Arquitetura Hardware/Software de um núcleo NCAP Segundo o Padrão IEEE 1451.1: Uma Prova de Conceito

José de Anchieta G. dos Santos[1], **Ivan Saraiva Silva**[2]

[1]Departamento de Informática e Matemática Aplicada – Universidade Federal do Rio Grande do Norte (UFRN)– Natal – RN – Brasil

[2]Departamento de Informática e Estatística – Universidade Federal do Piauí (UFPI) Teresina–PI–Brasil

{anchieta}@ppgsc.ufrn.br

{ivan}@.ufpi.edu.br

Abstract. *The IEEE 1451 standard is an IEEE and NIST initiative that aims standardize the software to communicate TIM and NCAP modules, which must to communicate among them using the standardized interface TII. This work presents a hardware/software architecture for the prototyping of a NCAP module in FPGA according the IEEE 1451.1. The NCAP is prototyped on DE2 Altera's board and the NIOS II embedded processor is used to support the software which is developed in C. The hardware behavior is described using VHDL language.*

Resumo. *O padrão IEEE 1451 é uma iniciativa do IEEE e do NIST que visa padronizar o software de comunicação entre os módulos TIM e NCAP, os quais devem se comunicar através da interface padronizada TII. Este trabalho apresenta uma arquitetura hardware/software para a prototipação de um módulo NCAP em FPGA de acordo com o padrão IEEE 1451.1. O NCAP é prototipado na placa DE2 da Altera e o processador embarcado NIOS II é utilizado para dar suporte ao software escrito em C. O comportamento do hardware é descrito usando a linguagem VHDL.*

1. Introdução

Os avanços na tecnologia dos sistemas micro-eletromecânicos (MEMS) fizeram com que os sensores pudessem realizar atividades além daquelas necessárias para gerar apenas uma correta representação dos dados. Esses sensores passaram a fazer parte de uma nova classe conhecida como sensores inteligentes [Oliveira 2004]. Eles se diferenciam do sensor comum pela capacidade de processamento *on-board* sobre os dados monitorados.

Nesse contexto o sensor inteligente está dividido em módulos TIM (*Transducer Independent Module*) e NCAP (*Network Capable Application Processor*). O TIM contém os transdutores (sensores e atuadores), o equipamento de conversão de sinal e de condicionamento do mesmo. O NCAP contém a parte de processamento de dados e comunicação do sensor em rede com o usuário final, sendo considerado o responsável por atribuir característica de inteligência ao sensor.

Em 1993, o IEEE e o NIST criaram uma iniciativa de padronização que é considerada como um significativo avanço na tecnologia de sensores inteligentes

[Wobschall 2002],[Viegas et al. 2005]. Ela deu origem ao padrão IEEE 1451 que é um meio padronizado para comunicação em redes de sensores e atuadores. Seu grande benefício é unificar diversos padrões existentes que variam dependendo do fabricante.

Apesar de ser considerado como um avanço o padrão IEEE 1451 possui dificuldades de ser aderido devido à falta de NCAPs disponíveis seguindo o mesmo [Viegas et al. 2005]. Embora já tenha se passado um certo tempo desde que essa informação foi notificada ainda há casos como o da *Smart Sensor System*. Essa empresa disponibiliza um módulo TIM segundo o IEEE 1451, porém não disponibiliza um módulo NCAP, mas apenas um software que permite o acesso às informações do TIM [System 2010].

O objetivo deste trabalho é o desenvolvimento de um módulo NCAP segundo o padrão IEEE 1451.1. Esse módulo é prototipado em FPGA e deve possuir capacidade de comunicação em rede tanto com outro NCAP através de uma interface RS-232 e com o módulo TIM através de uma interface serial padronizada de acordo com o IEEE 1451.2.

2. Padrão IEEE 1451.1

O padrão IEEE 1451.1 se baseia na linguagem orientada a objetos para especificar o software do NCAP. O seu objetivo é o desenvolvimento de um modelo de objeto comum para que módulos NCAPs possam trocar dados entre si. Apesar do padrão ser baseado na orientação a objetos ele deixa o desenvolvedor livre à utilização de outras linguagens.

