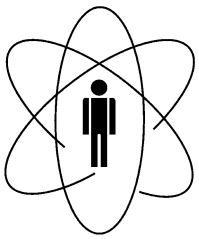
ISSN 0101-9201



CBPF - CENTRO BRASILEIRO DE PESQUISAS FÍSICAS

Rio de Janeiro

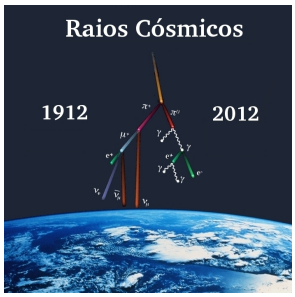
Notas Técnicas

CBPF-NT-002/12

março 2012

Breve introdução da comunicação de dados em alta velocidade através de barramento PCI *Express* e lógica programável (FPGA)

Fernando M.B. de Sousa & H. P. Lima Jr



Ministério da
Ciência, Tecnologia e Inovação

Breve introdução da comunicação de dados em alta velocidade através de barramento PCI *Express* e lógica programável (FPGA)

Fernando M. B. de Sousa* e H. P. Lima Jr

Centro Brasileiro de Pesquisas Físicas - CBPF

Rua Dr. Xavier Sigaud,

150 – Urca – Rio de Janeiro – RJ – Brasil

Este documento descreve o desenvolvimento de um sistema capaz de transmitir dados de um módulo eletrônico para um computador pessoal através do barramento serial PCI Express 1.0. Este barramento é de grande interesse para aplicações que exigem alta taxa de transferência de dados como, por exemplo, captura e processamento de imagens de alta resolução, e aquisição de dados em Física experimental. Utilizando-se este barramento na configuração de maior desempenho é possível atingir a taxa de 8 GB/s na transferência de dados entre um periférico e o computador.

1. INTRODUÇÃO

Com o objetivo de contribuir para a área de instrumentação científica, e aperfeiçoar as técnicas de aquisição de dados, está em desenvolvimento no CBPF o projeto MOPI - MÓDULO PROGRAMÁVEL PARA INSTRUMENTAÇÃO CIENTÍFICA E CONTROLE DE PROCESSOS. Trata-se de um módulo eletrônico que será capaz não somente de realizar aquisição de dados, mas também análise espectral de sinais contínuos, processamento digital de sinais, controle de experimentos e monitoração de processos industriais. O módulo utiliza o barramento PCI *Express* [1] como interface de comunicação com um computador, sendo conectado diretamente à placa mãe e podendo atingir 1 GB/s de velocidade na transferência de dados. Outra forma de comunicação com o módulo é através de uma interface USB [2], que permite transferências de até 12 MB/s. Todo o processamento digital, incluindo os protocolos de comunicação, é concentrado em um dispositivo lógico programável FPGA (*Field Programmable Gate Array*) [3].

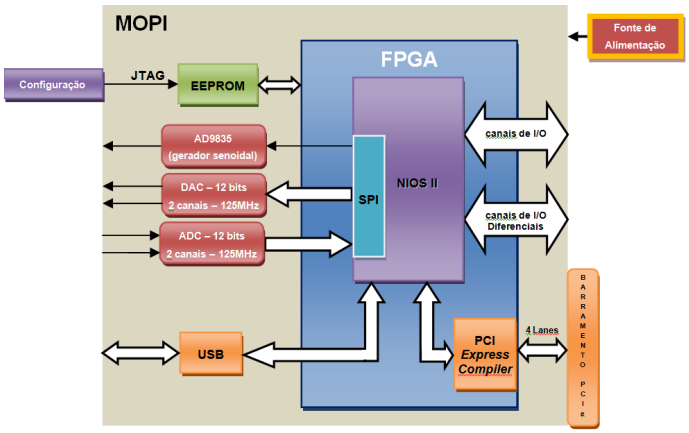


Figura 1: Diagrama em blocos do módulo MOPI.

Além destas características, o MOPI dispõe de um conversor analógico-digital (ADC) AD9627 [4] de 2 canais, 12 bits, um conversor digital-analógico (DAC) AD9116 [4], também de 2 canais e 12 bits, ambos operando na freqüência de 125 MHz e um chip dedicado à Síntese Digital Direta (DDS) [5], que gera sinais senoidais de alta freqüência (até 25 MHz) digitalmente para qualquer aplicação externa. Estão disponíveis também quatro canais digitais de I/O modo *single-ended* e quatro canais de I/O em modo diferencial (padrão LVDS), todos conectados à FPGA. A Figura 1 mostra o diagrama em blocos do módulo MOPI. No diagrama podemos ver os dois blocos principais da FPGA, isto é, o PCI *Express Compiler* [6] e o microprocessador NIOS II [7]. O primeiro é responsável pela comunicação de dados entre o MOPI e o computador via barramento PCI *Express*. O segundo bloco gerencia o funcionamento do MOPI a partir de informações que chegam através do PCI *Express Compiler*. A comunicação do NIOS II com o ADC, o DAC e o DDS é feita através do protocolo SPI (*Serial Peripheral Interface*) [8]. O NIOS II é programado em linguagem C e é responsável por tarefas como seleção das entradas ou saídas que estarão em funcionamento, freqüência de saída do DDS, início de aquisição de dados, entre outras. Devemos observar também que o módulo pode ser utilizado externamente ao computador através da interface USB disponível. Neste caso, o módulo deve ser alimentado por uma fonte de alimentação externa, capaz de fornecer 5 Volts - 3 Ampéres. A configuração da FPGA é feita através de uma entrada dedicada utilizando o protocolo JTAG [9] e uma memória EEPROM.

