TUTORIAL DE PROTOTIPAÇÃO UTILIZANDO O FLUXO DE PROJETO XILINX

Ricardo Guazzelli / Fernando Moraes
09/maio/2012

Este tutorial tem por objetivo apresentar o fluxo de projeto de prototipação da Xilinx, baseado no ambiente *PlanAhead*. Ao longo do tutorial é demonstrado como utilizar o *PlanAhead* para desenvolver, sintetizar e implementar circuitos a partir de arquivos RTL. Além disso, é demonstrado o uso do ambiente *ChipScope*, utilizado para validação de circuitos já carregados no FPGA.

Como estudo de caso, é utilizada a plataforma *HardNoC*[1], uma plataforma genérica para avaliação de NoCs (*Network-on-a-Chip*). A NoC usada nesse tutorial foi uma NoC Hermes configurada como *credit-based*, com dimensões 2x3.

SUMÁRIO

[1] http://www.inf.pucrs.br/moraes/my_pubs/papers/2012/spl_2012_hardnoc

1. Configurando o Ambiente

1) Inicialmente, é necessário carregar os arquivos fontes da HardNoc. Para carregar os arquivos execute (ou baixe da página da disciplina):

   $ wget http://www.inf.pucrs.br/moraes/prototip/HardNoC_tutorial.zip

   $ unzip HardNoC_tutorial.zip

   $ cd HardNoC_tutorial

2) Execute no terminal o comando abaixo para mapear os módulos das ferramentas do GAPH:
   $ source /soft64/source_gaph

3) Após este mapeamento, carregar o módulo com os caminhos das ferramentas XILINX:

   $ module load ise

2. Criando um Novo Projeto

1) Execute o PlanAhead (**$ planAhead**) - **Figura 1**.

Figura 1 - Interface inicial do PlanAhead.

2) Selecionando *Create New Project* a tela da Figura 2 é exibida. Selecionar *next.*

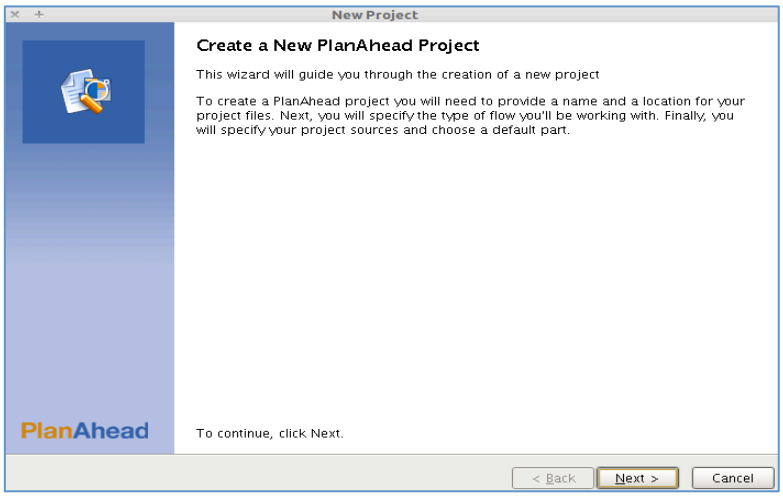


Figura 2 - Iniciando a criação de um projeto.

3) Na próxima tela é especificado o nome e o caminho do Projeto (**Figura 3**). Como exemplo, o nome do projeto foi definido como HardNoC e a pasta do usuário (/home/<username>) como o caminho do projeto. Clique em Next.

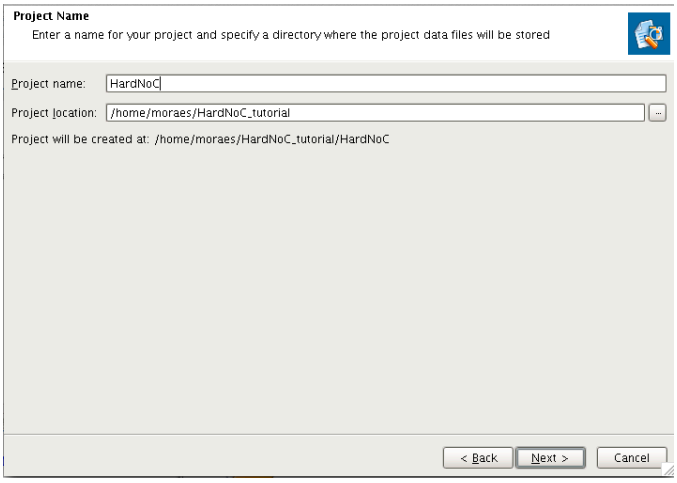
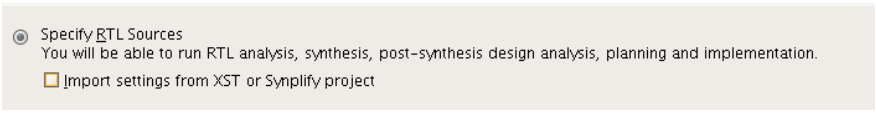


Figura 3 - Determinando o caminho do projeto.

4) Nesta janela indicamos que iremos importar os VHDLs - Clique em Next.



5) Nesta etapa iremos adicionar os arquivos .vhd que descrevem a HardNoC (**Figura 4**).

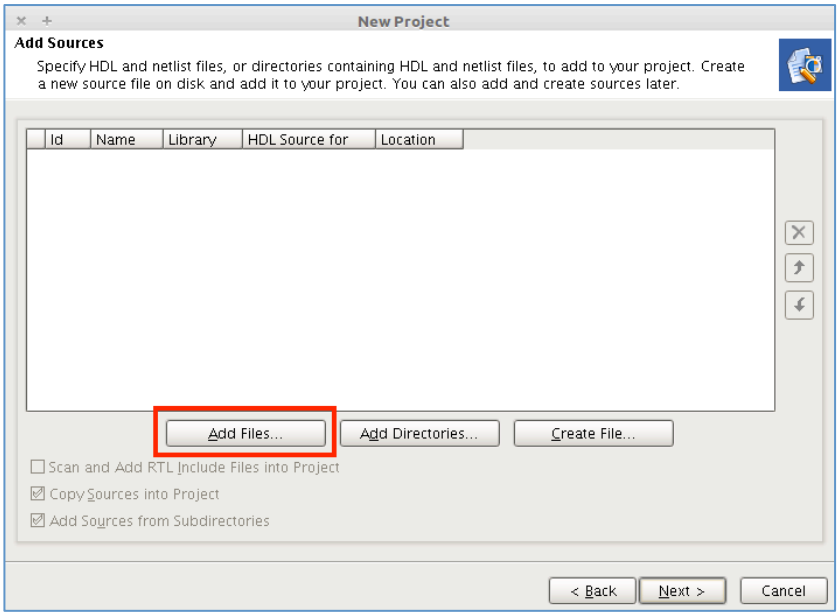


Figura 4 - Inserindo arquivos fontes.

- Selecione Add Files e adicione os arquivos .vhd que descrevem a HardNoC.

  É preciso adicionar dois arquivos .vhd existentes na raiz da pasta hardnoc_tutorial e todos os arquivos .vhd existentes nas subpastas **IPs** e **NOC**.

  Os arquivos selecionados serão exibidos na tela indicando suas propriedades (Figura 5). As opções ***Scan and Add RTL includes Files into Project*** e ***Copy Sources into Project*** estão selecionadas de acordo com o padrão do *PlanAhead*.

- Após incluir todos os os arquivos VHDL **clique em Next.**

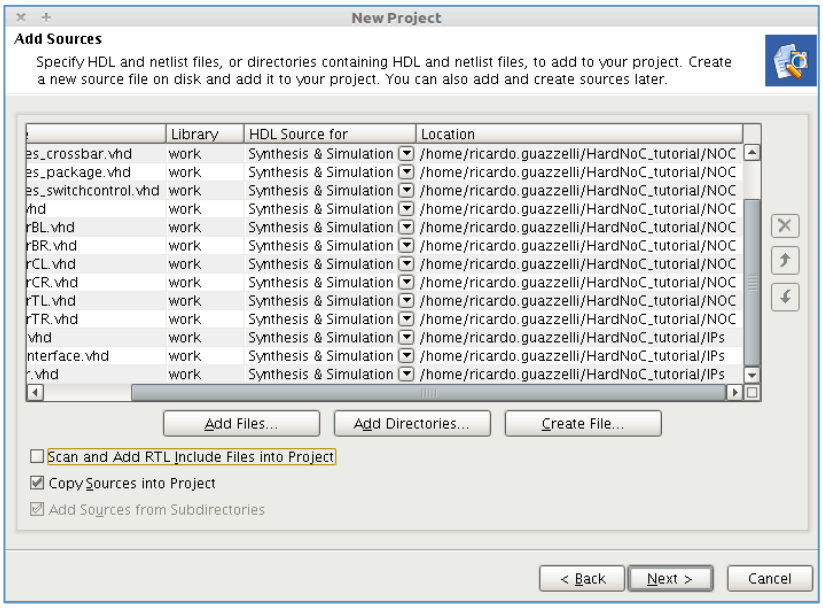


Figura 5 - Arquivos selecionados.

6) A próxima etapa é destinada a adicionar IP configuráveis. Estes IPs são módulos parametrizáveis gerados por outras ferramentas da Xilinx (Core generator). Como a HardNoC não usa nenhum IP configurável, não será necessário adicionar nenhum arquivo nessa etapa. *Clique em Next.*

7) Nesta próxima etapa adicionamos arquivos .ucf, que indicam restrições de design, pinos de I/O e parâmetros de floorplanning. O arquivo .ucf não será adicionada agora, pois será gerado durante este tutorial. *Clique em Next*.