Nele é definido um modelo de informação neutro que é composto por um conjunto hierárquico de classes que representam os blocos de objetos do NCAP. O modelo é neutro porque especifica a forma geral que deve ser feita a troca de informação entre NCAPs, independentemente da tecnologia de rede que será utilizada. Ele também define um modelo de dados, onde são especificados o tipo e a forma dos dados que devem ser utilizados pelo software de aplicação do NCAP [IEEE-Std-1451.1-1999 1999].

Existem 4 tipos especiais de classes que devem comportar todas as classes definidas por este padrão. A classe *Block* contém classes de suporte à configuração do sistema, bem como comunicação entre o software da aplicação e os transdutores. A classe *Component* contém construções comuns da aplicação, tais como informações estruturadas ou coleções de objetos específicos da aplicação. A classe *Service* contém classes que dão suporte a comunicação entre NCAPs distintos. A classe *Non-IEEE 1451* contém classes específicas do fabricante do NCAP e que não fazem parte do padrão. Essa divisão visa dar maior modularidade ao software.

De acordo com o padrão na classe *Block* existe uma classe chamada *BaseTransducerBlock* que contém operações de *IORead()* e *IOWrite()* para dar suporte a comunicação entre o TIM e o NCAP. A classe *Block* também contém as classes *FunctionBlock* que contém operações específicas da aplicação e *NCAPBlock* que contém funções de configuração do NCAP e que devem ser utilizadas durante a fase de inicialização do mesmo.

A classe *Service* contém classes como a *BasePublisherPort* que possui operações de configuração dos parâmetros utilizados na comunicação *Publish/Subscriber*, a classe *PublisherPort* que realiza a operação de publicação, a classe *SubscriberPort* que realiza operações de *AddSubscriber()* e *CallBack()*. A primeira operação é utilizada para

requisitar interesses através da rede e a segunda utilizada para notificar aos demais *subscribers* que uma publicação foi aceita. Existem também duas classes que dão suporte à comunicação Cliente/Servidor. A *BaseClientPort* que possui os parâmetros a serem configurados para esse tipo de comunicação e a *ClientPort* que contém as operações de *Perform()* e *Execute()* para que um NCAP possa solicitar que uma operação seja executada por outro NCAP remotamente.

A classe *Component* é responsável por dar suporte ao conjunto de estruturas que são utilizadas pelo NCAP. Ela possui classes como *Parameter* que define os tipos de parâmetros utilizados por um determinado NCAP e a classe *ParameterWithUpdate* que da suporte à atualização dos parâmetros do NCAP.

Como pode ser visto na descrição da classe *Service* o padrão IEEE 1451.1 define dois tipos de comunicação entre NCAPs, também chamada de comunicação de alto nível. Uma delas é a comunicação Cliente/Servidor e a outra é a comunicação *Publish/Subscriber*.

No tipo de comunicação Cliente/Servidor um NCAP funciona como cliente e o outro como servidor. Nesse tipo de comunicação o cliente deve interromper suas atividades até que receba o resultado da requisição feita ao servidor. De acordo com esse padrão o cliente deve invocar os serviços do servidor através da operação de *Execute*. Essa operação possui como argumentos o status do modo de execução, o endereço do NCAP servidor, os argumentos de entrada para o NCAP servidor e os argumentos de saída desejados. O NCAP servidor executa o serviço solicitado através da operação *Perform* e devolve o resultado através da rede.

Na comunicação *Publisher/Subscriber* os NCAPs não necessitam interromper suas atividades. O *Publisher* e o *Subscriber* não precisam saber da existência um do outro. O *Subscriber* publica seu interesse na *Subscriber Port* e o *Publisher* publica um determinado interesse que ele possui para todos os *Subscribers* da rede. Ao receber um interesse publicado pelo *Publisher* a *Subscriber Port* compara o mesmo com o interesse publicado pelo Subscriber utilizando o domínio, a chave e o qualificador de subscrição. Caso o interesse requisitado coincida com o interesse recebido, este é aceito. Caso contrário é descartado. Este é o tipo de comunicação utilizado pelo NCAP desenvolvido no presente trabalho.

