PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA

ARQUITETURAS RECONFIGURÁVEIS DE SISTEMAS DIGITAIS: Um Exemplo de Implementação

(O título mais geral, que foi encaminhado com o nome do avaliador, parece mais adequado, pois mais geral. Depois se pode mudar. Por enquanto deixa assim.)

Prof. Dr. Ney Laert Vilar Calazans
Orientador

Marcelo Sarmento

Proposta de trabalho de
conclusão do curso de
bacharelado em informática

Porto Alegre, 30 de Agosto de 2000

ARQUITETURAS AUTO-RECONFIGURÁVEIS EM SISTEMAS DIGITAIS

Marcelo Sarmento

Sumário

Lista de Figuras

1 Introdução

Durante as últimas décadas, o projeto de componentes digitais tem se tornado uma peça chave para a construção de sistemas computacionais. Neste período, muitos projetistas têm encontrado um problema freqüente, que consiste em estabelecer a relação correta entre generalidade e velocidade um componente pode ser construído visando versatilidade, possibilitando a execução de diferentes aplicações , ou visando desempenho, limitando o dispositivo a um numero menor de tarefas, mas com maior velocidade. Qual a relação entre generalidade e versatilidade acima? Se são a mesma coisa, usa um termo só!

Microprocessadores como o Intel Pentium [INT99] ou o Motorola Power PC [MOT98] são exemplos de componentes de propósito geral. Eles possuem um conjunto de instruções de programação codificadas, cuja combinação pode levar à execução (de virtualmente qualquer operação matemática ou algoritmo concebível). No lugar do texto entre parênteses, ficaria melhor: qualquer tarefa computável, segunda o uso de modelos teóricos tais como a máquina de Turing.

Por outro lado, componentes dedicados, geralmente conhecidos como ASICs (do inglês, Application Specific Integrated Circuits) ou Circuitos Integrados Específicos para uma dada Aplicação, são construídos visando a funcionalidade específica de uma determinada tarefa. Um chip de aceleração gráfica para a plataforma PC, por exemplo, pode desenhar linhas ou executar movimentação de objetos na tela 10 a 100 vezes mais rápido do que um microprocessador executando instruções genéricas.

Apesar do custo mais baixo de produção (ao contrário! Cuidado quando falar de custo! Custo de produção está relacionado ao volume de comercialização mais que à complexidade de projeto do circuito!) e da maior velocidade, a utilização dos ASICs fica restrita apenas à resolução ou operacionalização do problema para o qual eles foram desenvolvidos, visto que seu hardware é tradicionalmente fixo. Uma pequena modificação dos requisitos ou a inclusão de uma nova necessidade requer o desenvolvimento de um novo componente.

Contínuos avanços na tecnologia de desenvolvimento de circuitos integrados possibilitaram uma terceira opção, agregando a funcionalidade da reconfiguração característica dos sistemas que utilizam microprocessadores com o desempenho dos ASICs. Field –Programmable Gate Arrays, ou FPGAs, consistem em uma matriz de elementos agrupados em blocos lógicos configuráveis, que podem ser interconectados de um modo genérico por barramentos de conexões também configuráveis. Embora este sistema seja estruturalmente semelhante a uma PAL (Programmable Array Logic), as interconexões entre os elementos são implementadas por blocos de chaves programáveis pelo usuário. Em uma PAL, estas conexões são determinadas durante o período de

fabricação. (Depende da PAL, não? A maior diferença esta de fato na flexibilidade da interconexão e na densidade de integração, ou seja, complexidade)

Os FPGAs foram introduzidos no mercado em 1985 (de onde veio esta data? Acho que foi antes, coloca uma referência. Acho que a invenção foi em 1981 ou 1982, e a versão comercial é que veio em 1985) pela Xilinx Inc. Desde então, grande variedade de FPGAs foram desenvolvidos por outras companhias e tem sido amplamente utilizados, principalmente para desenvolvimento de produtos relacionados a áreas de telecomunicações e redes, em função do reduzido time-to-market (período entre o desenvolvimento do produto e sua inserção no mercado) que estes dispositivos oferecem.