1.1. O PROTOCOLO PCI *EXPRESS*

O PCI *Express* (PCIe) é um barramento serial ponto a ponto onde cada periférico possui um canal exclusivo de

*Electronic address: fsousa@cbpf.br

comunicação com o *Core* PCIe, que é o controlador PCI *Express* existente em cada dispositivo (ver Figura 2). No padrão PCI [10], anterior ao PCIe, o barramento é paralelo e compartilhado por todos os periféricos ligados a ele, o que pode criar congestionamento de dados. A compatibilidade, no nível de *software*, entre PCIe e PCI é mantida para assegurar que todas as aplicações e *drivers* existentes possam ser reutilizados. Conceitualmente, a principal diferença entre os dois padrões está no fato de que no PCIe a transmissão de dados é serial, enquanto no PCI a transmissão é paralela (até 64 bits). De uma forma geral, a arquitetura PCIe é definida em camadas (*layers*), como mostra a Figura 2, como será discutido em detalhes a seguir.

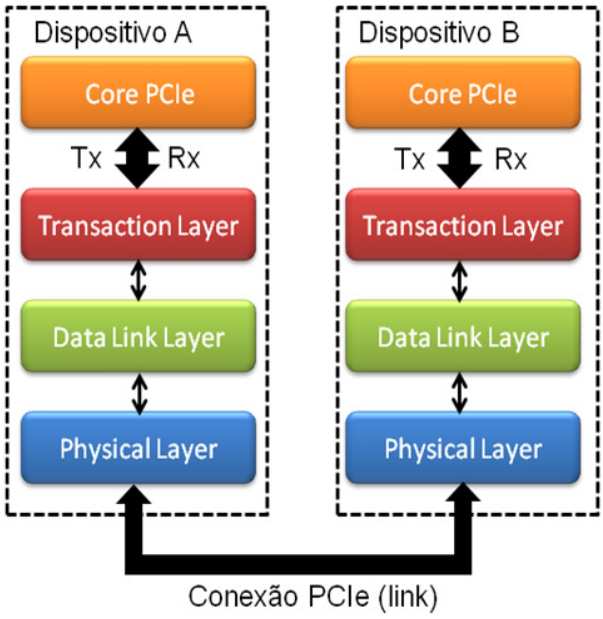


Figura 2: Representação das camadas do protocolo PCI *Express*.

A camada *Transaction Layer* recebe, através do *Core* PCIe, as requisições de leitura e escrita de um dos dispositivos e cria pacotes com essas informações para serem enviados ao PCIe-*link*. Essa camada também é responsável por receber pacotes de respostas vindos do PCIe-*link* e enviá-los ao dispositivo destinatário. Isso é possível porque cada pacote possui uma identificação única que o direciona para a origem correta. O papel principal da camada *Data Link Layer* (DLL) é entregar o pacote ao PCIe-*link*, assegurando a integridade dos dados. Para isso é utilizada uma técnica de detecção de erros chamada *Cyclic Redundancy Check* (CRC) [11]. A camada DLL irá automaticamente repetir um pacote que foi detectado como corrompido. Finalmente, a camada *Physical Layer* conecta os dispositivos ao PCIe-*link*, que podem ser compostos por 1, 2, 4, 8, 12, 16, ou 32 *lanes*, podendo ser representados da forma (x1), (x4), (x8), (x12), (x16), ou (x32). Cada *lane* é capaz de transmitir um byte nas duas direções ao mesmo tempo. Esta comunicação *full-duplex* é possível porque cada *lane* é composto por um par de sinais (Tx e Rx), sendo um para enviar e outro para receber os dados, conforme ilustra a Figura 3. Os sinais Tx e Rx são transmitidos em modo diferencial.

Teoricamente, a taxa de transferência de dados no PCI *Express* 1.0 é de 2,5 Gbits por segundo, por *lane*, por direção. O

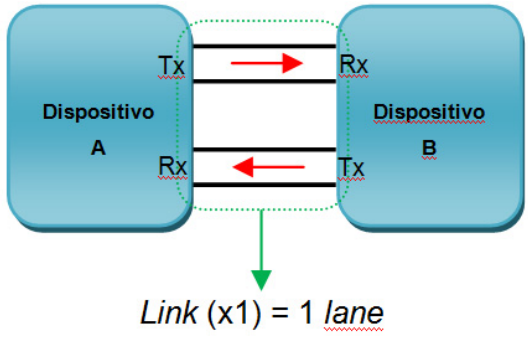


Figura 3: Representação de um *link* PCIe com apenas 1 *lane*.

