MAPEAMENTO DOS PERIFÉRICOS DA DE2i-150

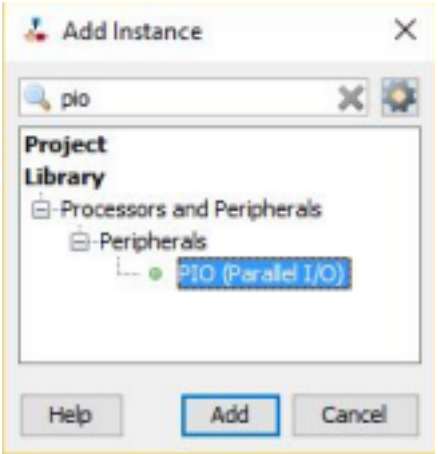
- TUTORIAL -

Passo 0: Antes de começar o mapeamento, certifique-se de estar utilizando a versão 17.1 do Quartus. Ela deve estar disponível na partição do Windows na maioria dos PCs do laboratório de hardware com o pacote de suporte à placa Cyclone IV para desenvolvimento do projeto. A gravação do .sof gerado na compilação também pode ser feita nessas máquinas, conectando o cabo do USB Blaster. (“a maioria dos PCs do laboratório de hardware” significa que alguns podem não estar com o suporte à Cyclone IV, portanto **é recomendável que instale na sua máquina o Quartus 17.1**, para o caso de os PCs utilizáveis estarem ocupados por outras equipes/alunos. Para quem quiser, disponibilizamos um tutorial de como instalá-lo).

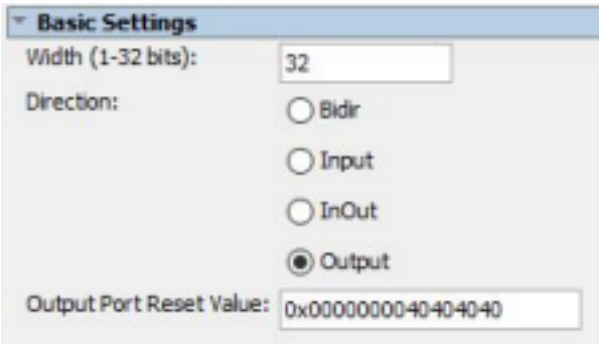
Passo 1: Abra o **pcihello.qar** com o Quartus 17.1 e siga os passos para restaurar o projeto disponibilizado.

Passo 2: Os dispositivos da placa serão mapeados em memória, logo a primeira coisa a se fazer é **reservar memória do FPGA**. Para isso no **Quartus** iremos no menu **Tools** e selecionamos a opção **Platform Designer**. Ao abrir a janela do Platform Designer, selecione o arquivo .qsys do projeto restaurado. Perceba que já existem dois itens mapeados (**inport** = 16 chaves e **hexport** = bloco de 4 displays de 7 segmentos).

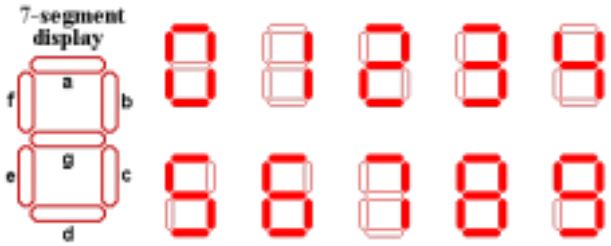
Passo 3: Clique no ícone  e digite “pio” e selecione **PIO (Parallel I/O)**.



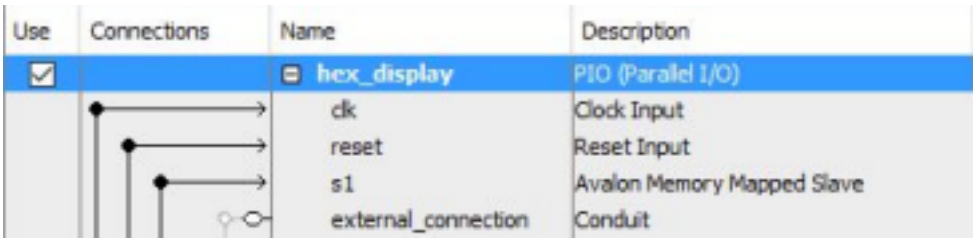
Passo 4: Informe o tamanho de memória que iremos reservar (em bits), o tipo de uso do dispositivo (entrada e/ou saída) e o valor inicial da memória reservada (indica o estado do dispositivo ao (re)iniciar a placa, em hexadecimal).



