Nestor Dias Pereira Neto

Desenvolvimento de um co-processador de vídeo em FPGA para integração com o Robot Operating System - ROS

Salvador

30 de abril 2020

Nestor Dias Pereira Neto

Desenvolvimento de um co-processador de vídeo em FPGA para integração com o Robot Operating System - ROS

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós Graduação em Engenharia Elétrica da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica.

Universidade Federal da Bahia - UFBA

Escola Politécnica

Programa de Pós-Graduação em Engenharia Elétrica

Orientador: Wagner Oliveira

Coorientador: Paulo César

Salvador

30 de abril 2020

Nestor Dias Pereira Neto
Desenvolvimento de um co-processador de vídeo em FPGA para integração com o Robot Operating System - ROS/ Nestor Dias Pereira Neto. – Salvador, 30 de abril 2020-
0p. : il. (algumas color.) ; 30 cm.

Orientador: Wagner Oliveira

Dissertação (Mestrado) – Universidade Federal da Bahia - UFBA
Escola Politécnica
Programa de Pós-Graduação em Engenharia Elétrica, 30 de abril 2020.

1. Palavra-chave1. 2. Palavra-chave2. 2. Palavra-chave3. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

Nestor Dias Pereira Neto

Desenvolvimento de um co-processador de vídeo em FPGA para integração com o Robot Operating System - ROS

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós Graduação em Engenharia Elétrica da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Engenharia Elétrica.

Trabalho aprovado. Salvador, 24 de novembro de 2012:

Wagner Oliveira
Orientador

Professor
Convidado 1

Professor
Convidado 2

Salvador
30 de abril 2020

Resumo

Segundo a **??**, 3.1-3.2), o resumo deve ressaltar o objetivo, o método, os resultados e as conclusões do documento. A ordem e a extensão destes itens dependem do tipo de resumo (informativo ou indicativo) e do tratamento que cada item recebe no documento original. O resumo deve ser precedido da referência do documento, com exceção do resumo inserido no próprio documento. (. . .) As palavras-chave devem figurar logo abaixo do resumo, antecedidas da expressão Palavras-chave:, separadas entre si por ponto e finalizadas também por ponto.

Palavras-chave: latex. abntex. editoração de texto.

Abstract

oihbqptipbõq4tnpot4photnj4yojnj4ynojp

Keywords: latex. abntex. text editoration.

Sumário

1 Introdução

Agrupar inúmeros “blocos"de softwares usados em robótica, fornecer drivers para hardwares específicos (sensores e atuadores), gerenciar troca de mensagens entre os nós que fazem parte do sistema, são as função do ROS. Essas características fezem com que o ROS seja reconhecido com um pseudo sistema operacional (PYO et al., 2017). Dessa maneira o ROS se tornou muito ágil no desenvolvimento de novas aplicações para robótica. Usando nós já desenvolvidos e testados por outros desenvolvedores podemos criar novos sistemas completos apenas gerenciando esses nós na rede interna do ROS. Essa abordagem fez com que o número de pacotes para o ROS cresça em uma taxa muito rápida, desde o ano de seu lançamento, 2007, até 2012 o ROS aumentou de 1 para 3699 pacotes (YAMASHINA et al., 2005).

Com essa distribuição de tarefas através de vários nós podemos criar sistemas cada vez mais complexos, apenas inserindo novos nós na rede ROS, essa rede é gerenciada pelo ROS Master, que é apensas mais um nó do sistema, mas com a função de ser um servidor de nome e serviços para o restante dos nós. Ele identifica os nós na rede, assim todos os nós podem se comunicar com os outros através de conexões peer-to-peer, Figura 1. Para desenvolver novas aplicações para integrarem o crescente grupo de pacotes ROS, o desenvolvedor deve respeitar os protocolos de comunicação da rede, as bibliotecas do ROS facilitam a vida do desenvolvedor, por já fornecer funções prontas para o desenvolvimento de novos códigos compatíveis e que possam se registrar na rede. Detalhes dos protocolos e interno podem ser visto em (ROS, 2011a), (ROS, 2018) e (ROS, 2011b).