PCIe utiliza um sistema de codificação chamado 8bits/10bits, onde são incluídos dois bits adicionais para cada byte de dados transmitidos. Estes bits adicionais eliminam a necessidade de linhas adicionais para enviar um sinal de *clock*, o que simplifica bastante o projeto e melhora a confiabilidade. É por causa desta característica que a taxa de 2,5 Gbits/s alcançada pelos transmissores equivale a uma banda nominal de 250 MB/s de dados em cada direção. Se o *link* possuir, por exemplo, 4 *lanes* (x4), teremos 250MB x 4 = 1 GB/s em cada direção. A Figura 4 mostra um *link* com 4 *lanes* (x4).

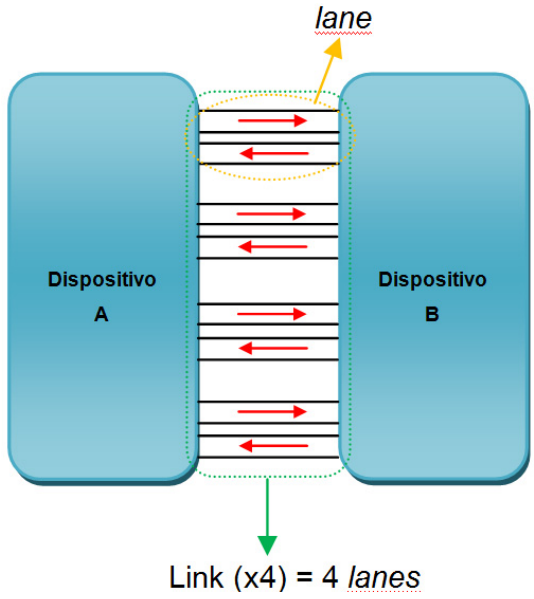


Figura 4: Representação de um *link* PCIe com 4 *lanes*.

2. DESENVOLVIMENTO

Nessa seção serão apresentadas as ferramentas e os procedimentos utilizados para implementar a transferência de dados entre o PC e a FPGA, utilizando o barramento PCIe. Para isso, será utilizado o Kit de desenvolvimento Arria II GX [12] e o *software* Quartus II [13]. A escolha destas ferramentas levou em consideração os custos envolvidos e as características funcionais similares entre as FPGA's ARRIA II e Cyclone IV, esta última adotada no projeto MOPI.

2.1. O Kit de Desenvolvimento Arria II GX

A Figura 5 mostra o módulo do Kit de desenvolvimento contendo a FPGA EP2AGX125EF35 da família Arria II GX.

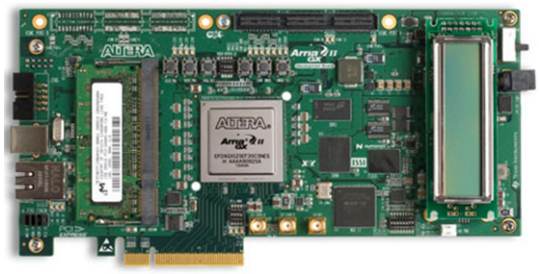


Figura 5: Kit Arria II GX utilizado para o desenvolvimento do *firmware*.

A partir deste kit é possível projetar e verificar o funcionamento do protocolo PCIe, na FPGA, com até 8 *lanes*. Este kit representa uma plataforma de *hardware* para o desenvolvimento e prototipagem de baixa potência e alto desempenho. O módulo fornece periféricos e interfaces de memória para facilitar o desenvolvimento e testes de projetos. Possui duas portas de alta velocidade (*mezzanine card* - HSMC) disponíveis para adicionar mais funcionalidade. Como pode ser observado na Figura 5, o kit pode ser instalado dentro do PC, diretamente conectado a um *slot* PCIe da placa mãe.

2.2. O *Software* de Desenvolvimento Quartus II

O *software* de desenvolvimento Quartus II é um ambiente de projeto que pode ser utilizado em todas as fases dos projetos com FPGA's. O Quartus II executa as etapas de *synthesis, placement* e *routing* do projeto lógico a ser sintetizado na FPGA. Utilizando a compilação incremental é possível obter uma redução no tempo de compilação. Através deste recurso são compiladas somente as partes do projeto, nas quais foram feitas alterações. A Figura 6 mostra a tela principal da ferramenta Quartus II.

O Quartus II possui diversos aplicativos destinados ao desenvolvimento das etapas de um projeto. O Editor de Blocos permite criar um arquivo tipo bdf (*block design file*), que é um ambiente gráfico onde pode-se inserir blocos e símbolos primitivos ou que representem uma lógica descrita em VHDL [14] ou Verilog [15]. Também é possível editar as informações deste projeto gráfico utilizando este aplicativo. A Figura 7 mostra o arquivo bdf (*top level*) do projeto aqui descrito.

