Todo list

Capítulo 1

Desenvolvimento

Este capítulo trata da concepção dos experimentos realizados. Nele serão descritos com detalhes cada um dos experimentos, ficando a parte de análise reservada ao capítulo **??**.

1.1 Introdução

Devido ao caráter experimental e exploratório do objetivo proposto na seção **??**, decidiu-se dividir o projeto em vários experimentos menores. Desta forma, além de garantir algum material mesmo que tudo dê errado, consegue-se simplificar o processo de pesquisa e desenvolvimento através dos pequenos passos e análises frequentes.

Como o objetivo final do projeto é a familiarização com as ferramentas e processos envolvidos na autoreconfiguração, decidiu-se começar estudando os elementos necessários para se realizar a reconfiguração dinâmica. O passo seguinte mais lógico é o de estudar como funciona as memórias dos sistema e de que jeito elas seriam melhor utilizadas. O último passo seria entender como funciona a autoreconfiguração em baixo nível, ou seja, como os dados devem ser entregues aos devidos componentes para que ela aconteça. Para cada um destes experimentos foi proposto um teste que validasse o completo entendimento do mesmo.

Figura 1.1: Foto ilustrativa do kit de desenvolvimento Kintex-7 KC705 extraida do site da Xilinx.

Para o desenvolvimento desse projeto, escolheu-se utilizar o kit de desenvolvimento da Xilinx® chamado Kintex-7 KC705. O único critério utilizado foi a disponibilidade dos equipamentos no início do projeto e a capacidade do dispositivo de realizar a reconfiguração parcial dinâmica. Este kit possui FPGA

modelo XC7K325T-2FFG900C, leitor de cartão de memória, conector PCIe®, memória DDR3, visor de 7-segmentos e porta ethernet, dentre outros.

Escolheu-se ainda, de forma arbitrária, o uso da linguagem VHDL para a descrição de *hardware* ao invés da Verilog.

1.2 Experimento 1 - Reconfiguração Dinâmica

De forma a dar validade a todo o projeto, foi preciso desenvolver um experimento para se entender o processo de desenvolvimento de sistemas reconfiguráveis dinamicamente e algumas peculiaridades do kit de desenvolvimento.

1.2.1 Fluxo de Ferramentas

A primeira coisa que se destaca no desenvolvimento de dispositivos dinamicamente reconfiguráveis é a diferença no fluxo de ferramentas, também conhecido como *software tools flow*, em relação ao fluxo tradicional (**??**). Esta diferença é motivada pela necessidade de construção de diversos *bitfiles* parciais. Como pode-se ver da figura 1.2a, o fluxo tradicional requer apenas o uso do programa ISE, e opcionalmente do XPS e do SDK, para a construção de um projeto de *hardware* e o iMPACT para a programação da FPGA. No fluxo para reconfiguração dinâmica mostrado na figura 1.2b, além das ferramentas do fluxo tradicional, faz-se necessário o uso da ferramenta XST para a síntese do *netlist* e do PlanAhead para a definição de partições e configurações. Note que estes fluxos não apresentam as únicas opções de fluxo de ferramentas, mas as que foram utilizadas neste projeto.

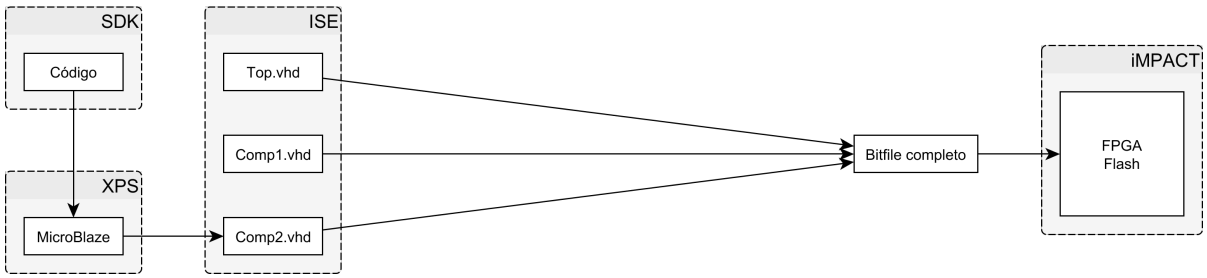
Reconfiguração parcial pede uma síntese utilizando o método "de baixo para cima" (*bottom-up*), mas uma implementação "de cima para baixo" (*top-down*) (**??**), ou seja, a implementação acontece construindo primeiro a interface com o sistema e depois os componentes auxiliares, mas a síntese precisa ser realizada no sentido oposto. Esta implementação é equivalente a se construir diversos projetos tradicionais com alguma lógica em comum, onde a síntese deve garantir que esta lógica em comum seja implementada da mesma forma para as diferentes configurações (**??**).

1.2.2 Peculiaridades

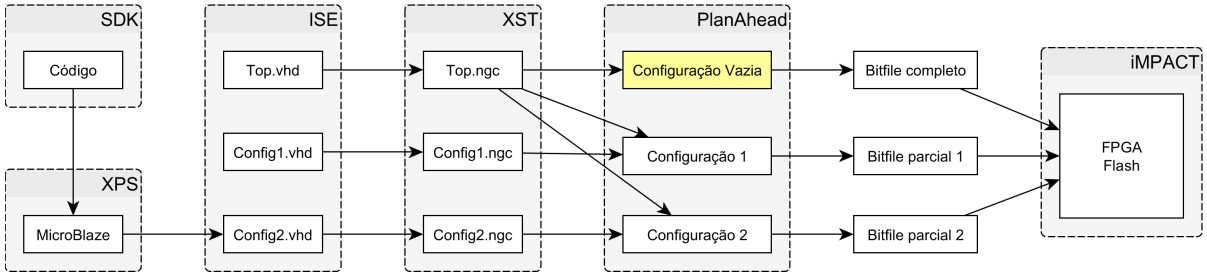
O kit de desenvolvimento utilizada apresenta algumas peculiaridades com relação aos kits comuns. A seguir serão apresentadas algumas destas peculiaridades.