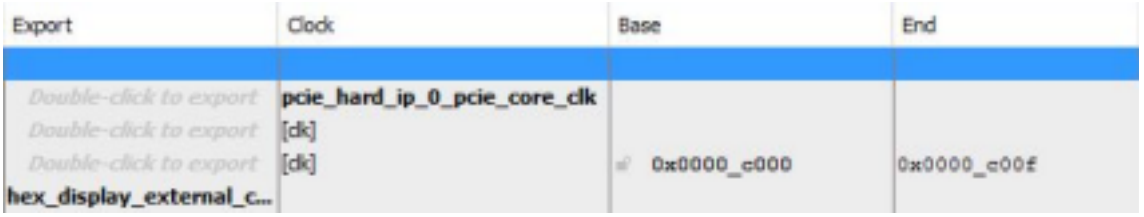
O exemplo acima é um mapeamento para display de 7 segmentos. Esses valores “40” repetidos para iniciar cada display com o número 0, apagando o bit relativo ao led do segmento do meio (40h = 01000000b). Então, como foi dito no Passo 4, isso é apenas o valor de inicialização, não é necessário colocar esse valor nos outros periféricos, de preferência inicialize com 0. É interessante usar o tamanho de 32 bits para todos os periféricos, isso vai ficar mais claro no desenvolvimento do driver PCI. (Cheque se os outros periféricos já mapeados têm esse tamanho)



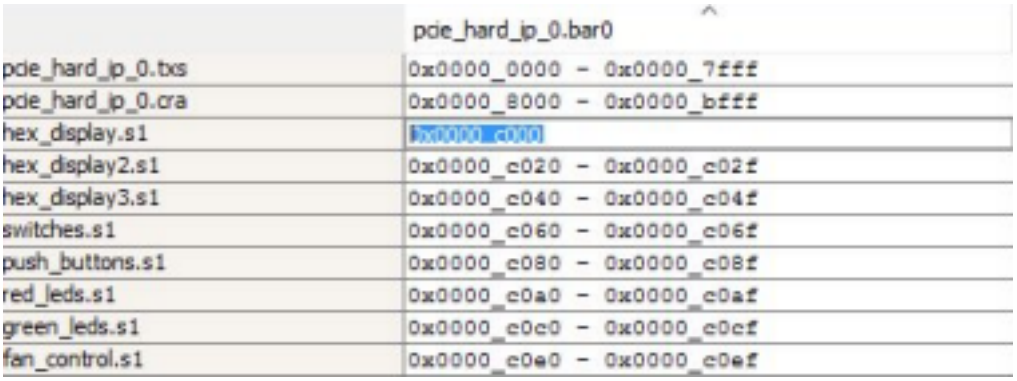
Passo 5: Vai ser adicionado uma linha com este novo item ao seu esquema principal. Renomeie o seu PIO de acordo com o que você deseja mapear:



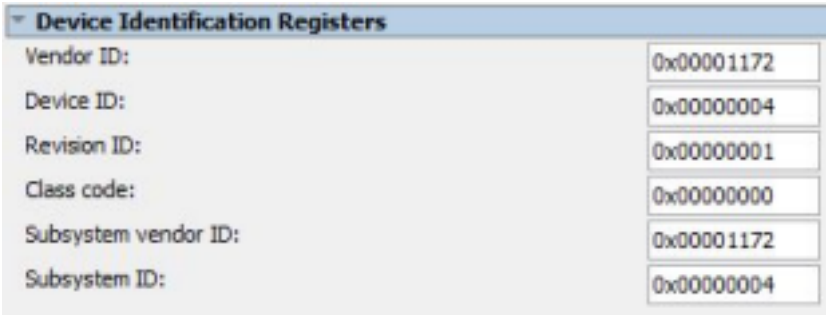
Passo 6: Esse PIO deve ser conectado ao núcleo do seu esquema. **Para isso conecte o clk, reset e s1 clicando nos fios ou nós ao lado desses itens**. Também dê dois cliques na célula da coluna **Export** na linha **external_connection**. Isso vai deixar um nome com o sufixo “_external_conection” e que será utilizado depois.