Através do *MegaWizard Plug-In Manager* [13] cria-se ou edita-se componentes pré-existentes, chamados de *megafunctions*, que executam funções específicas dentro da FPGA, tais como memórias, funções lógicas, multiplexadores etc. Após isso, esses arquivos podem ser instanciados no projeto. Para análise dos resultados pode-se utilizar o analisador lógico *SignalTap* [13], que permite a visualização dos sinais internos da FPGA em tempo quase real, através do protocolo JTAG [9].

2.3. O Processador NIOS II

O sistema de processamento Nios II, que pode ser implementado na FPGA, funciona de forma equivalente a um microcontrolador, onde estão inclusos um processador (CPU), uma combinação de periféricos e memória. No caso do Nios II, temos um núcleo de processamento Nios II (CPU), um conjunto de periféricos, memória *on-chip*, e interfaces para memória *off-chip*. Uma diferença entre o Nios II e outros sistemas de processamento é que o Nios II pode ser configurado. É possível adicionar, remover, ou editar recursos do sistema de maneira a atender as metas de desempenho e custo do projeto. O sistema Nios II é chamado *"soft"*, pois o núcleo do processador, assim como os periféricos, não são estáticos (fixos) na pastilha de silício, podendo ser direcionados para qualquer família de FPGA's do mesmo fabricante. O NIOS II, assim como qualquer microcontrolador, pode ser programado em linguagem C. A ferramenta utilizada para programação e compilação do NIOS II chama-se Eclipse [7].

3. PROJETO

Nessa seção será descrito o processo de leitura dos dados existentes em uma memória interna à FPGA pelo computador, através do *Windriver* [16], que é de um driver PCIe associado a um aplicativo de leitura e escrita de dados.

Como pode ser observada na Figura 8, onde é apresentado o diagrama em blocos do sistema, a fonte de dados para a memória FIFO é um contador descrito em VHDL. A memória FIFO foi implementada utilizando-se o *MegaWizard Plug-In Manager*, possuindo 16 bits de dados, 128 posições e bits indicadores que avisam quando está cheia ou vazia. Essas características associadas a um controlador, também desenvolvido em VHDL, possibilitam estabelecer o controle de fluxo de dados entre a FIFO e o NIOS II. Este último faz a leitura dos dados quando a FIFO informa que está cheia. Da mesma forma, os dados somente voltam a ser escritos na FIFO quando o controlador é informado que a FIFO está vazia. A Figura 9 mostra em detalhe esse processo.

A seguir serão descritos três dispositivos que foram implementados utilizando-se o aplicativo SOPC *Builder* [17],

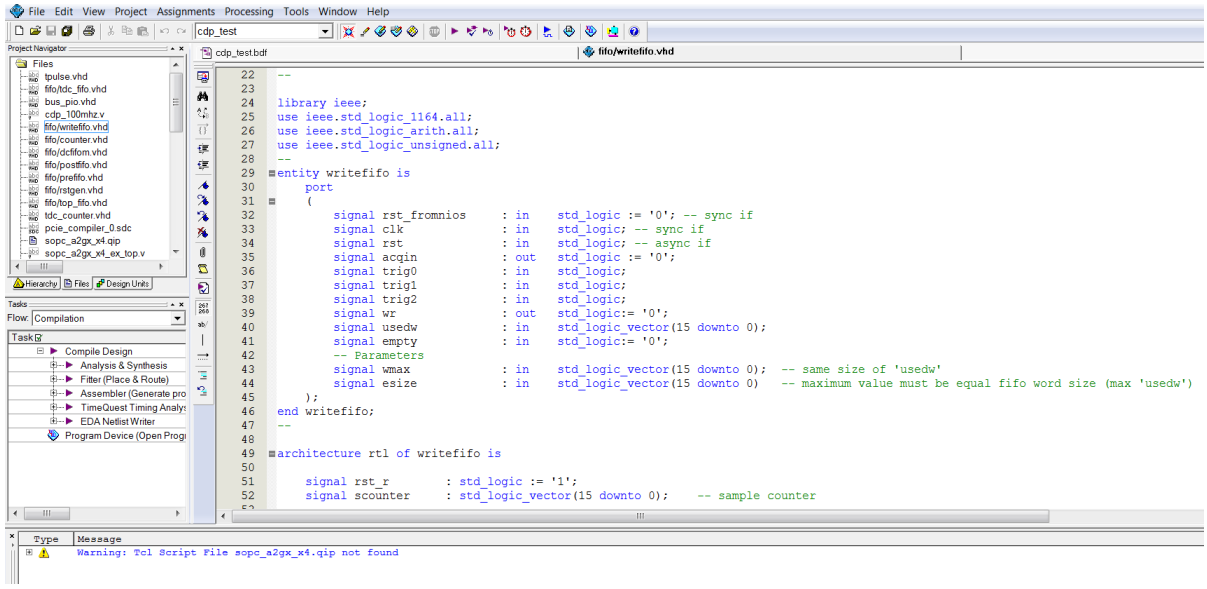


Figura 6: Ferramenta de desenvolvimento Quartus II - tela principal.

