UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

SISTEMA OPERACIONAL EM TEMPO REAL DE SoC RECONFIGURÁVEL PARA RASTREIO

FABRICIO SCHLAG

ORIENTADOR: JOSÉ CAMARGO DA COSTA

DISSERTAÇÃO DE MESTRADO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO: PPGENE.DM – 303/07

BRASÍLIA/DF: JULHO - 2007

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

SISTEMA OPERACIONAL EM TEMPO REAL DE SoC RECONFIGURÁVEL PARA RASTREIO

FABRICIO SCHLAG

DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE.

APROVADO POR:

_____
Prof. José Camargo da Costa,
(Orientador)

_____
Prof. Alexandre Ricardo Soares Romariz
(Examinador Interno)

_____
Prof. Pedro de Azevedo Berger
(Examinador Externo)

BRASÍLIA/DF, 11 DE JULHO DE 2007

FICHA CATALOGRÁFICA

SCHLAG, FABRICIO
SISTEMA OPERACIONAL EM TEMPO REAL DE SoC RECONFIGURÁVEL PARA RASTREIO [Distrito Federal] 2007.
xvii, 96p., 210 x 297 mm (ENE/FT/UnB, Mestre, Engenharia Elétrica, 2007).
Dissertação de Mestrado – Universidade de Brasília. Faculdade de Tecnologia.
Departamento de Engenharia Elétrica.
1. Sistema Operacional Embarcado 2. SoC Reconfigurável
3. RTOS 4. Rastreabilidade Animal e Vegetal
I. ENE/FT/UnB II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

SCHLAG, F. (2007). Sistema operacional em tempo real de SoC reconfigurável para rastreio, Publicação PPGENE.DM-303/07, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 96p.

CESSÃO DE DIREITOS

AUTOR: Fabricio Schlag.

TÍTULO: Sistema operacional em tempo real de SoC reconfigurável para rastreio.

GRAU: Mestre ANO: 2007

_____

Fabricio Schlag
QE 30, conjunto O, casa 25, Guará.
71.065-150 Brasília – DF – Brasil.

AGRADECIMENTOS

Agradeço a Deus por ter me concedido o querer e a persistência na busca de fazer realidade a visão dos meus sonhos.

Em particular ao meu orientador, Professor José Camargo, pelos conselhos e pela ajuda, sem os quais este trabalho não teria sido concluído. Aos professores Adson Ferreira da Rocha, Ricardo Jacobi e Janaína Gonçalves Guimarães.

Agradeço meus pais e irmãos por me proporcionarem a oportunidade de concretizar esse projeto.

A minha esposa pelo carinho e compreensão em todos os momentos.

Meu primo Junior pelo auxílio, apoio e amizade.

Aos meus familiares por me acolherem em suas residências.

A meus colegas que permitiram um ambiente saudável ajudando nos momentos de dificuldades e que contribuíram para a conclusão deste trabalho.

Ao departamento de Pós-graduação em Engenharia Elétrica por estarem sempre prontos em nos atender.

Ao CNPq, pelo apoio financeiro.

Enfim, agradeço a todos que direta ou indiretamente, me auxiliaram na realização deste trabalho.

Dedico este trabalho a minha família, minha esposa, que muito me apoiaram e incentivaram nessa árdua jornada e pela graça de Deus.

RESUMO

SISTEMA OPERACIONAL EM TEMPO REAL DE SoC RECONFIGURÁVEL PARA RASTREIO

Autor: Fabricio Schlag

Orientador: José Camargo da Costa

Programa de Pós-graduação em Engenharia Elétrica

Brasília, julho de 2007.

Esta dissertação de mestrado apresenta o estudo de um RTOS (*Real Time Operating System*) para fazer parte de um sistema em chip reconfigurável podendo ser utilizado em diversas aplicações. Os componentes do sistema em chip são: processador de 32 bits, um FPGA (*Field Programmable Gate Array*) mapeado em memória, blocos de RF (Rádio-Freqüência) e matriz de sensores de imagem tipo APS (*Active Pixel Sensor*).

Para a especificação e projeto do RTOS proposto foi utilizada uma solução em ambiente operacional linux (eCos) com a possibilidade de simular e validar a proposta em computador com arquitetura x86 oferecendo portabilidade a outros tipos de arquitetura com suas aplicações voltadas para sistema em chip reconfigurável.

ABSTRACT

TÍTULO

Author: Fabricio Schlag

Supervisor: José Camargo da Costa

Programa de Pós-graduação em Engenharia Elétrica

Brasília, July 2007.

This Master's Degree dissertation presents the study of a RTOS (Real Time Operating System) for a reconfigurable system on chip that can be used in several applications. The components of the system on chip are: a 32-bit RISC microprocessor, a memory-mapped FPGA (Field Programmable Gate Array), RF (Radio-Frequency) blocks, and an APS (Active Sensor Pixel) image sensor array.

A solution in Linux Operating System (eCos) was used for the specification and project of the proposed RTOS. The developed solution was simulated and validated in a x86 computer architecture. The portability of that solution to other architectures with applications based on reconfigurable systems on chip, was also verified.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SÍMBOLOS, NOMENCLATURA E ABREVIAÇÕES.

API	*Application Programming Interface*
ASIC	*Application-specific integrated circuit*
Bootloader	*Carregar o setor de boot para o sistema operacional*
CAD	*Computer-Aided Design*
CMOS	*Complementary metal–oxide–semiconductor*
CVS	*Concurrent Versions System (Sistema de controle de versões)*
DLL	*Dynamic Link Libraries*
DSP	*Digital Signal Processor*
eCos	*Sistema operacional para dispositivos embarcados*
EDA	*Electronic Design Automation*
FPGA	*Field Programmable Gate Array*
FSF	*Free Software Foundation*
GDB	*GNU Debugger*
GPL	*General Public License*
HAL	*Hardware Abstract Layer*
IP	*Intellectual Property*
MMU	*Memory Management Unit*
MPU	*Microprocessor Unit*
NoC	*Network on Chip*
PNM	*Programa Nacional de Microeletrônica*
POSIX	*Portable Operating System Interface based on Unix*
RAM	*Random Access Memory*
RDIF	*Radio Frequency Identification*
RF	*Rádio Freqüência*
ROM	*Read Only Memory*
RSSF	*Redes de Sensores sem Fio*
RTOS	*Real Time Operating System*
SoC	*System-on-chip*
UML	*Unified Modeling Language*
VLSI	*Very Large Scale Integrated*

1 – INTRODUÇÃO

A evolução da microeletrônica chegou a ponto de viabilizar sobre um único Circuito Integrado (CI) os circuitos de rádio-freqüência (RF), analógicos, digitais e outros componentes micros e nanofabricados resultando nos chamados Sistemas em Chip (SoC).

Na atualidade do estado da arte da microeletrônica, caminhamos para um número maior de transistores, estimando-se alguns bilhões em um único chip dentro de alguns anos [1]. Para uma integração destes bilhões de transistores, torna-se imperativo uma abordagem minuciosa de uma arquitetura que preencha alguns requisitos para um melhor desempenho, que serão apresentados ao longo do trabalho.

Os Sistemas Embarcados, sistemas computacionais de propósito especiais com funções pré-definidas, estão cada vez mais presentes nas tecnologias atuais [68]. O propósito destes sistemas é incluir um número cada vez maior de funções em um mesmo equipamento, como um chip, por exemplo, preocupando-se com o custo de projeto e de fabricação desses dispositivos no desenvolvimento de circuitos integrados.

A procura por uma nova abordagem de integração dos dispositivos de um chip envolve questões financeiras e tecnológicas. O mercado indica a necessidade para que os chips fabricados possuam uma estrutura reutilizável, o que favorece a diminuição do *time-to-market*[1]. Os componentes reutilizáveis são denominados núcleos IP (*Intellectual Property*), sendo blocos de propriedade individual que podem ser desenvolvidos pela empresa responsável pelo projeto do sistema ou de empresas terceirizadas. Quanto às questões tecnológicas, o assunto é bem amplo sendo que vários itens devem ser discutidos como: topologia interna, roteamento interno das informações, um sistema operacional que gerencie a arquitetura interna, sua performance, o consumo de energia, modularidade e prováveis interconexões.

As pesquisas envolvendo sistemas em chip (SoC) buscam por processadores, comunicação, memórias, blocos dedicados, periféricos, com a parte de software contendo aplicação, sistema operacional, compiladores, drivers, API's, linguagens computacionais,

[1] *Time-to-market* é o tempo necessário para que um produto seja fabricado e fique disponível para o mercado.

metodologias de projeto na especificação e modelagem, exploração do espaço de projeto, particionamento hardware/software, estimativas, simulação, síntese do hardware, síntese do software, síntese da comunicação, projeto voltado ao reuso de componentes visando baixa potência e com capacidade de reconfiguração.

A utilização de metodologia de projeto de hardware com suporte em software acelera o processo de desenvolvimento, reduz o tempo de desenvolvimento e o custo de projeto.

Outra importância neste foco de pesquisa é a interdisciplinaridade em diversos ramos de conhecimento, onde o estudo é articulado com afinidades em questões ambientais, agricultura, biomédica, engenharia, computação e demais áreas. Desse modo, este trabalho propõe uma solução para um sistema operacional embarcado do sistema em chip reconfigurável para rastreamento.

1.1 - MOTIVAÇÃO

Atualmente as pesquisas têm levado ao projeto de software e dispositivos embarcados com modelagem e especificação em alto nível com o propósito de especificar, validar, verificar, projetar e testar sistemas embarcados e de tempo real. A existência de diversos RTOS tem motivado os projetos de arquitetura de sistemas computacionais que envolvem partes de hardware e software.

Os trabalhos relacionados a sistemas em chip como o SCI [39] (Sistema de Controle de Irrigação) do grupo de pesquisa na Universidade de Brasília envolvendo a participação de diversas entidades (14 Instituições de Ensino Superior, a EMBRAPA e vários centros de pesquisa) no âmbito da NAMITEC (Tecnologias de Micro e Nanoeletrônica para Sistemas Integrados Inteligentes) motivam estudar este projeto.

Este trabalho pretende contribuir com o desenvolvimento de um rSoC (Sistema em Chip Reconfigurável ou em inglês, *Reconfigurable System-on-chip*). O ambiente de desenvolvimento de aplicações busca estabelecer um conjunto de objetos de hardware e software e uma metodologia de integração para o desenvolvimento e validação do sistema.

A Figura 1.1 ilustra uma necessidade das partes de softwares básicos (fundamentais) para o funcionamento do sistema em chip. O software básico é composto de um sistema operacional que favorece a parte de software de aplicação que depende do ambiente de desenvolvimento. Outros acréscimos como ambientes de simulação são consideradas por permitir projetos voltados a testabilidade.

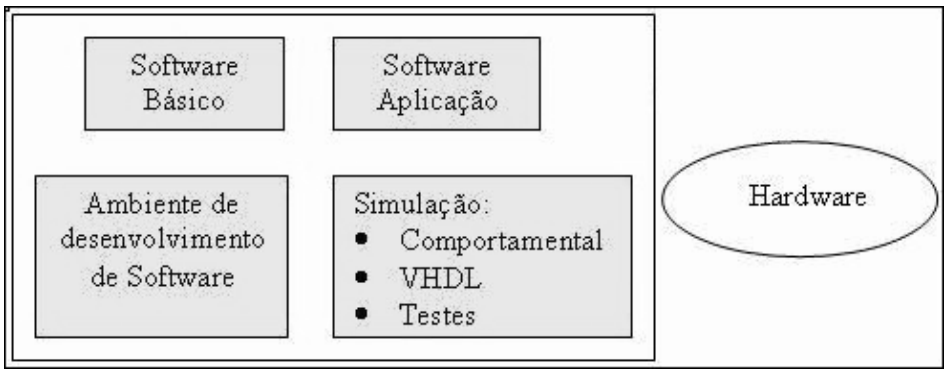

Figura 1.1: Blocos software do rSoC (explicar melhor e tirar a seta do hardware)

A parte de software é tão necessária como a parte de hardware para o projeto do rSoC. O principal produto é o sistema operacional que agregará suas ferramentas, seja o RTOS ou de um ambiente de produção de software de aplicação para as simulações e testes. A reconfigurabilidade do rSoC pode ser atuado pelo sistema operacional e pelo próprio hardware em questão.

1.2 - TRABALHOS RELACIONADOS

Este trabalho está sendo desenvolvido em conjunto com as pesquisas do projeto do rSoC para rastreamento e monitoramento, pelo grupo de pesquisa do departamento de Engenharia Elétrica da UNB. O grupo desenvolve projetos de circuitos integrados em ambiente CAD em conjunto com o projeto NAMITEC do Instituto do Milênio SCMN, Rede de Pesquisa em Sistema em Chip, Microssistemas e Nanoeletrônica e o Programa Nacional de Microeletrônica (PNM) [39].

Alguns destes trabalhos relacionados a sistemas em chip são apresentados neste item. Na dissertação de Costa [2] mostra o desenvolvimento de um sistema de comunicação sem fio e de processamento de dados em SoC com o interesse de aperfeiçoar

o aproveitamento de recursos hídricos na agricultura brasileira, permitindo a monitoração da umidade no solo para evitar o desperdício em práticas de irrigação.

O Sistema de Controle de Irrigação (SCI) é composto por uma estação-base, estações de campo e por uma rede de nós sensores. Possui uma estação-base que é a interface entre o usuário e as estações de campo concentrando as informações oriundas das mesmas, mostrada na Figura 1.2.

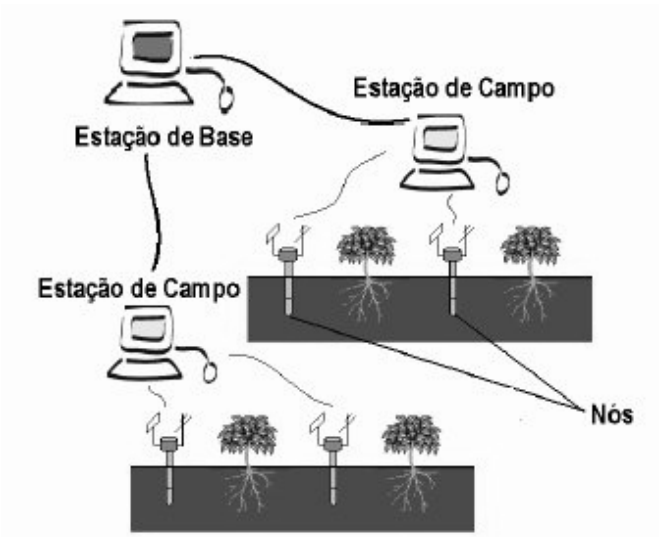

Figura 1.2: Sistema de Controle de Irrigação (SCI) [4]

De acordo com a Figura 1.2, as informações são enviadas pelas estações de campo para a estação-base por um link sem fio que são disponibilizadas para o usuário. De posse das informações o usuário pode programar o comportamento do sistema após a análise dos dados recebidos.

Melo apresenta em [3] o projeto de sistemas mistos digitais e analógicos utilizando um único ambiente de trabalho. No mesmo projeto do sistema de controle de irrigação foi simulado o nível comportamental de um processador RISC 16 bits como proposta de sua utilização. A simulação foi realizada em ambiente da ferramenta MATLAB®[2], para o desenvolvimento de um sistema em chip de comunicação sem fio.

[2] MATLAB® é um software de alto desempenho voltado para cálculos numéricos, matrizes, processamento de sinais e construção de gráficos.

Outro trabalho relacionado, de Povoa [4], apresenta o desenvolvimento de um software montador para um microprocessador RISC de 16 bits. O microprocessador RISC16, como é chamado, é parte do Sistema em Chip (SoC) que integra um sistema de controle de irrigação. Este projeto é baseado nos recursos de engenharia de software e especificação de implementação desse software abordando as etapas necessárias para o desenvolvimento completo do montador. Neste trabalho contém a especificação, o projeto, o desenvolvimento, testes do software montador além da especificação de um simulador. O montador foi chamado de Assembler16 sendo utilizado para desenvolvimento de aplicativos para o microprocessador RISC16. Futuramente o montador e o simulador serão integrados em uma única ferramenta de desenvolvimento de aplicações para o RISC16 que deverá conter também um compilador.

O trabalho de SILVA [5], tem por objetivos desenvolver softwares para sistema integrado em chip (SoC), de forma a habilitar o seu funcionamento até a execução de aplicações que validam todo o sistema. A especificação destes softwares para o sistema de controle de irrigação (SCI) é composta de programa de inicialização de sistema (*boot*), rotinas operacionais e software de aplicação, sendo desenvolvido em linguagem assembler (*assembly*). Um ponto importante deste trabalho é alocar a memória RAM para a execução das tarefas do processador como operações lógicas e aritméticas.

Em outro trabalho, Martins [6], sugere a especificação e documentação da implementação de ferramentas de desenvolvimento de aplicativos para um sistema em chip (SoC). Dois aplicativos são propostos: um ambiente visual e um simulador. O primeiro provê uma interface gráfica para o simulador e um software montador. Ele contém um editor de código assembler, um ambiente de depuração e simulação de aplicativos. O segundo programa é destinado aos projetistas de hardware e software. Ele é implementado com SystemC e oferece um modelo *register-transfer* do sistema.

O autor Carro [7] propõe uma modelagem e projeto de sistemas computacionais embarcados que são encontrados cada vez mais nas atividades cotidianas como equipamentos eletrônicos e comandos eletrônicos de veículos. Estas tecnologias envolvem restrições mais complexas que um sistema computacional tradicional, como limite de potência, custo e baixíssimo tempo de projeto, mas sem comprometer o desempenho final.

Neste trabalho é apresentada, a revisão das arquiteturas, modelos de computação e suporte de softwares necessários ao desenvolvimento de tais sistemas.

O crescimento da computação flexível em dispositivos embarcados motivou as pesquisas incorporando lógica reconfigurável com instruções do processador (ISP, *Instruction-Set-Processor*) em suas arquiteturas, podendo conseguir uma maior performance na execução das tarefas. Este tipo de tecnologia vem sendo discutido devido à necessidade de aprimorar formas para uma estrutura de gerenciamento em tempo real, facilitando o desenvolvimento de uma aplicação, contribuindo com o programador na complexidade do sistema e originando uma API mais clara para aproximar o desenvolvimento de um sistema operacional para sistemas reconfiguráveis [8].

Os Sistemas Embarcados, de acordo com o domínio de aplicação possibilitam tecnologias de sistemas em chip reconfiguráveis (rSoC). No entanto, o projeto de rSoC torna tarefas complexas em conseqüência da grande variedade de parâmetros em projeto, tais como processador, sistema operacional e barramento.

Em [9], [10] é apresentado um protótipo de um rSoC que esta sendo desenvolvido pela Universidade de Queensland na Austrália. Esta pesquisa contextualiza técnicas para projetos de sistemas embarcados em tempo real, pois, a necessidade em agregar mais complementos em suas execuções induz a redução dos custos. A sugestão é o desenvolvimento de plataformas genéricas para o projeto desenvolvendo sistemas em tempo real reconfiguráveis.

Atualmente existem dezenas de pesquisas de sistemas embarcados que referenciam projetos de sistemas operacionais em tempo real. Alguns deles são: [29], [52], [53], [54], [57], [58], [59], [60], [61], [62], [63].

O destino de todos esses trabalhos é empregado em diversos casos e como exemplo disso, as aplicações agropecuárias no Brasil através da Empresa Brasileira de Pesquisa Agropecuária - EMBRAPA [11] têm oferecido apoio ao desenvolvimento de sistemas para o rastreamento e controle de animais [12] e vegetais [13].

O Serviço de Rastreabilidade da Cadeia Produtiva de Bovinos e Bubalinos (SISBOV) têm como objetivo o controle e a rastreabilidade do processo produtivo no âmbito das propriedades rurais de bovinos e bubalinos. A adesão ao programa é voluntária para os produtores rurais, porém será obrigatório no caso da comercialização desses animais para os mercados que exijam a rastreabilidade [14].

Por exemplo, a necessidade de melhorar a qualidade da carne bovina é uma necessidade mundial em questões econômicas e de saúde. O animal é registrado em um banco de dados, no qual deverão ser armazenadas todas as informações relevantes à segurança alimentar que ocorrerem até o abate (durante sua vida), dentro do frigorífico e daí até sua entrega ao consumidor, verificando anormalidades em relação a possíveis doenças como a da “vaca louca” [15]. Também é possível ter um sistema eficiente para controle e levantamento de animais domésticos e silvestres [16].

No ramo de produtos alimentícios é possível o rastreamento em todos os estágios desde a produção, o armazenamento, o processamento, a logística e a prevenção no controle de pragas contribuindo tanto para os produtores quanto consumidores. É necessário que a fruta seja transportada e que se tenha o controle dos merecidos cuidados em relação a impactos, pressão e umidade, sendo assim se faz necessário um estudo para determinar as melhores condições para o transporte das mesmas. Cuidados em relação à temperatura requerida da refrigeração que deve sempre ser mantida, mesmo durante a manipulação da carga, toda uma preocupação em relação à umidade, ventilação e a carga estar sempre em ambiente seco, seguem o modelo da Figura 1.3 de um processo produtivo e controle na certificação oficial como a soja [13].