Sistemas computacionais podem fazer uso da capacidade de reconfiguração dos FPGAs de várias maneiras. Uma das técnicas que estão sendo pesquisadas atualmente é a utilização de FPGAs para construção de dispositivos de hardware cuja funcionalidade pode ser alterada durante o uso do sistema. Esses dispositivos, conhecidos como ASP (do inglês, Application Specific Processor) (o conceito está errado, um ASP é um conceito muito mais geral) [ADA97], em conjunto com um processador de propósito geral (ou GPP, do inglês General Purpose Processsor) e uma estrutura de memória compartilhada (opcional), constituem uma Arquitetura Reconfigurável.

Neste contexto, um único dispositivo de hardware (FPGA), pode executar uma seqüência de diferentes tarefas através da reconfiguração sob demanda dos seus blocos lógicos: o equivalente em hardware da execução de um programa, sua remoção da memória do computador e a execução de outro. Uma consideração a respeito da utilização deste modelo é feita em [MIT94]: a especialização destas arquiteturas reconfiguráveis tem melhor desempenho a nível de desempenho (em português, o termo desempenho tem exatamente o mesmo significado, prefira desempenho a performance. A frase melhor desempenho a nível de performance não faz qualquer sentido! Substitua por melhor desempenho) que o seu equivalente em microprocessadores convencionais ou workstations (não misture alhos com bugalhos, microprocessador é dispositivo, workstation é computador).

2 Objetivos

Segundo Villasenor e Smith [VIL97], “Uma arquitetura de hardware que tem a possibilidade de reconfigurar a si mesma dinamicamente à medida em que uma tarefa é executada, refinando a sua própria programação para obter um melhor desempenho é a forma mais desafiadora e potencialmente mais poderosa de um sistema computacional configurável”.

O objetivo deste trabalho é implementar uma aplicação que exemplifique a utilização de uma Arquitetura Auto-Reconfigurável, verificando as vantagens e desvantagens em relação aos sistemas convencionais.

Dado um determinado problema, pretende-se construir o algoritmo para a resolução desse problema, dividir o algoritmo em etapas não simultâneas (quando possível), e implementar cada uma destas etapas em seu equivalente em hardware.

Na Figura 1 (Nomes de Figura Tabela, Seção, etc, tudo com maiúscula. Troquei aqui, troca no resto do documento), é apresentado o formato convencional para o mapeamento do algoritmo em hardware. Utilizando uma linguagem de descrição de hardware, como HDL ou VHDL e sua posterior síntese, é possível programar um dispositivo FPGA para executar a função desejada. Neste modelo, o hardware executa o mesmo algoritmo até que o agente configurador seja acionado novamente. Como um FPGA tem um número limitado de blocos configuráveis, algoritmos mais complexos necessitam dispositivos maiores (e, por conseqüência, de custo mais elevados) ou múltiplos dispositivos.