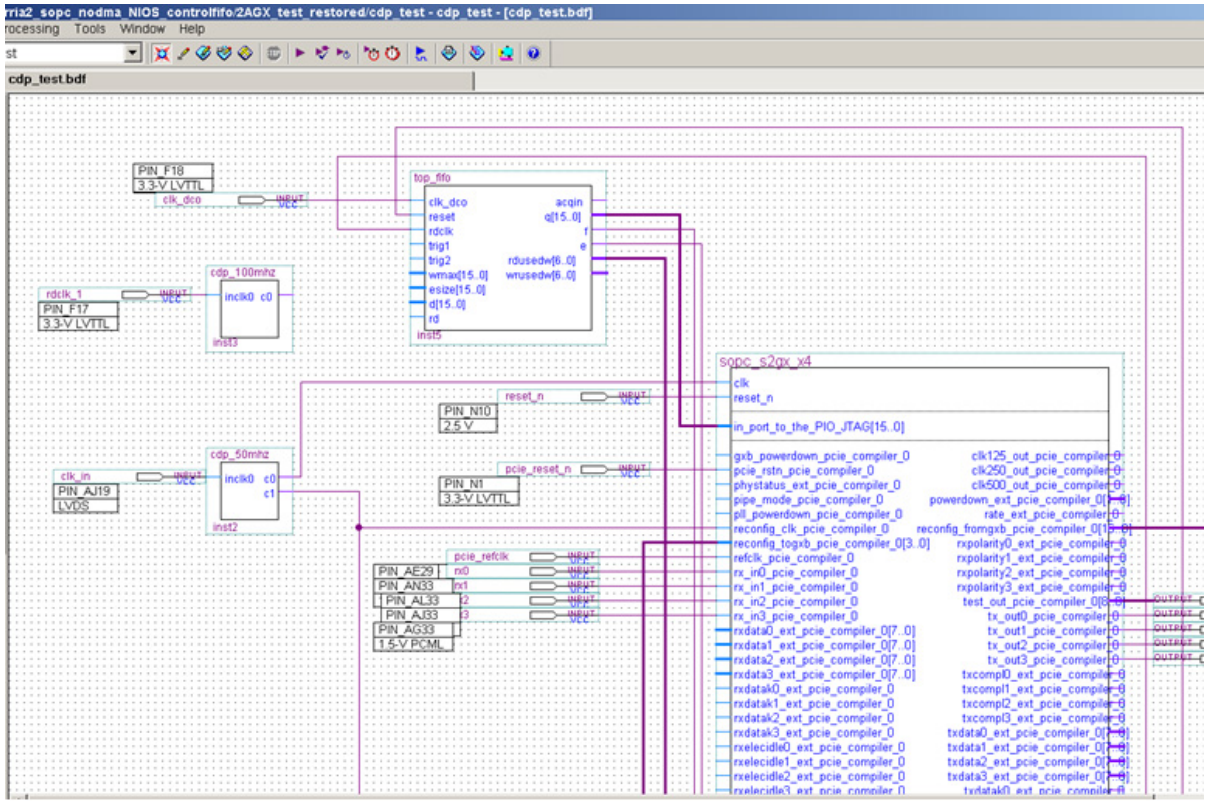


Figura 7: Arquivo *Top Level*, em bdf, do projeto.

através do qual, pode-se selecionar, configurar e interligar dispositivos mais complexos do sistema.

O NIOS II lê os dados da memória FIFO e os transfere para uma memória dual (memória que pode ser acessada por dois dispositivos). Em seguida, os dados da memória dual são acessados pelo PCIe *Compiler*. Esse dispositivo é chamado pelo fabricante de *MegaCore Function* e sua função básica é comunicar-se com o barramento PCIe do computador. A Figura 10 mostra, em destaque, as interconexões entre os dispositivos no SOPC *Builder*.

Até aqui, foi descrito o processo de implementação do *hardware* do sistema. É importante ressaltar que o NIOS II e sua configuração fazem parte deste *hardware* que será gravado na FPGA. O Quartus II possui um programador que se encarrega dessa tarefa.