8) Após a definição dos arquivos fontes, devemos especificar qual FPGA iremos usar no projeto (**Figura 6**). Nesse tutorial foi usado uma Xilinx Virtex-4 (xc4vlx25ff668-10) para prototipação. Abaixo seguem os parâmetros a inserir: **Product Category: General Purpose; Family: Virtex-4; Sub-family: Virtex-4 LX ; Package: FF668; Speed Grade: -10; Temp Grade: C. Selecione o segundo dispositivo da lista (xc4vlx25ff668-10) e *next.***

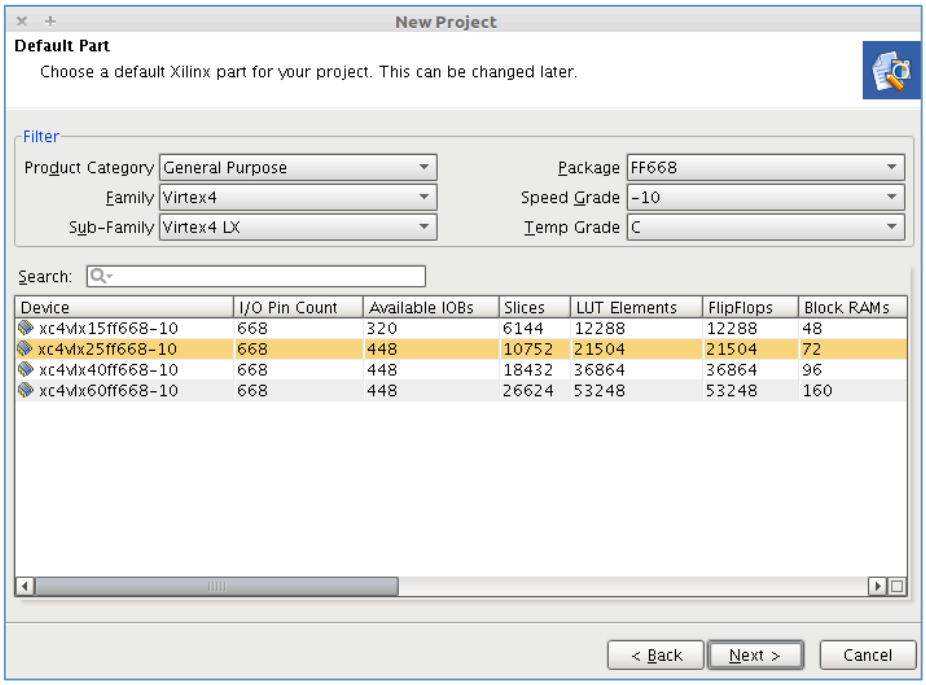


Figura 6 - Selecionando FPGA.

9) Concluídos os passos acima, o *PlanAhead* exibirá um sumário com informações das etapas anteriormente (**Figura 7**). **Verifique se todas as informações estão corretas Clique em Finish.**

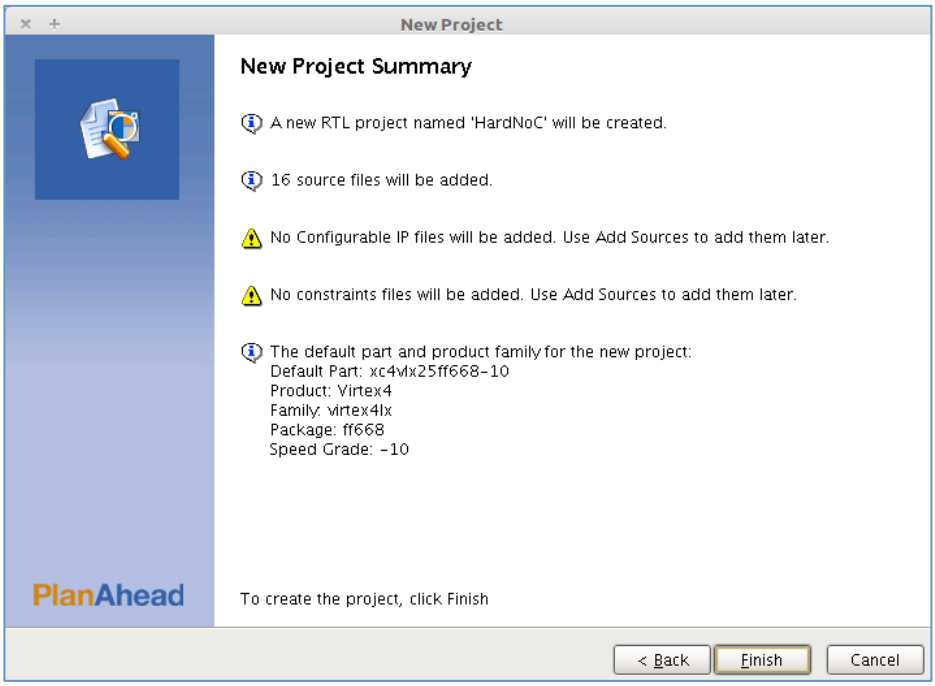


Figura 7 - Sumário do projeto.

O projeto será criado. A janela do projeto será carregada e informações e opções do projeto serão exibidas (Figura 8). Na janela ***Project Summary***, o planAhead exibe um sumário informando a *status* de projeto, incluindo informações sobre sínteses, implementações, recursos utilizados na FPGA e restrições de *timing*. Na janela ***Sources***, se encontram todos os arquivos pertencentes ao projeto. Finalmente, à esquerda se encontram as opções de síntese, implementação e programação/validação, que serão abordadas a seguir.

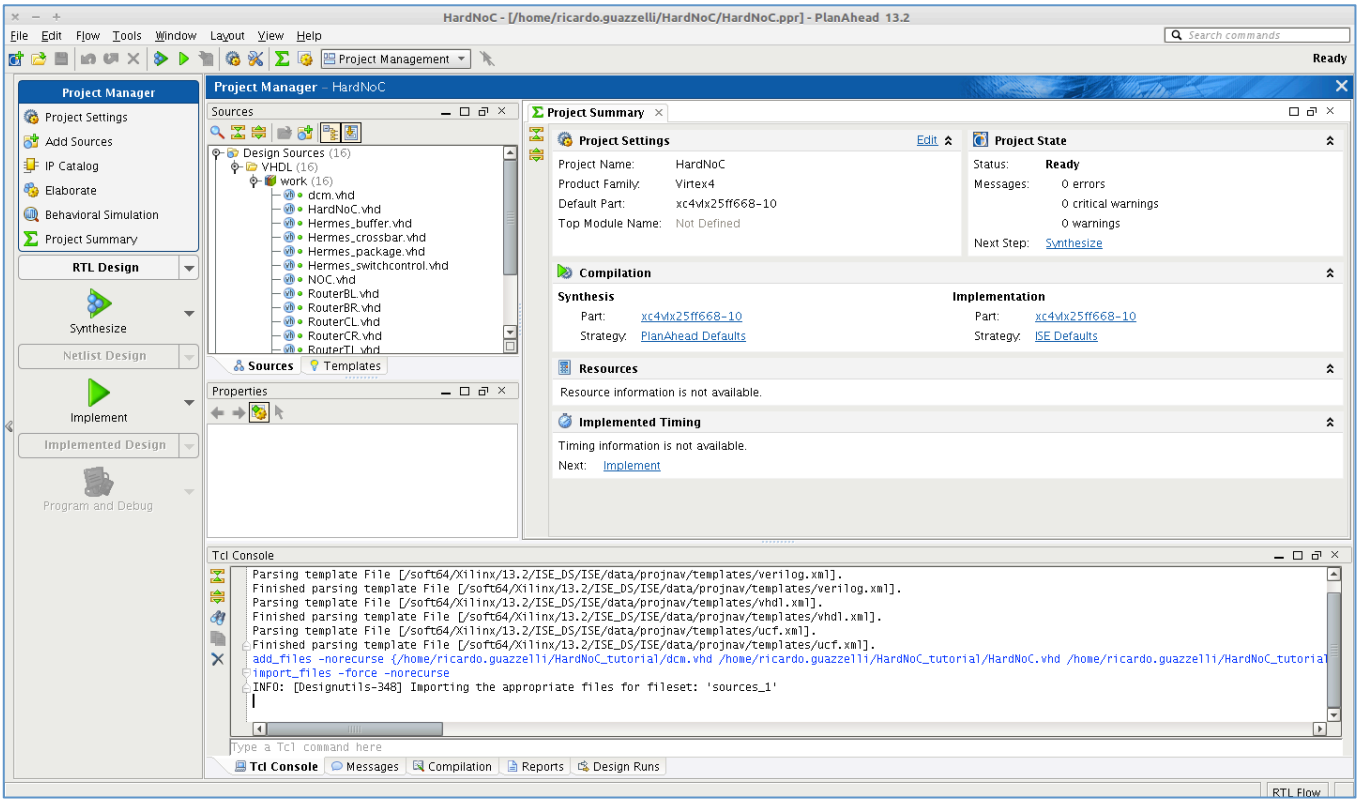


Figura 8 - Projeto criado.

3. RTL Design

A opção RTL Design tem por funcionalidade exibir um esquemático baseado nos arquivos RTL adicionados no projeto. Isto permite ao usuário visualizar e validar a hierarquia do projeto. Além disto, esta etapa permite a criação do arquivo de restrições de projeto, *.ucf* (*User Constraint File*).

1) Para carregar o esquemático clique em RTL Design (Figura 9).

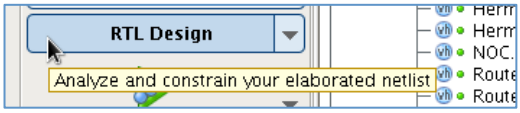


Figura 9 - RTL Design.

Surgirá uma janela solicitando o topo da hierarquia do circuito.

2) Clique em "*Select the top module of your design*" como indicado na Figura 10.

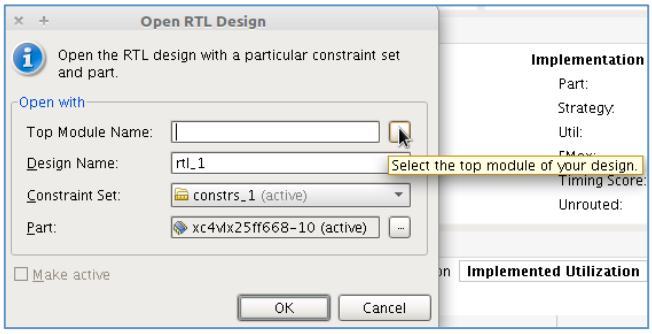


Figura 10 - Definindo topo da hierarquia.

3) Selecione a opção "*HardNoC*", e clique em OK duas vezes.