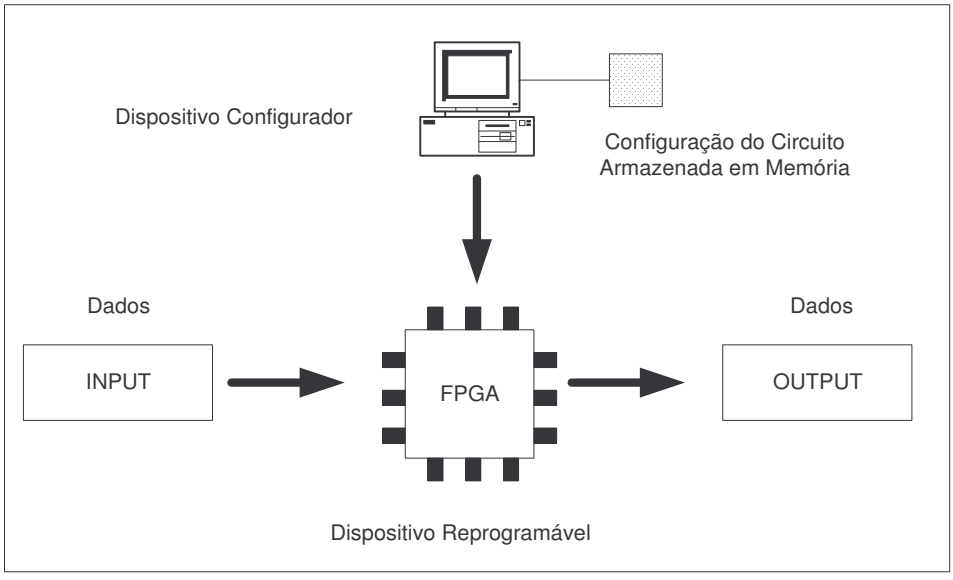

Figura 1 – Modelo convencional para programação de um FPGA

Na Figura 2, é apresentada a proposta de uma arquitetura auto-reconfigurável destinada à resolução do mesmo problema. Neste caso, o algoritmo é “quebrado” em partes

menores, sintetizado, e cada uma das suas partes é armazenada em uma memória de configuração.

O agente configurador faz a programação inicial, e a partir deste momento o próprio sistema determina qual o próximo módulo a ser carregado, seguindo a seqüência do algoritmo ou de acordo com a demanda dos dados recebidos.

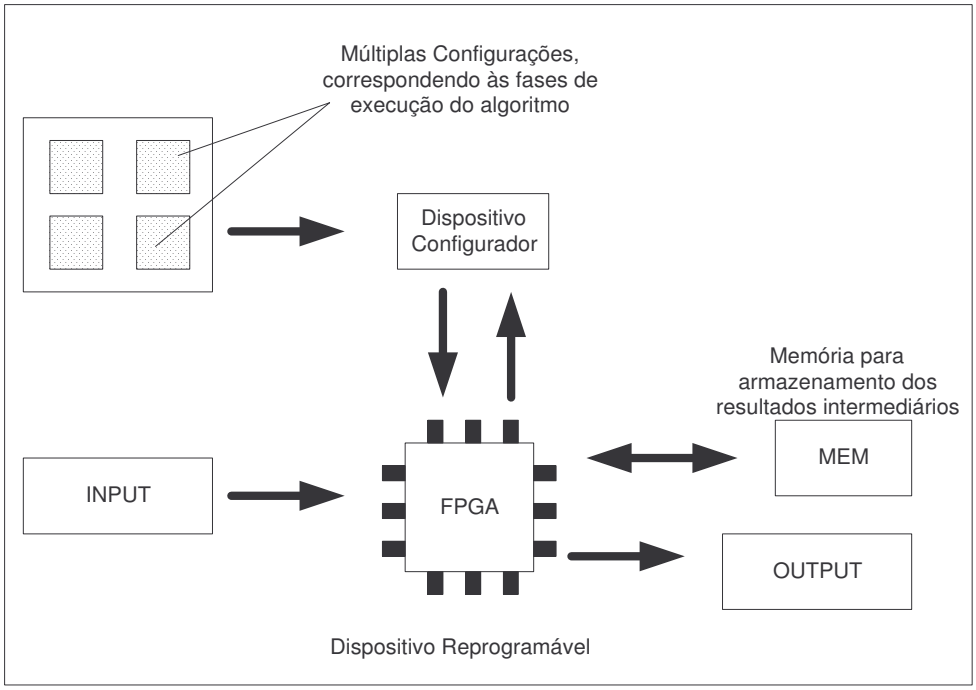

Figura 2 - Arquitetura Auto-Reconfigurável Proposta

Estas arquiteturas também são descritas como reconfiguráveis dinamicamente, segundo a classificação de Sanchez [SAN99]. Para uma implementação, podem ser usados tanto dispositivos programáveis parcialmente como a alocação de um mesmo dispositivo com configurações distintas em instantes de tempo diferentes.