3. Trabalhos Relacionados

Trabalhos interessantes na área de sensores inteligentes são os de [SAUSEN et al. 2007] e [Song and Lee 2008]. Neles é mostrado o estado da arte do padrão IEEE 1451, servindo como base para outros trabalhos da área. Dentre os trabalhos relacionados à este merece destaque o de [Kochan et al. 2004] que desenvolve um NCAP remotamente reprogramável usando microcontroladores. O trabalho de [Turchenko et al. 2005] é uma evolução do anterior que usa FPGA para dar suporte à interfaces de alta velocidade. Em [Mayikiv et al. 2007], é feita uma comparação entre algumas abordagens de desenvolvimento existentes para o módulo NCAP, onde são destacadas suas vantagens e desvantagens, nele também é dado destaque para aquelas usando microcontroladores.

4. Arquitetura Proposta

A arquitetura proposta para este trabalho está dividida em uma infra-estrutura de hardware e uma infra-estrutura de software. A infra-estrutura de hardware comporta todos módulos de hardware utilizados para dar suporte ao funcionamento do NCAP, bem como o simulador do TIM. E a infra-estrutura de software é responsável pelo comportamento do NCAP, pelo suporte ao padrão IEEE 1451.1 e aplicação específica. A seguir será feita a descrição das mesmas.

4.1. Infra-estrutura de Hardware

A figura 1 mostra a arquitetura deste trabalho. Através dela pode-se observar que a placa DE2 da Altera é utilizada para a prototipação do NCAP. O processador embarcado NIOS II é utilizado para dar suporte ao software que é escrito em C, onde se encontram o padrão e a aplicação específica. O Barramento Avalon é o responsável por comunicar o NIOS com o driver RS-232 que dá suporte à comunicação de alto nível, com a memória utilizada para armazenamento de dados e com o módulo de interface padronizada TII (*Transducer Independent Interface*), que é utilizado para comunicação com o simulador do TIM. Este em conjunto com a interface TII são descritos através da linguagem VHDL.

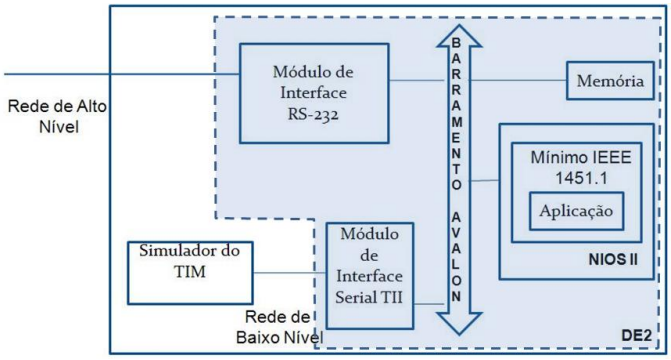

Figura 1. Arquitetura do NCAP proposto.

Na figura está claramente identificado que todos os componentes do NCAP estão situados dentro da zona tracejada em destaque. No caso o simulador do TIM também se encontra na mesma placa em que o módulo NCAP, porém não faz parte da proposta deste trabalho e foi implementado apenas para dar suporte às simulações e análises de resultados.

4.1.1. Interface TII

A interface serial padronizada TII é definida no padrão IEEE1451.2. Essa interface contém as portas de comunicação mostradas na figura 2. Neste trabalho ela é responsável por realizar a comunicação entre o módulo NCAP e o simulador do módulo TIM. De acordo com a figura a porta DIN é utilizada para envio de dados seriais do NCAP para o TIM e na porta DOUT os dados são enviados do TIM para o NCAP. A DCLK é utilizada para que o NCAP forneça seu clock para que o TIM possa trabalhar em sincronia com o mesmo. A NIOE é ativada durante o envio de mensagens do NCAP para o TIM. NTRIG é usada pelo NCAP para enviar um sinal de disparo para o TIM e este deve reconhecer

esse sinal através da porta NACK. O sinal de disparo é enviado pelo NCAP toda vez que ele deseja solicitar que o TIM realize alguma tarefa. A NINT é utilizada quando o TIM deseja enviar um sinal de interrupção para o NCAP. Neste trabalho ele é utilizado para que o TIM informe ao NCAP que alguma das grandezas medidas por ele ultrapassou um limiar pre-determinado. O NSDET é utilizado pelo NCAP para reconhecer a presença do módulo TIM, ou seja, ele deve sempre estar ativo. As demais portas são utilizadas como fontes de alimentação energética.