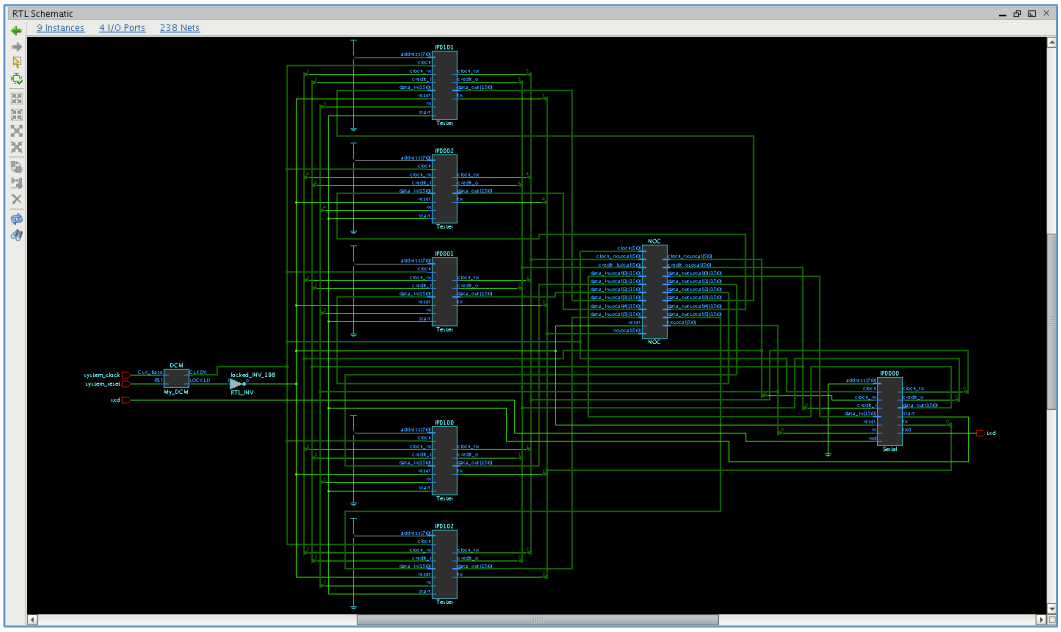


Figura 11 - Esquemático gerado.

Inicialmente serão exibidos apenas as instâncias pertencentes ao topo da hierarquia da HardNoC (Figura 11). Porém é possível visualizar lógicas que se encontram em camadas mais abaixo da hierarquia. Como exemplo, iremos acessar a lógica no interior da instância NOC.

4) Selecione o componente NOC.

5) Selecione Expand all logic inside selected instance nas opções à esquerda da janela ou duplo clique no componente.

Será exibido a lógica correspondente à NOC (Figura 12).

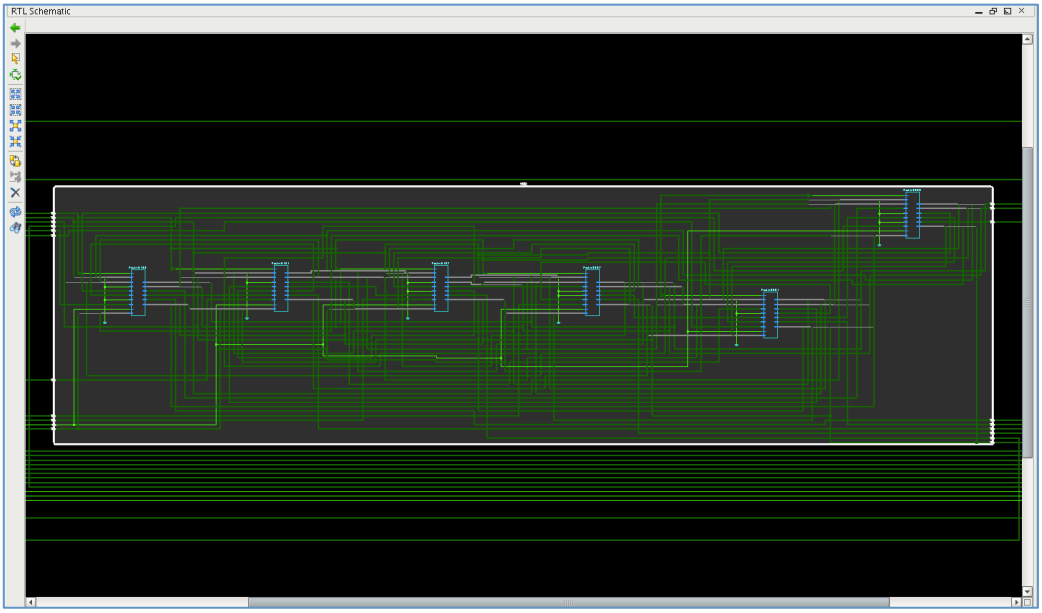


Figura 12 - Esquemática da NOC.

6) Para retornar ao esquemático anterior selecione Previous schematic  .

O esquemático pode ser usado para definir os pinos de E/S que serão usado na FPGA. Na *HardNoC*, apenas 4 sinais de interface necessitam ser especificados: *system_clock*, *system_reset*, *txd* e *rxd*. Abaixo segue a relação dos sinais com seus respectivos pinos:

Sinal de E/S na HardNoC	Pino na FPGA
system_clock	AE14 (Oscilador 100MHz)
system_reset	B6 (Push-button central)
txd	W1 (Serial out)
rxd	W2 (Serial in)

Observação: a documentação do kit indica a relação dos pinos da FPGA com os periféricos do kit.

Como exemplo, iremos relacionar o pino AE14 com o sinal *system_clock*.

7) Selecione no esquemático o sinal de I/O (Figura 13).

8) Na janela I/O Port Properties, preencha o campo Site com o pino desejo (Figura 14).

9) Clique em Apply.

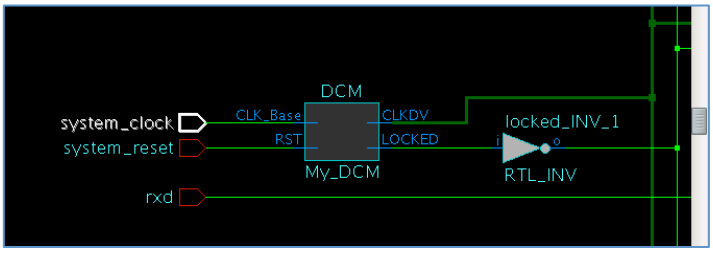


Figura 13 - Sinal de I/O no esquemático.

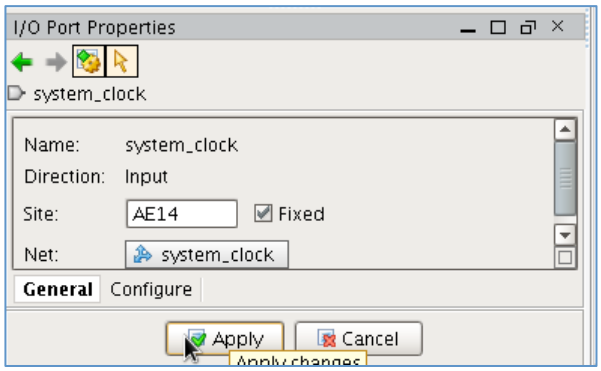
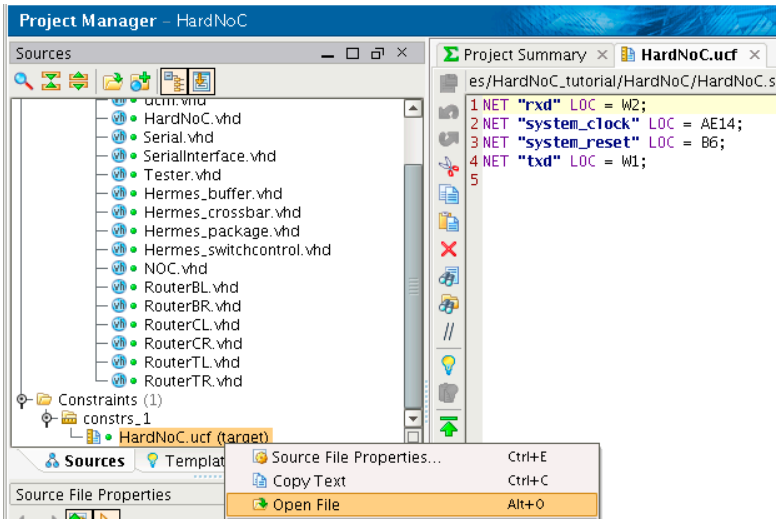


Figura 14 - Propriedades de I/O.

10) Repita o processo para outros 3 sinais, salve o projeto, e depois abra o arquivo *HardNoC.ucf* na janela de sources:

```
NET "rxd" LOC = W2;
NET "system_clock" LOC = AE14;
NET "system_reset" LOC = B6;
NET "txd" LOC = W1;
```



11)Inserindo restrições de temporização.

Ao se inserir restrições de temporização, o *PlanAhead* tentará implementar um circuito que atenda às restrições impostas e em caso de falha, irá informar em qual parte do circuito não foi possível atender a restrição. Como exemplo de restrição, iremos definir a frequência de entrada do circuito. Adicione as seguintes linhas no arquivo *ucf* (Figura 15):

```
NET "system_clock" TNM_NET = "TNM_clk_100MHz";
TIMESPEC TS_clk_100MHz = PERIOD "TNM_clk_100MHz" 10 ns HIGH 50 %;
```

Figura 15 - Definição de restrições de *timing*.

Time name net (TNM_NET) define um *label* para uma NET (sinal) que receberá uma restrição temporal. Mais de um sinal pode pertencer este *label*, podendo-se assim associar a um dado conjunto de sinais uma restrição comum de temporização. Na linha seguinte cria-se uma restrição temporal (TIMESPEC) sobre o *label* previamente definido. Neste caso a restrição é de PERÍODO, igual a 10 ns.

4. Síntese Lógica

Na etapa de síntese lógica é gerados o *netlist* (.ngc) correspondente ao circuito que será prototipado. O *netlist* contém os elementos básicos do FPGA, como LUTs, flip-flops, memórias, etc. Como iremos utilizar a ferramenta de *floorplanning*, devemos evitar que a síntese otimize a hierarquia do circuito.