Nessa proposta, não será prioritária a utilização de técnicas de reconfiguração parcial. Todo o projeto tem como premissa a reconfiguração total do dispositivo sempre que seja necessária a troca da função que ele deve executar. (Faltou uma pequena justificativa, afinal porque recusar uma arquitetura mais flexível em benefício de uma menos flexível? Diga que o que se está fazendo é uma hipótese simplificadora para viabilizar a realização do trabalho no prazo disponível, afinal, auto-reconfiguração já é bastante complicado, parcial poderia tornar-se inviável.)

3 Arquitetura do Sistema

Visando atingir os objetivos citados, o sistema a ser desenvolvido baseia-se em um computador hospedeiro (em português, já é termo consagrado) para ao desenvolvimento e síntese das imagens de hardware e de uma plataforma de prototipação que contenha o dispositivo reconfigurável (FPGA) e recursos de memória e interface com o usuário. A Figura 3 mostra a estrutura geral do sistema proposto.

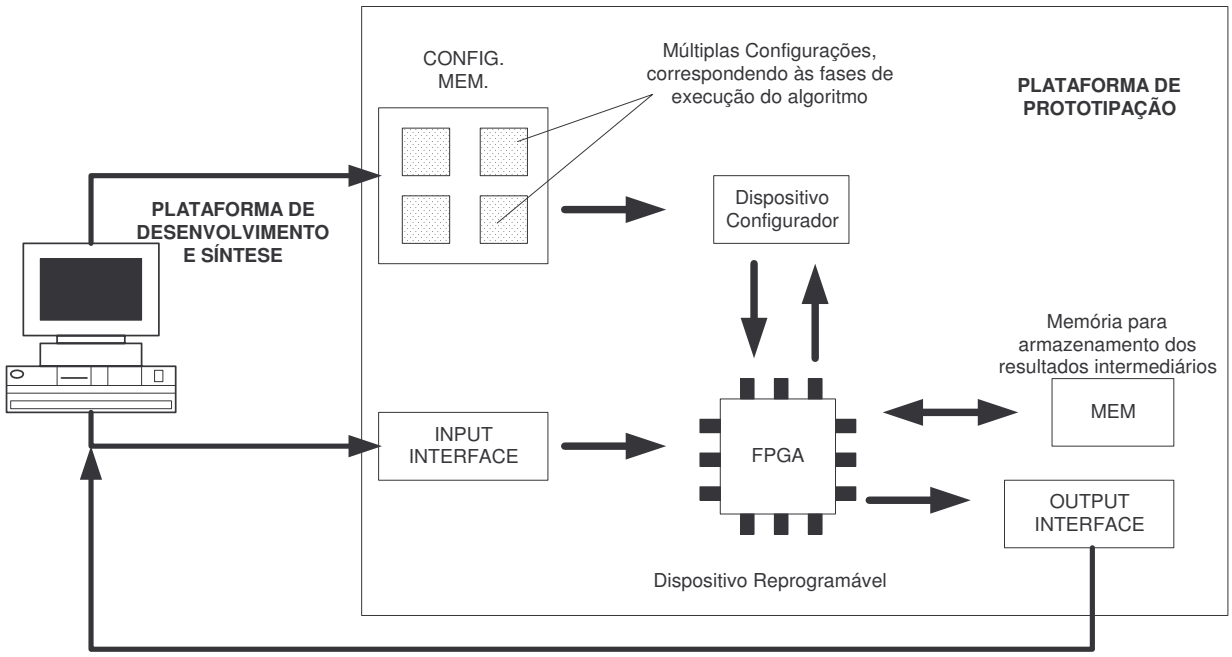

Figura 3 - Arquitetura do Sistema Proposto (Tira a terminologia em inglês da tua Figura, troca Input Interface por Interface de Entrada, etc. Troca disposisitivo Reprogrmável por Reconfigurável para separar o que se faz com hw do que se faz com sw. Não abrevie nomes da Figura Escreva Memória de Configuração)