Por se tratar de um hardware configurável o FPGA é ideal para processamento digitais de sinais, e segundo Meyer-Baese (2007) "os Field programmable gate array - FPGAs estão próximos a revolucionar o processamento digital de sinais, assim como os DSPs fizeram algumas décadas atrás". O potencial que os FPGAs possuem para melhorar o desempenho de sistemas que utilizam processamento digitais de sinal é conhecido já algum tempo, as possibilidades de paralelismo, criação de estruturas de DSP dedicadas à aplicação são recursos muito interessantes que a possibilidade do hardware configurado oferecem, mas em contra partida as facilidade de desenvolvimento encontradas em aplicações que fazem uso de softwares não são encontradas nas mesmas proporções no mundo do hardware, sendo assim:

- **Como estabelecer a comunicação entre o ROS e um sistema de processamento de vídeo embarcado em um FPGA?**

  Esse problema inicial nos leva naturalmente a outro:

- **Stabelecendo a comunicação entre o FPGA e o ROS a execução do processamento de vídeo de forma paralela, embarcada em um FPGA, pode melhorar a performance do sistema?**

1.1 Justificativa

Nos últimos anos novas técnicas para construção de robôs tem sido bastante estudadas, em especial uma área que tem sido bastante explorada é a robótica móvel. A principal características que tem sido buscada é cada vez fornecer mais autonomia ao sistemas robóticos o que vem tornado cada vez seus softwares mais complexos, o que aumenta a necessidade do uso de processadores cada vez mais poderosos, que consomem mais energia. Entretanto a busca por mais autonomia, diz respeito também às baterias, que são as fontes de energia da maioria dos robôs móveis, o que provoca uma verdadeira briga entre poder de processamento e baixo consumo.

Sendo assim, o FPGA pode ser uma ótima alternativa para solucionar os problemas de aumento do poder de processamento em conjunto com baixo consumo de energia. Meyer-Baese (2007) descreve algumas vantagens dos FPGAs modernos para uso em processamento digitais de sinais, como as cadeias de fast-carry usadas para implementar MACs de alta velocidade e o paralelismo tipicamente encontrado em dedign implementados em FPGA. Por essas características o FPGA necessita de frequências menores de trabalho para alcançar desempenho equivalente ou superior às soluções baseadas em processadores, tornando a dissipação de energia menor.

Foi encontrado até o memento uma única pesquisa que relaciona o ROS e FPGA para processamento de vídeo, nesse ponto a proposta deste trabalho difere da solução encontrado. No trabalho de Yamashina et al. (2005), são demonstradas três técnica para realizar a conexão entre o FPGA e o ROS, que se diferem da proposta por essa pesquisa. A ideia é utilizar um soft-core processor, que é um processador descrito em linguagem HDL embarcado no FPGA (CHU, 2012), esse processador ficará responsável por estabelecer a comunicação entre a rede TCP/IP e o hardware configurado no FPGA.

O FPGA que será utilizado para o desenvolvimento da pesquisa é o Cyclone IV da Intel. A Intel fornece o Nios II um soft-core porocessor para ser utilizado em conjunto com os seus FPGAs. O Nios II é disponibilizado em duas versões, a fast: que foi projetado para alta performance e a economy: projetado para ocupar um menor espaço dentro do FPGA(CHU, 2012).

A versão do Nios II econômica foi escolhida por não necessitar de uma licença adicional para seu uso, uma alternativa para RTOS é o FreeRTOS, que é um sistema operacional de tempo real de código aberto(BARRY, 2016b) e para stack TCP/IP a Lightweight TCP/IP stack - LwIP, que é uma versão do pacote de protocolos TCP/IP

de código aberto para ser usado em sistemas embarcado(NONGNU, 2018). Todas as ferramentas de softwares necessárias para desenvolver o projeto são de uso livre, o objetivo é fazer uso de o maior número de ferramentas sem custos adicionais, como licenças e softwares pagos.