1) Clique à direta de Synthesize (Figura 16).

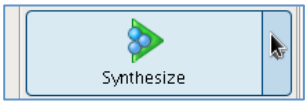


Figura 16 - Síntese

2) Selecione Synthesis Settings.

Será exibida uma janela com as opções de síntese.

3) Selecione *Choose Strategy* (Figura 17).

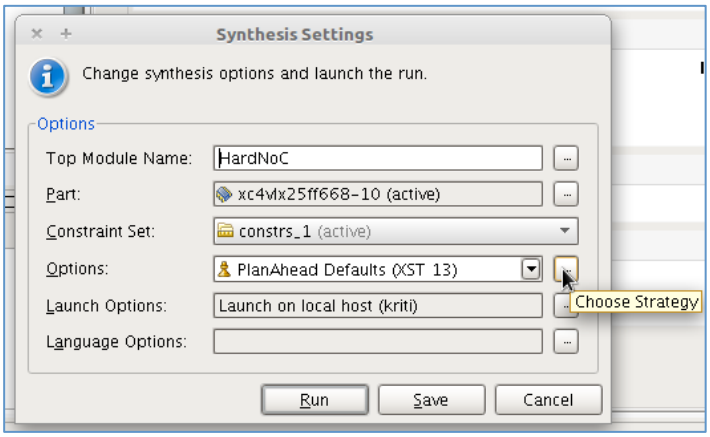


Figura 17 - Opções de síntese.

Serão exibidas as opções de estratégia de síntese (Figura 18). Nessa janela é possível determinar quais otimizações serão usadas durante a síntese;

- **Para evitar a otimização de hierarquia selecione *rebuilt* na opção -netlist_hierarchy.**
- **Defina a codificação das máquinas de estado: *one-hot* na opção -fsm_encoding.**

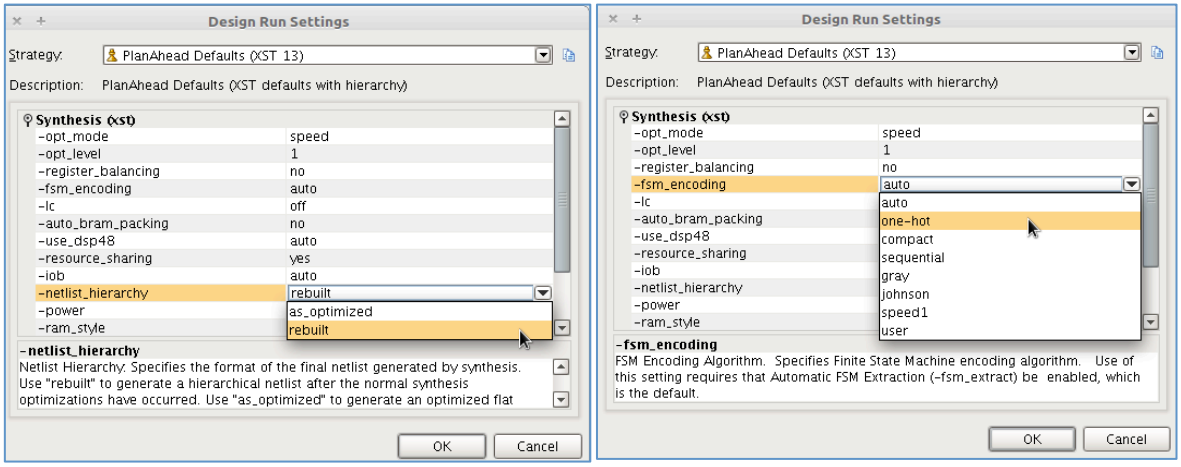


Figura 18 - Opções de síntese lógica.

4) Selecione OK e RUN

A síntese será iniciada. Caso hajam erros na descrição do circuito, a síntese será interrompida e o terminal **"Messages"** indicará onde ocorreu o erro. Caso contrário, será exibida a janela abaixo (Figura 19).

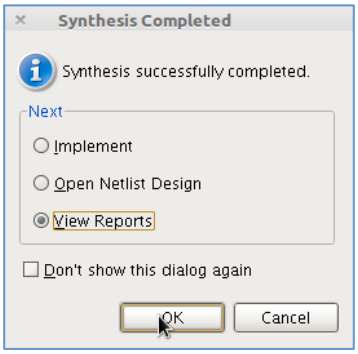


Figura 19 - Síntese completa.

5) Selecione View Reports e clique em OK.

A lista de relatórios será exibida (Figura 20).

6) Acesse o arquivo "XST Report" gerado.

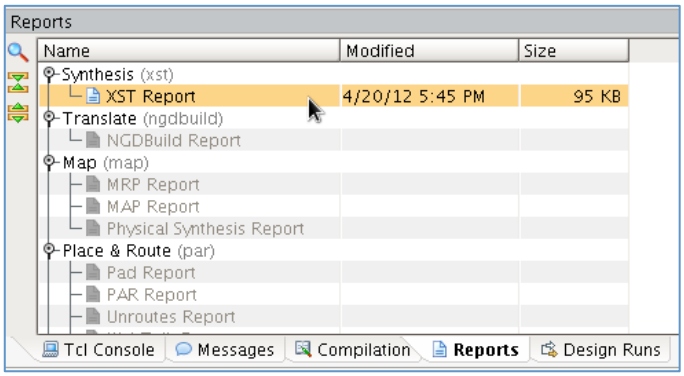


Figura 20 - Relatório de Síntese Lógica gerado.

Analisando o relatório, responda:

- **Qual é a porcentagem estimada de uso de área do dispositivo? (device utilization summary).**
- **Qual a frequência estimada? (timing summary)**

Finalizando os passos anteriores, está finalizada a etapa de síntese lógica.

5. Floorplanning

Com a síntese lógica completa, teremos uma estimativa dos recursos necessários para prototipar o circuito. Usando a ferramenta de *floorplanning*, é possível determinar qual a área que cada componente do circuito ocupará na FPGA.

1) **Para carrega a ferramenta de *floorplanning* selecione Open Netlist Design (em Netlist Design).**
2) **Clique em OK.**

A janela **Device** será exibida e a aba **Netlist** (Figura 21). **Device** indica a posição de LUTs, BlockRAMs e DSP Slices na FPGA. Por sua vez, **Netlist** indica os componentes do projeto a serem mapeados na FPGA. Caso não mapeado, o símbolo  será exibido ao lado do nome do componente. Se o componente já foi mapeado, o símbolo  indicará tal informação.

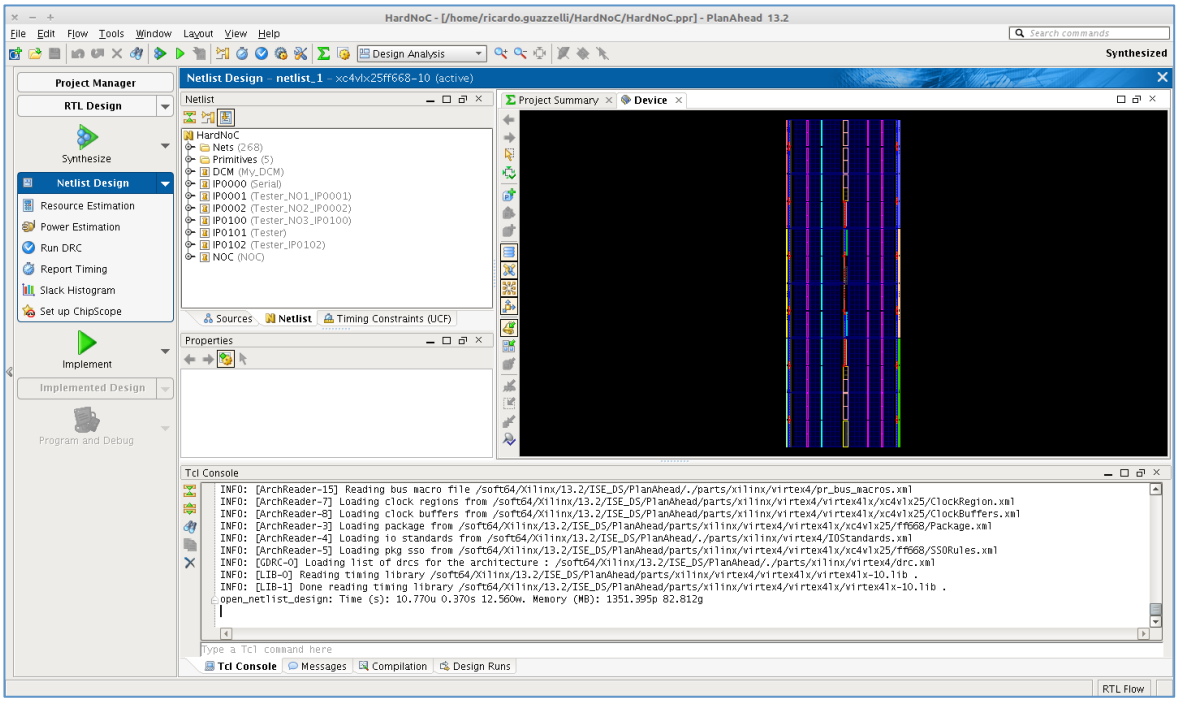


Figura 21 - Floorplanning.

Componentes podem ser mapeados individualmente ou em conjunto. Como exemplo, iremos mapear cada roteador da NoC com seu respectivo IP. Para mapear componentes em conjunto:

3) **Selecione um roteador com o seu respectivo IP (Router0000 e IP0000) pressionando ctrl (Figura 22).**

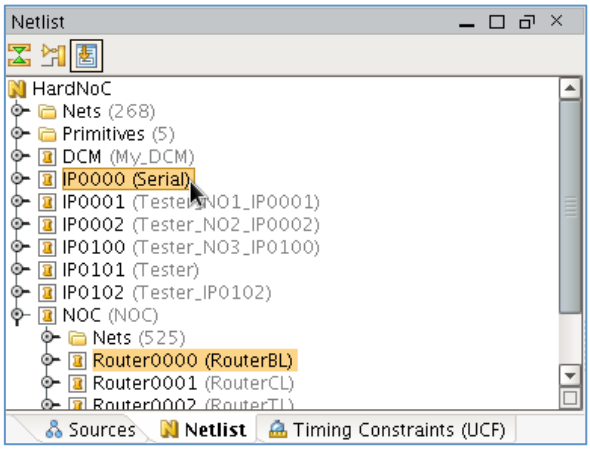


Figura 22 - Mapeamento conjunto.