Passo 7: Para mapear a memória que será acessível ao driver no Linux, é necessário definir seu intervalo de endereço. O pcie_hard_ip_0 utiliza endereços desde a posição 0x0000_0000 até 0x0000_BFFF. Portanto, os mapeamentos dos periféricos deverão iniciar a partir do endereço 0x0000_C0000. Na aba **Address Map**, você é capaz apenas de definir o início do endereço (ele define apenas um intervalo de 16 bits). Então, para que se possa utilizar leituras e escritas de 32 bits em seu driver do Linux, é importante que seus mapeamentos sigam um padrão de intervalos em 32 bits (0x0000_C000, 0x0000_C020, 0x0000_C040, …), mesmo que seu PIO use bem menos bits, como mostra a imagem a seguir (não é preciso preencher com os mesmos intervalos):



Passo 8: Na tela principal do seu esquema de conexões, dê um clique duplo em **pcie_hard_ip_0**. Você deve verificar essas informações:

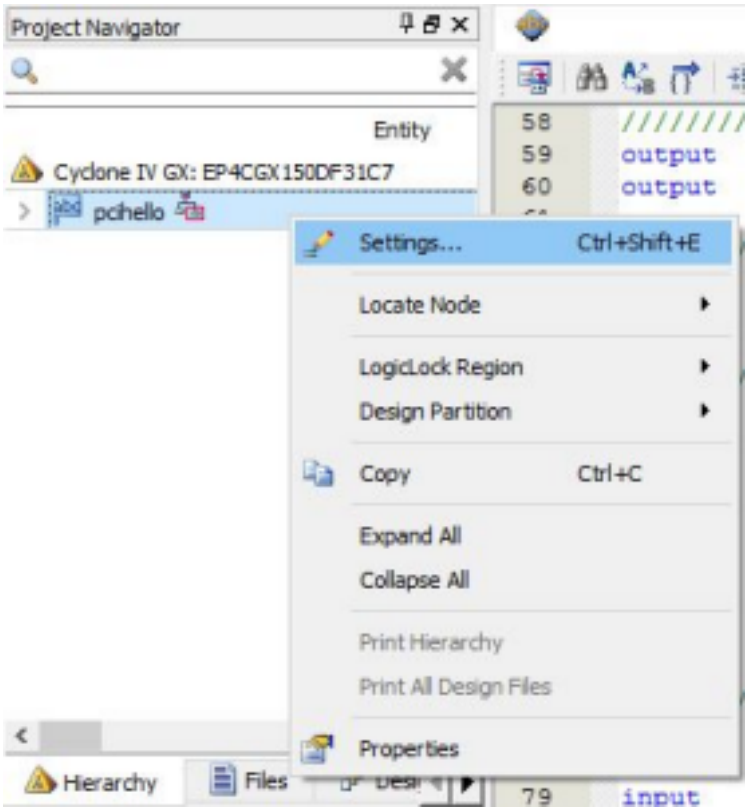


Caso você modifique algo nessas numerações, tenha essas alterações anotadas, esses códigos serão necessários no momento de escrever o driver da PCI. (Só para constar, não alteramos nada nesses itens durante nossos testes, recomendamos