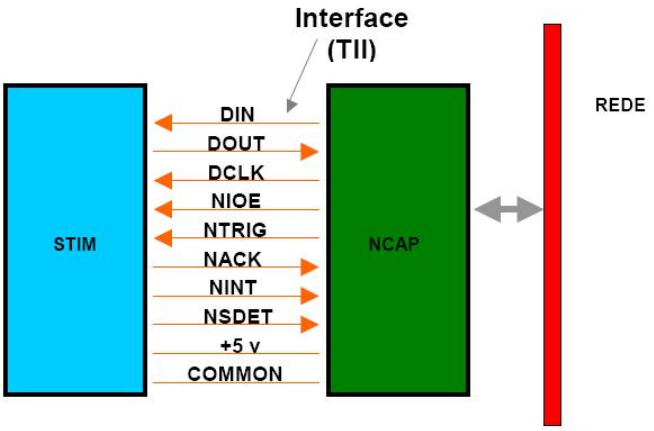

Figura 2. Interface padronizada TII.

O envio de mensagens entre o NCAP e o TIM também é feito seguindo o padrão IEEE 1451.2. De acordo com ele as mensagens devem ser enviadas ao TIM iniciando-se com o envio da função a ser realizada, em seguida do canal transdutor que deve executar a função e caso a função seja de escrita em um canal transdutor (atuador) o dado a ser escrito é enviado, senão é feita a leitura do canal (sensor). Para tanto segue-se o modelo da figura 3, onde o endereço da função é passado serialmente do bit mais significativo para o bit menos significativo, depois o mesmo ocorre com o endereço do canal transdutor. O bit mais significativo do endereçamento de função identifica se ela é uma escrita ou uma leitura, o zero representa leitura e o um representa uma função de escrita.

Endereço de função Byte mais significativo								**Endereço de canal** Byte menos significativo							
r/w	Código de função							Número de canal							
msb							lsb	msb							lsb

r =leitura, w= escrita

Figura 3. Endereçamento de funcão e de canal.

4.1.2. NIOS II

O NIOS II é um processador *soft-core* configurável, em contrapartida aos microcontroladores, que são elementos de processamento fixo. Nesse contexto, configurável quer

dizer que características podem ser adicionadas ou retiradas para que objetivos de desempenho e custo sejam alcançadas [Altera 2010].

O NIOS permite que um software desenvolvido na linguagem C/C++ possa ser prototipado e executado em uma FPGA. O ambiente de desenvolvimento do NIOS II é baseado no compilador GNU C/C++ e na IDE (*Integrated Development Environment*) do Eclipse que provê um ambiente familiar e conceituado para o desenvolvimento de software. Usando essa IDE é possível projetar e simular aplicações para esse processador.

4.1.3. SOPC Builder

A integração dos componentes do sistema é feita utilizando o SOPC Builder. O SOPC Builder é uma ferramenta que automatiza a tarefa de integração de hardware, pois ele permite gerar um completo SOPC (*System-On-a-Programmable-Chip*) em menos tempo do que utilizando métodos de integração tradicionais. Usando métodos tradicionais torna-se necessário escrever manualmente toda a linguagem de descrição do hardware que define a junção de todos os componentes do sistema. Enquanto que usando o SOPC Builder torna-se necessário apenas especificar os componentes do sistema através de uma GUI (*Graphical User Interface*) e o SOPC Builder gera todas as interconexões lógicas automaticamente.