O tempo de desenvolvimento de projetos em FPGA é maior em relação a projetos puramente de software, por isso, a pesquisa busca ao final do projeto produzir um sistema genérico que possa ser usado em outras aplicações com poucas ou até mesmo nenhuma alteração se tornando uma alternativa para integrar o ROS a um FPGA, de forma simples e de baixo custo, possibilitando outras aplicações desta solução.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver uma solução para estabelecer comunicação entre *Field Programmable Gate Array - FPGA*, configurado como um co-processador de vídeo e o *Robot Operating System - ROS* avaliando o impacto desta aplicação ao sistema.

1.2.2 Objetivos Específicos

- Estudar os assuntos relevantes ao projeto: Verilog HDL, RTOS, Nios II, TCO/IP Stack, ROS;
- Conhecer com detalhes os protocolos da rede TCP/IP usada para comunicação interna dos nós e serviços ROS;
- Desenvolver plataforma com Nios II como base para o andamento do projeto;
- Implementa um sistema operacional de tempo real - RTOS na plataforma base;
- Estabelecer comunicação entre o ROS e o sistema Nios II (embarcador no FPGA) através da tecnologia Gigabit Ethernet;
- Testar aplicações de processamento de vídeo em hardware em conjunto com ROS;
- Avaliar a performance com a inclusão do FPGA ao sistema.

1.3 Organização

No primeiro caítulo...

Parte I

Referenciais teóricos

2 Field Programmable Gate Array - FPGA

2.1 Cyclone IV - Intel

2.2 Soft processor NiosII

2.3 Kit de desenvolvimento DE2-115

3 Robot Operating System - ROS

3.1 Sistema multiagentes

4 FreeRTOS

O FreeRTOS é um sistema de tempo real de grande sucesso desde de seu início, é suportado por mais de 35 architectures e "foi baixado uma vez a cada 175 secundos no ano de 2018" (**??**). O FreeRTOS é um projeto de *free* e de código aberto disponível sobe licença MIT. O FreeRTOS foi a escolha para esse trabalho por não possuir restrições ao seu uso e também por sua popularidade, o que facilita a busca por informações de seu uso uma tarefa menos árdua.

4.1 Entendendo o FreeRTOS

FreeRTOS é um kernel de um sistema operacional de tempo real para sistemas embarcados, permite que aplicações sejam organizadas como coleções de independentes threads mantendo os requisitos de tempo real do sistema. Foi desesenvolvido em 2003, inicialmente por Richard Barry e posteriormente mantida pela *Real Time Engineers Ltd.*, em 2017 o projeto FreeRTOS passou aser adminitrado *Amazon Web Services*.

O kernel básico é formado por

4.2 Gerenciamento de Memória

4.3 Tasks

4.4 Interrupções

4.5 FreeRTOS+TCP

«««< HEAD

4.5.1 Estrutura de arquivos

4.5.2 Adicionar os arquivos TCP/IP ao projeto FreeRTOS

======= FreeRTOS+TCP é um TCP/IP *stack* de código aberto para o FreeRTOS, ou seja, é um extenção do do FreeRTOS que fornece uma interface progamação socket para sistemas embarcados.

4.5.3 Orcanização do código fonte

FreeRTOS-Plus-TCP [Contains the source files that implement the TCP/IP stack] | +-include [Contains the header files for the TCP/IP stack] | +-portable | +-Compiler | +-Compiler$_x$[$Containsstructurepackingheaderfilesfor Compiler_x$]|+−$Compiler_y$[$Containsstructurepackin$ −$Compiler_z$[$Containsstructurepackingheaderfilesfor Compiler_z$]|+−$BufferManagement$[$Sourcefiles$ −$NetworkInterface$+−MCU_x[$Containsanetworkdriverforthe MCU_x familyofmicrocontrollers$]+ −$MCU_y$[$Containsanetworkdriverforthe MCU_y familyofmicrocontrollers$]+−$MCU_z$[$Containsanetwor$

The FreeRTOS+TCP Directory Structure »»»> 28dbe8d2c9be3feb73c102331b38736ec41f54ab

Parte II

Desenvolvimento

5 Hardware

6 Software

Parte III

Resultados

7 estudos funturos

7.1 Pellentesque sit amet pede ac sem eleifend consectetuer

8 Conclusão

Capítulo conclusão