O programa de gerenciamento da memória dual pelo

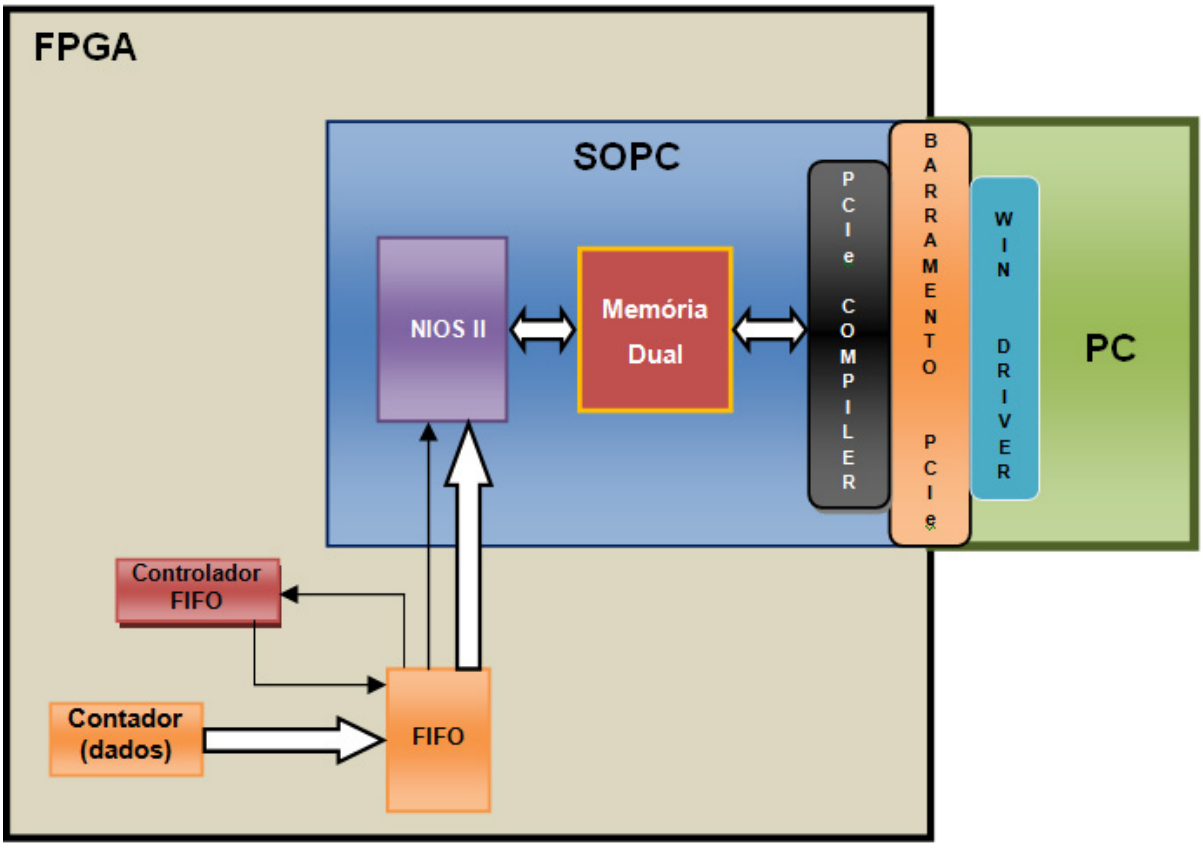


Figura 8: Diagrama em blocos do sistema.

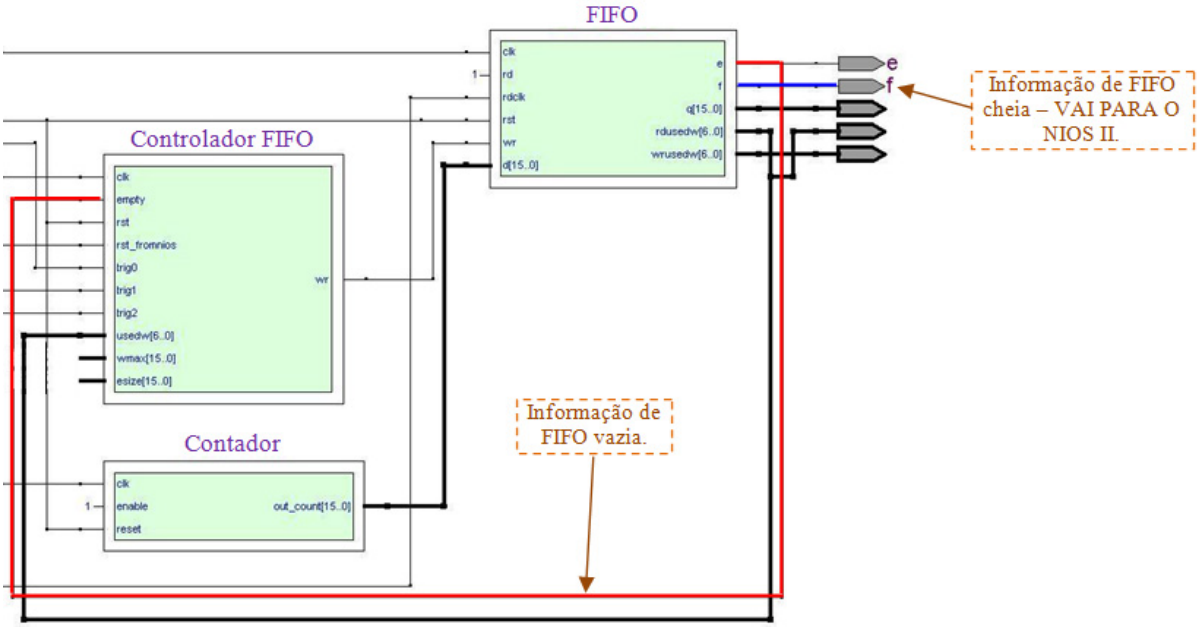


Figura 9: Diagrama em blocos do fluxo de dados entre a FIFO e o NIOS II.