4.2. Infra-estrutura de Software

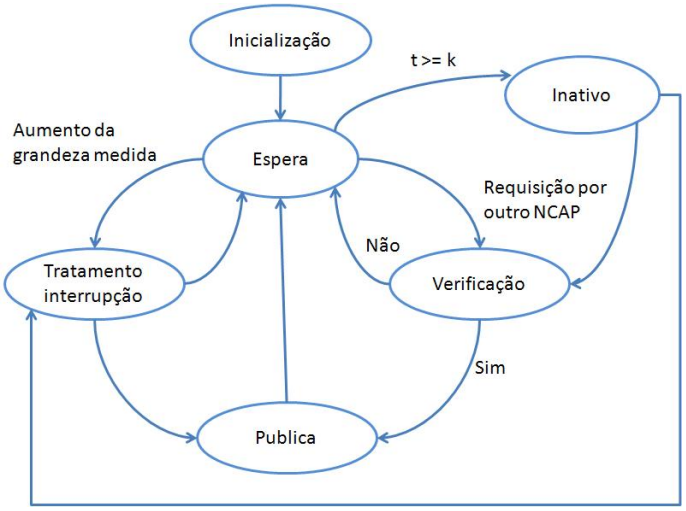

Figura 4. Máquina de estados do software do NCAP.

O software do NCAP trabalha de acordo com a máquina de estados da figura 4. Na inicialização é feita a solicitação de leitura das TEDS (*Transducer Eletronic Data Sheets*) armazenadas no simulador do TIM e todas as configurações necessárias ao funcionamento do NCAP. No estado de espera o NCAP fica à espera de solicitações de dados à partir de outro NCAP. Caso isso aconteça ele vai para o estado de verificação, onde verifica o dado solicitado e se possuir publica o mesmo para o NCAP requisitante. Também no estado de espera pode ocorrer uma interrupção por parte do simulador do TIM. Essa interrupção existe quando o TIM deseja enviar algum aviso de urgência, como no caso da

temperatura aumentar bruscamente. Nesse caso o NCAP deve publicar essa informação através da rede. Passado um determinado tempo sem receber requisição ou interrupção o NCAP entra no estado inativo, no qual ele permanece até a próxima ocorrência de uma das mesmas.

O tipo de comunicação de alto nível utilizado é o *Publish/Subscriber*, pois é o que mais se adéqua à realidade das redes de sensores. A comunicação de baixo nível é feita através de operações como *IORead* e *IOWrite* definidas no padrão. A aplicação específica irá verificar a variação de temperatura do sensor, calcular qual a variação deve ser feita no atuador do aquecedor e escrever esse valor no mesmo para manter a temperatura ideal.

5. Cenário de Uso

O NCAP proposto faz parte de um sensor inteligente, que por sua vez deve estar inserido em uma rede de sensores. Deste modo o cenário de uso para o NCAP em questão seria uma rede de sensores, onde cada sensor possui capacidade para medir uma grandeza específica. Um sensor chamado de *sink*, normalmente situado próximo à central de observação, faz requisições aos demais sensores da rede chamados *sources* e estes respondem com os dados requeridos caso seja capaz de medir a grandeza requisitada.

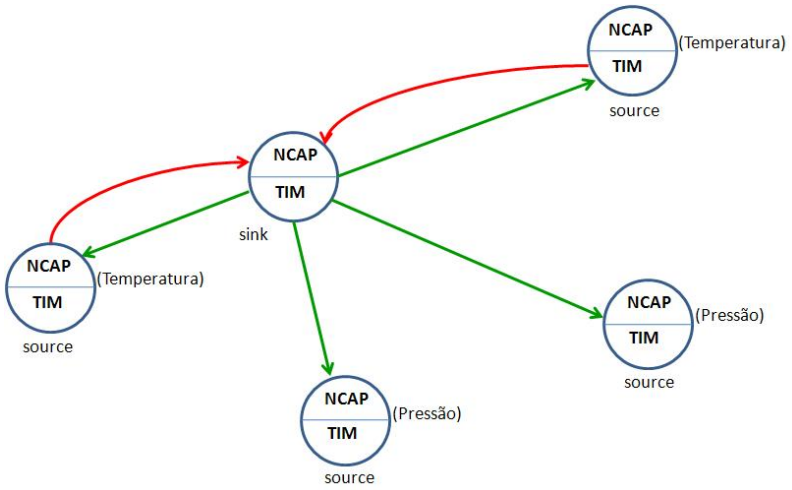

Figura 5. Cenário de uso para o NCAP.

