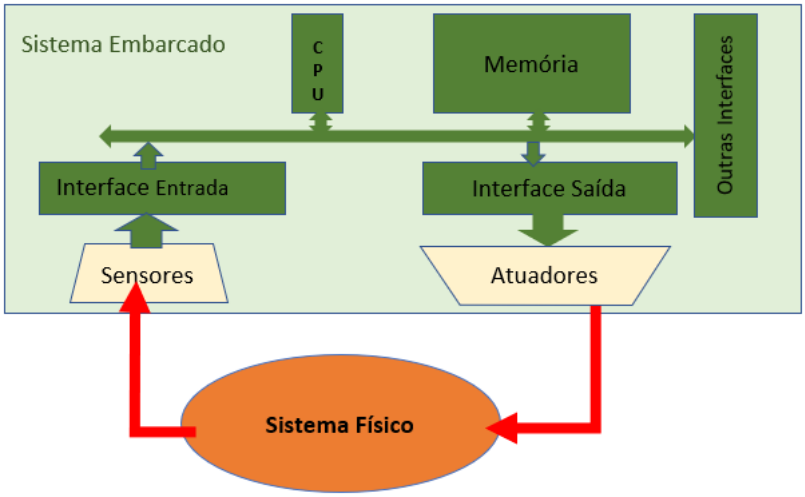

Figura 3.5 - Sistema Ciber-físico, destacando o sistema embarcado.

3.7 Proposta de uma metodologia para projetos de sistemas embarcados

Diversos trabalhos de pesquisa e desenvolvimento são realizados visando a criação, a análise, a definição de metodologias e de ferramentas para projeto de sistemas embarcados. Uma simples busca na Internet por "*embedded system*" era listar uma enorme relação de páginas acadêmicas e comerciais oferecendo recursos e apresentando exemplos de projetos nesta área. No contexto desta disciplina, propõe-se utilizar o modelo de metodologia de projeto apresentado na Figura 3.6.

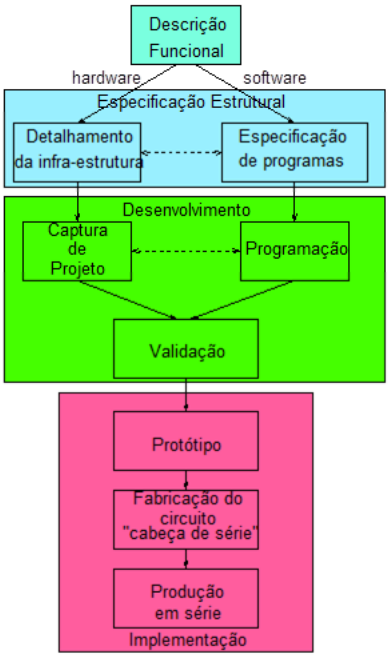

Figura 3.6 - Um modelo de desenvolvimento de sistema embarcado

Nas seções seguintes são descritas as principais etapas deste modelo.

3.7.1 Descrição Funcional

Diferentemente do projeto de sistemas computationais tradicionais, como *maniframe, 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 embarcado 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 embarcado 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 são apresentados os principais itens que devem constar na descrição funcional de um sistema embarcado.

3.7.1.1 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.

3.7.1.2 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 alojar e processar o programa de tratamento de eventos, 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;

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 3.5 . 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.

3.7.1.3 Descrição da Infra-estrutura para o projeto

Onde são identificadas as restrições para o desenvolvimento e os recursos disponíveis para ele.

3.7.2 Especificação

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 embarcados são muito maiores com, porém, menos soluções alternativas. Os sistemas embarcados 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 embarcado 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.

A especificação é muito importante para o sucesso de um projeto. A melhor definição da necessidade da especificação para o projeto foi feita por Young et al. (1985) e lembrada por Lee & Seshia (2017):

> "Um projeto sem especificações não pode ser certo ou errado, só pode ser imprevisível (no original: surprising)".

As subseções a seguir detalham a especificação dos algoritmos de processamento e a especificação da estrutura. 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.

3.7.2.1 Especificação de Algoritmos

Deve ser elaborado para **CADA** evento o algoritmo de tratamento deste evento. 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 correção.

3.7.2.2 Especificação Estrutural

Entende-se por especificação estrutural a descrição tanto das características elétricas e temporais como das restrições físicas de cada bloco funcional mencionado na Seção 3.7.1.2. Como o projeto de um sistema embarcado 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.

Em seguida, deve-se especificar 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 serem realizados, durante o desenvolvimento, para cada item definido no detalhamento estrutural.

3.7.3 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 s*oftwar*e, há também distintos recursos de apoio ao desenvolvimento, desde a geração de códigos de máquina até testes.