4) **Clique com o botão direito sobre um dos componentes selecionados.**

5) **Selecione Draw PBlock.**

6) **Na janela Device, use o mouse para determinar a área do componente (Figura 23).**

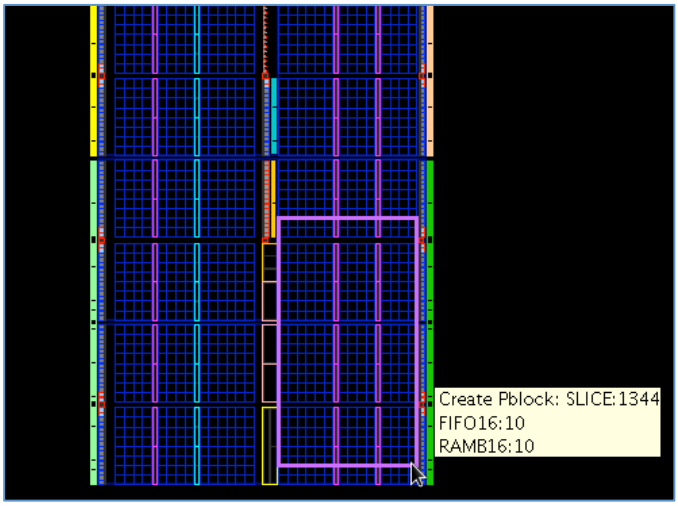


Figura 23 - Desenhando bloco.

7) **Clique em OK.**

Router0000 e o *IP0000* serão mapeados na área determinada. A linha **verde** na Figura 24 indica a conexão entre o *IP0000* com blocos de I/O indicados no arquivo .ucf. Caso esta linha não esteja visível no seu projeto:

8) **Clique em Show I/O Nets**  .

Tendo o bloco posicionado como na Figura 24, é necessário averiguar se o bloco possui componentes necessários para implementação. Na janela *PBlock Properties* (Figura 25) é possível visualizar o número de componentes na área especificada e o número de componentes necessárias para implementação. Deve-se salientar que o "número de componentes necessárias" é apenas uma estimativas gerada pela síntese. Na etapa de implementação esse número pode aumentar e o *PlanAhead* pode acusar erros caso o bloco não suporte o componente. Para evitar isso, é aconselhável mapear os componentes em blocos maiores que o estimado pela síntese.

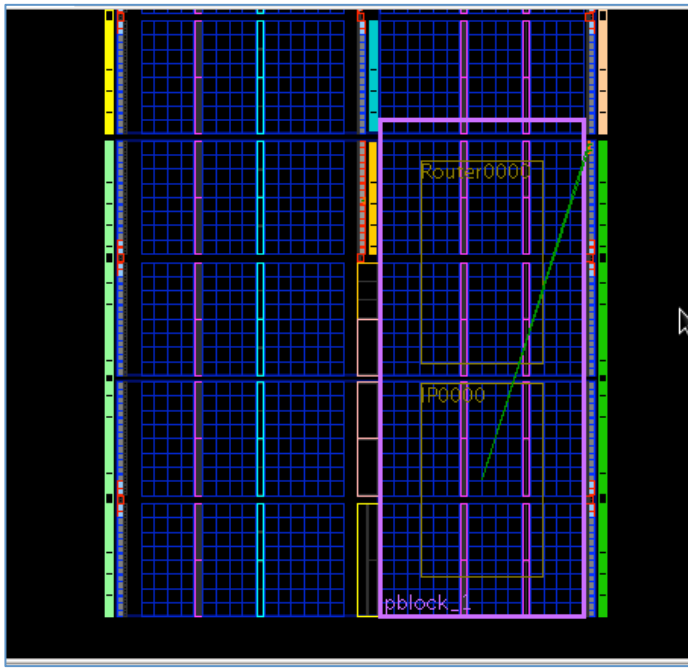


Figura 24 - Router 0000 e IP 0000 mapeados.

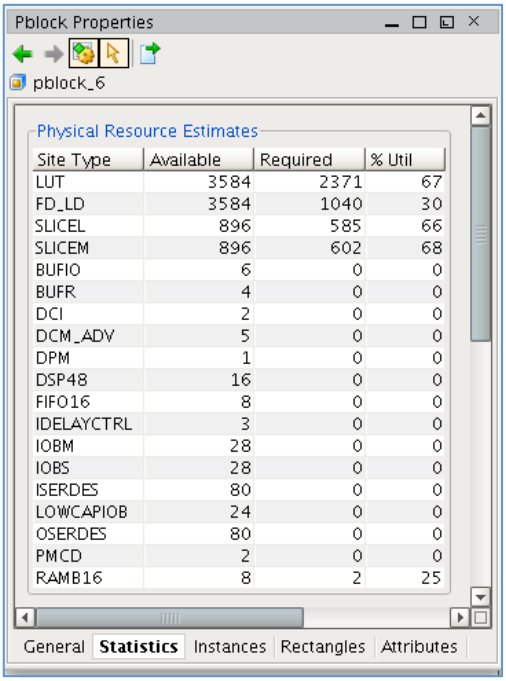


Figura 25 - Recursos físicos do bloco.

9) Repita esse processo com os os outros roteadores/IPs. Posicione os roteadores de acordo com os seus roteadores vizinhos.

O *floorplanning* final deve ser semelhante ao da Figura 26. As linhas **laranjas** indicam as sinais que conectam um bloco a outro. Quanto mais espessa a linha, mais sinais há entre os blocos. Portanto, é aconselhável um posicionamenteo adjacente entre estes blocos, facilitando o roteamento do circuito.

10) Salvar e observar as alterações no arquivo UCF.

O que foi inserido no arquivo UCF?

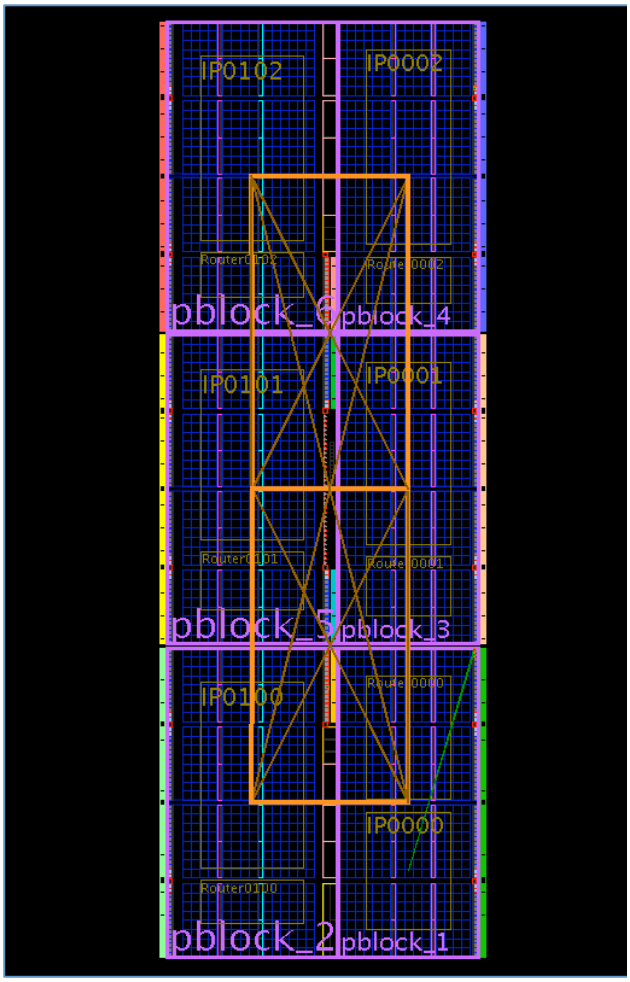


Figura 26 - Floorplanning completo do circuito HardNoC.

6. Criando um módulo do ChipScope

A ferramenta ChipScope permite visualizar os sinais internos do FPGA durante a execução para um determinado conjunto de entradas. Para efetuar uma validação via ChipScope, é necessário inicialmente especificar quais sinais queremos observar no circuito.

1) Para criar um módulo do ChipScope selecionar *Set up ChipScope* nas opções de *Netlist Design*.

Uma janela será exibida. Clique em Next. Na próxima janela, iremos definir quais sinais internos da HardNoC queremos observar. Como exemplo, iremos observar os sinais ***IP0000/data_out*** e ***IP0000/tx*.**

2) Selecione Add/Remove Nets, como indicado na Figura 27.

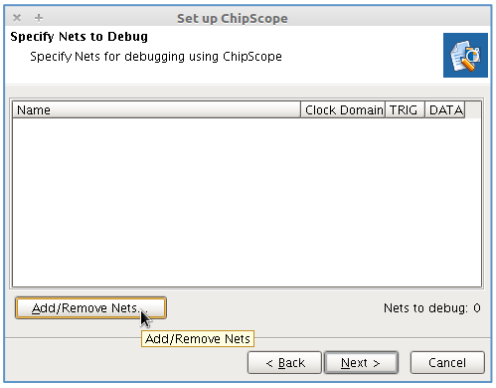


Figura 27 - Adicionando sinais ao ChipScope.

3) No campo Find Net Criteria, pesquise pelo sinal IP0000/data_out (Figura 28).

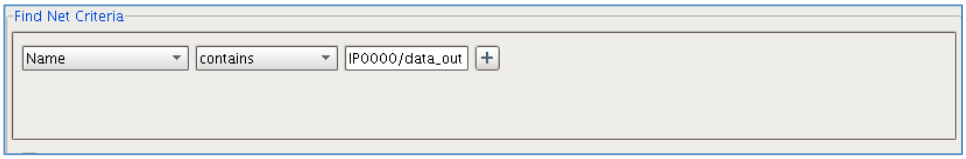


Figura 28 - Pesquisando sinais internos.