Como pode ser observado na figura 5 a rede de sensores é formada por um nó *sink* que tem acesso direto aos demais nós da rede. Essa rede não segue uma abordagem *multihop* para entrega de dados ao *sink*. O objetivo desse cenário de uso é testar o correto funcionamento do módulo NCAP e o uso do padrão IEEE 1451.1. Na seção 7 que apresenta as conclusões e trabalhos futuros a este é deixada a implementação da comunicação através de uma interface Ethernet que possibilite simular uma rede de sensores contendo nós roteadores.

Na simulação são usadas duas placas DE2, de acordo com a figura 6, uma delas simulando um NCAP *sink* e outra simulando um NCAP *source*. O NCAP *sink* requisita a leitura de dados de pressão e temperatura do outro. Este deve responder apenas quando possuir a grandeza que é capaz de medir, no caso a temperatura. Eventualmente a temperatura pode aumentar demais, nesse caso o *source* deve enviar uma notificação ao *sink*, onde o conteúdo da notificação é o valor da temperatura.

Durante a comunicação os NCAPs utilizam as funções do padrão da seguinte maneira. Inicialmente a placa simulando o *sink* utiliza a operação *AddSubscriber()* para solicitar seus interesses através da rede. Essa operação tem como argumentos a chave de subscrição *key*, o domínio de subscrição *domain* e o qualificador de subscrição *qualifier*. A placa que simula o *source* ao receber interesses vindos da rede verifica se possui aquele tipo de interesse e caso possua realiza a operação de *publish()* para enviar os dados. Essa operação envia a chave de publicação *key*, o domínio de publicação *domain*, o tópico de publicação *topic* e o conteúdo da publicação. Ambos, o *sink* e o *source*, ficam sempre monitorando os canais transdutores do seu TIM através de solicitações de leitura usando a operação de *IORead()*, que recebe como argumento o endereço do canal transdutor a ser lido. Caso queira realizar uma escrita em algum canal transdutor é usada a operação *IOWrite()* que tem como argumentos o endereço do canal transdutor e o dado a ser escrito.

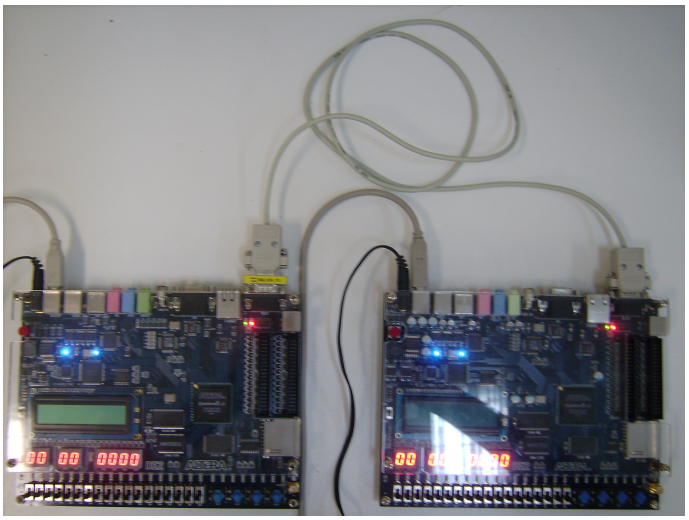

Figura 6. Placas DE2 se comunicando através de uma interface RS-232.

6. Resultados

O módulo NCAP desenvolvido para este trabalho ainda não se encontra completamente desenvolvido. Ainda falta finalizar o módulo de comunicação entre os NCAPs através da interface RS-232. Apesar de ainda não estar completo já existem alguns resultados obtidos a partir de simulações dos módulos de interface TII e do simulador do TIM. As operações de escrita e leitura solicitadas ao TIM a partir do NCAP estão funcionando, bem como a operação de interrupção vinda do TIM.

Uma operação de leitura utiliza 41 ciclos de clock do sistema para disponibilizar o dado ao software. A operação de escrita utiliza 37 ciclos. Desde a ocorrência de uma condição de interrupção o simulador do TIM leva 18 ciclos de clock para disponibilizar o valor da grandeza medida ao software. Como já foi explicado antes, uma interrupção ocorre sempre que uma grandeza medida ultrapassa um limiar pre-determinado.