4 Definição das Atividades

Os dispositivos que compõem uma Arquitetura Reconfigurável evoluem constantemente. Novas tecnologias de construção de circuitos integrados (CIs) têm possibilitado a construção de CIs com densidade cada vez maior de portas lógicas. FPGAs podem chegar hoje a comportar de 50 mil até mais de 3 milhões de portas, operando em velocidades de até 200MHz [VIR99]. Sendo assim, é importante a realização de uma análise detalhada sobre estas tecnologias e como elas podem ser utilizadas nesse trabalho. Como resultado deste estudo espera-se chegar a escolha mais apropriada, seguindo tendências atuais como por exemplo: portabilidade, baixo custo, facilidade de uso, etc.

Para melhor descrever e definir as atividades que serão necessárias para a implementação desta proposta, durante as etapas de desenvolvimento, o mesmo foi dividido em macro-atividades, listadas a seguir.

- Pesquisa e Definição da Plataforma de Prototipação a ser Utilizada;
- Pesquisa e Definição do Modelo de Inserção dos dados na Memória de Configuração;
- Pesquisa e Definição do Protocolo de (Re)configuração (Como havíamos combinado);
- Pesquisa e Definição do Hardware Adicional necessário;
- Pesquisa e Definição da Interface com a Aplicação;
- Pesquisa e Definição de Uma Aplicação Exemplo para a Arquitetura Proposta;
- Escrita do Volume Final de TC1.

4.1.1 Pesquisa e Definição da Plataforma de Prototipação a ser Utilizada

Atualmente, existem inúmeras plataformas de prototipação disponíveis no mercado, das quais duas são utilizadas nas disciplinas da graduação do curso de Bacharelado de Informática, e por este motivo, são as candidatas imediatas para este projeto:

- Xess XS40/XS95 [XES99]
- AEE 1199^{A} [AEE98]

(Diga que será feita uma avaliação de requisitos para verificar se estas placas serverm ou não para o trabalho. Cite que plataforma distinta pode ser necessária.).

Estas duas plataformas possibilitam a prototipação de projetos envolvendo a programação de dispositivos FPGA de maneira flexível com baixo custo. No entanto, a sua área física é reduzida, o que limita a inserção de hardware adicional caso necessário.

Outra característica comum a ambas é a utilização da família de FPGAs XC4000 da Xilinx Inc. [XIL99a].

O hardware básico a ser utilizado é a questão mais importante deste trabalho, visto que desta definição dependem todas as outras atividades.

4.1.2 Pesquisa e Definição do Modelo de Inserção dos dados na Memória de Configuração

Esta Seção tem como objetivo abordar as formas de alimentação e alocação dos dados originados do hospedeiro, onde serão desenvolvidas as configurações do FPGA, para a memória que vai armazenar estas configurações na plataforma de prototipação.

É necessário que se empregue um recurso de segmentação desta memória, para que cada configuração fique “isolada” em uma área específica. A ativação de cada um destes segmentos deve ser possível tanto do host principal como da própria aplicação em execução.

Outro detalhe a ser definido é a possibilidade de compartilhar a memória de configuração com a memória de armazenamento de dados.

Como as placas em geral não contam com uma memória específica para o armazenamento das configurações, poderá ser necessário determinar o hardware adicional a ser implementado para que esse recurso esteja disponível.