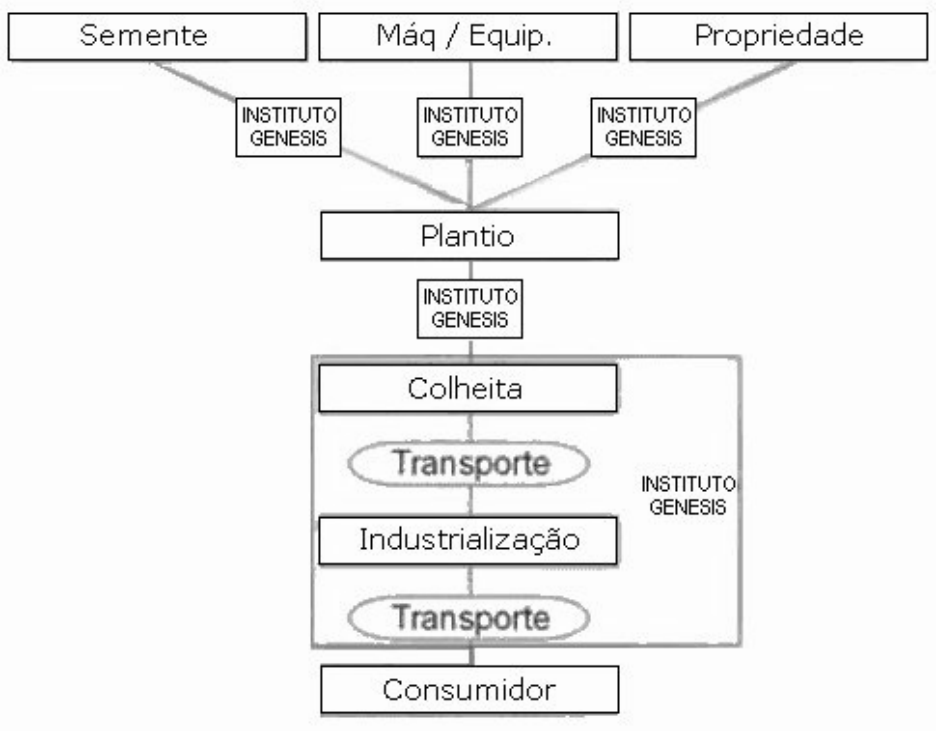

Figura 1.3: Processo Produtivo e Pontos de Controle na Certificação Oficial da soja [13]

Atualmente as aplicações citadas tiveram a solução utilizando diversas tecnologias em RFID, RSSF, GPS, etc [47], [48]. As soluções apresentadas no monitoramento de animais utilizando tecnologia *onboard*[3] geralmente gerados em produtos maiores e mais caros. Uma das propostas deste projeto está na melhoraria desses sistemas e no desenvolvimento de diversas funcionalidades em um único chip.

1.3 - DEFINIÇÃO DO PROBLEMA

Um sistema em chip (SoC) possui diversos subsistemas que estão interconectados para desenvolver funções específicas. Devido à alta complexidade existem muitas vantagens como redução do tamanho, baixo custo, o consumo de energia e aumento de potência. Em função das dificuldades e necessidades apresentadas é que se identifica a possibilidade de realizar pesquisas baseados em Sistema em Chip Reconfigurável (rSoC).

Um rSoC permite flexibilidade no desenvolvimento de ASIC (Circuito Integrado de Aplicação Específica), resultando em alto desempenho com baixo consumo de energia e com um incremento de ser reprogramado para aplicações variáveis.

[3] *Onboard*: diversos circuitos presentes em uma só placa de circuito impresso.

O rSoC, com capacidade de reprogramação tanto em hardware como em software é importante para sistemas heterogêneos, oferecendo recursos de configuração para quando necessário, ora em hardware e ora em software.

No caso do hardware para a matriz reconfigurável, o FPGA (*Field Programmable Gate Array*) embarcado proporciona agilidade e facilidade para programar, mas não oferece portabilidade em relação ao tamanho. Outro exemplo de hardware é o DSP (Processador Digital de Sinais) oferecendo maior capacidade de configuração e tem um bom desempenho, mas incide em maior consumo de potência dificultando aplicações com dispositivos portáveis.

Em relação ao software, o RTOS oferece uma capacidade de reprogramação da aplicação em um sistema embarcado. Os sistemas operacionais embarcados apresentam como solução o controle dos dispositivos e permitem um ambiente de auxílio de programação para as aplicações.

1.4 - OBJETIVOS

Este trabalho tem como objetivo estudar e propor (sugestão: especificar e configurar) um sistema operacional embarcado para um sistema em chip heterogêneo. Este tipo de sistema operacional deverá atender aos requisitos de portabilidade, reconfigurabilidade, operar em tempo real e suportar as limitações de recursos, espaço de processamento, consumo de energia e transmissão de dados.

A proposta do sistema operacional definido levará em consideração a característica de adaptação e principalmente fazer o reuso do software para atender as necessidades da aplicação. Além das características citadas, o sistema operacional deverá oferecer condições de interação com o ambiente da aplicação do SoC proposto.

1.5 - METODOLOGIA/ESTRUTURA DO DOCUMENTO

Esta dissertação se encontra dividida em cinco capítulos. No capítulo 2 serão apresentados conceitos de sistemas embarcados no contexto de sistemas em chip. É

mostrado uma comparação com os principais sistemas operacionais existentes e favoráveis ao projeto do rSoC. O resultado dessa comparação foi selecionar o eCos, o sistema operacional em tempo real objeto de pesquisa com suas características determinantes ao projeto.

O capítulo 3 descreve o projeto do SoC reconfigurável com a especificação, a implementação do sistema operacional embarcado pensando na arquitetura proposta.

Os resultados e discussão estão sendo apontados no capítulo 4 bem como os trabalhos futuros. O capítulo 5 apresenta as conclusões e propõe a utilização dos resultados para o projeto do rSoC com o sistema operacional em tempo real.

2 - SISTEMAS EMBARCADOS

Durante o processo de desenvolvimento deste trabalho foram analisados as principais referências existentes para o estudo do sistema operacional para o SoC reconfigurável. A pesquisa foi baseada na revisão bibliográfica dos projetos existentes em RTOS para sistemas embarcados e nos estudos sobre os principais sistemas operacionais disponíveis em código aberto para futuras adequações às exigências do projeto.

Os sistemas de computação podem ser gerados com finalidade especial para executar uma função dedicada, como os sistemas embarcados (em Inglês, *Embedded Systems*), que estão cada vez mais presentes e mais complexos. Um sistema embarcado resulta de uma combinação de hardware e software para desempenhar uma ou mais funções específicas e com predefinição dessas tarefas, sendo composto principalmente por processador, memória, interface para periféricos e blocos dedicados [17]. Os componentes são interligados por uma estrutura de comunicação que pode variar de um barramento a uma rede complexa (NoC – Network-on-Chip) [18].

Os sistemas embarcados estão em toda parte, como os telefones celulares e centrais telefônicas, equipamentos de redes de computadores, como roteadores, os *hubs*, *switches* e *firewalls*, as impressoras, os dispositivos de armazenamento, controladores do motor e do anti-bloqueio dos freios ABS em automóveis, calculadoras, eletrodomésticos, como fornos microondas, máquinas de lavar, aparelhos de TV, DVD *players*, equipamentos médicos, videogames, PDAs, Sistemas em Chip e outros com previsão de estarem mais presentes no nosso cotidiano [19].

Esses sistemas mesmo tendo características em comum, ficam longe de equipamentos de propósitos gerais. Para cada aplicação, pode existir uma combinação bastante distinta de módulos de hardware e software que atenda de maneira adequada requisitos de desempenho, custo, consumo de potência e área ocupada.

O projeto de sistemas embarcados exige cada vez mais a inovação de uma aplicação dependendo do software. Embora a plataforma de hardware de um celular possa ser similar à de um controle de freios ABS, definitivamente o software não é o mesmo. A

automação do projeto de hardware é determinada pelo reuso de plataformas e componentes e a automação do projeto do software ganha importância para a diminuição do tempo de projeto, sem sacrifício na qualidade da solução. Esta automação deve idealmente cobrir o software aplicativo, o RTOS [20], as interfaces entre os processos [21] e os acionadores dos periféricos. Também aqui é essencial o reuso de componentes de software previamente desenvolvidos, de modo que o projeto do sistema embarcado concentrando-se apenas na configuração e integração dos mesmos [22].

O projeto Molen [23], mostra a idéia de um processador reconfigurável com a utilização de um micro-código e um hardware configurado. A implementação é feita em FPGA (Virtex II da Xilinx). Seu principal objetivo é de aumentar as pesquisas científicas no aproveitamento de código (IP, Propriedade Intelectual) nos conjuntos de instruções através de operação por matrizes e projetar um roteamento e rede relacionados com hardware reconfigurável.

A reutilização de projetos é essencial para o crescimento de co-projeto de hardware e software de acordo as previsões da ITRS (*International Technology Roadmap for Semiconductors*) [1]. A idéia surge da engenharia de software em que o reuso é uma tecnologia fundamental. No entanto, para fornecer bibliotecas e aplicações para reuso no desenvolvimento de software, algumas iniciativas de códigos-fonte (ex.: linux, gcc, mysql) têm surgido durante a última década [24].

Uma idéia básica é distribuir as bibliotecas ou código fonte de aplicação (normalmente grátis) e permitir que qualquer programador use, modifique, depure e aprimore-as, criando componentes comuns em qualquer tipo de programa gerando uma enorme redução no custo do desenvolvimento do software. Diversas iniciativas tentam portá-las para desenvolvimento de hardware, com o objetivo principal de desenvolver uma plataforma sintética descrita em SystemC[4] de uma arquitetura aberta tendo processador e alguns periféricos básicos [25]. Um conjunto de ferramentas de desenvolvimento de software (compiladores, depuradores) e sistema operacional em tempo real tem sido desenvolvido também. Isso avalia as vantagens e desvantagens de um modelo de código aberto de projeto de sistemas eletrônicos.

[4] Linguagem de descrição de hardware como VHDL ou Verilog que utiliza o contexto C++.

2.1 - SISTEMAS OPERACIONAIS

Um Sistema Operacional (SO) é um conjunto de ferramentas necessárias para que um computador possa ser utilizado de forma adequada. O SO faz o papel de intermediário entre o aplicativo e a camada física do hardware, ou seja, é um conjunto que permite a abstração do hardware [26].

Desta forma, na ausência desses sistemas, todo software desenvolvido deveria saber se comunicar com os dispositivos de hardware do computador de que precisasse. Quando temos um sistema operacional, é ele quem realiza a comunicação com os dispositivos, como a placa de som, placa de rede, disquetes e outros. Assim, um software que seja feito para funcionar neste sistema não precisará de informações específicas do equipamento.

Cada sistema operacional pode ter uma maneira própria e distinta de comunicar-se com o hardware, razão pela qual é bem comum que software feito para um sistema operacional não funcionem em outro, principalmente no caso de linguagens compiladas.

Outra função do SO é gerenciar os recursos com função de identificar os dispositivos que estão ociosos e ocupados, como por exemplo, dividir o tempo de uso da CPU entre os vários processos, alocar e gerenciar o uso de memória principal e secundária.

Algumas definições de sistemas operacionais geram bastante controvérsia, Andrew Tanenbaum [27] considera que só a parte do sistema que roda sobre modo *kernel* constitui o sistema operacional e os demais softwares básicos são ferramentas de sistema. No entanto, outros consideram os sistemas operacionais como o conjunto de *kernel* e ferramentas de sistema [69].

Um sistema operacional pode ser diferenciado de outro por ser um sistema de código aberto ou código proprietário, pela estrutura do seu *kernel*, ou núcleo (responsável por oferecer recursos do sistema como escalonamento de processos), gerenciamento de memória, sistema de arquivos e manipulação de entrada e saída de dados.

2.2 - SISTEMAS OPERACIONAIS EM TEMPO REAL

Sistemas Operacionais em Tempo Real (RTOS) estão bem difundidos com mais de 55 tipos de ofertas, desde os mais simples, com menor exigência no controle de tarefas, aos mais complexos que utilizam maior compartilhamento de dados. A escolha do RTOS certo para uma determinada aplicação requer uma avaliação no equilíbrio entre desempenho de execução, flexibilidade de configuração, custo em licenças de utilização, apoio técnico, ferramentas de desenvolvimento de aplicação, etc..

A utilização de um RTOS no projeto de sistema em chip como o SoC reconfigurável pretende incluir um sistema operacional *open source*, livre de *royalty*[5], sistema operacional em tempo real para aplicações embarcadas e oferecer qualidade de configuração. Esta abordagem permite que o sistema operacional seja personalizado para características de aplicações precisas, resultando no melhor possível desempenho e recurso de área do chip otimizado e favorece a reutilização do seu código.

O gerenciamento de um sistema em chip reconfigurável é uma tarefa desafiadora, de modo que o sistema operacional deve estar sintonizado com as necessidades existentes neste tipo de abordagem. Para ajudar nesta tarefa, o sistema operacional deverá estar apto a distribuir os recursos dos sistemas entre diferentes aplicações existentes. Além de possuir a tarefa de monitorar o sistema como um todo, existe a necessidade de delegar tarefas para os componentes envolvidos em determinadas requisições. Para tal, este tipo de sistema operacional deve manter, para cada componente, uma descrição com as funcionalidades dos mesmos.

O crescimento de equipamentos computacionais flexíveis em dispositivos embarcados tem levado as pesquisas a incorporar a lógica reconfigurável junto com as instruções do processador na arquitetura (ISP, *Instruction Set Processor*), para que consiga um melhor desempenho na execução das tarefas. Dessa forma, as tarefas podem ser priorizadas na melhor escolha possível através do gerenciamento dos processos e dos dispositivos de hardware. (dar exemplo!)

[5] *royalty* livre é a inexistência da cobrança feita pelos proprietários de direitos autorais, patentes, ou marca registrada de um determinado produto, para a sua utilização.

A tecnologia ainda não favorece esse tipo de configuração devido à necessidade de aprimorar formas para o gerenciamento em tempo real, ou seja, uma infra-estrutura que permite facilitar o desenvolvimento de uma aplicação ajudando o programador na complexidade do sistema e promovendo uma API mais clara com a tentativa de aproximar o desenvolvimento de um Sistema Operacional para sistemas reconfiguráveis. Dessa forma, uma pesquisa da Bélgica, sugere a criação de um RTOS, o OS4RS (*Operating System for Reconfigurable Systems*) para o gerenciamento da estrutura de hardware na Figura 2.1 [8].

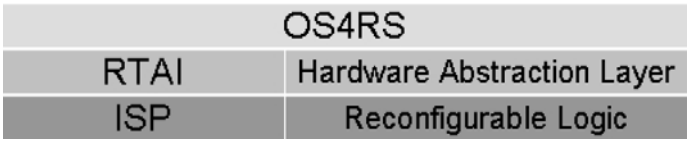

Figura 2.1: RTOS com a camada de abstração de hardware [8]

O gerenciamento de um sistema operacional para a organização e competitividade da execução das tarefas como multiplicação, execução concorrente e aplicações heterogêneas é base para a utilização de um RTOS, permitindo uma reconfiguração no momento da execução de uma tarefa, nesse caso o RTAI (*Real Time Application Interface*) [57], que é um pacote de extensão do linux para gerenciamento em tempo real. A escolha é bem colocada pela disponibilidade de código fonte, *kernel* do linux (GNU), construção em módulos e facilidade em compilar e alterar. Deste modo, existe a possibilidade para um possível gerenciamento em tempo real, utilização de computação distribuída com um sistema operacional que pode ser adequado para pequena capacidade de memória.

Em projeto de sistema heterogêneo, ou seja, com possibilidade de fazer uso de diversas aplicações e situações com a utilização de um chip reconfigurável, as tarefas tradicionais do sistema operacional ficarão mais complexas por permitir que se ajuste às necessidades do sistema. Como o sistema operacional é um intermediador (HAL, *Hardware Abstration Layer*) entre o hardware e software, são exigidos do mesmo algumas características [8]:

a) Criação/Exclusão de tarefas: Esta é a tarefa principal do sistema operacional, aliada ao fato de poder configurar parcialmente o hardware e colocá-lo no estado inicial para servir de ponto de partida para qualquer outra tarefa que for designado.

b) Realocação dinâmica de tarefas heterogênea: O problema de realocação de tarefas quando lidamos com sistemas flexíveis e diferenciados propósitos, que consiste em permitir a migração de tarefa do hardware para o software em tempo real (*run-time*).

c) Comunicação entre tarefas: Uma solução correta seria passar toda a comunicação (hardware para o software e software para o hardware) através do sistema operacional que gerencia o chip, ver na Figura 2.2. Em sistemas heterogêneos, esta solução necessita de eficiência uma vez que o conjunto gastaria tempo considerável copiando os dados entre os elementos de processamento.

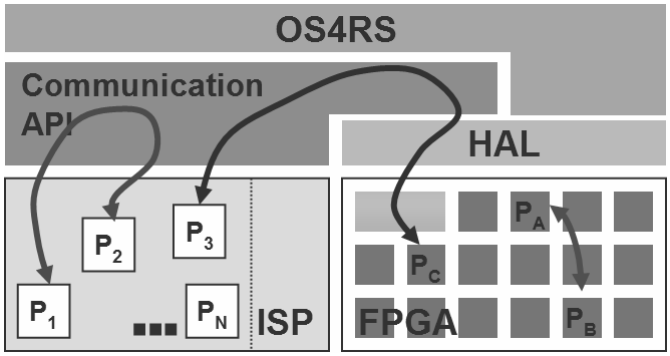

Figura 2.2: Comunicação entre tarefas no RTOS [8]

d) Depuração de erros: é importante a ser considerado, quando trabalhamos com co-projeto de hardware/software. O sistema operacional deve prover suporte para "*debug*" de tarefas do hardware, com isto, as bibliotecas referentes ao hardware e documentação devem estar atualizadas.

e) Verificação/Testes: Para estar informado do comportamento das tarefas do hardware, em termos de uso de recursos de comunicação e de segurança, o sistema operacional deve ter acesso a várias partes do chip. É essencial que um componente de uma determinada parte do chip (divisão em blocos) execute a tarefa de rastrear estes dados e forneça ao sistema operacional as estatísticas e sinais de segurança.

A possibilidade de programarmos um sistema operacional para um SoC nos leva a questionar qual a maneira de desenvolvê-lo, se a partir de um novo, o que aumentaria muito o *time-to-market*, ou buscar uma solução de código aberto. A solução mais aceitável

seria uma padronização de sistema operacional já existente, o sistema operacional GNU/Linux[6] surge como uma solução viável.

Dessa forma, há uma necessidade de sistema operacional portável para a arquitetura reconfigurável, buscando sistemas operacionais de código aberto e grátis para preencher esta lacuna. Desde que foi criado o projeto GNU/LINUX, o mesmo desperta interesse das maiores empresas dos diferentes ramos de atuação. Para aplicações de sistemas embarcados, o atrativo principal é o mesmo que o tornou popular entre as maiores corporações deste planeta, a sua livre utilização. A opção por uma distribuição do projeto GNU vem oferecer uma solução estável, usada por muitos desenvolvedores e a possibilidade de estar mais perto do hardware, por seu código fonte ser aberto, livre para modificações e redistribuições. O Linux possui uma ampla compatibilidade com os padrões API POSIX (*Portable Operating System Interface*), ou seja, segue as indicações do IEEE para sistemas abertos e portáveis.

2.2.1 - Comparação de RTOS para dispositivos embarcados

As variáveis consideradas na escolha de um sistema operacional para plataformas reconfiguráveis são: menor tempo de resposta, baixo custo, economia de energia, robustez e segurança.

A seguir são apresentados exemplos de RTOS:

- Código aberto:

  eCos [20], FreeRTOS [52], Nut/OS [60], Prex, RTAI [57], RTEMS, TLinux, SHaRK, TRON Project, RTLinux [58] e uCLinux [63];

- Código proprietário:

  BeOS, ChorusOS, ITRON, LynxOS [62], MicroC/OS-II, Nucleus, OS-9, OSE, OSEK/VDX, OSEKtime, pSOS, QNX, RMX, RSX-11, RT-11, RTOS-UH, VRTX, VxWorks, Windows CE, µnOS, RTX - Windows Real-time Extension, Phar Lap ETS.

[6] Projeto GNU: GNU's not UNIX. Sistema operacional baseado em Unix distribuído gratuitamente podendo ser modificado e copiado sob determinadas condições;

Na Tabela 2.1 são apresentados algumas características de Sistema Operacional em Tempo Real de código aberto relevantes para a definição do RTOS do rSoC.