Na versão que se encontra desenvolvido o NCAP, ele trabalha com uma frequência de máxima de 50MHz e seu hardware possui 4204 elementos lógicos, ocupando 13% em área da FPGA, no caso a Cyclone II EP2C35F672C6 da Altera.

7. Conclusões

Este trabalho apresenta uma arquitetura composta por uma infra-estrutura de hardware e uma infra-estrutura de software como abordagem de desenvolvimento de um módulo NCAP. Esse módulo segue o padrão IEEE 1451.1 definido pela IEEE em conjunto com o NIST. As linguagens utilizadas para o desenvolvimento foram a linguagem C para o software e a linguagem VHDL para a descrição do comportamento do hardware. Um simulador do comportamento de um módulo TIM também foi implementado para dar suporte aos testes e análises do NCAP, que é prototipado na placa DE2 da Altera. O módulo desenvolvido trás como contribuição o incentivo ao uso do padrão IEEE 1451, o qual apesar de ser considerado como um avanço na área de sensores inteligentes ainda é pouco aderido devido à falta de módulos NCAPs disponíveis seguindo este padrão.

Apesar de estar em fase de conclusão o presente trabalho deixa trabalhos futuros interessantes. Um deles seria o desenvolvimento do módulo de interface Ethernet para dar suporte a comunicação entre vários NCAPs. Outro seria a implementação de um protocolo de roteamento para o NCAP desenvolvido.

Referências

Altera, C. (2010). Altera corporation. http://www.altera.com/. Acessado em: 3 de Março de 2010.

IEEE-Std-1451.1-1999 (1999). Standard for a smart transducer interface for sensors and actuators - network capable application processor (ncap) information model. IEEE Instrumentation and Measurement Society, TC-9,The Institute of Eletrical and Eletronic Engineers, Inc.

Kochan, R., Lee, K., Kochan, V., and Sachenko, A. (2004). Development of a dynamically reprogrammable ncap [network capable application processor]. volume 2, pages 1188–1193 Vol.2. Instrumentation and Measurement Technology Conference, 2004. IMTC 04. Proceedings of the 21st IEEE.

Mayikiv, I., Stepanenko, A., Wobschall, D., Kochan, R., Sachenko, A., and Vasylkiv, N. (2007). Remote reprogrammable ncaps: Issues and approaches. pages 109–113. Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 2007. IDAACS 2007. 4th IEEE Workshop on.

Oliveira, A. L. (2004). Modelo e algoritmos para organização de redes de sensores sem fio hierárquicas. Master's thesis, Departamento de pós-graduação em Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte - MG.

SAUSEN, P. S., CAMPOS, M. d., and SALVADORI, F. (2007). O estado da arte do padrão ieee 1451 aplicado às redes de sensores. *Conselho em Revista*, (36):36 – 36.

Song, E. and Lee, K. (2008). Understanding ieee 1451-networked smart transducer interface standard - what is a smart transducer? *Instrumentation & Measurement Magazine, IEEE*, 11(2):11–17.

System, S. S. (2010). Smart sensor system. http://www.altera.com/. Acessado em: 30 de Março de 2010.

Turchenko, I., Kochan, R., Kochan, V., Sachenko, A., Maykiv, I., and Markowsky, G. (2005). Network capable application processor based on a fpga. volume 2, pages 813–

817. Instrumentation and Measurement Technology Conference, 2005. IMTC 2005. Proceedings of the IEEE.

Viegas, V., Dias Pereira, J., and Silva Girao, P. (2005). Using a commercial framework to implement and enhance the ieee 1451.1 standard. In *Instrumentation and Measurement Technology Conference, 2005. IMTC 2005. Proceedings of the IEEE*, volume 3, pages 2136–2141.

Wobschall, D. (2002). An implementation of ieee 1451 ncap for internet access of serial port-based sensors. In *Sensors for Industry Conference, 2002. 2nd ISA/IEEE*, pages 157–160.