4.1.3 Pesquisa e Definição do Modo de Configuração do Dispositivo Programável

Na família XC4000, várias possibilidades de programação são oferecidas. O objetivo desta tarefa é determinar a melhor opção deste conjunto, que atenda os requisitos do sistema. Os pontos principais que serão considerados são: facilidade de implementação, velocidade de programação do dispositivo em cada modo e adequação do modo à necessidade de auto-reconfiguração.

4.1.4 Pesquisa e Definição do Hardware Adicional Necessário

Como as placas normalmente não contam com uma memória específica para o armazenamento das configurações, poderá ser necessário determinar o hardware adicional a ser implementado para que esse recurso esteja disponível.

4.1.5 Pesquisa e Definição da Interface com a Aplicação

É necessário que a plataforma da arquitetura reconfigurável tenha condições de receber e enviar dados de/para um agente externo. Entre as opções para essa interface estão teclado, vídeo ou comunicação através de porta serial usando o protocolo RS-232. A definição desta interface será feita nesta etapa.

4.1.6 Pesquisa e Definição de Uma Aplicação Exemplo

Definidos os pontos anteriores, será necessário desenvolver uma aplicação que demonstre a funcionalidade do sistema proposto. Esta fase do projeto pretende determinar um problema, estabelecer a sua solução através de um algoritmo, dividir esse algoritmo em etapas seqüenciais e implementar cada uma dessas etapas em seu equivalente em hardware.

5 Atividades

Esta Seção apresenta o cronograma das atividades, as interdependências e os recursos necessários para a correta execução deste trabalho.

5.1 Cronograma de Atividades

O cronograma das atividades foi dividido nas seguintes tarefas

1. Pesquisa e Definição da Plataforma de Prototipação a ser Utilizada;
2. Pesquisa e Definição do Modelo de Inserção dos dados na Memória de Configuração;
3. Pesquisa e Definição do Protocolo de (Re)configuração (Como havíamos combinado);
4. Pesquisa e Definição do Hardware Adicional necessário;
5. Pesquisa e Definição da Interface com a Aplicação;
6. Pesquisa e Definição de Uma Aplicação Exemplo para a Arquitetura Proposta;
7. Escrita do Volume Final de TC1.

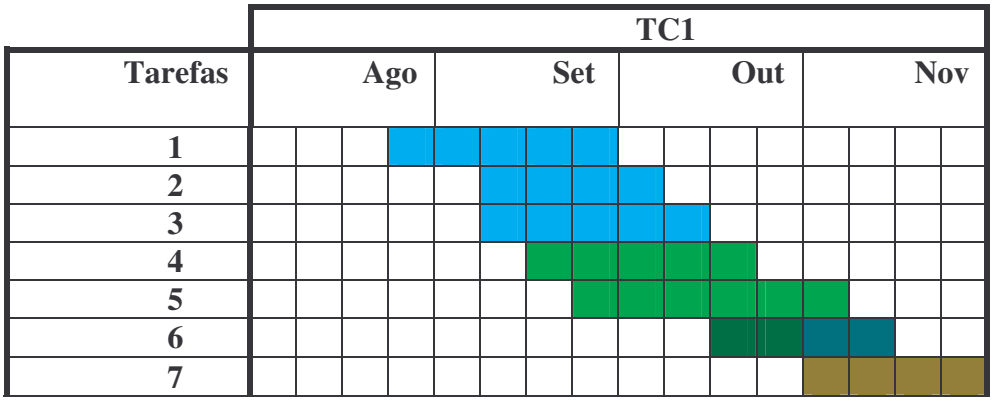

Figura 4 – Cronograma de Atividades e Dependências

5.2 Interdependência

As tarefas são seqüenciais e sua relação de dependência é mostrada na Figura 3. (Não percebi esta relação de dependência, pelo menos não na Figura 3. Revisa esta parte.)