Tabela 2.1: Comparação dos RTOSs

Open Source	**Características**	**Plataformas de hardware:**
eCos	Pode ser usado para baixa capacidade de memória RAM. Distribuição: Linux e Windows Compartilhamento de todo espaço de endereçamento.	ARM, CalmRISC, MIPS, PowerPC, x86.
FreeRTOS	Utilizado por microcontroladores. *Kernel* tempo real portável. Código aberto. Existe um arquivo de configuração para definir o escalonamento de tarefas.	ARM, x86, PIC, MSP430, 8052.
Nut/OS	Webserver embarcado, TCP/IP.	ATmega128.
Prex	Gerenciamento de memória, comunicação de inter-processos. Distribuição BSD [65].	IBM PCs and GameBoy Advances.
RTAI [57]	Extensão em tempo-real do *kernel* do linux. HAL (camada de abstração de hardware).	ARM, x86, MIPS, PowerPC.
uClinux [63]	Linux embarcado para microcontroladores. Sem necessidade de microprocessadores com unidade de gerenciamento de memória.	Linux/PalmPilot.
OS4RS [8]	- Sistema operacional para sistema reconfigurável; - API: Interface para aplicação de aplicações; - RTAI: Real Time Application Interface; ISP: *intructor-set-processor*	ARM, x86.

A comparação entre os RTOS's, demonstra que a maioria deles apresenta vantagens cruciais para os requisitos do projeto, tais como: contêm ferramentas de desenvolvimento livre, são de fácil utilização, possuem código fonte livre embarcado e royalty livre, fornecem exemplos de soluções pré-configuradas e alguns podem ainda, serem trabalhados em ambiente Windows.

É importante ressaltar que os principais aspectos no estudo de RTOS são: a capacidade de memória e, consequentemente, o compromisso dos elementos que compõem o hardware e sua aplicação.

2.2.2 - eCos

Um RTOS relativamente novo no meio de tantos existentes é o Sistema Operacional Embarcado Configurável ou como é chamado, eCos™ (*Embedded Configurable Operating System*) [29].

O projeto do eCos, teve início pela *Cygnus Solutions* [30], fundindo depois com a *Red Hat* [56] e atualmente mantida pela eCosCentric [55].

Ele oferece algumas vantagens como flexibilidade em ajustá-los às exigências da aplicação e aos recursos de hardware, isenção de taxa de licença do software e ferramentas livres para desenvolvimento visando o baixo custo.

O eCos oferece um excelente desempenho em tempo real, baixa latência, o que ocupa poucos recursos do sistema e gera respostas mais rápidas, e ainda comportamento determinístico de forma que os serviços agregados ao sistema quando em operação são executados em um tempo previamente definido tornando-o bastante viável. Uma arquitetura em camadas oferece portabilidade e reutilização do código fonte. Bem similar ao linux, a distribuição de pacotes do *kernel* do eCos permite total acesso ao seu código fonte que pode ser adicionado ao proposto sistema.

A idéia principal da "*Cygnus Solutions*" era de oferecer alta qualidade e desenvolvimento de software de código aberto. Esse projeto foi desenvolvido por Michael Tiemann, David Henkel-Wallace, e John Gilmore, colaboradores para o projeto GNU.

2.2.2.1 - Características do eCos

Eventualmente, hoje esses esforços foram adicionados ao grupo de colaboradores do projeto GNUPro, com os seguintes pacotes de desenvolvimento:

- GCC – compilador de alta otimização ANSI-C;
- Binutils – Utilitários binários;
- G++ - compilador C++ ANSI;

- GDB – depurador de código;
- GAS - GNU assembler;
- LD - GNU *linker* ;
- Cygwin™ - ambiente Unix para Windows;
- Insight – interface de usuário gráfica (GUI) para o GDB;
- Source-Navigator – ferramenta de compressão de código fonte;

As discussões tiveram início por volta de 1997, com o objetivo de trazer um baixo custo, com alta qualidade de produtos embarcados para o mercado. O eCos foi proposto para complementar as ferramentas do GNUPro, oferecido pelos produtos da Cygnus (Red Hat).

O eCos é um sistema operacional em tempo-real especialmente desenvolvido para sistemas embarcados. Como o linux, o eCos tem licença GPL (General Public License) e seu foco são aplicações para dispositivos eletrônicos de consumo, telecomunicações, sistemas automotivos e sistemas em chip. Ele é provido de um sistema em tempo real de código aberto apoiado pelo projeto GNU, permitindo o acesso em todos os aspectos de sistema em tempo real. Seu código pode ser examinado, adicionado e modificado de acordo com as necessidades do usuário. Os direitos são concedidos e protegidos pela licença eCos, dando permissão ao desenvolvimento e distribuição de aplicações. Caso o usuário venha favorecer com alguma contribuição, ela pode ser analisada e utilizada (para dispositivos de hardware e componentes) com a finalidade de beneficiar a comunidade eCos.

Foi projetado para ser portável em vários tipos de arquiteturas e plataformas como 16, 32 e 64 bits, MPUs (microprocessadores), MCUs (microcontroladores) e DSPs (processadores digitais). O *kernel* do eCos, bibliotecas e componentes em execução são organizados na camada de abstração de hardware (HAL) [31], ou seja, uma estrutura de software para manipular a estrutura de hardware. Esse conceito permite que o eCos seja executado em qualquer plataforma como, ARM, CalmRISC, FR-V, H8, IA32, M68K, Matsushita AM3x, MIPS, NEC V8xx, PowerPC, SPARC, SuperH.

O eCos suporta aplicações com requisitos de tempo real, latência mínima durante as interrupções, mecanismos de sincronização (semáforos, variáveis condicionais, etc.), vários algoritmos de escalonamento incluindo escalonamento em vários níveis e mecanismo de gerenciamento de interrupções.

O eCos disponibiliza ainda suporte a dispositivos, relógio em tempo real, gerenciamento de memória, gerenciamento de exceções, bibliotecas matemáticas e as API's POSIX e uITRON (Normalização das API's pela IEEE). O eCos pode ser programado na linguagem de programação C. O tamanho do eCos é bem variado podendo ir de algumas centenas de kilobytes até algumas dezenas de megabytes, isto graças ao fato do eCos ser extremamente configurável, ou seja, é possível incluir, alterar ou retirar qualquer item que corresponde as bibliotecas associadas a arquitetura que será portado o *kernel* do eCos e escrever programas para ser executado em conjunto. Assim, uma quantidade de código mínima pode ser gerada para rodar o sistema.

Em relação à arquitetura do eCos, é importante analisar a estrutura de componentes para projetos embarcados. Ela foi projetada para minimizar a utilização de memória, permitindo que os usuários controlem o sincronismo baseado em exigência de tempo real e utilização de linguagens de programação C e C++. Esta estrutura é especificamente destinada a sistemas embarcados e suas exigências. Uma quantidade enorme de funções para uma aplicação pode ser construída a partir dos componentes de software reutilizáveis ou dos blocos de software construídos.

Os desenvolvedores podem selecionar os componentes que satisfazem as necessidades básicas da aplicação e configurar reservadamente cada componente, selecionando o item em particular. Um exemplo disso está na programação de uma rotina de configuração, selecionando um número de níveis de prioridade e satisfazendo o sincronismo das tarefas.

A filosofia do eCos é otimizar a quantidade de funções dos componentes de implementação de controle para as limitações do sistema, ou seja, adequar o eCos para realizar funções no menor tempo ou no menor número de passos possível A Tabela 2.2 apresenta a seguinte diretiva de compilação condicional, que verifica qual a condição de

definição de estrutura para o caso de, por exemplo, uma definição “INCLUDE_FUNCTIONALITY” estar previamente definida.

Tabela 2.2: Modelo de código de diretiva de compilação condicional

```
1 #ifdef INCLUDE_FUNCTIONALITY
 2
 3     ... <condição>
 4
 5     #else
 6
 7     ... <condição>
 8
 9     #endif
```

O eCos foi projetado para atender os seguintes itens fundamentais que acabam forçando as companhias de dispositivos embarcados de produtos a desenvolver suas próprias tecnologias em tempo real:

- Custo livre de royalty, porque não existe cobrança pelo seu uso e da licença pública geral (GPL);
- Portabilidade, que oferece através das arquiteturas de chip de maneira apropriado para o desenvolvimento de software de alto volume nas aplicações em eletrônica de consumo, telecomunicações, automotivo e entre outras aplicações embarcadas;
- Código aberto, resultado de esforços de uma comunidade crescente de desenvolvedores na internet.

A configuração do eCos é possível a partir de alteração do código, como as bibliotecas em arquivo texto, e através do ambiente gráfico de configuração.

2.2.2.2 - Arquitetura do eCos

O eCos é baseado em uma arquitetura de software em camadas que abstrai os detalhes da plataforma de hardware de alto nível de aplicação, oferecendo uma excelente portabilidade, enquanto que, continua mantendo o desempenho em baixo nível dos drivers e serviços.

A Figura 2.3 mostra um exemplo de como os blocos do *kernel* são construídos. Junto com alguns componentes disponíveis e opcionais para o sistema eCos, podem ser incorporados a funcionalidade necessária a específica aplicação [32]:

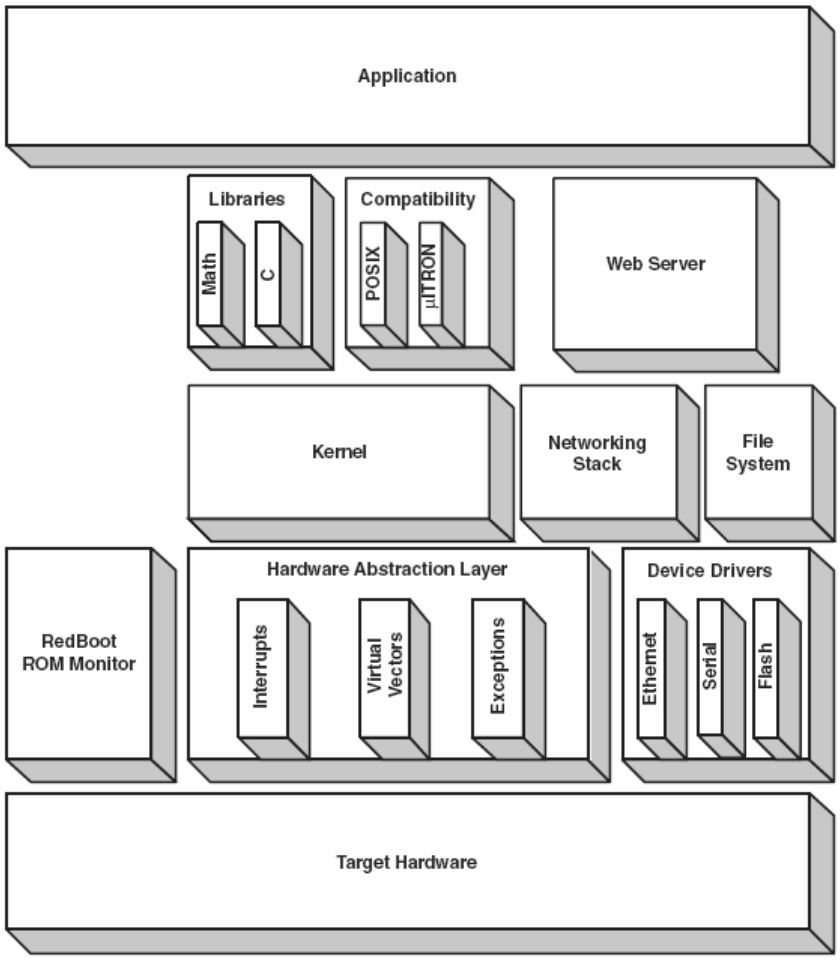

Figura 2.3: Arquitetura de software em camadas do eCos [32]

O aspecto principal do sistema eCos é que ele permite configuração do sistema. Algumas ferramentas são fornecidas como depurador de código para gerenciar alguma complexidade, permitindo que os componentes, que são os módulos dos arquivos que compõem a estrutura para compor o *kernel* sejam alterados como necessário (adição ou remoção). A ferramenta gráfica (*ConfigTool*) oferece facilidade em construir o produto final principal de uma configuração do eCos (*kernel*) e o resultado dessa construção é ligado ao código da aplicação.

A estrutura de componentes do eCos é um conjunto de ferramentas que permite o usuário configurar o sistema com os pacotes necessários à aplicação específica. Na estrutura dos componentes estão inclusos: ferramenta de configuração gráfica e por linha de comando, estrutura de disposição de memória e a administração de pacotes. Estas ferramentas são usadas para controlar e construir uma imagem de configuração do eCos [32]. Uma configuração, mantém a escolha dos pacotes que foram selecionados sendo gravado num arquivo com o nome seguido da extensão *'.ecc'*.

A terminologia do eCos compreende a estrutura dos componentes, a ferramenta de configuração (*ConfigTool*), a estrutura de diretórios do repositório, as opções de configuração, os componentes e os pacotes, as plataformas destino e os *templates* (modelos de configuração parcial).

O repositório de componentes é uma estrutura de diretório do eCos que contém todos os pacotes de instalação. Ela inclui uma ferramenta de administração para adicionar novos pacotes, atualizar e remover os pacotes velhos dentro do repositório. O diretório principal, eCos, possui os arquivos de distribuição do eCos. Já o subdiretório que contém o repositório componente são os pacotes (*packages*). Um arquivo de banco de dados, "ecos.db" (situado no diretório dos pacotes), é mantida pela ferramenta da administração do pacote e contém os detalhes sobre os vários pacotes no repositório componente.

Esse arquivo de banco de dados deve ser, ocasionalmente editado para o caso de existir uma camada HAL para uma plataforma de hardware própria, deve-se assim editar o banco de dados (*ecos.db*) para que o novo HAL seja reconhecido e controlado pela ferramenta de configuração.

Uma parte dos pacotes do *kernel* do eCos é vista na Figura 2.4 através da ferramenta de configuração. A utilização da Linguagem de Definição de Componentes (CDL) é resultado da hierarquia de configuração dos pacotes dos componentes para as opções de configurações e sub-opções. Os blocos gerados são agrupados aos itens subordinados.

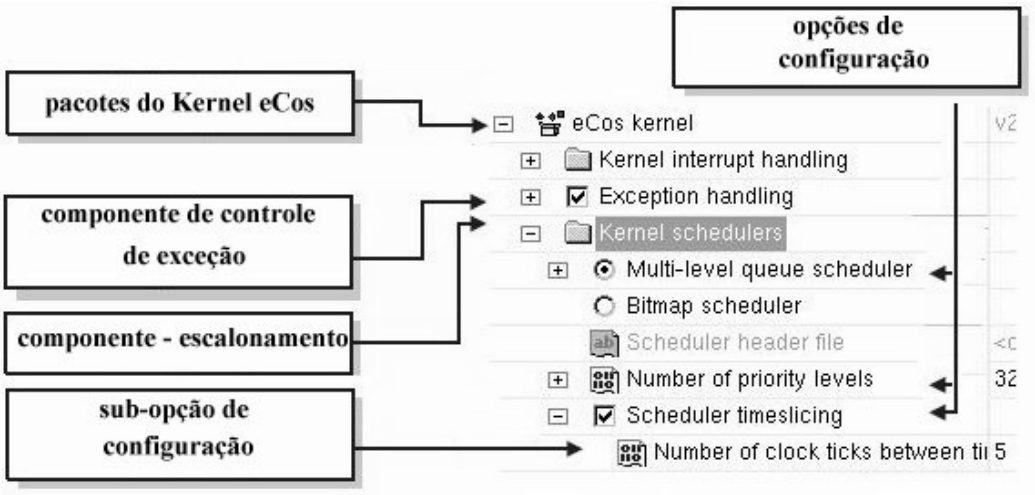

Figura 2.4: Exemplo dos blocos de configuração dos pacotes

O componente é uma opção de configuração que encapsula opções mais detalhadas. Eles podem ser habilitados e desabilitados, dependendo de sua necessidade em particular. Na estrutura dos componentes, pode se ver uma *checkbox* para habilitar ou desativar o componente. A principal vantagem é a otimização do tempo e da imagem do eCos a ser gerada.

A Figura 2.5 dá uma visão geral da estrutura do diretório do repositório de componentes de modo hierárquico, com a organização própria de instalação do eCos.

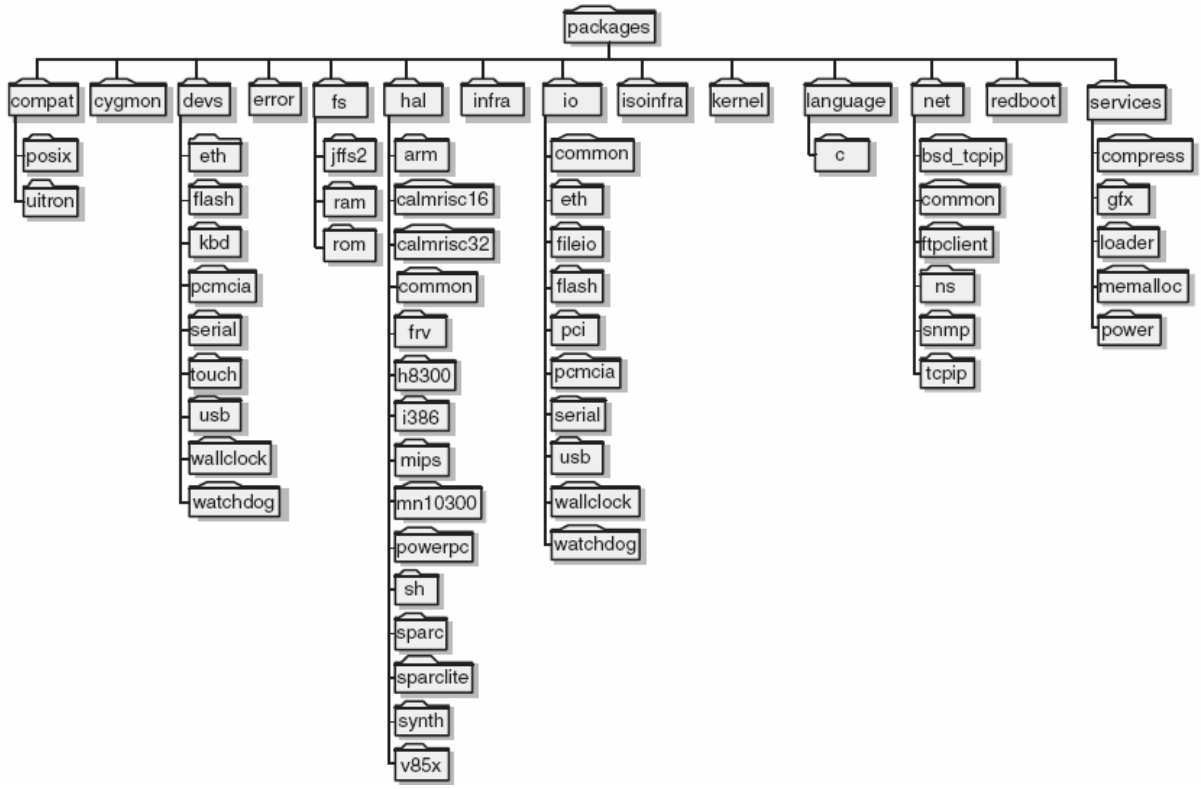

Figura 2.5: Estrutura do repositório de componentes

A opção de configuração é fundamental para configurabilidade do eCos. Na verdade, a configuração é feita para a geração de um único eCos e suas opções podem ser alteradas.

Outra forma de otimização é em relação aos componentes de hardware. Por exemplo, no caso de não ser usado uma porta *Ethernet* para uma aplicação haverá uma economia de memória para o sistema.

Já um pacote (*package*) é um tipo de componente pronto para a distribuição. Nele existem todos os arquivos necessários de código fonte, arquivo de cabeçalho, arquivo de descrição de configuração, documentação e outros relevantes. Um pacote possui geralmente um arquivo simples podendo ser instalado pela ferramenta apropriada ou alterado no futuro quando forem feitas mudanças, existindo ainda formas de controlar as versões dos arquivos.

Já o *template*, é uma configuração parcial que oferece um ponto de partida inicial e uma forma de se criar uma configuração automatizada. É uma combinação de hardware destino e um conjunto de pacotes.

Quando uma nova configuração é criada, um *template* é usado como base para somar as necessidades gerais da aplicação e as opções de configuração podem ser ajustadas de acordo com as especificações sendo mostrada todos os pacotes inclusos em cada *template* no conjunto de ferramentas.

2.2.2.3 - *Kernel*

O *kernel* é o elemento principal de todo o eCos, fundamental para qualquer sistema operacional e também chamado de núcleo. Ele oferece a funcionalidade necessária ao núcleo para desenvolvimento de aplicações *multi-threads* ou escalonamento de processos, como relacionado a seguir:

a) Habilidade em criar novos processos no sistema, tanto durante o início como no momento de execução;

b) Controlar os diversos processos, como suas prioridades de tarefas;

c) Escolha de programação, determinando qual o processo estaria em execução no momento;

d) Uma escala de primitivas de sincronização, permitindo os processos interagir e compartilhar os dados com segurança;

e) Integração com suporte do sistema para interrupções e exceções.