4) Clique em Find.

O sinal desejado será exibido no campo **Find Results** (Figura 29)**.**

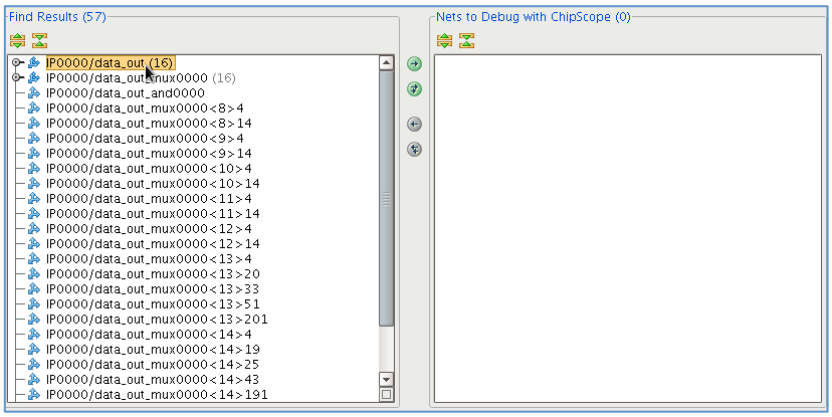


Figura 29 - Selecionando sinal.

5) Clique em  **para indicar qual sinal será avaliado.**

6) Repita os passos 4 e 5 pesquisando pelo sinal IP0000/tx.

Após a seleção, os dois sinais devem ser exibidos no campo ***Nets to Debug with ChipScope*,** como indicado na Figura 30.

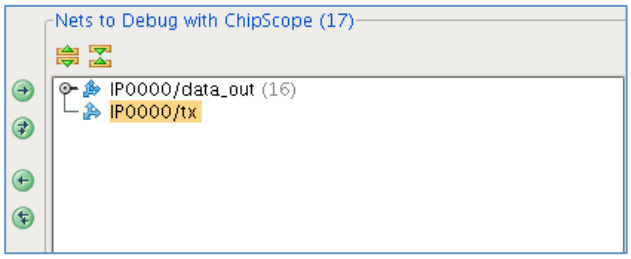


Figura 30 - Sinais selecionados.

7) Após selecionar os sinais, para isto clique em OK.

Será exibido a janela anterior, porém com os sinais selecionados **com seus respectivos domínios de clock** (Figura 31).

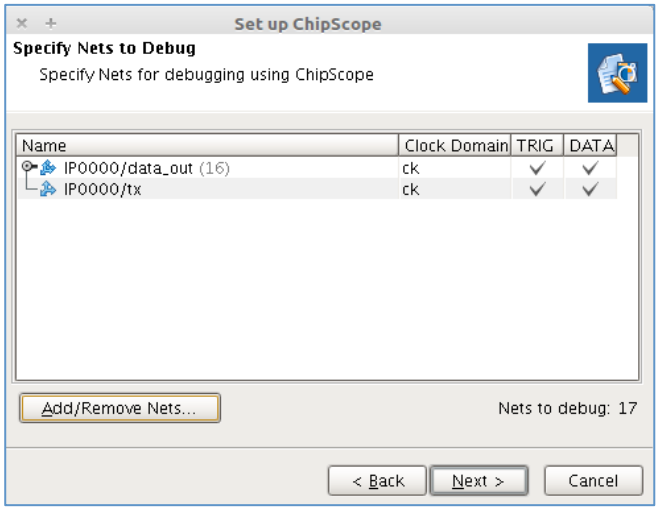


Figura 31 - Sinais selecionados com domínios de clock.

8) Clique em Next.

Será exibido um sumário do módulo a ser criado (Figura 32).

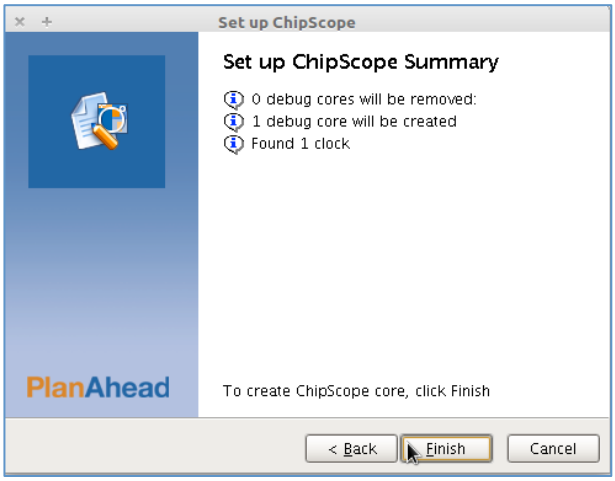


Figura 32 - Sumário do ChipScope.

9) Clique em Finish.

O PlanAhead criará o módulo do ChipScope e a aba ChipScope será adicionada ao projeto (Figura 33).

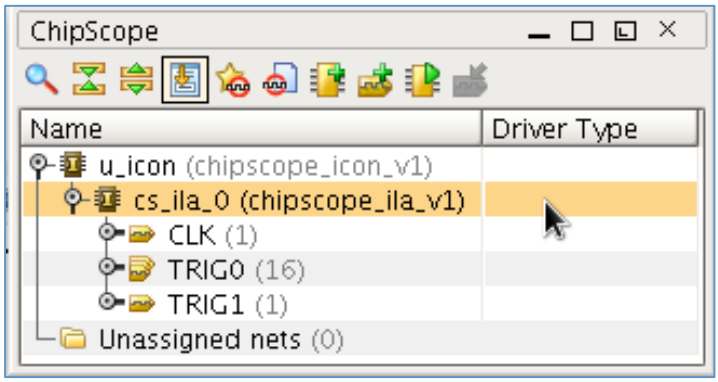


Figura 33 - Aba do Chipscope.

10) Clique com o botão direito do mouse sobre cs_ila_0 (chipscope_ila_v1).

11) Selecione Implement ChipScope Debug Cores.

O PlanAhead irá implementar o módulo do ChipScope no projeto e a etapa de síntese física pode ser iniciada.

7. Síntese Física (Implementação)

Nesta etapa, o *netlist* gerado nas etapas anteriores (circuito HardNoC e componentes do Chipscope) terá seus elementos posicionados (*placement*) e conectados (*routing*) no FPGA. Da mesma forma que na síntese lógica, a implementação possui opções de otimização (área, velocidade, roteamento, *timing*, etc).

1) Para visualizar as opções de implementação Clique à direita em *Implement* (Figura 34).

Figura 34 - Implementação.

2) Selecione *Implementation Settings*.

Será exibida uma janela com opções de implementação.

3) Clique em *Choose Strategy* (Figura 35).

Será exibida uma janela com as opções de estratégia de implementação (Figura 36). No nosso caso, não modificaremos nas opções definidas.

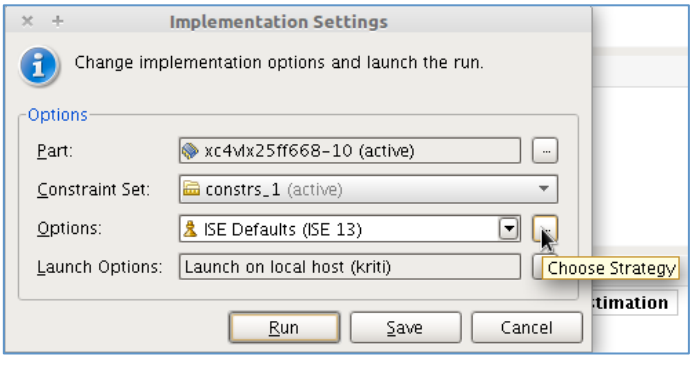


Figura 35 - Opções de implementação.

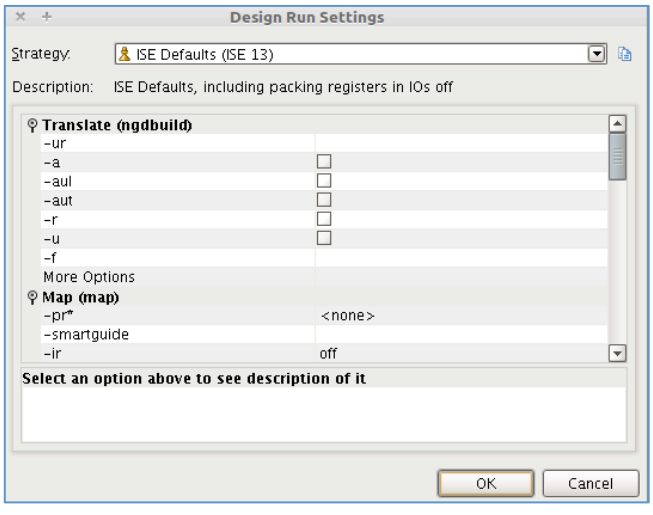


Figura 36 - Opções de estratégia de implementação.

4) **Clique em OK e depois em RUN**

A implementação será iniciada. Caso hajam erros, a implementação será interrompida e será indicado no terminal **Messages** onde ocorreu o erro. Caso contrário, será exibida a janela da Figura 37, e a implementação física está assim concluída.

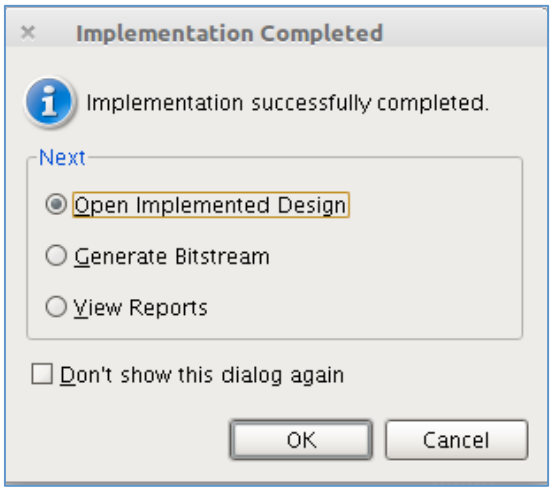


Figura 37 - Implementação completa.