NIOS II foi desenvolvido em linguagem C, utilizando-se o compilador Eclipse.

Em outras palavras, após a configuração do *hardware* na FPGA é necessário gravar o programa no NIOS II. O fluxograma com a rotina de funcionamento do programa no NIOS II é mostrado na Figura 11.

Após a gravação do projeto na FPGA, já é possível ler os dados gerados pelo contador, através do *Windriver*. Os resultados serão apresentados na próxima seção.

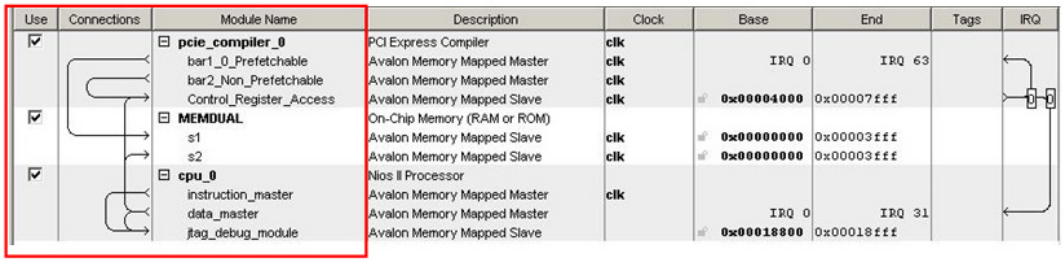


Figura 10: Interconexões entre o PCIe *Compiler*, memória dual e o NIOS II.

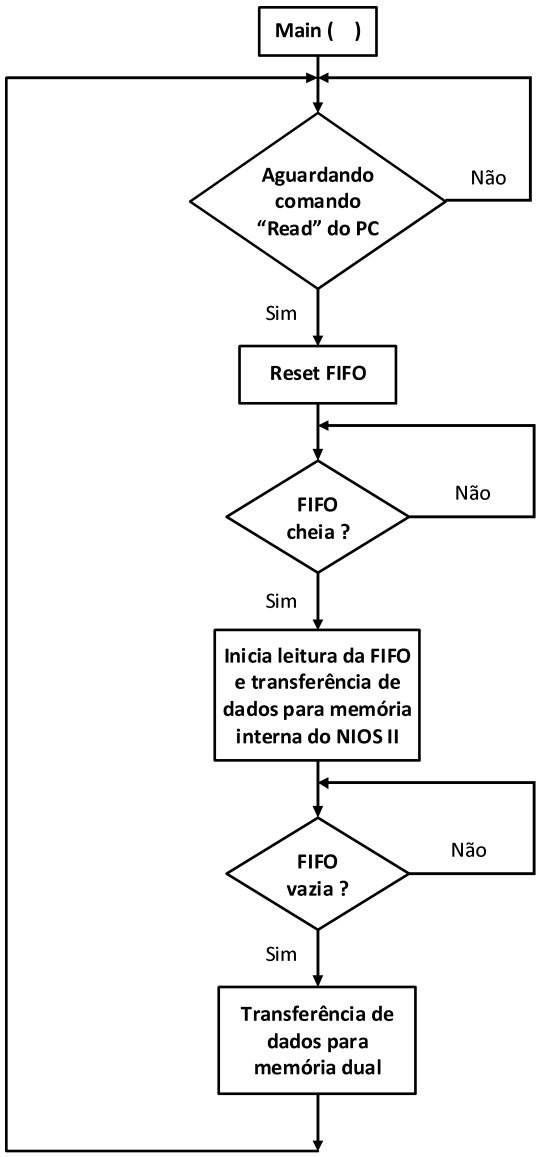


Figura 11: Fluxograma de rotina de funcionamento do programa do NIOS II.

4. RESULTADOS

Os primeiros resultados da comunicação de dados entre o Kit de desenvolvimento Arria II GX e o PC foram obtidos utilizando-se o *WinDriver*. A Figura 12 mostra o painel do *Windriver* sobreposto ao ambiente Eclipse. Essa Figura mostra que os dados lidos da memória dual pelo *Windriver* (via barramento PCI Express) estão de acordo com a leitura de dados desta mesma memória pelo Eclipse, através da porta JTAG. O *Windriver* mostra os dados na base numérica hexadecimal e da direita para esquerda, ao passo que o Eclipse os apresenta, verticalmente, na coluna "Dados"' e na base numérica decimal. A Tabela 1 mostra a correspondência entre as duas bases numéricas.