Em alguns sistemas operacionais o *kernel* permite recursos adicionais, como por exemplo, alocação de memória e fazer com que os drivers dos dispositivos façam parte do núcleo, mas o eCos não faz isso. Ele controla a alocação de memória através de um módulo separado, assim como os drivers dos dispositivos.

Os módulos do *kernel* do eCos são opcionais para seu funcionamento, sendo possível escrever aplicações de processos simples como o RedBoot [49]. Tipicamente algumas aplicações são baseadas em torno de uma rotina central de decisão, checando

continuamente todos os dispositivos e gerando as ações apropriadas às ocorrências de entrada/saída.

A funcionalidade do *kernel* do eCos pode ser usada de duas maneiras, oferecendo sua própria API em C, funções como "*cyg_thread_create*" e "*cyg_mutex_lock*". Eles podem ser chamados diretamente através do código da aplicação ou por outros pacotes. Também existem diversos pacotes ou módulos que oferece compatibilidade com as API's existentes como os processos:

- POSIX®: nome coletivo de padrões de uma família de padrões da IEEE para definir uma interface de programação de aplicativos de software compatíveis com variáveis do sistema operacional Unix.
- µITRON: projeto que tem como objetivo a padronização do RTOS e relacionar as especificações para sistemas embarcados em VLSI [33].

2.2.2.4 - Camada de Abstração de Hardware

Os sistemas operacionais normalmente utilizam uma camada de abstração de hardware (HAL, do inglês *Hardware Abstraction Layer*). Essa camada é responsável por transformar as instruções especificas de um dispositivo em instruções genéricas. O HAL é essencialmente importante para sistemas embarcados, pois os mesmos são executados em uma variedade muito grande de hardware. Por exemplo, os processadores ARM, muito utilizados na produção de sistemas embarcados, possuem diversas famílias que apesar de serem semelhantes cada qual tem características particulares. [34]

Outro exemplo da funcionalidade do eCos está na Figura 2.6, demonstrando a construção do eCos baseada nos requisitos de hardware e suas características possíveis.

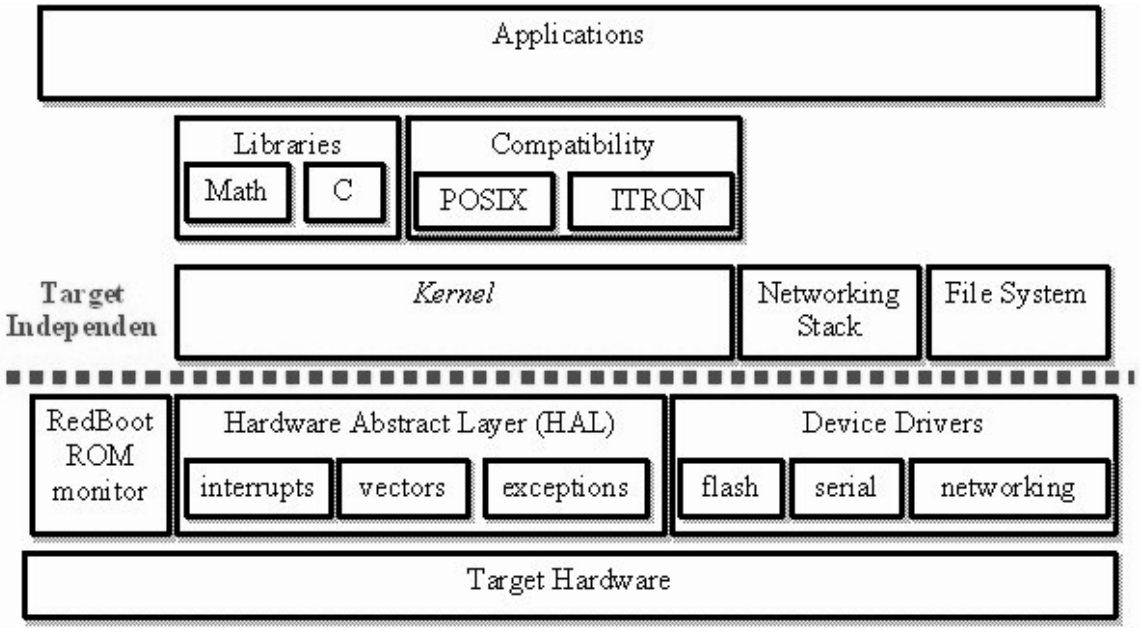

Figura 2.6: Arquitetura do eCos [32]

A camada de abstração de hardware encapsula características específicas de cada arquitetura suportada pelo *kernel*, para que ele e outros componentes possam ser implementados em sistemas embarcados.

2.2.2.5 – Redboot

RedBoot ([29], [49]) é um ambiente completo de inicialização e verificação para sistemas embarcados. Ele é baseado na camada de abstração de hardware do eCos, com requisitos de confiabilidade, compactação, configurabilidade e portabilidade. O Redboot permite baixar e executar aplicações embarcadas via porta serial ou placa de rede *ethernet*, incluindo aplicações embarcadas de linux e eCos. Pode ser usado tanto para desenvolvimento de produtos (suporte na depuração) como para produtos aplicados em campo (atualização por flash e reinicialização por rede). Em sua versão mais atual, suporta os protocolos de comunicação Ipv4, UDP, TCP, BOOTP, ICMP e no nível físico somente *Ethernet*.

O RedBoot pode ser usado para comunicar-se com o GDB (depurador GNU) para verificação de aplicações via porta serial ou *ethernet*, podendo interromper uma aplicação em execução caso tenha sido iniciada pelo depurador GDB.

2.2.3 - Projeto GNU/Linux

O termo linux se refere a qualquer sistema operacional baseado em Unix que utiliza o *kernel* ou o núcleo linux, considerado o mais preeminente exemplo de código aberto e software livre conhecido, onde o código está disponível para qualquer pessoa utilizar, estudar, modificar e distribuir novamente.

Já o GNU/Linux, refere-se a qualquer sistema operacional tipo Unix que utiliza o *kernel* linux e ainda os programas do projeto GNU. O que acontece é que geralmente os programas estão sempre associados ao sistema do núcleo do linux e sendo chamado apenas de linux ao invés de anunciar pelo termo GNU/Linux [35]. O *kernel* do sistema operacional GNU/Linux foi criado por Linus Torvalds [67].

O conjunto de ferramentas GNU é um termo dado às ferramentas de programação produzidas pelo Projeto GNU. Estes projetos formam um sistema integrado que é usado para desenvolver aplicações e sistemas operacionais, sendo componentes vitais para o desenvolvimento do *kernel* do linux e do BSD (*Berkeley Software Distribution*) [65], o Sistema Operacional Unix da Universidade de Berkeley, como uma ferramenta padrão ao desenvolvimento de sistemas embarcados.

Componentes do software GNU:

- GNU *make* - automação do processo de compilação;
- GCC - compiladores de várias linguagens de programação;
- GNU *Binutils*
- GDB – GNU Debugger, que corresponde ao depurador do projeto GNU;
- GNU *autotools* - *Autoconf, Autoheader, Automake, Libtool* –

correspondem aos geradores de arquivos *makefile*;

GNU make: utilitário que automatiza o processo de converter um arquivo em outro formato. O *make* identifica e executa programas que são necessários para processar esses arquivos, geralmente compila-se um código fonte e o converte em código objeto, que o

transforma em executável ou uma biblioteca. Podem ser utilizados os "*makefiles*" para determinar as dependências e gerar o arquivo de saída.

GCC: Conhecido como o *GNU Compiler Collection*, é um compilador de alta otimização ANSI-C produzido pelo projeto GNU. É distribuído pela FSF (*Free Software Foundation*) [66] sob a licença GPL (*General Public License*) correspondendo ao compilador padrão para softwares livres como sistemas operacionais baseados em UNIX e sistemas operacionais da Apple Mac OS X [7].

GNU Binutils: é o GNU *Binary Utilities*, uma coleção de ferramentas de programação para manipulação de código objeto em vários formatos de arquivos objetos possuindo alguns comandos predefinidos.

GNU Debugger (GDB): é o depurador padrão de softwares com grande portabilidade baseado em UNIX, funciona em diversos tipos de linguagens de programação como C e C++. [36]. Pode-se ainda utilizar um depurador com interface de usuário gráfico (GUI) para o GDB;

GNU Build System: conhecido como *Autotools*, bastante usado em softwares livres para automatizar a portabilidade de pacotes de código fonte em diversos sistemas baseados em UNIX. Os principais são: *Autoconf*, *Autoheader*, *Automake* e *Libtool*.

2.3 - SOC RECONFIGURÁVEL

A necessidade de ter um SoC reconfigurável é permitir que os componentes integrados no chip tanto em hardware quanto em software ofereçam opções de configurações. Também permitir respostas mais rápidas em mudanças de tarefas, algumas vezes podendo ser oportuno a diversos tipos de aplicações.

O rSoC é composto de chip monolítico, com tecnologia de fabricação CMOS (analógica/digital) programável, adaptativa, comunicação sem fio, sensor de imagem APS embutido, FPGA embutido mapeado em memória, barramento de alta eficiência, recurso

[7] Mac OS X: Sistema operacional criado pela *Apple* destinado aos computadores *Macintosh.*

de pipeline para o processador, memória (ROM, RAM e cachê), interface analógica/digital e projeto voltado a IP (*Intellectual Property*) como mostrada na Figura 2.7 [37].

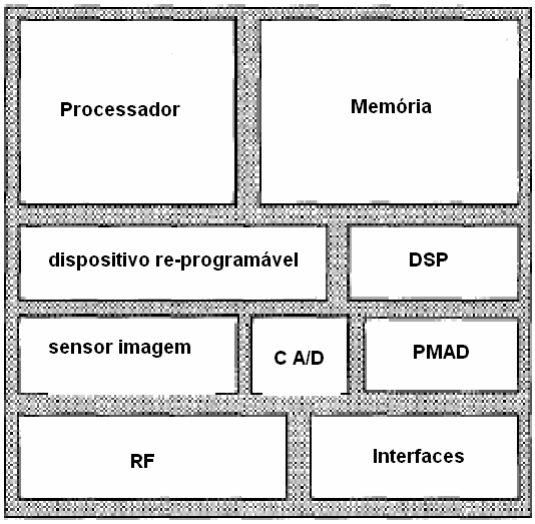

Figura 2.7: Blocos do rSoC

A parte de software é composta de um sistema operacional adaptado à reconfiguração, software de aplicação e estruturas de desenvolvimento (montador, simulador e emulador).

O objetivo da pesquisa em sistema operacional de tempo real (RTOS) é possibilitar que o SoC seja feito para uma diversidade de aplicações oferecendo adaptações e garantir a otimização do hardware [38].

O projeto de SoC reconfigurável (rSoC), do qual faz parte esta pesquisa, dá continuidade a sistemas em um único chip em microeletrônica. As atividades de pesquisas em sistemas em chip tiveram início a partir da construção do SoC para controle de irrigação do projeto NAMITEC [39], cuja motivação foi a necessidade de controlar a qualidade hídrica do solo. A aplicação do rSoC é fazer monitoramento e rastreamento de animais ou vegetais (fruticultura).

Uma utilidade para o uso do rSoC é o acompanhamento de animais existentes em parques ecológicos, matas, fazendas, galpões, etc. Neste caso, a presença de dispositivos para captura e tratamento de imagem é necessária, bem como um RTOS para auxiliar a operação em mudanças drásticas de tarefas.

A Figura 2.8 apresenta uma visualização de aplicação, o modelo do trabalho de dissertação é baseado na pesquisa de Mignolet, que refere-se ao desenvolvimento em ambiente reconfigurável por hardware e software [40].

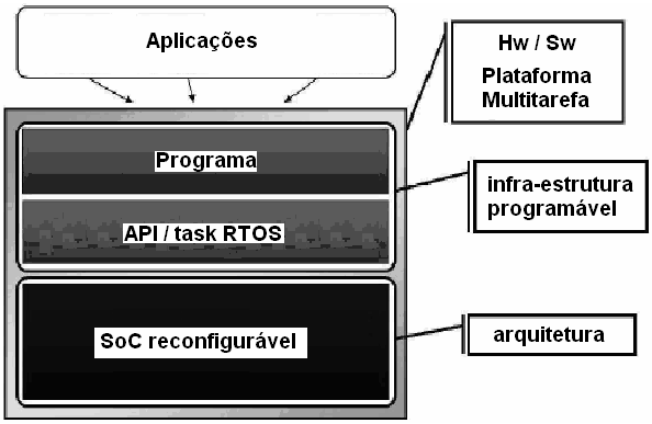

Figura 2.8: Relocação de tarefas em ambientes reconfiguráveis (modificado – Mignolet, 2002)

A primeira camada representa a plataforma reconfigurável, o SoC reconfigurável com o desafio de selecionar a medida certa dos blocos reconfiguráveis e o desenvolvimento de interconexões que irão promover a comunicação entre as diferentes partes. Uma NoC (*Network on Chip*) fornece uma solução para comunicações em SoCs complexos.

2.3.1 - Arquitetura do projeto do rSoC

Uma outra contribuição deste trabalho foram as pesquisas em definir a arquitetura (processador, memória, interfaces, etc.) e as aplicações para o projeto do rSoC. A arquitetura do sistema é proposta na Figura 2.9, e seus blocos principais descritos nas subseções. O projeto do rSoC será baseado em IP, que se enquadram nos padrões VSIA e usará a disponibilidade livre do OCP (*Open Core Protocol*) como os protocolos de barramentos.

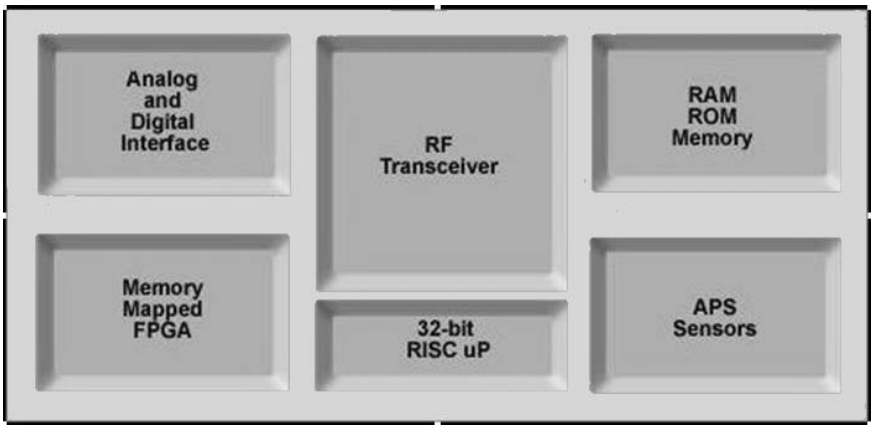

Figura 2.9: Arquitetura do rSoC

A parte do microprocessador poderá se basear em uma arquitetura RISC (*Reduced Instruction Set Computer*) de 32 bits com um circuito de hardware dedicado de multiplicação, operação aritmética de pontos fixos, hierarquia de memória e pipeline eficiente, que deverá ser escolhido. Suas instruções de pontos fixos aumentam o desempenho do sistema e diminuem a complexidade do caminho dos dados.

Considerando essas características, existem algumas opções de núcleo disponíveis, mas a escolha deve ser limitada a uma dessas arquiteturas. Um delas é o *IP-suite* para o microprocessador SoC XiRisc [41], um modelo HDL8 de RISC 32 com conjunto de instrução configurável da arquitetura e do caminho dos dados. Foi implementado em algumas arquiteturas de SoC usando a tecnologia 0.18µm e testado com algoritmos de DSP [42]. Outra opção é o processador polimórfico MOLEN CCM (*Custom Computing Machine*) baseado na arquitetura de paradigma de co-processador que foi sintetizado e prototipado numa plataforma de FPGA que foi validado usando uma aplicação MPEG-2 [23].

Uma memória RAM estática deverá ser usada para as instruções e os dados do processador por não ter necessidade de atualizar (*refresh*) e pode funcionar mais rápido do que a memória RAM dinâmica. Tem referência também de instrução cachê de 64 KB e cachê de dados de 64 KB [42], sendo implementado em tecnologia de 0,18µm com área total de 9 mm^2.

[8] *Hardware Description Language (HDL),* ou Linguagem de Descrição de Hardware.

A Tabela 2.3 representa uma área estimada de um sistema em chip, sendo que na primeira coluna são apresentados os componentes e na seguinte os suas respectivas dimensões.

Tabela 2.3: Área estimada do SoC

Componentes	**Área Estimada (mm^2)**
PA (Amplificador de Potência)	10
Memória:	
- Cachê de instrução e dados	*9*
- Cachê de configuração	*2*
Core do microprocessador	2
Blocos configuráveis	16
Interfaces	5
***Core* (total)**	**44**
PADS	**26**
Chip	**70**

O bloco é composto ainda por um arranjo de sensores APS (*Active Pixel Sensor*) de uma matriz de pixel de 256x256, circuitos lógicos em linhas e colunas, amplificadores, conversor A/D para digitalização da imagem adquirida pelos sensores, e um buffer para armazenar a imagem antes que seja enviada ao microprocessador. Uma aplicação de vídeo de 256 cores com uma taxa do frame de 30 hertz poderá ser considerada. Um algoritmo da compressão como o MPEG pode reduzir o tamanho do vídeo, o que acarreta o aumento do processamento de imagem [43].

A reconfigurabilidade é uma das principais características nesse sistema, nesse caso o projeto do rSoC incluirá um FPGA mapeado em memória, de forma que, parte do endereçamento da memória será usada para acessar o FPGA.

O FPGA pode ser configurado em rotinas específicas que geralmente demandam cálculos sucessivos, como tratamento, zoom, compressão e reconhecimento das bordas adquiridas pela matriz APS, e também seleção de freqüência do transceptor RF. Essas rotinas irão ajudar como se fossem co-processadores no FPGA, acelerando o processamento e reduzindo o carregamento computacional do núcleo do processador principal.

É possível utilizar regras de *pipeline PiCoGA* (*PiCo Gate Array*,) como unidade configurável integrada no núcleo do processador que consiste em um conjunto de linhas da matriz, cada uma representando um estágio de um *pipeline* customizado. Essa aproximação foi usada por Lodi em [42], onde cada linha do *PiCoGA* implementada em tecnologia 0,18um ocupando 1,9mm2. A potência média foi de 16mW/linha.

Um transceptor de RF é proposto por Soares em [44] e pode ser adaptado permitindo comunicação entre as unidades. Durante a transmissão, o processador seleciona a freqüência de operação usando um registrador de seleção de freqüência *(Frequency Selection Register*), de modo que a freqüência originada do oscilador externo é convertida.

Com o objetivo de processar corretamente os sinais elétricos do sistema para o projeto do rSoC, um conversor analógico/digital (A/D, ADC ou CAD) será usado. O CAD será tipo sigma/delta, com resolução de 10 bits, permitindo assim a conversão de cada pixel definida por três conjuntos (por exemplo, as cores) de valores na faixa de 0 – 255, na taxa de conversão de um segundo. A interface analógica consistirá nos circuitos responsáveis para o condicionamento dos sinais analógicos para a conversão AD. A taxa de sinal-ruído (SNR) nos circuitos analógicos será mantido acima de 60 decibéis, seguindo um regra de polegada de 6 decibéis por bit da conversão AD.

O sistema operacional para o chip reconfigurável tem como importantes propósitos, a otimização no desempenho do hardware e diminuição do custo. Deve oferecer execução concorrente e respostas em tempo real. Deve ser flexível e tolerante a falhas. Em relação à flexibilidade permitida pelo hardware e a reconfigurabilidade do sistema será feita através do RTOS [8], mais especificadamente, o eCos [29].

2.3.2 - Aplicações

Hoje em dia é muito comum que grandes empresas possuam sistemas de controle e rastreamento de produtos para Lojas Inteligentes ou Empresas de Logística, numa infra-estrutura de rádio-freqüência, por RFID, etc [45].

A rastreabilidade é o primeiro passo para atender as novas demandas dos consumidores do mundo todo, que estão se tornando cada vez mais exigentes quanto à

qualidade e à inocuidade dos alimentos. A criação de um cadastro único dentro do sistema de controle nacional e a identificação individual dos animais através de uma marca sendo por fogo, tatuagem, brinco ou microchips são simplesmente as primeiras etapas do processo que envolve a rastreabilidade. Porém, a sua implantação está variando de país para país, de acordo com os hábitos alimentares dos consumidores e a sua classificação no mercado mundial como importador ou exportador.

Como referência de projeto anterior em sistemas em chip, o SCI [4] (sistema de controle de irrigação) tem como objetivo melhorias para qualidade do solo no controle de irrigação dentro os diversos atributos como temperatura e umidade, surge a idéia de aproveitar do crescimento em SoC, como também utilização de transistores CMOS, a melhoria e complexidade em ferramentas para projeto de circuitos e necessidade em estudos em Tecnologia da Informação para Sistemas Agroindustriais.