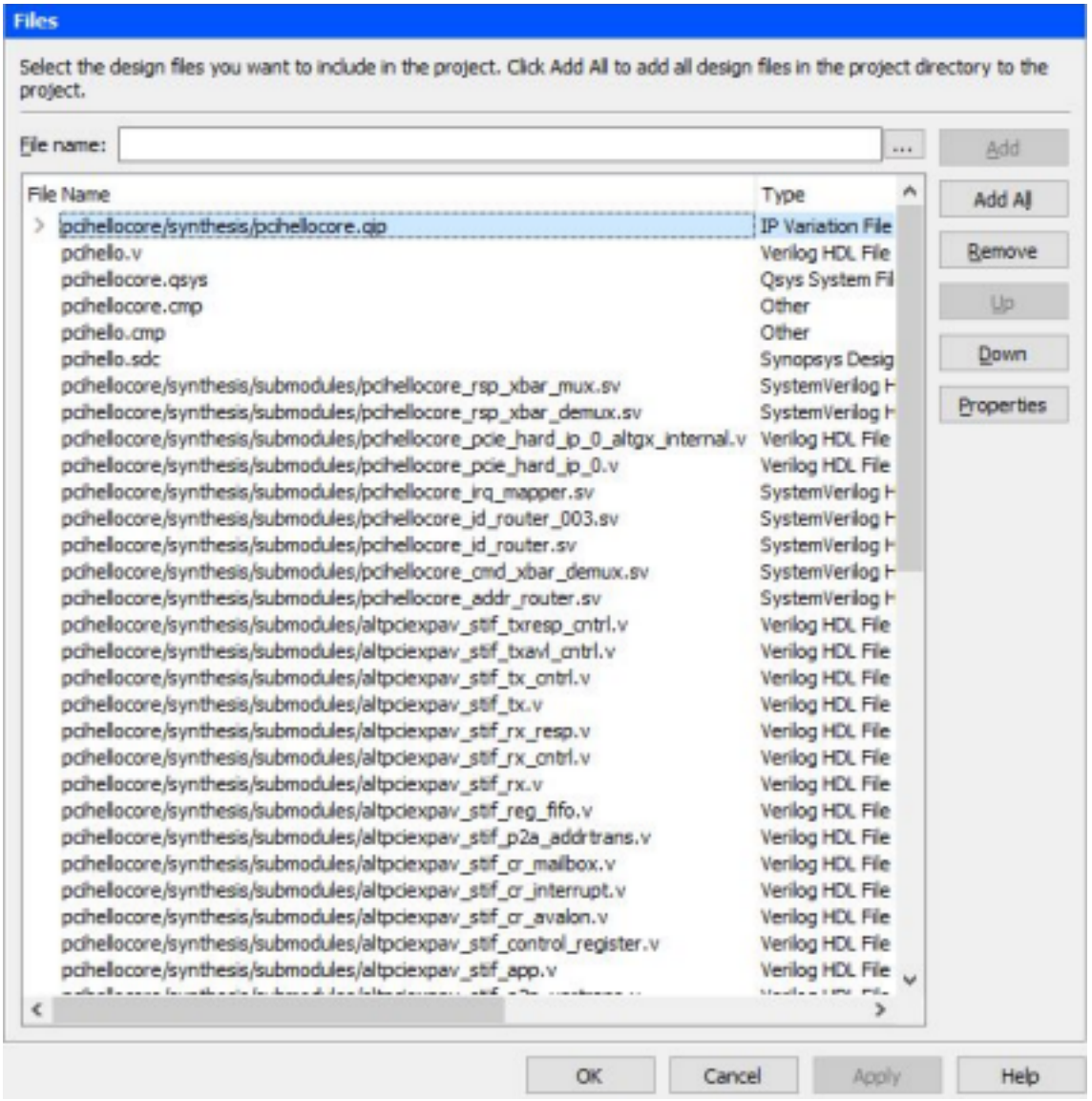
deixá-los assim por enquanto).

Passo 9: Vá no menu **Generate** e selecione **Generate HDL...**. Na seção **Synthesis** selecione **Verilog** e depois basta clicar em **Generate**. O Platform Designer irá gerar alguns arquivos na pasta “pcihellocore” do diretório do seu projeto. Após a geração desses arquivos, sem erros (ignore os *warnings*), minimize a janela do Platform Designer e volte para janela principal do Quartus.

Passo 10: Vá ao menu de configurações do seu projeto e selecione a categoria “**Files**”:



Clique na reticências e procure o arquivo **pcihellocore.qip**, adicione-o ao projeto. Utilizando os botões Up/Down, selecione o arquivo **pcihello.v** e mova-o para cima, como mostra a figura abaixo. Depois clique em **Apply** e **OK**.



Passo 11: Abra o código Verilog (**pcihello.v**) . Nele haverá uma estrutura que associa os barramentos da memória a wires. Então para cada periférico a ser mapeado, deve-se criar barramento do tipo wire com tamanho respectivo ao periférico que se deseja controlar (deve ser o mesmo tamanho escolhido da hora de reservar a memória). Nesse primeiro momento, prefira criar wires de tamanho fixos de 32 bits, depois pode-se diminuir seus tamanhos para conter a quantidade necessária desde que sejam um desses três tamanhos: 8 bits, 16 bits e 32 bits. Por exemplo, mesmo os botões utilizando 4 bits, é necessário criar um barramento de wire de 8 bits, pois as funções de leitura e escrita no driver da PCI que será desenvolvido no Linux permite apenas operações com os três tamanhos de dados já citados.

```
wire [31:0] hex_bus;
```