3.7.3.1 Captura de Projeto

Esta etapa é caracterizada pela elaboração do desenho esquemático do circuito, da listagem de ligações (net list) e de materiais (*BOM – bill of material*). 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 (Texas 85). Todos os elementos devem ser devidamente identificados. Recomenda-se utilizar nomes que lembrem os 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 = uF e VDC

   Transistor = Nome

   Diodo = nome

   Cristal = Hz

2. **Linhas (*Net*)** 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.

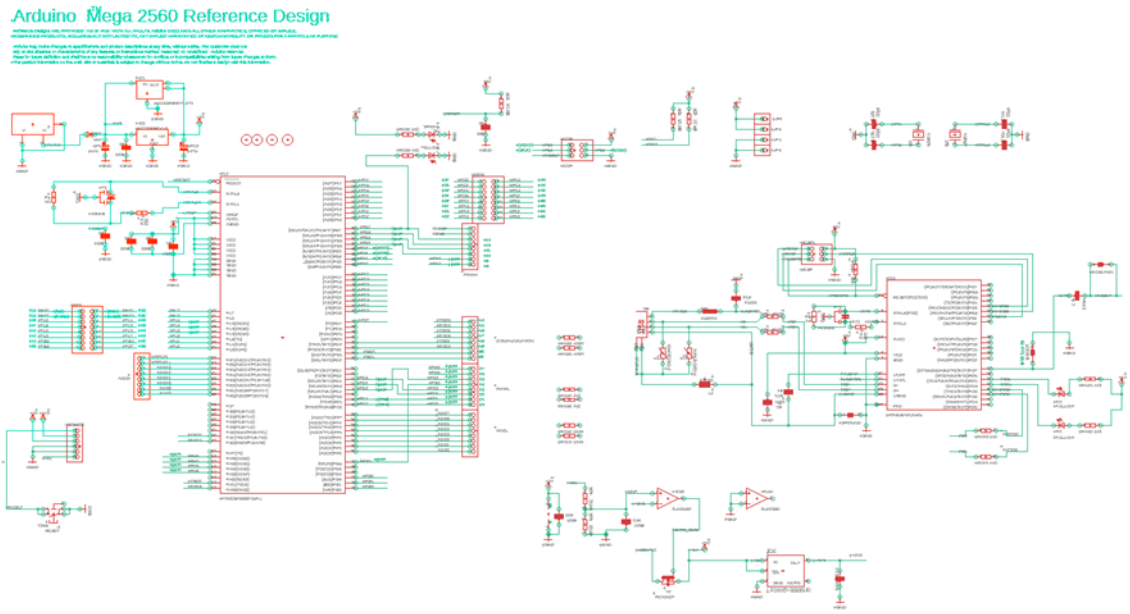

Figura 3.7 - exemplo de circuito esquemático (Eagle)

3.7.3.2 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. A Figura 3.8 exemplifica um ambiente de desenvolvimento de programas em C para a família de processadores STM32, com área de código fonte, listagem de variáveis, breakpoint etc.

3.7.4 Validação

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

As principais atividades de engenharia terminam na validação. As demais ações, de implementação, prototipagem, fabricação e produção em série não serão aqui consideradas.

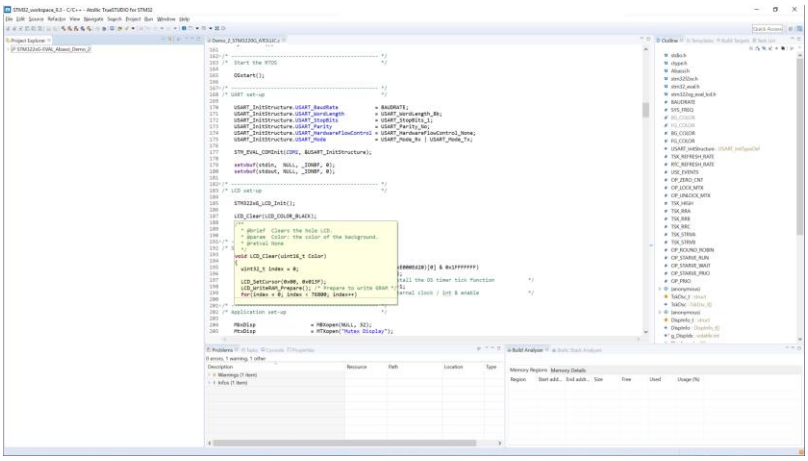

Figura 3.8 - Tela do IDE Atollic - True Studio.

3.8 WEB-Grafia comentada

http://www.iso.org. Site oficial da International Organization for Standardization, onde é possível adquirir diversas normas. Acessado em agosto de 2019.

Sobre Embedded System

http://www.embedded.com/. Site especializado em sistemas embarcados. Acessado em agosto de 2019.

http://www.microchip.com/. site da Microchip, fabricante dos microcontroladores PIC e AVR (da incorporada Atmel). Acessado em agosto de 2019.

Sobre EDA

https://en.wikipedia.org/wiki/Electronic_design_automation. Site com uma boa introdução à ferramentas de automação de projetos eletrônicos. Acessado em agosto de 2019.