5) **Concluído a etapa de implementação, devemos gerar o arquivo *bitstream*, que será carregado na FPGA contendo a HardNoC e o módulo do ChipScope criado. Para gerar o arquivo .bit selecione *Generate Bitstream*. e clique OK 2 vezes.**

O arquivo .bit será criado e uma janela indicará o fim da geração (Figura 38).

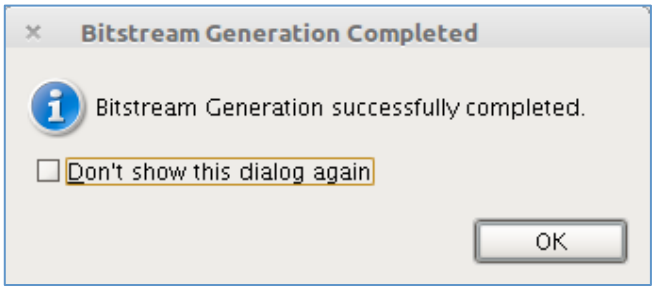


Figura 38 - Arquivo .bit gerado.

Verificar se o *bitstream* foi gerado, localizando-o no diretório:

```
<user>/HardNoC_tutorial/HardNoC/HardNoC.runs/impl_1
```

CRIADO O BITSTREAM SAIR DO PLANHAEAD. Neste ponto, cada grupo deve realizar as etapas seguintes na máquina onde está a placa com Virtex-4, dado que temos uma só plataforma destas no grupo.

NÃO PRECISA IR FISICAMENTE NA MÁQUINA DE TESTES!

REALIZAR ssh –X <user>@GAPHL11 ou ssh –X <user>@10.32.162.130

8. Prototipação/validação via ChipScope

Estando o terminal na máquina de testes, reconfigure o ambiente, abra novamente o planAhead, e abra o projeto novamente. Utilizaremos *ChipScope* para carregar o arquivo .bit e efetuar a validação. A inicialização do *ChipScope* pode ser efetuada de duas formas.

1) **Pelo PlanAhead:**

- Clique em *Program and Debug* (**Figura 39**).
- Selecione *ChipScope Analyzer*.

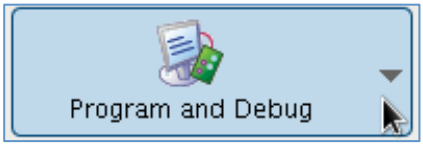


Figura 39 - Iniciando ChipScope via PlanAhead.

2) **Pelo terminal (com os módulos do ISE carregados) executando:**

- $ analyzer.

O *ChipScope* será carregado e a tela inicial é exibida (Figura 40).

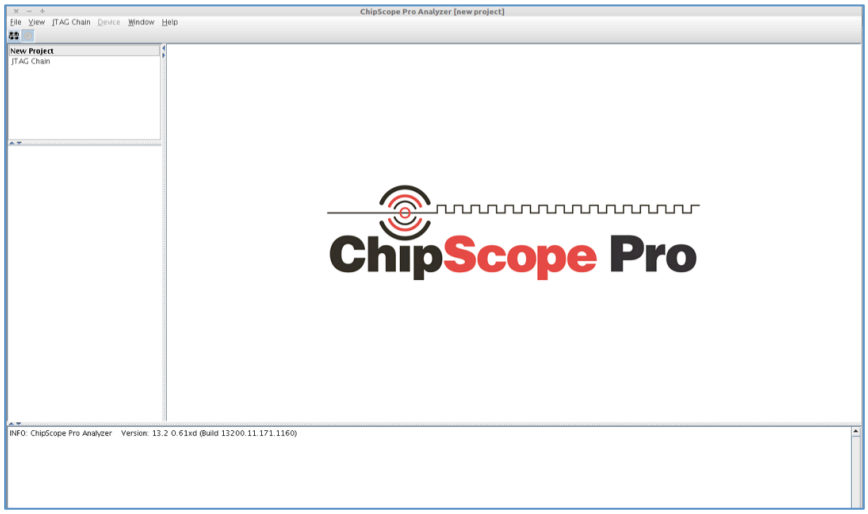


Figura 40 - Interface inicial do ChipScope.

Antes de utilizar o ChipScope, conecte a placa de prototipação ao PC utilizando um *Plataform Cable USB*. O LED de *status* do *Plataform Cable USB* deve estar **verde**, indicando a conexão foi estabelecida.

3) **Localizar o FPGA na placa, clicando em *Open Cable/Search JTAG Chain***  **.**

O ChipScope tentará localizar a FPGA. Caso a conexão seja estabelecida, a janela abaixo (Figura 41) é exibida.

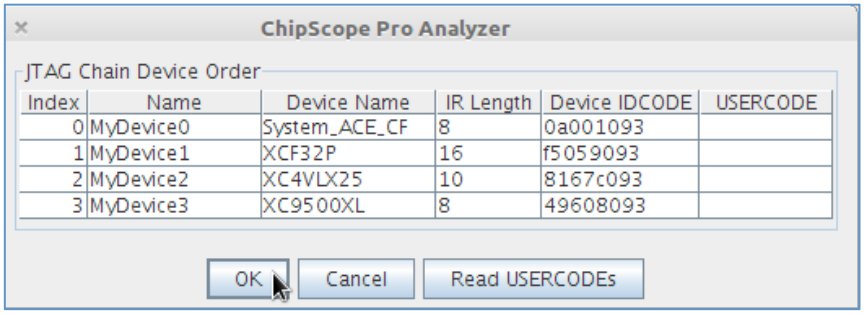


Figura 41 – Dispositivos FPGA reconhecidos na placa de prototipação.

4) **Clique em OK.**

Os dispositivos reconhecidos serão exibidos na parte superior esquerda (JTAG Chain). Agora poderemos carregar o .bit gerado para a FPGA.

5) **Clique com o botão direito sobre DEV:2 MyDevice (XC4VLX25) (Figura 42).**

6) **Selecione Configure.**

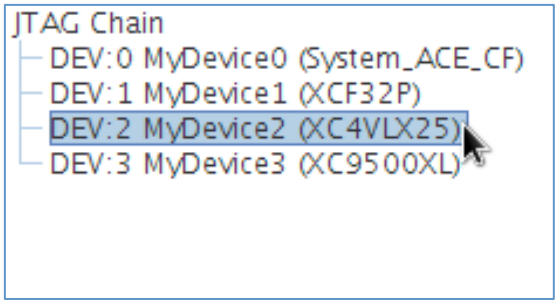


Figura 42 - Configurando FPGA.

Será exibido uma janela para especificar qual .bit será carregado e qual .cdc será usado (Figura 43).

7) **A princípio o Chipscope irá carregar os dados do último projeto sintetizado. Caso isto não ocorra, pode-se definir na própria janela da Figura 43 os dados manualmente:**

a. Clique em *Select New File* em *JTAG Configuration*.
b. Selecione o .bit gerado pelo PlanAhead. O arquivo se encontra na pasta <nome_do_projeto>.runs, dentro da pasta do projeto com o nome *HardNoC.bit*.
c. Clique em Open.
d. Clique em *Select New File* em *Design-level CDC File*.
e. Selecione o .cdc gerado pelo ChipScope. O arquivo se encontra também na pasta <nome_do_projeto>.runs, dentro da pasta do projeto com o nome degub_nets.cdc.
f. Clique em Open.

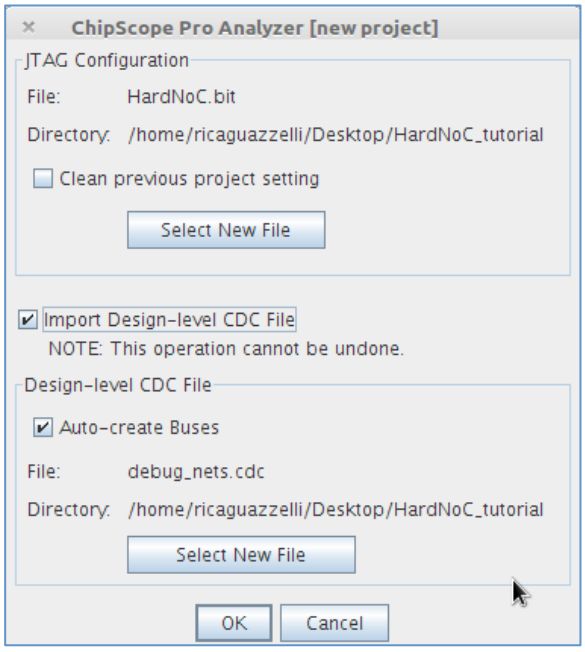


Figura 43 - Especificando arquivos.

8) **Clique em OK.**

A HardNoC será carregada na FPGA e o LED DONE no kit indicará que o carregamento foi efetuado com sucesso. Será exibido o módulo do ChipScope UNIT:0 MyILA0 (ILA) em DEV:2 MyDevice (XC4VLX25).

9) **Na janela Waveform:**

Serão exibidas duas janelas. Na janela **Waveform** será exibida as amostras coletadas pelo ChipScope. Enquanto na janela **Trigger Setup** é possível definir opções de captura de amostras. Isto auxilia o usuário definir quando o módulo do ChipScope deve iniciar a captura das amostras.

Como estamos monitorando dois sinais (*IP0000/tx* e *IP0000/data_out*), apenas será exibidos dois *Triggers* com suas respectivas opções de captura (Figura 44):

- **Value: valor a ser comparado com o sinal.**
- **Radix: base numérica definido no campo Value.**
- **Function: tipo de comparação a ser feito entre Value e o sinal:**
  - **"==" : trigger será disparado quando o valor do sinal for igual ao valor inserido em Value.**
  - **"<>" : trigger será disparado quando o valor do sinal for diferente ao valor inserido em Value.**
  -

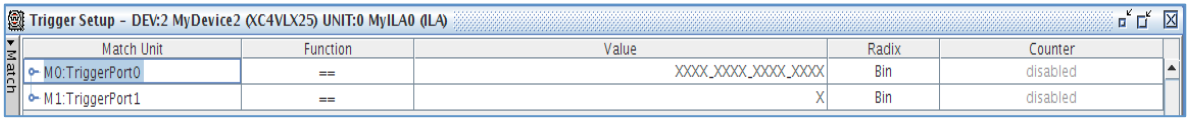


Figura 44 - *Triggers* de captura.