Passo 12: Criados os wires, associamos as entradas e saídas do bloco **`pcihellocore u0`**, e associamos aos respectivos wires. Essas entradas e saídas estão ligadas aos intervalos que reservamos através da *external connection* que exportamos no fim do passo 6. (Não esqueça da vírgula essas entradas e saídas, e do "_export" no fim de seu nome)

```
pcihellocore u0 (
    .pcie_hard_ip_0_rx_in_rx_datain_0          (PCIE_RX_P[0]),
    .pcie_hard_ip_0_tx_out_tx_dataout_0        (PCIE_TX_P[0]),
    .pcie_hard_ip_0_powerdown_pll_powerdown     (PCIE_WAKE_N),
    .pcie_hard_ip_0_powerdown_gxb_powerdown     (PCIE_WAKE_N),
    .pcie_hard_ip_0_refclk_export              (PCIE_REFCLK_P),
    .pcie_hard_ip_0_pcie_rstn_export           (PCIE_PERST_N),
    .hex_display_external_connection_export         (hex_bus),
    .hex_display2_external_connection_export       (hex_bus2),
    .hex_display3_external_connection_export       (hex_bus3),
    .switches_external_connection_export             (sw_bus),
    .push_buttons_external_connection_export       (push_bus),
    .red_leds_external_connection_export            (red_bus),
    .green_leds_external_connection_export        (green_bus),
    .fan_control_external_connection_export         (fan_bus)
);
```

Passo 13: Para finalizar, procure a MACRO referente ao mapeamento da pinagem da FPGA de cada periférico (está presente no código) e dê um **assign** no wire associado a essa MACRO (se for saída: **`assign FPGA_PINS = bus[inter:valo]`**; se for entrada: **`assign bus[inter:valo] = FPGA_PINS`**):

```
assign HEX0 = hex_bus[ 6: 0];
assign HEX1 = hex_bus[14: 8];
assign HEX2 = hex_bus[22:16];
assign HEX3 = hex_bus[30:24];
```

(Cheque se os intervalos das atribuições fazem sentido com as pinagens dos periféricos)

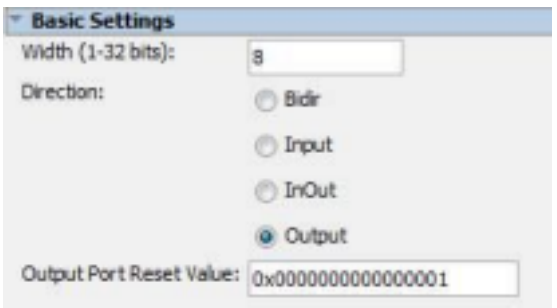
Passo 13.1: Lembrem-se de, antes de compilar, mudar o seguinte código:

```
assign FAN_CTRL = 1'b0;
```

para:

```
assign FAN_CTRL = 1'bz;  ou  assign FAN_CTRL = 1'b1;
```

Do jeito que estava antes, assim que se programava a placa, a ventoinha era desligada. Mas essa ventoinha aparentemente é o cooler da placa. Não é recomendável trabalhar com a placa com este cooler desligado por muito tempo.



Quando vocês fizerem o mapeamento do controle dessa ventoinha, lembrem-se de colocar um valor de reset de modo a deixar ela ligada (figura ao lado).

Passo 14: Compile e ignore os *warnings*. *(quem compilar sem warning ganha 5 pontos no projeto final)*

Passo 15: Conecte o cabo USB branco. Vá em **Tools >> Programmer**. Clique em **Hardware Setup** e selecione **USB-Blaster**. Depois clique em **Add File…** e selecione o **pcihello.sof** que acabou de ser gerado na compilação (é importante que só haja um arquivo .sof na lista de arquivos). Clique em **Start** e grave a placa.

Passo 16: Reinicie a placa, entre agora pelo **Lubuntu ou Debian** da placa (com os periféricos necessários, como teclado, mouse e monitor). Abra o terminal e digite `lspci`. Deve aparecer um item como a seguir:

01:00.0 Non-VGA unclassified device: Altera Corporation Device 0004 (rev 01)

Passo 17: Essa linha acima indica que a gravação deu certo. Caso esta linha não apareça, tente ver se esqueceu algum passo da geração dos arquivos HDL do QSYS ou tente regravar a placa.

OBS: As pinagens da FPGA e suas associações com MACROs estão no arquivo **pcihello.qsf** dentro no diretório do projeto. Para facilitar o trabalho das equipes que forem mapear o display LCD ou outro periférico, disponibilizamos o arquivo **de2i_150_qsys_pcie.qsf** e um HTML **Pinagem DE2i_150.htm** para que sirvam de suporte nessa etapa.