O projeto visa o controle e rastreio no carregamento de frutas ou vegetais, com a intenção de avaliar a qualidade da carga transportada com a presença de um chip ou vários chips junto com a mercadoria, podendo enviar os resultados coletados desse chip para uma central também residente no caminhão de transporte. O rSoC terá diversos atributos, o desenvolvimento acontece tanto em nível de hardware e como para software para o produto em si, como a descrição, modelagem, estrutura, testes, simulações, fabricação do chip e ainda com as dependências no desenvolvimento de produtos voltados para o controle e gerenciamento dos resultados obtidos com a coleta dos dados, surgindo até a necessidade de um software de gerenciamento.

Para que todo esse trabalho seja concretizado, precisa-se que diversos elementos estejam bem adequados. No caso do rastreio voltado para o transporte, o caminhão deverá conter um módulo de monitoramento auxiliado pela bateria do veículo, os rSoC deverão fazer o processamento de tempo em tempos e um protocolo de comunicação para emissão dos resultados será utilizado.

No Controle de Frutas e verduras, para o produtor saber da avaliação em relação a produtos químicos permitidos, o sistema deverá ser um gestor de acesso rápido e fácil de modo automático, além de rastrear, será um assessor e consultor jurídico do produtor. Em

uma eventual ameaça sanitária ou se um supermercado precisar de informações especificas do produto, será possível rastreá-lo num instante.

Dentre as aplicações do rSoC, um possível rastreamento animal é apresentado na Figura 2.10. O movimento dos animais poderão ser detectados por sensores que ativam o sistema de RFID (*Radio-Frequency IDentification*), identificação por rádio-freqüência [46]. O RFID estará posicionado em alguma etiqueta no animal que enviará informações por uma comunicação sem fio. Pode também monitorar e analisar o comportamento do animal através dos comportamentos dinâmicos conseguidos através dos sensores.

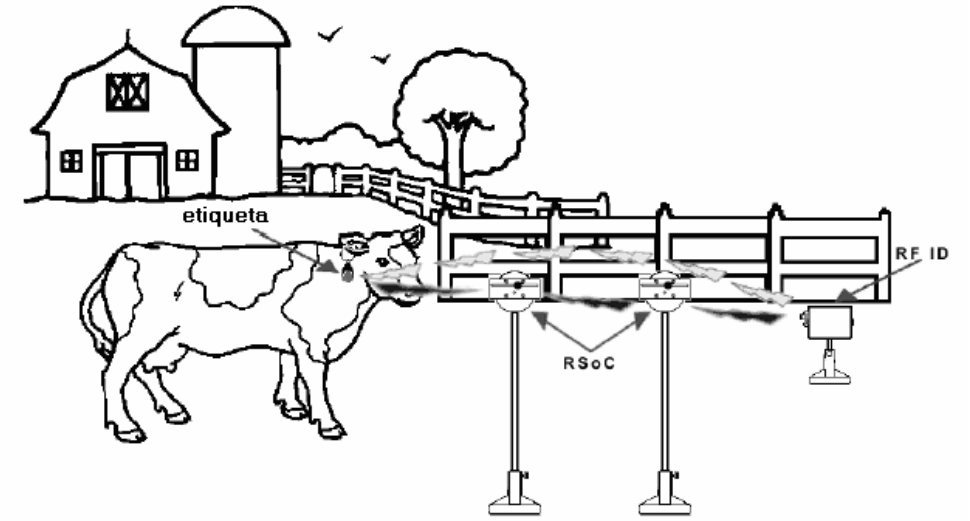

Figura 2.10: Rastreio animal

Para a aplicação de rastreio vegetal, no quesito de transporte, idealiza o rastreio de frutas desde a sua produção até chegar ao consumo do alimento, com a preocupação maior de melhorar a qualidade no transporte.

O desenho na Figura 2.11 mostra uma pseudo-fruta (fruta falsa) dentro de um caminhão correspondendo a um SoC, com as características de monitoramento de imagem com dispositivo de captura APS, módulo de alimentação e eventualmente um atuador (ex: alarme para avisar sobre condições criticas das frutas ou da carga da bateria).

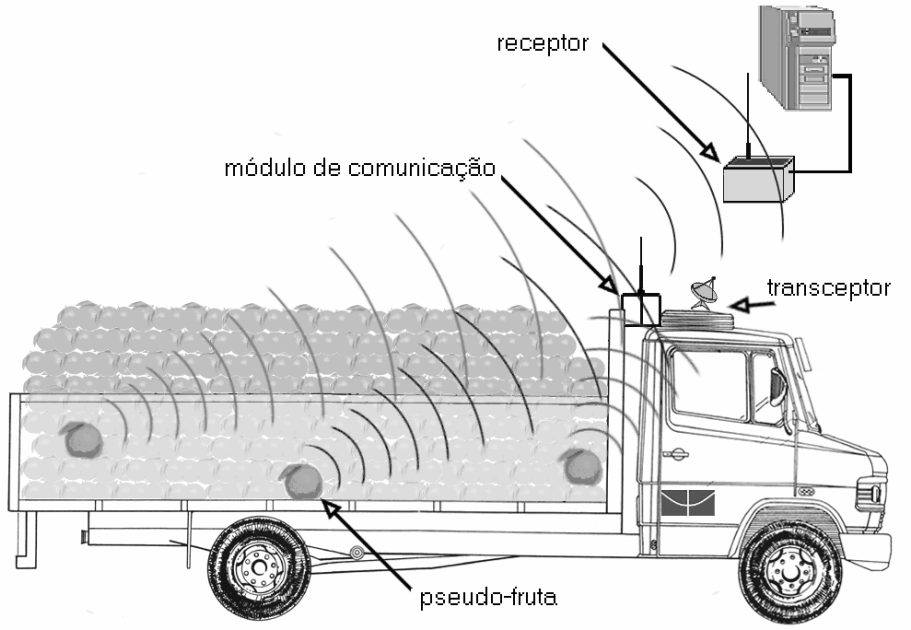

Figura 2.11: Rastreio vegetal

Na especificação existem sensores discretos (não integrados) de temperatura, de umidade, de vibração e de pressão. Um módulo de comunicação e um repetidor/transceptor no container poderão ter comunicação com estações de supervisão terrestres ou via satélite.

2.3.2.1 - Reconfigurabilidade em relação à aplicação

A utilização do RTOS vem ajudar no ambiente de descrição do rSoC, funcionando como um middleware, auxiliando na transmissão de dados e informações entre os dispositivos. Seu objetivo é mascarar a complexidade de possíveis aplicações para o rSoC e fornecer um modelo de trabalho mais produtivo para os programadores de aplicação.

O sistema operacional eCos é o conjunto de ferramentas necessário para o hardware proposto intermediando entre o aplicativo e a camada física. O processo de gerenciamento de recursos [27] é carregar primeiramente o sistema operacional da memória secundária para a memória principal ativando o sistema (boot) através de um programa em uma posição específica da ROM (disco).

A Figura 2.12 propõe um modelo de reconfiguração de sistema com a utilização de um sistema operacional em tempo real e suas ferramentas para a arquitetura proposta. O resultado da junção da aplicação com o eCos poderá ser enviado a um dispositivo para

validação. Futuramente é previsto embarcar o sistema diretamente para o rSoC.

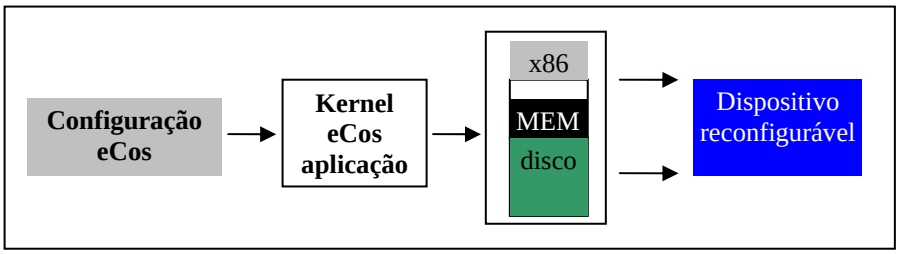

Figura 2.12: Reconfigurabilidade do sistema

Um exemplo, para rastreio de frutas e verduras com o sistema configurado para monitoramento em trânsito seria verificar a temperatura e a qualidade das frutas em relação ao balanço no transporte realizando em vários tempos. Neste caso as configurações podem ser pré-estabelecidas para mudar a sua configuração em função do tempo ou uma condição verificada no sistema diferente do normal.

A importância de sistemas configuráveis é prever a utilização de recursos mínimos do sistema garantindo um maior aproveitamento no processamento dos dados e prolongando o tempo de vida.

3 - IMPLEMENTAÇÃO DO RTOS

As aplicações que utilizam ambientes embarcados buscam a integração do sistema em um único chip (SoC). Sendo que os requisitos de área, potência e desempenho são mais críticos. O projeto do SoC na forma de um ASIC (circuito integrado para aplicação específica) pode ser determinante para o resultado, o que eleva os custos de projeto e fabricação.

A arquitetura de hardware de um SoC é composta por processador, memórias, interfaces para periféricos e blocos dedicados. Os componentes são interligados por uma estrutura de comunicação que pode variar de um barramento a uma rede complexa (*NoC – network-on-chip*). O processador de 32 bits oferece alta capacidade de processamento, permitindo a reconfiguração através de um FPGA mapeado em memória.

Em sistemas contendo componentes programáveis, o software de aplicação pode ser composto por diversos processos. São distribuídos entre diferentes tarefas e comunicam-se através de mecanismos variados. Um sistema operacional de tempo real (RTOS – Real Time Operating Systems) é necessário, pois oferecem serviços como comunicação e escalonamento de processos.

O uso de um sistema operacional para controlar o rSoC oferece características que tornam o sistema apto a utilizar um algoritmo para escalonar tarefas com requisitos temporais. Outras características são tratar as aplicações em tempo-real, ser robusto, eficiente e consumir poucos recursos. A possibilidade em projetar o RTOS para arquiteturas diferentes é favorável oferecendo portabilidade ao desenvolvimento.

A Figura 3.1 descreve como foi executada as implementações com o eCos para sua validação. A ferramenta de configuração permite a padronização de acordo com a aplicação gerada para facilitar o ajuste do sistema. O eCos permite que o sistema gerado, somado à aplicação, utilize o projeto GNU para compilar, depurar, gerar o código binário e portar o resultado.

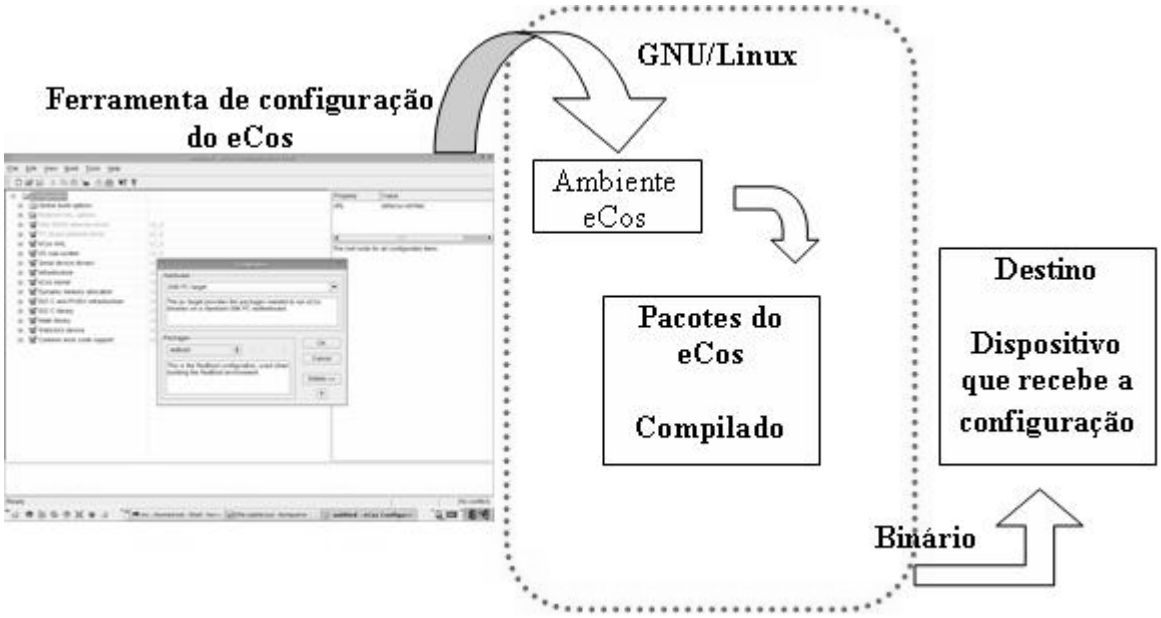

Figura 3.1: Ambiente de desenvolvimento do eCos

3.1 - IMPLEMENTAÇÃO DO eCos EM PLATAFORMA X86

A implementação do eCos foi feita através de uma simulação utilizando computador com arquitetura x86. O eCos oferece portabilidade a outros tipos de arquitetura. Este desenvolvimento resulta em um sistema operacional em tempo real com suas aplicações podendo ser utilizadas para sistema em chip reconfigurável.

A configuração e validação do eCos através de suas características ressalta a utilidade de um sistema operacional embarcado resultando em mecanismos de inicialização de sistema e aplicações em linguagem C.

As implementações são executadas através dos seguintes requisitos de sistema:

- Arquitetura Intel [28] padrão, testado no ambiente operacional com as distribuições Red Hat, SuSE e Debian, Kurumin e também no Microsoft© Windows NT, 2000 e XP;
- O processo de instalação lista os componentes do eCos e o conjunto de ferramentas de compilação;
- Exigência de no mínimo 64 Mb de memória RAM e processamento de 350 MHz.

Antes da instalação, discute-se porque ter um outro sistema operacional em tempo real embarcado, mesmo com muita solução em tempo real comercialmente disponível, sendo que mais de 50 por cento de todos os projetos embarcados são construídos hoje por meio de proprietário, ou seja, produzidos por encomenda. Os colaboradores originais dos eCos, foram a Cygnus [30] e posteriormente a Red Hat [56]. Entretanto, a Cygnus atua na venda de microprocessadores que alocam companhias de fabricação do produto para desenvolver o eCos e satisfazer a exigência de uma solução em tempo real aberta de técnica avançada.

A instalação do RTOS eCos pode ser feita usando o gerenciador de pacotes do linux, ou pelo comando exemplificado anteriormente. No primeiro caso, basta localizar os pacotes necessários:

a) ecos: corresponde ao arquivos do *kernel* do eCos e também o arquivo de configuração (*ConfigTool*) e a linguagem de componentes;

b) ecosconfig: são arquivos para execução do eCos por linha de comando;

c) ecos-doc: é a documentação de referência do eCos e os exemplos de aplicação em linguagem C;

O eCos é projetado para a compilação com um conjunto de ferramentas do projeto GNU/Linux. Existem pacotes pré-construídos disponíveis para determinados tipos de arquiteturas discriminadas no pacote de instalação do eCos. Diversos colaboradores visam desenvolver soluções para novas arquiteturas.

As instalações requerem um espaço em disco de pelo menos de 10 MB para armazená-lo e concluir o processo. Em ambiente Windows é necessário o sistema de emulação UNIX, o Cygwin^{TM}. Para o conjunto de ferramentas do eCos (*Toolchain*) é usado um processo de instalação simplificada que baixa os arquivos fontes e sua documentação.

O Cygwin é um emulador do sistema operacional UNIX com uma coleção de ferramentas de software livre desenvolvidas originalmente pela *Cygnus Solutions* fazendo com que várias versões do Microsoft Windows emulem um ambiente Unix. Ele se apresenta como um terminal texto do tipo *bash shell* que contém grande parte da funcionalidade e dos aplicativos disponíveis no UNIX. O Cygwin disponibiliza as

ferramentas nativas para Windows os programas dos pacotes GNU (utilitários, compiladores e depuradores), linguagem *tcl/tk*, GNU *make*, entre centenas de outros, além dos comandos tradicionais do UNIX tais como o *ls*, o *clear*, o *cd*, e demais.

Estes programas permitem que sejam escritos códigos originalmente para UNIX podendo ser facilmente portado para Windows. Isto é possível principalmente porque o Cygwin possui uma biblioteca dinâmica – chamada *cygwin.dll*.

O Cygwin é uma ferramenta essencial para este trabalho, caso não seja usado o sistema operacional linux, porque os programas GNU utilizados são distribuídos na forma de código fonte, e este código fonte só pode ser compilada em um sistema baseado em UNIX. O processo de instalação do Cygwin está apresentado no apêndice A.

A instalação do eCos em ambiente linux é diferenciada por suas distribuições, algumas já possuem os pacotes do eCos e permite uma instalação mais fácil do que outras como pode ser visto em apêndice B.

As implementações do eCos foi baseado em distribuição linux Kurumin, que é uma distribuição GNU/Linux popular no Brasil baseada nas distribuições Knoppix, Kanotix e Debian. Inicialmente com a intenção de promover maior portabilidade podendo ser executado por uma mídia de CD. Além disso, é um código leve, que permite maior facilidade no uso, maior detecção de hardware e mantém compatibilidade com os pacotes Debian. Por exemplo, as distribuições podem ser encontradas nos CDs ou em sites da Internet [29]. Outro recurso é a facilidade de adicionar pacotes usando o gerenciador de pacotes do linux, nesse caso, basta localizar os pacotes necessários (ecos, ecosconfig e ecos-doc) e instalá-los.

Após a instalação do eCos pode-se iniciar o desenvolvimento com o processo de configuração apresentada na tela com a ferramenta *ConfigTool* do eCos como pode ser visto no instante de selecionar um *template* na Figura 3.2. O *template* nesse caso foi para plataforma i386 com opção de utilizar o RedBoot.

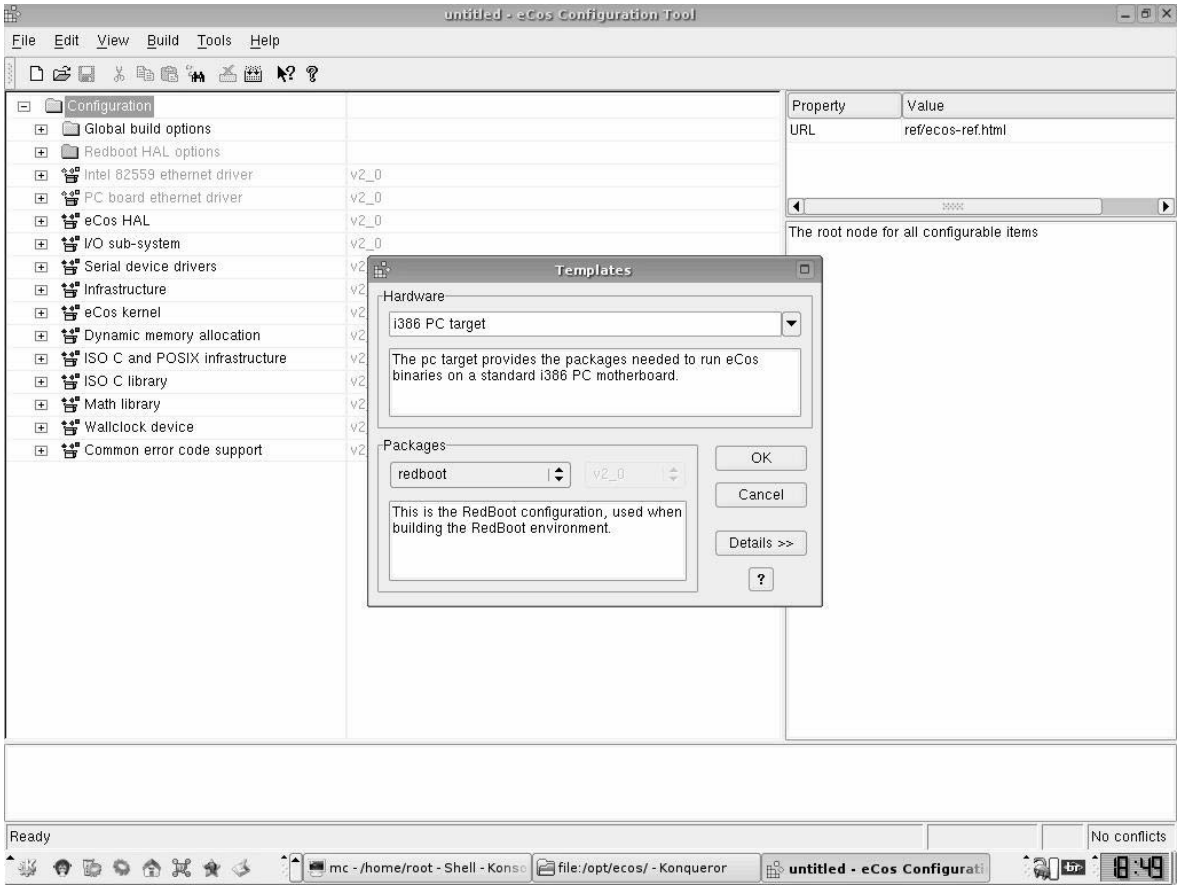

Figura 3.2: Ferramenta de configuração do eCos, templates

O primeiro passo foi selecionar os pacotes necessários para construir a imagem do RedBoot. Foi usado o template para a seleção de construção “Build – Templates” da janela de configuração. Subseqüente foi necessário selecionar *i386 PC target* e na opção “packages” e selecionado “*redboot*”, ao confirmar com o botão OK. Em seguida, pode aparecer a caixa de diálogo de “*Resolv Conflits*“, bastando confirmar em “*continue*” visto na Figura 3.3.