5.3 Recursos

Abaixo seguem alguns recursos necessários par ao desenvolvimento do trabalho

- Laboratório
  - placas de prototipação XS40 – XSTEND;
  - placas de prototipação AEE-1199[A];
  - analisador lógico HP 1663E;
  - osciloscópio Tektronix TDS 220;
  - software para síntese e simulação de linguagens de descrição de hardware, tal como VHDL;
  - microcomputadores PC's e Estações de Trabalho;
  - compiladores C/C++;
- Fontes de Pesquisa
  - biblioteca central;
  - biblioteca do IPCT ;
  - internet.

6 Referências Bibliográficas

Tua bibliografia está completamente furada, bem como a notação. Usaste no texto [LLL99] e aqui usas [99]??????? Além do mais, não parece have relação entre as referências mencionadas aqui e as mencionadas no texto! Suponho que não tiveste tempo de gerá-las, certo?

[1] Smart Homes. [http://www.smart-homes.nl/engels/smarthome.html].

[2] Adapt. Domótica. [http://www.domotica-dip.com].

[3] Baumann, F., Jean-Bart, B., Kung, A., Robin, P.. Eletronic Commerce Services for Home Automation. Trialog. [http://www.trialog.com/emmsec9-97.pdf].

[4] Kung, A., Raither B.. Eletronic Commerce Services Expand Home Automation Capabilities. Trialog. [http://www.trialog.com/emmsec6-99.pdf]

[5] X10 Home Page. [http://www.x10-beta.com/ActiveHome/].

[6] AM14A/AM15A/LM14A X10 Specification. [ftp://ftp.x10-beta.com/pub/manuals/xtc797.doc].

[7] CM10A/CM11A X10 Programming Specification. [ftp://ftp.x10-beta.com/pub/manuals/protocol.doc].

[8] Beta Version of ActiveHome Software. Version 2.0b2.0.69. [ftp://ftp.x10-beta.com/pub/applications/activehome/x20b2069.exe].

[9] ROBERT BOSCH GmbH. CAN Especification Version 2.0. Stuttgart. 1991. [http://www.bosch.de/k8/can/docu/can2spec.pdf].[http://www.can-cia.de/CAN20B.pdf].

[10] Controller Area Networking - The Future Of Industrial Microprocessor Communications ?. IPE CAN Home Page - University of Magdeburg. http://141.44.61.248/NT/CAN/Welcome.html

[11] BRAISZ, H., SCHUHMANN, N., GLAUERT, W.. A VHDL Model of a Parameterizable CAN Controller as an Example of a Reuse Library Component. International Workshop on Logic and Architecture Synthesis - IWLAS98. Institut National Polythechnique of Grenoble, France. 15-16 December, 1998. [http://www.e-technik.uni-erlangen.de/~braisz/CAN_IWLAS98_slides.ps].

[12] EKIZ, H., KUTLU, A., POWNER, E.T. Design and Implementation of CAN/CAN Bridge. School of Engineering University of Sussex. International Symposium on Parallel Acvhitectures, Algorithms and Networks – I-SPAN'96. pp. 507-513.

[13] ETSCHBERGER, Ing. K.. CAN-based Higher Layer Protocols and Profiles. [http://www.stzp.de/papers/icc97/icc97_e.html].

[14] STAFFAN, Nilsson. Controller Area Network - CAN Information,. [http://www.algonet.se/~staffann/developer/CAN.htm].

[15] INTEL . Stand-Alone Controller Area Network. [http://developer.intel.com/design/auto/82527.htm].

[16] KVASER CAN School. [http://www.kvaser.com/can/edu/index.htm].

[17] NATIONAL . The COP888EB ROM based microcontrollers. [http://www.national.com/pf/CO/COP888EB.html].

[18] PHILIPS. Controller Area Network – CAN. [http://www.semiconductors.com/can/].

[19] SCHOFIELD , M. J.. Controller Area Network - an introduction to the serial communications bus. [http://www.omegas.co.uk/CAN/index.html].

[20] Smart Distributed System - SDS. [http://content.honeywell.com/sensing/prodinfo/sds/]