Inicialmente, o campo **Value** estará com todos os bits em **X** (*don't care*). Ou seja, independente do valor do sinal, o *Trigger* será disparado. Como queremos visualizar a entrada de pacotes na NoC, iremos estabelecer para o *trigger* ser disparado quando o sinal *IP0000/data_out* e *IP0000/tx* for diferente de zero.

10) Trigger deve ser configurado como apresentado na Figura 45.

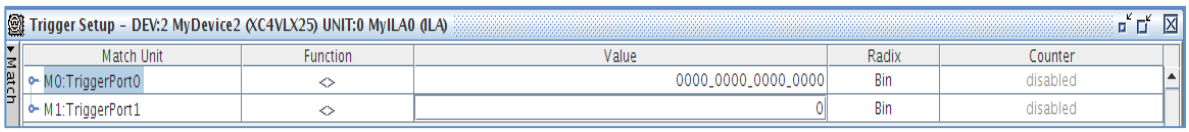


Figura 45 - *Triggers* configurados.

11) Clique em Apply Settings and Arm Trigger → íncone:  .

Na parte inferior da janela Trigger Setup, o ChipScope informará que está aguardando o disparado do *trigger* (Figura 46).

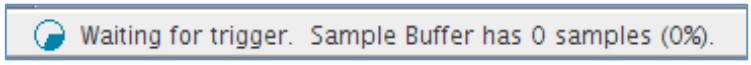


Figura 46 - ChipScope aguardando disparado do *trigger*.

Com o ChipScope "armado", iremos nos comunicar com a FPGA para visualizarmos a nossa iteração com a HardNoC.

Para comunicação, usaremos o terminal serial que está na pasta da HardNoC adquirida via SVN. O software necessita que o Java JDK esteja instalado e uma biblioteca para comunicação serial do Java (*librxtx-java*). Caso a biblioteca não esteja instalada, abra o terminal e execute:

$ sudo aptitude install *librxtx-java*

chmod 777 /dev/ttyS0

12) Neste ponto o projeto HardNoC já está no FPGA, aguardando dados pela serial. Conecte o cabo serial cross-over para estabelecer a conexão física entre a porta serial do PC com a do kit ML403. Acesse o diretório do terminal serial (HardNoC_tutorial/terminal-serial/linux), e execute:

$ cd terminal-serial/linux

$ ./serial para iniciar.

A janela do terminal será exibida (Figura 47).

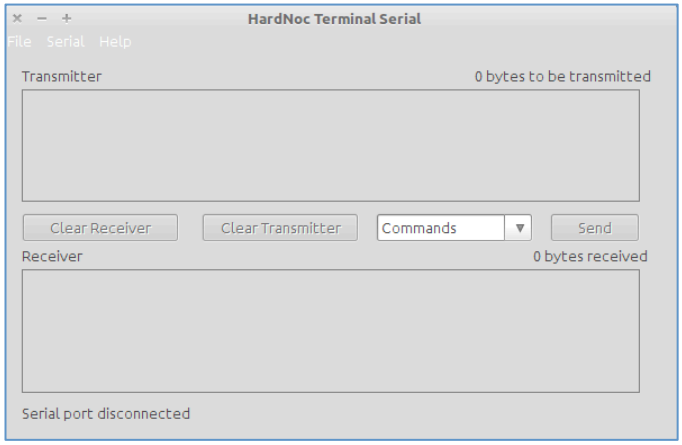


Figura 47 - Terminal Serial

13) Selecione Serial → Configure na parte superior esquerda da janela do terminal serial. Será exibido uma janela com configurações da serial (Figura 48). Configure de acordo com os parâmetros abaixo, e depois OK.

- Port: /dev/ttyS0
- Baud Rate: 4800
- Byte Size: 8 bits
- Parity: No
- Processor: 16 bits
- Stop Bits: 1,0

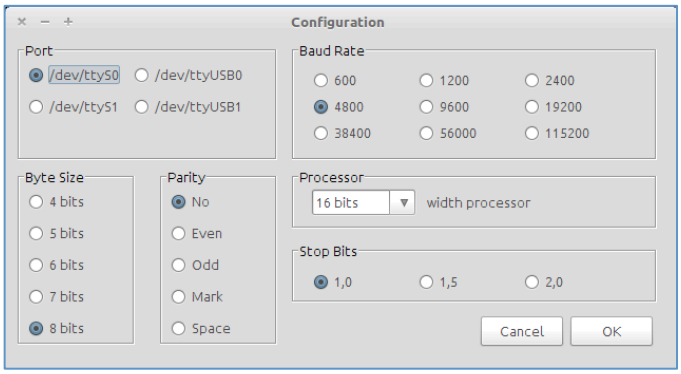


Figura 48 - Configurações da Serial.

14)Selecione Serial → Connect.

Os botões da interface são liberados. Para iniciar a comunicação, é necessário reiniciar a HardNoC e sincronizar a interface serial da HardNoC. Para isso pressione o push-button central do kit ML403 para reiniciar a HardNoC. (quadrado vermelho - Figura 49). – esta ação só é necessária se reiniciar o FPGA.

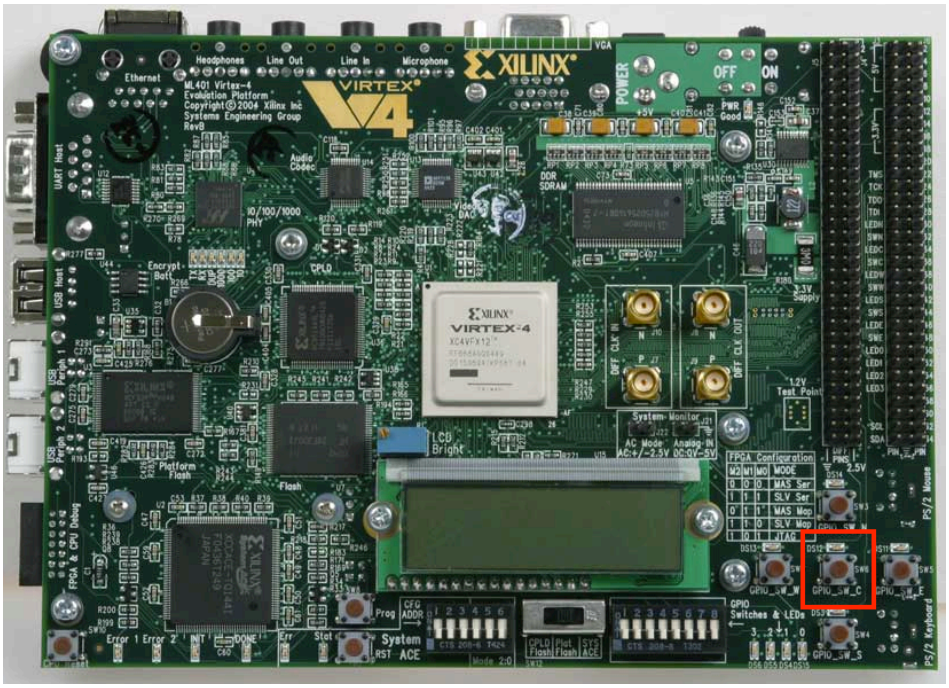


Figura 49 - Kit ML403.

15) Selecione o arquivo sync no campo Commands (Figura 50).

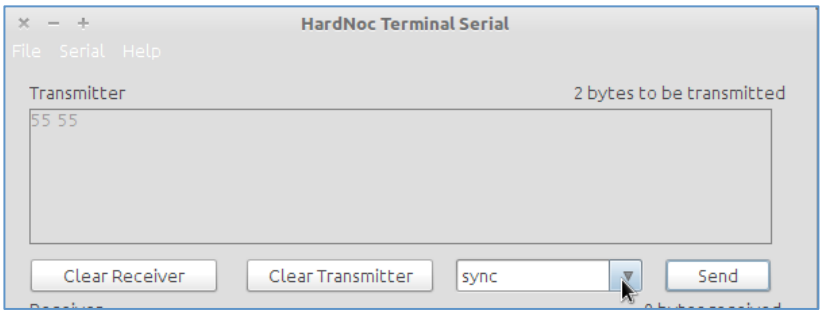


Figura 50 - Sincronizando a interface serial.

16) Clique em Send.

A interface será sincronizada. Para verificar se tudo ocorreu como esperado, iremos solicitar a leitura da memória de um IP. Como exemplo, iremos nos comunicar com o *IP11*.

17) Selecione o arquivo debug11 no campo Commands, e clique em Send.

Caso a comunicação ocorra com sucesso, será exibido no **Receiver** 32 bytes em zero. Caso contrário, repita os passos anteriores.

Voltando para a tela do ChipScope, podemos ver que o trigger dos sinais foram disparados e as amostras estão exibidas na janela **Waveform** (Figura 51).

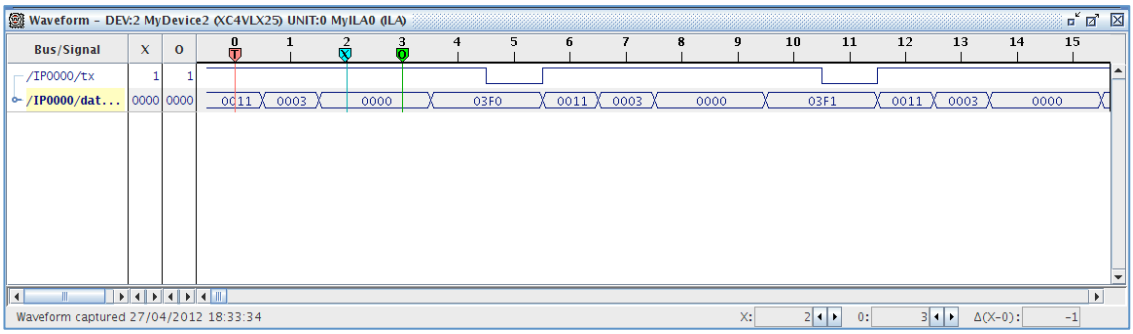


Figura 51 - Amostras Coletadas.

Outros testes:

Traffic01 → send

Start → Send

Debug11 → Send -- informa latências e número de pacotes

FIM DO TUTORIAL