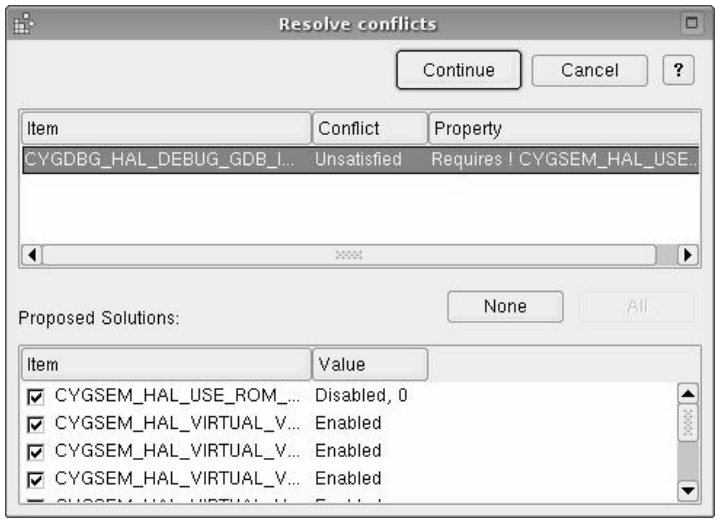

Figura 3.3: Janela de conflitos na configuração do eCos

As opções de configuração para a construção do RedBoot foi alterada importando o mínimo necessário em 'File – Import' e selecionando o arquivo apropriado com extensão "*.ecm"* que representa o arquivo de configuração mínima do eCos localizado em "$ECOS/ecos_2.0/packages/hal/i386/pc/v2_0/misc/redboot_FLOPPY.ecm". Do mesmo modo que existe a opção para *Floppy*, existe também o pacote para ROM.

Outra alteração foi inserir o endereço IP e configuração da porta serial para monitor de ROM do RedBoot (RedBoot ROM monitor) em opção de configuração.

A partir da janela do "eCos Configuration Tool", em "*Configuration > redboot ROM monitor > Build RedBoot ROM ELF image > RedBoot networking > default IP address*" é necessário especificar o IP no formato exemplo: 192, 168, 1, 102. Coloca-se vírgula ao invés de ponto. Verificar também se a opção "*Do not try to use BOOTP*" está desmarcada. Então, a partir da opção "*RedBoot Networking*" selecione "*DNS support – Default DNS IP*" ajustando para o endereço IP para qualquer DNS que seja acessível da sub-rede, no formato exemplo "201.10.128.3", ver que agora se usa ponto ao invés de vírgula.

Depois foi selecionado o canal de comunicação para receber os arquivos binários para o monitor ROM RedBoot a partir de "*Configuration - eCos HAL – i386 architecture – i386 PC target*". Foi usado o "canal 0" para a porta "serial 0", /dev/ttyS0 deixando a opção de Saída para a tela do PC ("*Output to PC screen*") selecionada.

A seguir é necessário salvar o arquivo em "*File – Save as*" e colocar um nome como "*ecos.ecm*" no diretório $ECOS. Ele cria automaticamente 3 subdiretórios:
- built: arquivos make e Obj;
- install: arquivos binários de saída;
- mlt: referente à memória.

O próximo passo foi construir a imagem do RedBoot a partir da ferramenta de configuração através dos parâmetros configurados. Após isso, existirá o arquivo binário 'redboot.bin' no diretório '$ECOS/ecos_config_install/bin'.

A criação do arquivo de inicialização para o disquete foi executada em seguida com o propósito de portar o eCos juntamente com a aplicação para validar a implementação. O resultado foi uma visualização de mensagem mostrada em tela.

3.2 - EMULAÇÃO DO eCos EM X86

A simulação feita permite garantir eficiência na implementação do eCos, que utiliza também o RedBoot (*bootstrap firmware*) [49], que é um ambiente de inicialização embarcado no dispositivo de hardware que pode estar em uma ROM, EPROM ou EEPROM. [29].

O eCos é um sistema operacional embarcado em tempo real de código aberto, sendo portável a uma variedade de arquiteturas. Ele oferece características de baixo custo em desenvolvimento e soluções de desenvolvimento de sistemas embarcados livres de royalty.

Os sistemas operacionais de aplicações nesta área são escritos usando um conjunto de API's usadas pelo eCos que são ligadas ao *kernel* resultando em um binário simples contendo a combinação da aplicação com o código do *kernel* representado na Figura 3.4 a utilização do eCos e o projeto GNU.

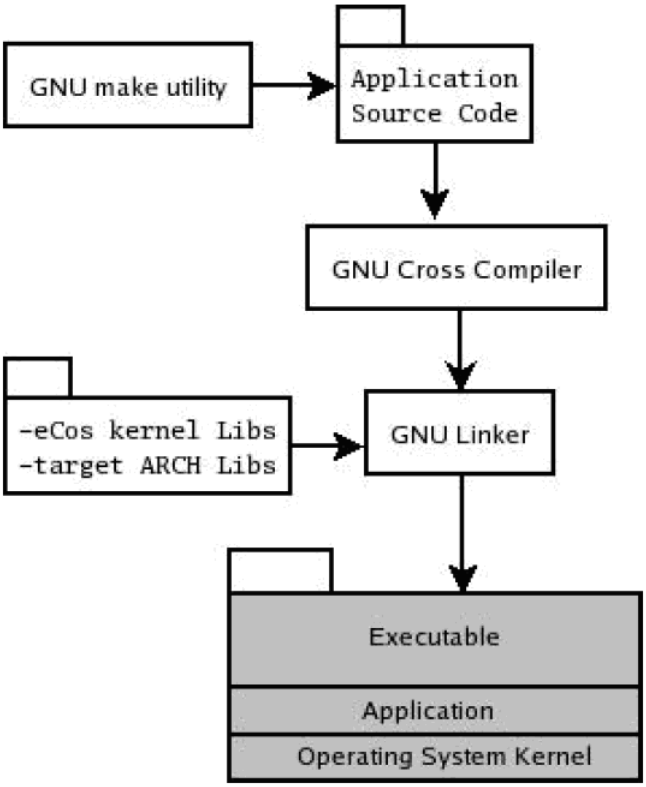

Figura 3.4: Fluxo do processo de desenvolvimento do eCos [49]

Uma aplicação no eCos não precisa ser escrita especificadamente usando a API do eCos, desde que o sistema operacional suporte as API's uITRON e POSIX, que também inclui completamente características do padrão C, como a biblioteca "math.h".

Assim como o Unix onde qualquer rotina é um arquivo, qualquer parte do eCos é um pacote ou módulo. Alocação de memória é controlada por um módulo separado e os *drivers* dos dispositivos serão organizados em módulos separados.

Vários pacotes são combinados e configurados através da ferramenta de configuração do eCos (*ConfigTool*) para ajustar as exigências da aplicação.

Em diversos sistemas operacionais o *kernel* pode oferecer funções de alocação de memória juntamente com os *drivers*, mas o eCos, a alocação de memória ou os dispositivos (*drivers*) são controlados por pacotes separados. Vários pacotes são combinados e configurados pela tecnologia de configuração do eCos para as exigências da

aplicação. É possível escrever uma aplicação de uma simples *thread*[9] sem utilizar qualquer funcionalidade do kernel, por exemplo o RedBoot. As aplicações estão baseadas em um laço central que verifica continuamente todos os dispositivos e a ação apropriada quando ocorre um evento. Nesse caso as funções existentes são executadas concorrentemente [29].

O desenvolvimento em arquitetura PC x86 foi realizado com opção de configuração em Floppy e RAM para os *templates* das arquiteturas testadas. Foi usada a opção para portar em RAM também, pois as demais arquiteturas não portariam para disco flexível.

O diretório de trabalho na emulação das arquiteturas foi a partir do "*/work*" para uma melhor divisão das implementações e subdiretórios seguintes com os nomes apropriados ao tipo de plataforma destino.

O desenvolvimento para arquitetura PC x86, por exemplo, com opção de portar para "Floppy" ou "RAM" e localização dos pacotes em "*/work/pc*", assim sucessivamente para as demais plataformas. A necessidade de configurar para memória RAM devido às demais arquiteturas como a ARM não puderem ser desenvolvidas para o disco flexível (Floppy).

3.3 - DESENVOLVIMENTO E VALIDAÇÃO

A validação do eCos foi através de uma simulação. Inicialmente foram instalados os pacotes necessários descritos nos apêndices A e B que correspondem às instalações em ambiente Windows e Linux, mas as atividades deste trabalho foram executadas em sistema operacional linux.

O funcionamento do eCos foi realizado através do seu ambiente gráfico do arquivo de configuração ou através da linha de comando. A implementação do eCos foi realizada na arquitetura x86 em conjunto com um simples programa em ANSI C para uma impressão de tela [50].

[9] *Thread* ou linha de execução: o processo pode ser dividido em uma ou mais tarefas e ser executado simultaneamente.

A validação do sistema do eCos foi através da utilização de dados de entrada, enquanto monitora os dados de saída com a criação de uma configuração mínima do eCos através de sua ferramenta. O sistema comporta programas em C e usa a tela para o resultado obtido.

Na seqüência da execução após gerar o arquivo de configuração foi criada a estrutura de diretórios, compilando os pacotes e posteriormente o programa exemplo foi agregado ao eCos para realizar a simulação.

Com o resultado da simulação foi criado um disco de inicialização que pode ser usado pelo mesmo computador para gerar a simulação com a inicialização do mesmo e o resultado da aplicação na tela.

No desenvolvimento das implementações com o eCos foram utilizados processadores Pentium e/ou AMD com arquitetura x86 contendo o sistema operacional linux com a distribuição Kurumin [51]. Foram instalados os pacotes do RTOS eCos baseado no arquivo exemplo do pacote [29], e suas bibliotecas relacionadas à arquitetura i386 que é considerada a plataforma de referência para análise dos resultados.

A Figura 3.5 apresenta a plataforma utilizada pelo ambiente eCos na manipulação da configuração e gerando os arquivos do RTOS com os requisitos do sistema. A configuração com os requisitos da aplicação é compilada resultando no arquivo que alimentará o ambiente que vai atuar no domínio da aplicação. Durante esse processo a plataforma utiliza recursos de depuração e simulação para validar as implementações.

Em paralelo, o usuário utiliza um compilador agregado ao eCos, juntamente com os parâmetros de compilação para criar o conjunto que irá para a plataforma destino. O arquivo gerado pode ser utilizado em memória ROM, flash ou um simples disquete de 3/2 polegadas.

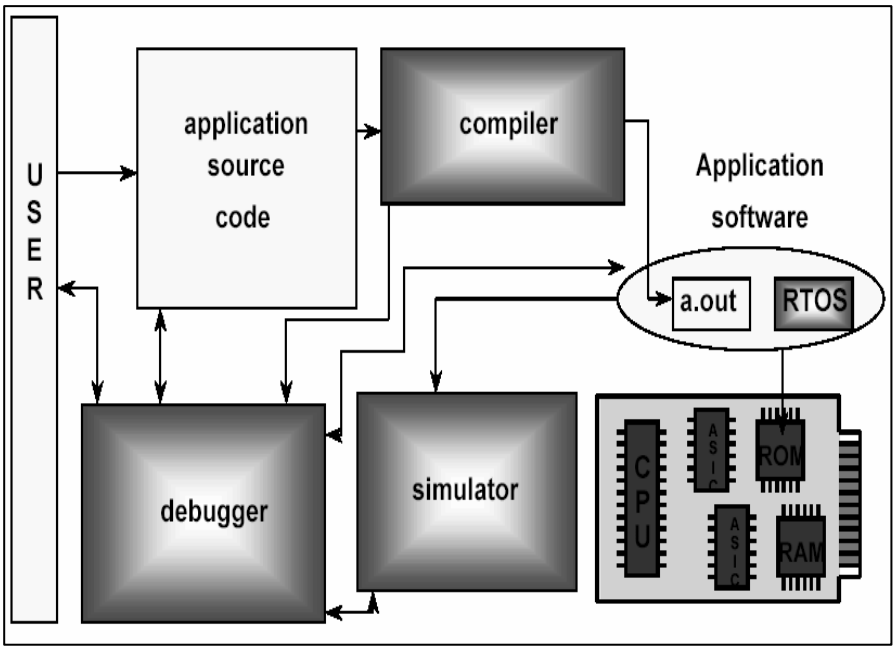

Figura 3.5: Modelo de desenvolvimento com o eCos [64]

A simulação feita para validar a implementação do eCos utiliza a ferramenta RedBoot (*bootstrap firmware*) [49] em um ambiente de inicialização para sistemas embarcados.

A configuração é realizada no ambiente de desenvolvimento host (computador PC comum com drive de disquete 3/2'') em conjunto com a ferramenta (*ConfigTool*) do eCos. Uma aplicação exemplo foi desenvolvida e enviada ao dispositivo que permitiu a sua inicialização. Neste caso foi o próprio computador (*host*) para demonstrar o processo conforme apresentado na Figura 3.6.

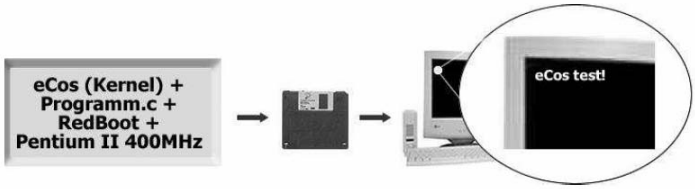

Figura 3.6: Simulação do eCos

Para construir aplicações para o eCos é necessário preparar o compilador, o depurador e seus utilitários que permitirão o processo de compilação dos programas a partir do código fonte, carregá-los e executá-los para o dispositivo destino.

A proposta de reconfigurar o sistema é apresentada conforme a Figura 3.7. Nesse caso o rSoC teria algumas pré-configurações estabelecidas e parametrizáveis que permitem alterações do contexto. A sugestão é que no momento de execução, com mudança de valores adquiridos do sistema, ele possa satisfazer outra configuração.

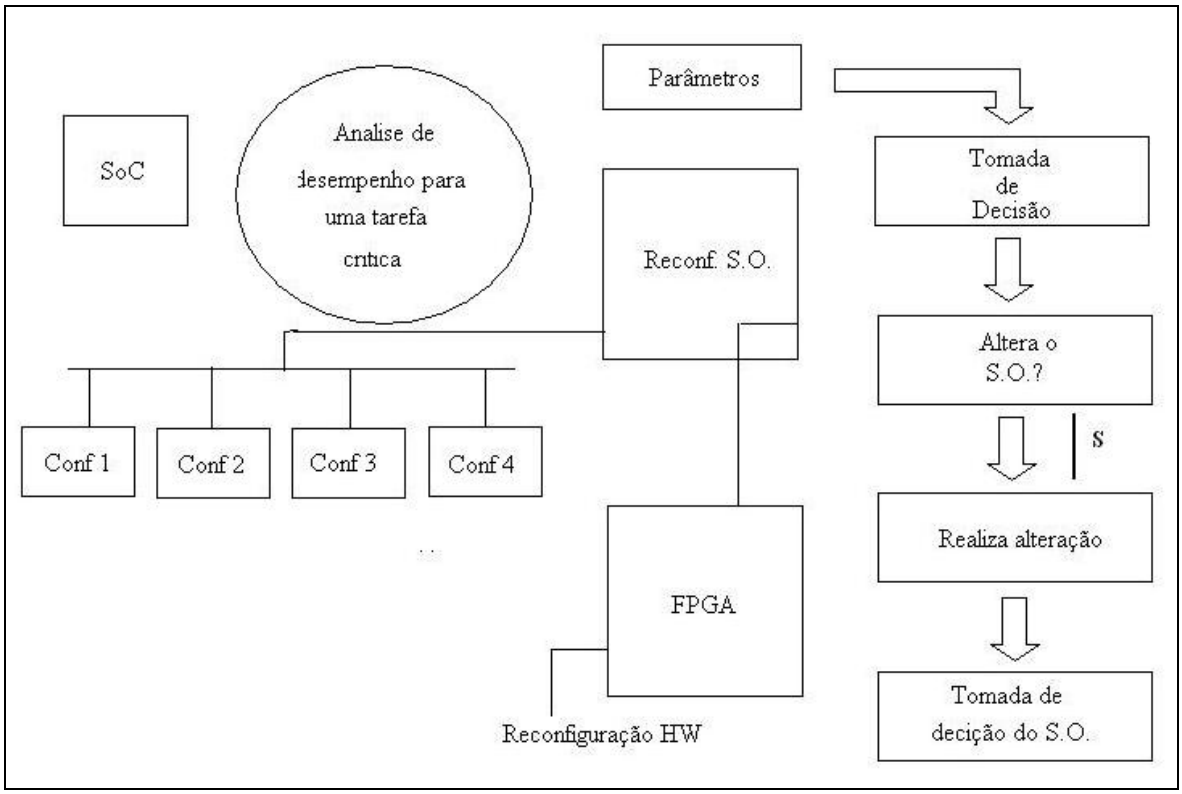

Figura 3.7: Configurabilidade do rSoC

O projeto do rSoC é um ambiente de projeto com a parte de Sistemas Digitais baseado em *Hardware/Software Co-design*. Este projeto tem como objetivo desenvolver um ambiente que suporte sistemas digitais complexos, considerando arquiteturas heterogêneas compostas por componentes de software que consistem em componentes programáveis de propósitos gerais e componentes de hardware que compreendem uma aplicação específica.

A aplicação do rSoC é para reconhecimento e rastreio de animais e vegetais, e mesmo não tendo a evolução das partes do projeto como sensor de imagem, processador e memória, o estudo em RTOS pode por meio de simulações em arquitetura x86, avaliar o compromisso de requisitos do projeto.

O software eCos é utilizado para desenvolvimento através de linguagem C no sistema operacional linux. A vantagem do eCos pelo projeto é justificada por requerer baixa capacidade de memória, o compartilhamento do endereço de memória, pela facilidade de obter soluções e ser um sistema de código livre.

De acordo com a implementação do protótipo apresentado em apêndice B, foi usado o RTOS eCos para a inicialização da ferramenta Redboot na plataforma destino por um disquete ou gerar o arquivo em memória *flash*. O trabalho verifica e valida todo o processo de desenvolvimento do ambiente eCos.

3.4 - RTOS PARA o rSoC

A proposta deste trabalho é apresentar o projeto de desenvolvimento com o sistema operacional em tempo real eCos como proposta de utilização no sistema em chip reconfigurável. O projeto do hardware com portas de comunicação de dados permite uma mudança ou no RTOS embarcado ou parte do RTOS com aplicações para as configurações desejadas. A utilização do eCos possibilitará o auxílio no desenvolvimento do projeto rSoC.

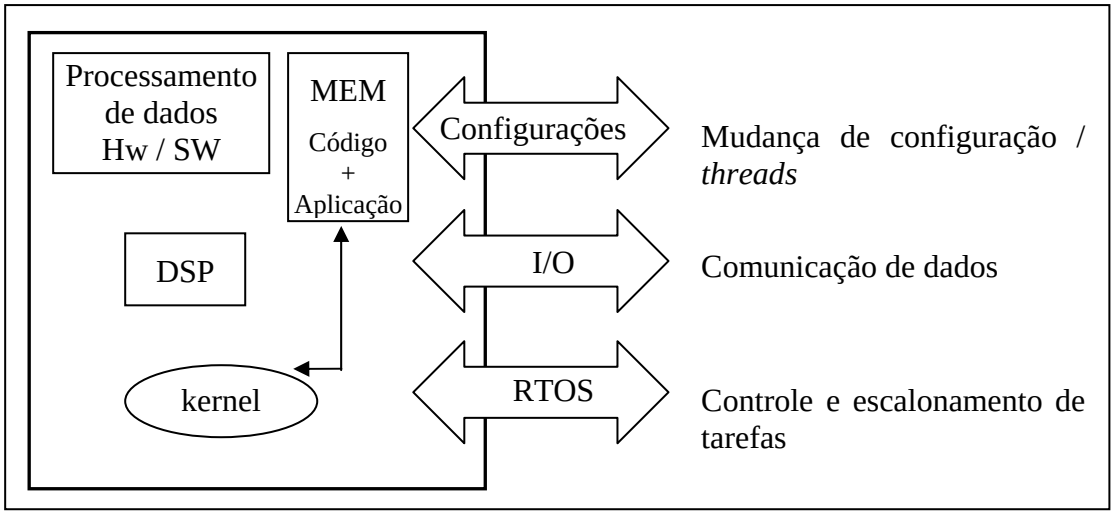

Figura 3.8: Projeto do rSoC com o eCos

O projeto do RTOS para o rSoC conforme apresentado na Figura 3.8 objetiva utilizar processador de 32 bits. Caso esta configuração seja proposta a sua adequação com a respectiva arquitetura deverá ser de acordo com as necessidades previstas pela aplicação.