Sobre CI

https://semiwiki.com/. Site com várias informações sobre circuitos integrados. Acessado em agosto 2019.

3.9 Twitters para seguir

@ABR_EmbeddedIoT. Advantech Brasil – Embed Your Mind. Líder global no mercado de computação #embedded, Advantech oferece uma ampla gama de serviços design-in e diversas soluções de #IoT.

@arduino. Arduino. O Arduino é uma plataforma de hardware, software e conteúdo de código aberto com uma comunidade mundial de mais de 30 milhões de usuários ativos.

@Arduino24x7. Arduino Updates. As últimas notícias relacionadas #Arduino, projetos, dicas, hacks e muito mais informações sobre Arduino.

@Arm Arm Discussões oficiais, notícias e blogs do ecossistema Arm de parceiros, projetistas e desenvolvedores.

@ArmMbed Arm® Mbed™ sobre sistemas operacionais, serviços em nuvem e ferramentas

@CircuitPython CircuitPython . Sobre Python em microcontroladores e a empresa Adafruit.

@ComputerSociety IEEE ComputerSociety. É a organização com membros líderes mundiais em tecnologia em computação.

@EDNcom. EDN.com. A voz do engenheiro – Rede de uma comunidade de eletrônica.

@ElectronicDesgn. Electronic Design. Sobre tecnologias emergentes e soluções de design

@embedded_com. Embedded.com. Informações para projetistas e desenvolvedores de sistemas embarcados.

@embedded_comp Embedded Computing #Embedded Projetos computacionais: Blogs, notícias & artigos sobre microeletrônica, software & estratégias #IIoT #IoT @embedded_world #IoTWorld #ew20 #IOTSWC19 #armtechcon #sps_live

@IEEEorg. IEEE. Twitter do IEEE

@IEEEPress. IEEE Press. Editora de livros de engenharia

@IEEESpectrum. IEEE Spectrum. Periódico com as notícias e análises sobre as últimas tecnologias.

@IEEEXplore. IEEE Xplore. fornece acesso em texto completo à literatura técnica de mais alta qualidade do mundo em engenharia elétrica, ciência da computação e eletrônica.

@IEEExtreme. IEEEXtreme. é um desafio global em que equipes de alunos participam de um período de 24 horas entre si para resolver um conjunto de problemas de programação.

@lhc_campinas. LHC. Hackerspace localizado no centro de Campinas Brazilian Hackerspace located in Campinas

@micropython MicroPython. Python implementado em microcontroladores e sistemas com restrições

@planetarduino. Planet Arduino. Planet Arduino é uma agregação de weblogs públicos de todo o mundo escritos por pessoas que desenvolvem, jogam, pensam na plataforma Arduino e seus filhos.

@Raspberry_Pi. Raspberry Pi. A conta oficial do Twitter para a Fundação Raspberry Pi. Notícias e informações sobre o mini PC de baixo custo.

@sparkfun. SparkFun Electronics Tecnologia abertas, recursos, eletrônica DIY (faça você mesmo).

@wolfSSL wolfSSL Embedded SSL Sobre segurança em sistemas embarcados. #IoTsecurity, #SmartCities, #Automotive. #TLS13 #SecureYourConnectivity

3.10 Referências Bibliográficas

Cooper, Robert. Winning at new products, accelerating the process from idea to launch. Cambridge: Basic Books, 2001.

Gajski,D , Kuhn, R. ''Guest's editors introduction''. Computer. pag 11-14. Dez 1983

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

Katz, R.H. "Information Management for Engineering Design ". SPRINGER-VERLAG. 1985

Lee, E. A. and S. A. Seshia. Introduction to Embedded Systems - A Cyber-Physical Systems Approach. Second Edition, MIT Press, 2017.

McCluskey, E.J. . ''Logic Design Principles - with Emphasis on Testable Semicustom Circuits''. Prentice Hall. 1986

Schneider, C.A. & A.R.de Souza. “Palestra: Alinhamento Conceitual e Prática do Processo de Inovação Tecnológica”. novembro 2011. disponível em

http://www.ipdmaq.org.br/Portal/Principal/Arquivos/Downloads/Documentos/DETI/Tutorial%2001.pdf. Acessado em agosto 2019.

Texas Instruments Incorported Semiconductor Group. "1981 Suplement to the TTL Data Book for Design Engineers". 1988.

Vahid, Frank e Tony Givargis. “Embedded System Design: a Unified Hardware/Software Introduction”. John Wiley & Sons. 2002.

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

Wikipedia Non-recurring_engineering. Disponível em: https://en.wikipedia.org/wiki/Non-recurring_engineering. Acessado em agosto 2019.

Young, W., W. Boebert, and R. Kain.. Proving a computer system secure. Scientific Honeyweller. 1985. 6(2). 18–27.