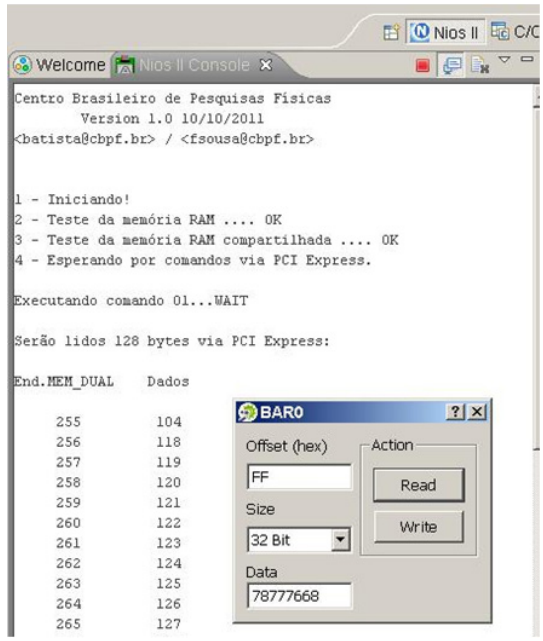


Figura 12: Leitura de dados pelo *Windriver*.

TABLE 1: Correspondência entre valores mostrados pelo Eclipse e lidos pelo *Windriver*.

Valores na base Decimal	Valores na base Hexadecimal	
255	FF	Posição da memória (*offset*)
104	68	Dado
118	76	Dado
119	77	Dado
120	78	Dado

5. CONCLUSÃO

Esta nota técnica descreve o desenvolvimento de um projeto capaz de estabelecer a comunicação de dados entre módulos de instrumentação científica e computadores pessoais utilizando o protocolo PCI *Express*. Foi utilizado um kit de desenvolvimento comercial que possui como dispositivo principal uma FPGA. O projeto foi implementado baseado no compartilhamento dos dados armazenados em duas memórias RAM, sendo uma delas do tipo FIFO. Em uma primeira etapa foi mostrado o controle de leitura e escrita da memória FIFO efetuado pelo NIOS II e um controlador desenvolvido em VHDL. Posteriormente foi possível verificar, através da transferência de dados da memória dual, o funcionamento da comunicação entre o kit e o PC via barramento PCI *Express*. Todo o processo foi gerenciado pelo NIOS II e mostrou-se bastante satisfatório.

Foram apresentados os resultados preliminares da leitura de dados pelo PC, e também o projeto de desenvolvimento de um módulo para instrumentação científica (MOPI) baseado nessa tecnologia.

[1] Ravi Budruk, Don Anderson, and Tom Shanley. *PCI Express System Architecture*. MindShare, Inc, 2nd edition, 2003.

[2] Jan Axelson. *USB Complete*. Lakeview Research LLC, 3rd edition, 2006.

[3] Stephen Bronw and Jonathan Rose. *FPGA and CPLD Architectures: A Tutorial*. *University of Toronto*, 1996.

[4] Analog Devices. *Datasheet*. http://www.analog.com.

[5] G. N. Santos e P. D. Batista. *Caracterização de um dispositivo sintetizador digital de sinais (DDS) como um gerador de sinais*. *Nota Técnica–CBPF*, 2011.

[6] Altera Corporation. *IP Compiler for PCI Express - User Guide*, 2011.

[7] Altera Corporation. *Nios II Processor Reference Handbook*, 2011.

[8] Analog Devices. *Intefacing High Speed ADCs via SPI – AN877*. http://www.analog.com.

[9] Altera Corporation. *IEEE 1149.1 JTAG Boundary-Scan Testing in Altera Devices - AN39*. 2005.

[10] Tom Shanley and Don Anderson. *PCI System Architecture*. MindShare, Inc, 4rd edition, 1999.

[11] William H. Press. *Numerical Recipes:The art of Scientific Computing*. Cambridge University Press, 3rd edition, 2007.

[12] Altera Corporation. *Arria II GX FPGA Development Kit - User Guide*, 2009.

[13] Altera Corporation. *Quartus II Handbook Version 11.0*, 2011.

[14] Stephen Bronw and Zvonko Vranesic. *Fundamentals of Digital Logic with VHDL Design*. McGraw-Hill, 2nd edition, 2005.

[15] Deepak Kumar Tala. *Verilog Tutorial*. http://www.asic-world.com/verilog/veritut.html.

[16] Jungo Ltd. *WinDriver PCI/PCI Express/PCMCIA - Quick Start Guide*, 2011.

[17] Altera Corporation. *SOPC Builder - User Guide*, 2010.

NOTAS TÉCNICAS é uma publicação de trabalhos técnicos-científicos.
Pedidos de cópias desta publicação devem ser enviados aos autores ou ao:

Centro Brasileiro de Pesquisas Físicas
Área de Publicações
Rua Dr. Xavier Sigaud, 150 – 4º andar
22290-180 – Rio de Janeiro, RJ
Brasil
E-mail: socorro@cbpf.br/valeria@cbpf.br
http://www.biblioteca.cbpf.br/index_2.html

NOTAS TÉCNICAS is a journal of technical reports.
Requests for copies of these reports should be addressed to:

Centro Brasileiro de Pesquisas Físicas
Área de Publicações
Rua Dr. Xavier Sigaud, 150 – 4º andar
22290-180 – Rio de Janeiro, RJ
Brazil
E-mail: socorro@cbpf.br/valeria@cbpf.br
http://www.biblioteca.cbpf.br/index_2.html