A ferramenta de configuração do eCos oferece uma variedade de *templates* e pacotes que estão associados aos requisitos do sistema em chip. Estes *templates* estão relacionados aos diversos tipos de arquitetura já pré-estabelecidos para aperfeiçoar sua utilização. Por exemplo, o *template* da arquitetura ARM possui pacotes (código existente) para a sua implementação, o que possibilita uma facilidade na integração da aplicação com os componentes de hardware.

Outras funcionalidades apresentada pelo eCos são: otimização de recursos, inserção de rotinas específicas e formas de validação de testes. Estas funcionalidades contribuem ainda mais para o desenvolvimento da arquitetura do rSoC.

Os procedimentos para desenvolver o eCos a partir da arquitetura definida para o rSoC estão apresentadas na Figura 3.9 e relacionadas nos itens a seguir:

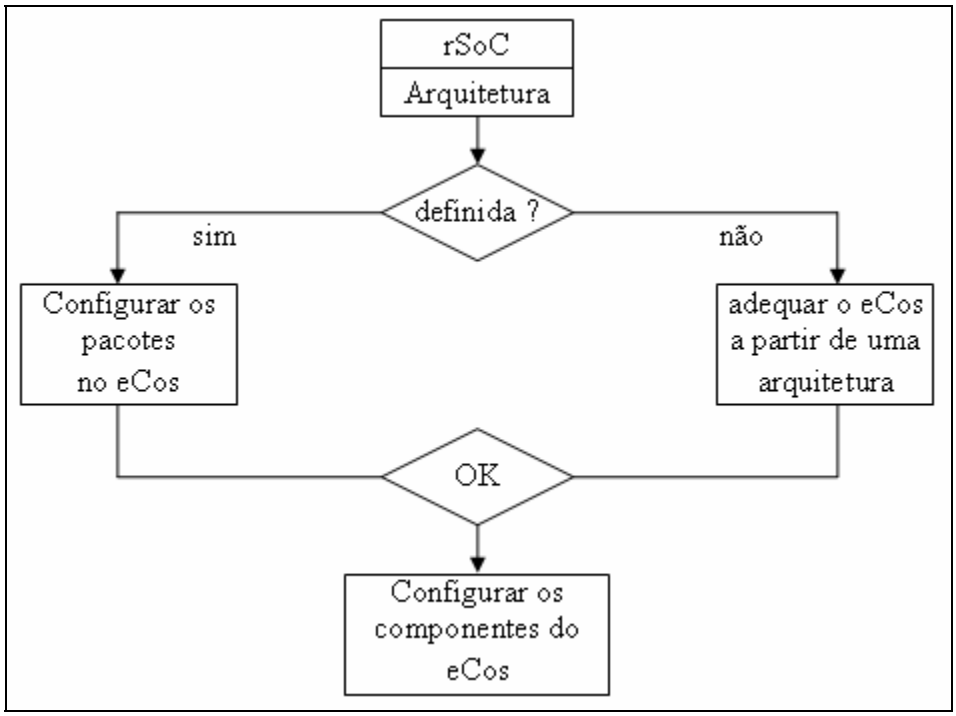

Figura 3.9: Projeto do rSoC com o eCos

1- Identificar a arquitetura nas bibliotecas dos pacotes existentes;

2- Caso exista a biblioteca da arquitetura específica para o desenvolvimento trabalhar na adequação e configuração do eCos para o rSoC;

3- Caso não exista a biblioteca para a arquitetura em questão deve-se aproximar a partir de uma arquitetura existente e adequar ao máximo das características da arquitetura;

4- Caso os pacotes estejam já relacionados à arquitetura, deve-se trabalhar na estrutura dos componentes habilitando ou não cada componente como anteriormente relatado na seção 2.2.2.

O diagrama anterior descreve uma construção básica de um arquivo de configuração do eCos. Os itens indicam um roteiro para o desenvolvimento baseado em uma arquitetura existente nos pacotes de referência (*templates*) ou se eventualmente não tenha suporte a arquitetura desejada é possível buscar um *template* próximo das definições escolhidas.

4 - RESULTADOS E DISCUSSÃO

Neste capítulo são apresentados e analisados os resultados obtidos durante a avaliação do sistema operacional em tempo real para o sistema em chip reconfigurável que será utilizado na arquitetura em desenvolvimento do rSoC para rastreio de animal e vegetal.

4.1 - ESTUDO DO RTOS

Os elementos principais de análise através de comparação com os demais RTOS's existentes para a definição do eCos foram as características dos sistemas operacionais embarcados que permitem o reuso do código e têm o código aberto.

Uma descrição desejada do rSoC foi realizada no decorrer das pesquisas e apontou como requisito mínimo de um sistema reconfigurável em ter um RTOS para garantir os itens de configurabilidade de sistema.

O RTOS eCos permitiu que fosse implementado em arquiteturas x86, um computador com sistema operacional linux, e portar para a mesma arquitetura x86 para sua validação (Figura 3.2, apresentada no capítulo 3). Através deste desenvolvimento ainda foram realizadas outras opções de configuração possibilitando direcionar suas utilidades para o projeto do rSoC podendo ser validado em outras arquiteturas com uma aplicação básica de mostrar uma simples impressão em tela.

O fato de não existir ainda uma certeza em relação à arquitetura do processador do rSoC foi necessário avaliar a possibilidade para o desenvolvimento para diferentes plataformas, utilizando a mesma configuração e aplicação a partir da implementação em x86, conforme desenvolvimento.

Os testes abordados nas atividades (apresentados no capítulo 3) mostraram as condições críticas de espaço. A questão de espaço é um requisito relevante em relação ao consumo de potência, que proporciona um compromisso entre o tipo de memória e a potência consumida para o desenvolvimento do hardware.

4.2 - ANÁLISE EM DIFERENTES SITUAÇÕES DE ARQUITETURAS

Uma característica importante na fase de análise foi verificar o comportamento de ambiente proposto em diferentes situações de arquiteturas. Foi analisado, para cada arquitetura, o tamanho dos arquivos de configuração, verificando também o tamanho do pacote compilado, já que cada configuração tem sua estrutura própria de arquivos.

A avaliação do RTOS foi feita através de implementações em diferentes situações (conforme estudos realizados no processo de definição da arquitetura do rSoC), o que possibilitou o desenvolvimento tanto em arquiteturas x86, quanto ARM, RISC entre outras. O resultado verificou a particularidade de cada um, relacionando requisitos básicos como ocupação de memória em disco, como tamanho do código compilado, o código binário e a apresentação dos passos nesse desenvolvimento, que estão apresentados no apêndice C.

A seguir na Tabela 4.1 é apresentado um arranjo comparativo com o tamanho dos arquivos gerados em cada etapa e arquiteturas correspondentes. As arquiteturas apresentam diferenciações em PC e PC_GIGA tendo como opções portar o resultado para disquete (*Floppy*) ou memória RAM na ferramenta de configuração do eCos (*ConfigTool*).

Tabela 4.1: Análise do eCos em diferentes arquiteturas

Arquitetura	Arquivo ecos.ecc (bytes)	Saída (a.out)	
		Compilado (bytes)	Imagem (bytes)
PC			
Floppy	433301	693591	55424
RAM	433265	679750	51488
PC_GIGA			
Floppy	433259	671626	55424
RAM	433173	693463	51520
ARM			
RAM	404761	810704	66624
Power PC			
RAM	434110	389089	86352

A coluna “Arquivo ecos.ecc” informa o tamanho do arquivo de configuração de cada arquitetura. O tamanho do arquivo de cada arquitetura é diferente pois cada *template* dessas arquiteturas manteve seus itens de configuração de acordo com a sugestão do eCos. A necessidade em adicionar ou em retirar funções ou itens de configuração faz com que o tamanho do arquivo seja proporcional a sua complexidade.

O resultado compilado e a imagem do programa binário também são pontos de análise. Ao compilar uma configuração, esta poderá se relacionar com um maior número de bibliotecas (dependência de pacotes) e resultar em um arquivo maior.

Na verdade, os *templates* sugerem uma configuração padrão com recursos ativos ou não. Isso permite que o usuário manipule as opções para que a configuração gere um *kernel* com maior limitação de recursos, porém com maior portabilidade.

4.3 - ANÁLISE DOS RESULTADOS OBTIDOS

Os testes foram realizados em computador PC com o sistema operacional linux Kurumin. A atividade foi executada com a utilização de um programa em C, com o resultado de impressão em tela. Para a arquitetura PC, foi carregado o *template* apropriado, para arquitetura i386, que é uma configuração mínima sugerida pelos pacotes do eCos, mas que garante o seu funcionamento.

A comparação do resultado pode ser analisada com o desenvolvimento seguinte, o PC_GIGA, que pode constatar a diferença em tamanho do programa código fonte, o resultado compilado do arquivo de configuração e suas bibliotecas e o código binário.

Alguns erros foram encontrados como a impossibilidade de compilar o “mipsisa32” por não inserir os parâmetros adequados. Nesse caso, foi possível encontrar outro dispositivo MIPS. O *template* corresponde ao dispositivo “atlas_mips32_4Kc” oferecendo os pacotes necessários para executar o eCos em um “Placa MIPS Atlas com processador Mips32 4Kc”. Outro item que se evidencia ao criar uma configuração é poder utilizar um *template* de outra arquitetura e fazer as modificações adequadas, desde que carregando o *template* correspondente, como foi o caso para o MIPS.

5 - CONCLUSÃO

Como contribuição o sistema operacional em tempo real eCos permite ajudar na especificação do projeto de desenvolvimento do rSoC para rastreio podendo simular suas aplicações e verificar que os resultados podem variar de acordo com os requisitos do RTOS a ser gerado.

A comparação entre os sistemas operacionais para o projeto permitiu sua escolha, como análise de um RTOS já existente e ferramentas que podem auxiliar em desenvolvimento. O sistema operacional embarcado escolhido ofereceu recursos para que os trabalhos em SoC sejam executados com adaptação e reuso do software voltado para as necessidades da aplicação.

Os atributos utilizados no estudo e o projeto do sistema operacional como escalonamento de tarefas, prioridade de execução, habilidade de manipular as funções, controle de preempção e o gerenciamento de processos são considerados vantagens para o sistema operacional em tempo real desejado.

Outra medida prevista para contornar ou superar a dificuldade de não ter uma arquitetura física do rSoC implementada foi justamente apresentar possibilidade de implementar em outras arquiteturas e analisar o comportamento.

A avaliação de funcionalidade do eCos foi efetuada numa plataforma x86 com uma estrutura que procurava representar a arquitetura básica do rSoC com áreas de memória reservadas para dispositivos específicos já que não existe uma arquitetura definida para o chip. A aplicação (realizada com sucesso em tempo real) foi mostrar um comando de impressão em tela da biblioteca “stdio.h” ao reiniciar o sistema, tendo a unidade de disco flexível como o dispositivo de entrada.

Durante o desenvolvimento deste trabalho foram realizadas comparações dos tamanhos dos códigos gerados para uma aplicação padrão a partir do eCos em plataformas distintas. Espera-se que os códigos gerados no rSoC tenham tamanhos ainda menores compatíveis com as aplicações desejadas.

5.1 - TRABALHOS FUTUROS

Em relação a atividades futuras, fazer uma comparação do objeto de estudo deste trabalho utilizando outros sistemas operacionais em tempo real existentes. Avaliar situações no mesmo domínio dos requisitos do projeto e que tenham o uso dos sistemas operacionais embarcados ou com outras finalidades específicas.

Outras propostas serão implementações em dispositivos programáveis e utilizar depuração de código a partir do eCos. O desenvolvimento do eCos para suporte ao ambiente do rSoC em dispositivo programável como FPGA podendo oferecer outros resultados, como descrição de projeto de hardware e validação do eCos junto a aplicação no dispositivo.

A validação do eCos para o rSoC proposto pode ser usada como recurso de depuração para ajudar no desenvolvimento de projeto. O projeto do rSoC encontra-se em desenvolvimento e as simulações e resultados deste trabalho proporcionam uma contribuição ao projeto de sistema em chip reconfigurável. Tais resultados deste trabalho contribuem para testes direcionados a softwares de aplicação prevendo a programação de aplicativos e simulações.

REFERÊNCIA BIBLIOGRÁFICA

[1] The International Technology Roadmap For Semiconductors. 2001 - Edition. Design. http://www.itrs.net/Links/2006Update/2006UpdateFinal.htm acessado em 12/05/2006.

[2] Costa, J. D., Implementação de um proprocessador RISC 16-bits CMOS num sistema em chip, dissertação de mestrado, Universidade de Brasília, 2004.

[3] Melo, M. D., Costa, J. C.; Jacobi, R. P. . Modelagem Funcional de um Processador RISC 16 Bits em Sistema em Chip para Aplicações Sem Fio. In: X Workshop Iberchip, 2004, Cartagena de Indias. Archivos del X Workshop IBERCHIP, 2004.

[4] Povoa, L. S.R., SoC para controle de Irrigação: Ambiente para o Desenvolvimento de Aplicações. Publicação Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 152p. 2004.

[5] Silva, A. L., Desenvolvimento de softwares para a comunicação de sistemas integrados em chip (SoC), aplicados a sensores para controle de irrigação. Projeto Final de Graduação, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 81p. 2005.

[6] Martins, A. J. O., Ambiente de Desenvolvimento de Softwares para um SoC: Desenvolvimento de um Simulador em SystemC. Projeto Final de Graduação, Publicação UnB.LabRedes.PFG.13/2005, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília , DF, 53p., 2005.

[7] Carro, L.; Wagner, F., Capítulo 2 das Jornadas de Atualização em Informática. In: XXII JAI 2003. (Org.). Sistemas Computacionais Embarcados. Campinas: Sociedade Brasileira de Computação, 2003, v. 1, 45-94.

[8] Nollet V., Coene P., Designing an Operating System for a Heterogeneous Reconfigurable SoC- IEEE Computer Society, IPDPS'03, 2003.

[9] Bergmann , N.., Williams J., Waldeck, P.: Egret: A Flexible Platform for Real-Time Reconfigurable Systems on Chip. Engineering of Reconfigurable Systems and Algorithms, 2003, 300-303.

[10] Bergmann, N.: Enabling Technologies for Reconfigurable System-on-Chip. Proceedings of the IEEE International Conference on Field-Programmable Technology, 2002, 360–363.

[11] Embrapa, http://www.embrapa.br/ acessado em 21/08/2006.

[12] Oxxen, Certificação Animal, http://www.oxxen.com.br/index.html acessado em 21/03/2006.

[13] Instituto Gênesis, http://www.institutogenesis.org.br/ acessado em 10/04/2006.

[14] Localiza Animal, http://www.localizaanimal.com.br acessado em 10/04/2006.

[15] GS1 Brasil, http://www.eanbrasil.org.br acessado em 10/04/2006.

[16] ABC - Associação Brasileira de Criadores, http://www.abccriadores.com.br/ acessado em 10/04/2006.

[17] Oyamada, M. S. ; Wagner, F. R. ; Carro, L.; Correa, E. F. ; Gervini, A. . Análise de Desempenho e Consumo de Potência na Comunicação Interprocessos em Software Embarcado. In: X IBERCHIP Workshop, Cartagena de Indias, 2004.

[18] Micheli, G. and Benini, L., Networks-on-Chip: a New Paradigm for Systemson-Chip Design. In: Proceedings of the Design, Automation, and Test in Europe Conference. Paris. IEEE Computer Society Press, 2002.

[19] Wolf, W., Computers as Components - Principles of Embedded Computing System Design. Morgan Kaufmann Publishers, San Francisco, CA. 2001.

[20] Gauthier, L., Yoo, S. e Jerraya, A.A., Automatic Generation and Targeting of Application Specific Operating Systems and Embedded Systems Software. DATE'01 – Design, Automation and Test in Europe, Munich, Alemanha. Proceedings, IEEE Computer Society Press, 2001.

[21] Böke, C., Combining Two Customization Approaches: Extending the Customization Tool TEReCS for Software Synthesis of Real-Time Execution Platforms. AES'2000 – Workshop on Architectures of Embedded Systems, Karlsruhe, Alemanha, Jan. 2000.

[22] Shandle, J. e Martin, G., Making Embedded Software Reusable for SoCs. EEDesign, Mar. 1, 2002.

[23] [Molen] MOLEN, http://ce.et.tudelft.nl/MOLEN/ acessado em 21/03/2006

[24] M. Bolado, J. Castillo, H. Posadas, P. Sánchez, E. Villar, C. Sánchez, P. Blasco, H. Fouren, Using Open Source Cores in Real Applications, XVIII Conference on Design of Circuits and Integrated Systems, DCIS 2003.

[25] Calazans, N. L. V., Moreno, E., Hessel, F., Rosa, V., Moraes, F. G., Carara, E., From VHDL Register Transfer Level to SystemC Transaction Level Modeling: a comparative case study. In: 16TH Symposium on Integrated Circuits And Systems Design, SBCCI´2003. Los Alamitos: IEEE Computer Society Press. São Paulo, Brazil, 2003, 355-360.

[26] Polpeta, F. V., Fröhlich, A. A., Junior, A. S. H., and D'Agostini, T. S., Portabilidade em Sistemas Operacionais Baseados em Componentes de Software. In First Brazilian Workshop on Operating System, pages 1466–1475, Salvador, Brazil, 2004.

[27] Tanenbaum, A. S., Modern Operating Systems. Prentice-Hall International, Inc. 1992.

[28] Intel, http://www.intel.com/portugues/ acessado em 10/04/2006.

[29] eCos, http://ecos.sourceware.org/ acessado em 15/02/2006.

[30] Mantenedor de pacotes do projeto GNU, http://www.cygnus-solutions.com acessado em 10/04/2006.

[31] Yoo, S., Jerraya, A.A., Introduction to Hardware Abstraction Layers for SoC. DATE'03 – Design, Automation and Test in Europe. Munich, Germany, Proceedings, IEEE Computer Society Press, 2003.

[32] Massa, Anthony J., Embedded Software Development with eCos - Prentice Hall – 2003.

[33] ITRON, http://www.itron.gr.jp acessado em 11/04/2006.

[34] Laplante, P. A., Real-Time Systems Design And Analysis, 3ª Edição, IEEE Press, 2004.

[35] Projeto GNU, http://www.gnu.org/ acessado em 11/06/2006.

[36] GNU GDB Debugger, http://www.gnu.org/software/gdb/gdb.htm acessado em 11/04/2006.

[37] Silva, J.L., Costa, K.A.P., Centro Universitário Eurípides de Marília – UNIVEM Programa de Pós-Graduação em Ciência da Computação Marília, S.P., Brazil. RtrASSoc–Adaptable Superscalar Reconfigurable Programmable System on Chip– The Embedded Operating System – EOS, 2003.

[38] Akgul, T., Kuacharoen, P., Mooney, V. J. and Madisetti, V. K., School of Electrical and Computer Engineering Georgia Institute of Technology. A Debugger RTOS for Embedded Systems.

[39] Tecnologias de Micro e Nanoeletrônica para Sistemas Integrados Inteligentes – NAMITEC, do Programa do Instituto do Milênio MCT/CNPq.

[40] J-Y. Mignolet, S. Vernalde, D. Verkest, R. Lauwereins, “Enabling hardware-software multitasking on a reconfigurable computing platform for networked portable multimedia appliances”, Proceedings of the International Conference on Engineering Reconfigurable Systems and Architecture, 116-122, Las Vegas, June 2002.

[41] XiRisc, http://xirisc.deis.unibo.it/ acessado em 03/06/2006.

[42] Lodi A., Toma M., Campi F., Cappelli A., Canegallo R., Guerrieri R.: A VLIW Processor With Reconfigurable Instruction Set for Embedded Applications. IEEE Journal of Solid-State Circuits 38 (2003) 1876 – 1886.

[43] Djaja, S., Chapman, G. H., Cheung, D. Y. H., and Audet, Y. "Implementation and Testing of Fault-Tolerant Photodiode-based Active Pixel Sensor (APS)," Proceedings of the 18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems 53 – 60, 2003.

[44] Soares, R. R. P., Sintetizador de Freqüências em Tecnologia CMOS para um Transceptor de RF a 900MHz em um Sistema em Chip. Dissertação de Mestrado, Departamento de Engenharia Elétrica, Universidade de Brasília. Brasília, DF, 2005.

[45] Pinto, J. S., Moura, G. S., Oliveira, L. e Marcondes, S. N. “O Desafio da Implantaçâo do Smart Tag: Etiquetas Inteligentes no Varejo”, VIII Semead - FEA – USP, 2006.

[46] Garfinkel, S., Adopting Fair Information Practices to Low Cost RFID Systems. In Ubiquitious Computing International Conference Privacy Workshop, September 2002.

[47] Gwyn, H., Food Safety: origin certification and traceability. XIII World Meat Congress. Belo Horizonte. 2000.

[48] Sene Jr I. G.; Barbosa T. M. G. de A.; Carvalho H. S.; da Rocha A. F.;Nascimento F. A. O. Monitoração da temperatura corporal baseada em uma rede de sensores sem fios. In: Congresso Brasileiro de Informática em Saúde (CBIS), 10., Florianópolis – SC. Anais., Florianópolis – SC:SBIS, 2006.

[49] RedBoot and eCos, An Application Specific Operating System Weqaar Janjua Arizona State University – Polytechnic Campus.

[50] eCos em PC x86, http://ant.comm.ccu.edu.tw/~spooky/ecos/ecos.html acessado em 11/02/2007.

[51] Guia do Hardware, http://www.guiadohardware.net/gdhpress/kurumin/ acessado em 11/02/2007.

[52] FreeRTOS.orgTM, http://www.freertos.org/ acessado em 14/02/2006.

[53] velOSity fornecido por Green Hills Software, Inc. http://www.ghs.com/html/velosity.html acessado em 03/06/2006

[54] RT-Linux fornecido por New Mexico Tech http://www.rtlinux.org acessado em 21/06/2006.

[55] eCosCentric, http://www.ecoscentric.com/ acessado em 25/02/2007.

[56] Red Hat, http://www.redhat.com/ acessado em 21/06/2006.

[57] RTAI, https://www.rtai.org/ acessado em 27/07/2006.

[58] RTLinux free, http://www.rtlinuxfree.com/ acessado em 15/05/2006.

[59] QNX Software Systems Europe http://www.qnx.com/products/os/qnxrtos.html acessado em 19/11/2006.

[60] Nut/OS, http://www.ethernut.de/en/software.html acessado em 12/01/2007.

[61] Windows CE, http://msdn2.microsoft.com/en-us/embedded/aa731407.aspx acessado em 12/01/2007.

[62] LynxOS, www.lynuxworks.com/ acessado em 12/01/2007.

[63] uCLinux, www.uclinux.org/ acessado em 02/04/2006.

[64] Keutzer, K., et.al. Lecture 26a: Software Environments for Embedded Systems Computer Science 252, Spring 2000.

[65] Sistema Operacional BSD, http://www.bsd.org/ acessado em 21/08/2006.

[66] FSF, Fundação de Software Livre, http://www.fsf.org/ acessado em 21/08/2006

[67] Linus Torvalds, http://en.wikipedia.org/wiki/Linus_Torvalds acessado em 11/07/2007.

[68] Sangiovanni-Vincentelli, A., Martin G.. Platform-based Design and Sofware Design Methodology for Embedded Systems. “IEEE Design and Test of Computers”, pp. 23-33, Novembro-Dezembro, 2001.

[69] Silberschatz, A.; Galvin, P. B. Operating system concepts. 6.ed. John Wiley & Sons, Inc., 2001.

APÊNDICES

A - INSTALAÇÃO DO ECOS (WINDOWS)

Esse apêndice apresenta a instalação do Cygwin e a instalação do eCos em ambiente Windows.

A.1 - CYGWIN

Acesse o site http://www.cygwin.com, e clique no link "*Install or update cygwin now!*" ou faça o download do arquivo de instalação. As instruções estão em http://ecos.sourceware.org/cygwin.html.

Salve o programa de instalação (setup.exe) em C:\temp, por exemplo.

Após terminar o download, execute o arquivo C:\temp\setup.exe.

A tela apresentará a seguir na Figura A.1 o início da instalação. O processo é bastante intuitivo e o usuário poderá retornar a instalação executando o mesmo arquivo.

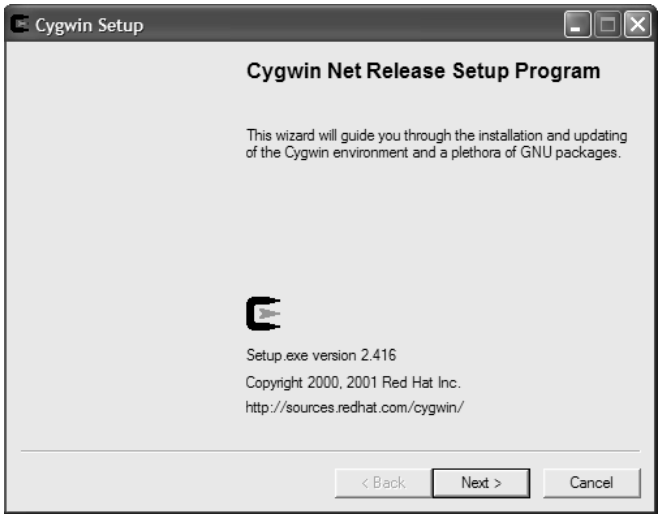

Figura A.1: Instalação do Cygwin

A próxima tela deve-se manter selecionado *Install from Internet*, e clicar em *Next*. Em seguida, escolher o diretório para instalação. Sugestão: C:\cygwin ou D:\cygwin. Na opção *Default Text File Type* deve-se marcar *Unix* e clicar em *Next*. O próximo passo é

determinar a localização dos arquivos temporários (C:\temp). Após a instalação esses arquivos devem ser excluídos.

Em seguida selecionar *Direct Connection* para determinar qual vai ser a fonte de download. Após isso na Figura A.2, deve-se escolher os pacotes a serem instalados, clicando na opção *view* para mostrar todas as opções e escolher os seguintes pacotes: gcc, make, sharutils, tcltk, wget.

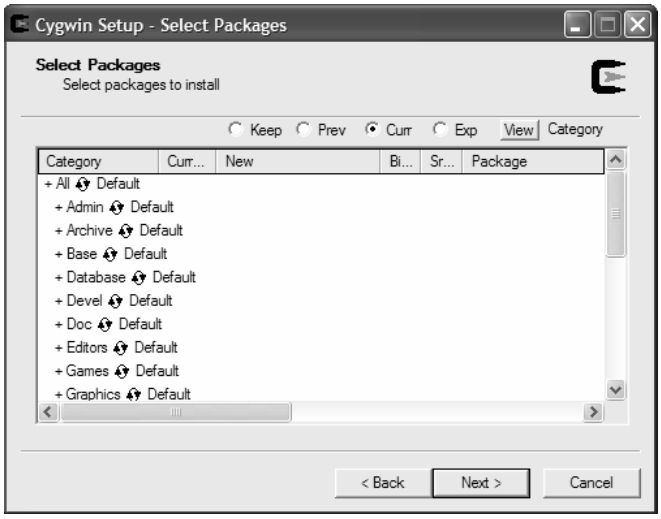

Figura A.2: Seleção de pacotes do Cygwin

A.2 - INSTALAÇÃO eCos NO CYGWIN

O Cygwin já instalado nessa etapa permite que a instalação do eCos possa ser iniciada.

No modo *shell* opção *bash* (linha de comando), com o seguinte comando para download: (Obs.: deve-se ter o pacote *wget* do *cygwin* instalado e verificar a restrição para baixar arquivo por FTP);

```
# wget --passive-ftp ftp://ecos.sourceware.org/pub/ecos/ecos-install.tcl
# sh ecos-install.tcl
```

A instalação é recomendada no diretório /opt/ecos, para ser acessada por qualquer usuário.

Pode ser usado também o pacote “ecos-2.0.cygwin.tar.bz2”, para descompactar usar:

```
# tar -jxvf ecos-2.0.cygwin.tar.bz2
```

Voltando a opção de usar o arquivo pego por FTP (wget), deve-se instalar o pacotte Tcl shell do cygwin, executar novamente o setup e escolher o pacote correspondente.

O pacote instalado de ferramenta de desenvolvimento i386-elf para a plataforma Intel x86 vem em três partes:

1. Coleção de compiladores GNU (GCC);
2. Depurador GNU (GDB);
3. Utilitários binários (assembler e linker GNU);

Essas ferramentas são usadas em conjunto (GNU: gcc, make, gcc, gdb).

B – INSTALAÇÃO E SIMULAÇÃO DO ECOS EM LINUX

A instalação do RTOS eCos pode foi feita primeiramente utilizando uma distribuição linux que foi testada em algumas versões como o Debian, o Red Hat e Kurumin sendo que, este último foi utilizado para os resultados obtidos.

As instalações em diferentes distribuições linux correspondem as versões do eCos a partir da 2.0 *pre-release*. Dependendo do caso, é preciso instalar outros pacotes que dependam do processo de instalação. Um desses pacotes é o TCL/TK e Xserver na máquina host. Esses pacotes estão disponíveis em qualquer distribuição do linux, assim como o carregador de inicialização e execução do sistema.

Outra alternativa e também considerada o modo mais prático para a instalação e desenvolvimento do eCos é utilizar a instalação por linha de comando:

wget –passive-ftp ftp://ecos.sourceware.org/pub/ecos/ecos-install.tcl

Esse processo faz o download do arquivo “ecos-install.tcl”, considerado uma rotina de instalação do eCos e os pacotes GNU. Basta executar a ferramenta de instalação pelo comando “sh ecosinstall.tcl” irá apresentar um lista de *mirrors* e depois selecionar as plataformas a serem instaladas, como por exemplo, a “i386_elf” mostrado na figura a seguir, necessário porém, certificar se os *PATHs* e o caminho do diretório de trabalho ($ECOS) estão coerentes.

B.1 - Figura da instalação do eCos

C – VALIDAÇÃO/TESTES (QUADRO COMPARATIVO)

C.1 - PC I386

O item apresenta o desenvolvimento para arquitetura PC x86 com opção de configuração em *Floppy* e RAM.

Comando para criar a configuração:

ecosconfig new pc

Comando para abrir o arquivo de configuração:

configtool ecos.ecc

A alteração das configurações é necessária na Figura C.1, mesmo com o “template pc”.

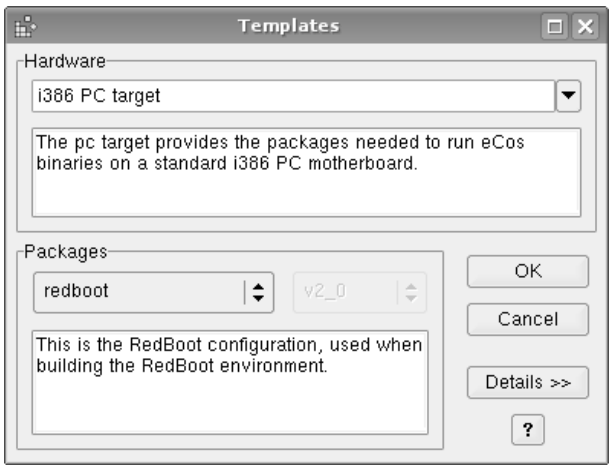

Figura C.1: Configuração de *template* no eCos

eCos HAL --> i386 Architecture--> i386 pc target --> startup type --> Floppy / RAM
eCos HAL --> i386 Architecture--> i386 pc target --> Default console channel --> 2

Após essa alteração, salvar o arquivo e sair da ferramenta de configuração.

Nessa etapa o arquivo de configuração *ecos.ecc* possui 434110 bytes.

ecosconfig check (verificação de inconsistência do eCos)

ecosconfig tree (gerando a estrutura de diretórios e o *makefile*)

make (compilando)

O programa em C é criado na mesma pasta do “ecos.ecc”, representado na Tabela C.1.

Tabela C.1: Programa em C para i386

```
#include <stdio.h>
 int main (void)
{
        printf(" Teste arquitetura i386 \n");
        return 0;
 }
```

Compilando o programa e fazendo a ligação com o eCos:

i386-elf-gcc -g -D_ECOS -I ./install/include/ teste.c -L ./install/lib/ -Ttarget.ld -nostdlib

O comando de compilação tem algumas opções padrões GCC como, por exemplo, o –g que habilita a depuração. O IBASE permite que os arquivos possam ser ligados à plataforma. Nessa etapa foi gerado um arquivo de saída: a.out

i386-elf-objcopy -O binary a.out (gerando o arquivo binário)

Os resultados foram:

- Arquivo de configuração “ecos.ecc” ficou com 433301 bytes;

- Com opção de Floppy e após compilação ficou com 693591 bytes programa compilado e acoplado à aplicação;

- O programa binário ficou com 55424 bytes;

- Com opção de RAM ficou com 6709750 bytes (compilado) e 51488 bytes (arquivo binário).

C.2 - PC GIGA I386

Apresenta o desenvolvimento para arquitetura PC x86 usando *template* para *Giga ethernet*, para os destinos *Floppy* e RAM. É utilizado para portar em RAM devido às demais arquiteturas não portarem para disco flexível, como na opção anterior.

ecosconfig new pc_giga

Abrir o arquivo de configuração:
configtool ecos.ecc

Alterar as configurações:
eCos HAL --> i386 Architecture--> i386 pc target --> startup type --> Floppy
eCos HAL --> i386 Architecture--> i386 pc target --> Default console channel --> 2

Após essa alteração, salvar o arquivo e sair da ferramenta de configuração.

ecosconfig check (verificação de inconsistência do eCos)
ecosconfig tree (gerando a estrutura de diretórios e o *makefile*)
make (compilando)

O programa em C é criado na mesma pasta do "ecos.ecc", representado na Tabela C.2.

Tabela C.2: Programa em C para i386 giga ethernet

```
#include <stdio.h>
 int main (void)
 {
        printf(" Teste arquitetura i386 pc GIGA Ethernet \n ");
        return 0;
 }
```

Compilando o programa e fazendo a ligação com o eCos:

i386-elf-gcc -g -D_ECOS -I ./install/include/ teste.c -L ./install/lib/ -Ttarget.ld -nostdlib

Tamanho do ecos.ecc:

Opcao RAM: 433173

Opcao Floppy: 433259

Arquivo de saída gerado: a.out

671626 (RAM)

693463 (floppy)

i386-elf-objcopy -O binary a.out

55424 bytes programa binário. (floppy)

51520 bytes (RAM)

434110 2007-05-29 11:21 ecos.ecc

Aparentemente o programa gerado a partir do template PC_GIGA foi menor que o *template* PC, devido a algumas configurações já carregadas e pré-selecionadas nos *templates*.

C.3 - ARM (ARM-ELF)

É utilizado o programa *configtool* com opção de alterar o *template* para "*ARM development board (INTEGRATOR) with ARM7TDMI*"

Verificar as configurações:

eCos HAL --> ARM Architecture--> ARM INTEGRATOR evaluation board --> startup type -->RAM (ROM /RAMROM)

ecosconfig check

ecosconfig tree

make

O programa em C é criado na mesma pasta do “ecos.ecc”, representado na Tabela C.3.

Tabela C.3: Programa em C para arm

```
#include <stdio.h>
 int main (void)
{
        printf(" Teste arquitetura para ARM \n ");
        return 0;
 }
```

#arm-elf-gcc -g -D_ECOS -I ./install/include/ teste.c -L ./install/lib/ -Ttarget.ld -nostdlib

A mensagem “teste.c:6:4: warning: no newline at end of file”, e verifica-se a pasta com o arquivo gerado “a.out”.

809864 2007-05-29 11:26 a.out
434110 2007-05-29 11:21 ecos.ecc

Existe várias plataformas de arquitetura ARM, desse modo foi escolhido o “ARM Evaluator 7T board”:

cd /work/arm

ecosconfig new e7t
(ecos.ecc = 404974)

O *template* corresponde ao dispositivo “e7t” oferecendo os pacotes necessários para executar o eCos em um “ARM Evaluator 7T board” (aka AEB-2).

Abrir o arquivo de configuração
configtool ecos.ecc

Verificar as configurações, não fazer alteração.

ecosconfig check
ecosconfig tree
make

Utilizar o programa em C apresentado na Tabela C.3

arm-elf-gcc -g -D_ECOS -I ./install/include/ teste.c -L ./install/lib/ -Ttarget.ld -nostdlib

Mensagem: “teste.c:6:4: warning: no newline at end of file” não sendo possível gerar programa binário, nesse caso não está visível o procedimento descrito na tabela de comparação.

C.4 - MIPS (MIPSISA32-ELF)

Utilização do *template* “atlas_mips32_4Kc”.

configtool

Alterado o *template* para “Mips Atlas board with Mips32 4 Kc processor”.

eCos HAL --> MIPS Architecture--> Atlas evaluation board --> startup type -->RAM

Tamanho do arquivo ecos.ecc = 408698.

Abrir o menu “build --> Template” e escolher: “MIPS Atlas board with Mips32 4Kc processor”

Salvar com o nome ecos.ecc e sair.

Verificar as configurações, não fazer alteração.

ecosconfig check
ecosconfig tree
make

O programa em C é criado na mesma pasta do “ecos.ecc”, representado na Tabela C.4.

Tabela C.4: Programa em C para mips

```
#include <stdio.h>
 int main (void)
{
        printf(" Teste arquitetura para MIPS \n ");
        return 0;
 }
```

mipsisa32-elf-gcc -g -D_ECOS -I ./install/include/ teste.c -L ./install/lib/ -Ttarget.ld -nostdlib

O tamanho do arquivo foi de 408627 bytes para o ecos.ecc.

C.5 - POWER PC (POWERPC-EABI)

ecosconfig new
ecosconfig new TARGET powerpc-eabi
    Unknown target TARGET
configtool
Aqui mudei o templates para “PowerPC simulator”
Resolvendo os conflitos - Abrir o arquivo de configuração

Verificar as configurações, não fazer alteração.

[eCos HAL --> PowerPC Architecture--> PSIM minimal simulator --> startup type -->RAM (como única opção)]

ecosconfig check
ecosconfig tree
make

O programa em C é criado na mesma pasta do “ecos.ecc”, representado na Tabela C.5.

Tabela C.5: Programa em C power pc

```
#include <stdio.h>
 int main (void)
{
        printf(" Teste arquitetura para power pc \n ");
        return 0;
 }
```

powerpc-eabi-gcc -g -D_ECOS -I ./install/include/ teste.c -L ./install/lib/ -Ttarget.ld -nostdlib
powerpc-eabi-objcopy -O binary a.out

86352 2007-05-29 10:57 a.out
389089 2007-05-29 10:49 ecos.ecc

/*
ecosconfig new

power pc
389089 2007-05-24 18:41 ecos.ecc
Target: psim

powerpc-eabi-gcc -g -D_ECOS -I ./install/include/ teste.c -L ./install/lib/ -Ttarget.ld –nostdlib

831048 2007-05-24 18:45 a.out (compilado)
86360 2007-05-24 18:46 a.out (binario)

C.6 - CRIAÇÃO DO BOOT PARA O FLOPPY

Esta etapa consiste em construir a imagem do RedBoot a partir da ferramenta de configuração através dos parâmetros configurados.

Depois que foi construído, deverá ter o arquivo binário 'redboot.bin' no diretório '$ECOS/ecos_config_install/bin'.

O utilitário “dd” do linux é utilizado para gravar a imagem do eCos para o floppy, através de linha de comando devendo ir para o diretório da imagem em “'$ECOS/ecos_config_install/bin”, executar o comando:

'dd conv=sync if=redboot.bin of=/dev/fd0'

Depois que terminar o processo, uma mensagem é mostrada indicando os registros de gravação na figura a seguir.

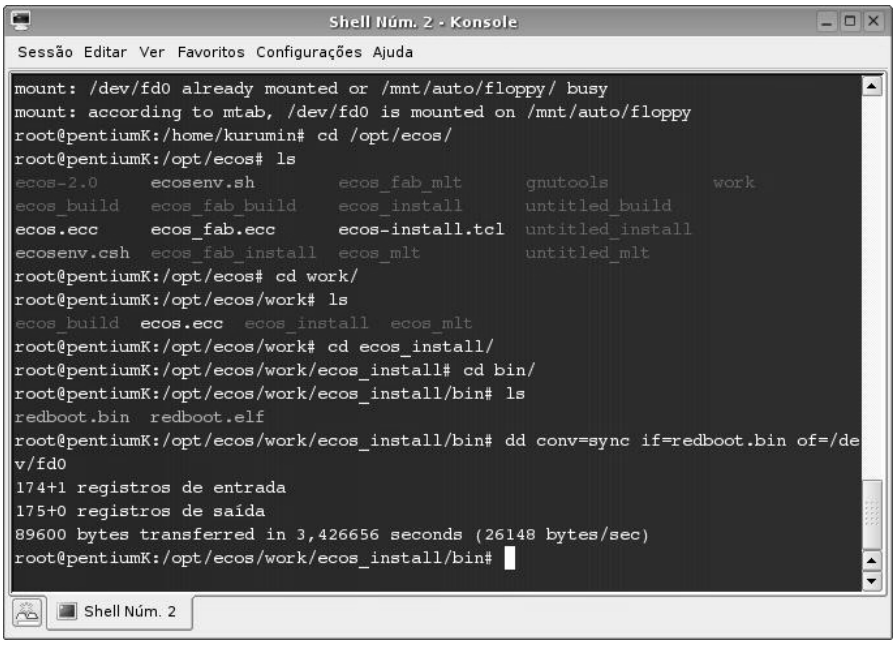

Figura C.2: Criação do disquete de boot com o eCos para PC i386

A seguir, faz-se a reinicialização com o disquete de boot, ele gera a mensagem na tela, certamente não ira executar mais nada além da aplicação com a saída do resultado “Test eCos!” na tela do computador.