1.2.2.1 Relógios

Diferentemente das FPGAs comuns, a que está presente neste kit contém um relógio diferencial, ou seja, dois sinais compoem tal relógio. A razão para tal é a presença de circuitos sensíveis a interferência, tais como *transceivers*, que são muito menores em sinais diferenciais. O kit disponibiliza duas opções de relógio: o SYSCLK e o USER_CLOCK. O primeiro possui uma frequência fixa de oscilação de 200 MHz.



(a) Foto ilustrativa do *software-flow* tradicional. Note que o uso do microcontrolador MicroBlaze é opcional, tornando os primeiros blocos, SDK e XPS, também opcionais.



(b) Foto ilustrativa do *software-flow* para a reconfiguração dinâmica. Assim como no caso tradicional, o uso do SDK e do XPS são opcionais. Note que o bloco em amarelo indica a configuração padrão, que será programada inicialmente. A escolha da configuração padrão é arbitrária.

Figura 1.2: Comparação dos fluxos de ferramentas. Note que estes fluxos não apresentam as únicas opções de fluxo de ferramentas, mas as que foram utilizadas neste projeto.

O segundo possui uma frequência original de 156.250 MHz, mas pode ser programado através de uma interface I^2C para ter frequências entre 10 MHz e 810 MHz. Por motivos de simplicidade, utilizou-se o SYSCLK. Para poder se trabalhar com o sinal diferencial, construiu-se, utilizando as ferramentas do ISE, um componente para tratamento do sinal de relógio. Este componente recebe o sinal diferencial, reduz sua frequência para 20 MHz, que corresponde ao menor valor suportado pelas PLLs da placa, e emite um sinal convencional.

1.2.3 Teste

Para se entender mais a fundo o fluxo de projeto, nada melhor que construir um projeto. Para isso, implementou-se o sistema esquematizado na figura 1.3. Este sistema contém o "Top" para interfaceamento com a FPGA, o "Clocks" para tratamento do sinal de relógio, a "Static", que possui um lógica estática para demonstrar que a reconfiguração de uma partição não interfere com outra, e a "Dynamic", que possui a lógica a ser alterada dinâmicamente.

Estrutura de Pastas A questão da organização do projeto em pastas bem específicas é sempre bem mencionado na literatura (**??????**). Os manuais recomendam a seguinte estrutura de pastas.

```
Projeto/
  Source/          //codigos-fonte organizados segundo particao
  Implementation/  //contem pastas para cada config. dinamica gerada
  Synth/           //contem pastas com os arquivos .xst e .prj
```

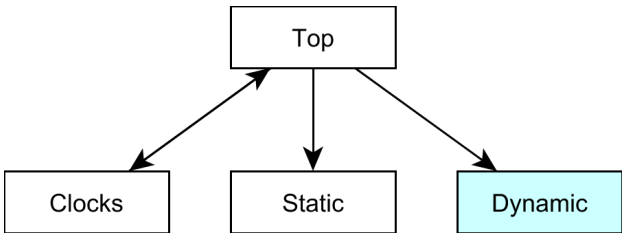


Figura 1.3: Foto ilustrativa do sistema desenvolvido para o teste de validação do experimento 1. Ele é composto por uma parte estática e uma parte dinâmica. O elementos em branco são estáticos e os em azul são dinâmicos.

```
Tools/          //ferramentas para automacao da sintese
PlanAhead/      //pasta para o projeto do PlanAhead
```

Esta estrutura de pastas foi obedecida por ajudar a manter o ambiente de desenvolvimento limpo.

1.2.3.1 Comportamento

Como explicado anteriormente, o projeto de um sistema parcialmente reconfigurável pode ser visto como vários projetos completos com partes em comum. Seguindo essa lógica, dois projetos com comportamentos diferentes foram construídos usando como base a figura 1.3. O comportamento individual de cada módulo ou componente será descrito a seguir. Este passo está ilustrado no fluxo de ferramentas da figura 1.2b como ISE.

O componente “Static” possui uma entrada para um relógio pulsando a 2 Hz e uma saída para um LED. Seu comportamento apenas faz com que o LED pisque a uma frequência de 2 Hz, o que permite observar seu funcionamento durante a reconfiguração do componente “Dynamic”.

O componente “Dynamic” possui dois comportamentos distintos. O primeiro deles é o de um simples contador crescente de 4 bits. O segundo é uma máquina de estados que alterna os 4 bits de saída entre "1100"e "0011"a cada pulso de relógio. Este componente possui uma entrada para um relógio pulsando a 1 Hz e uma palavra de 4 bits de saída. A frequência de operação deste componente foi escolhida para ser a metade da frequência da “Static” para poder ser visualmente comprovado que “Static” não para de funcionar quando “Dynamic” está sendo reconfigurado.

O componente “Clocks” recebe os sinais diferenciais de relógio e o transformam em um sinal comum. O bloco lógico utilizado para isso foi construído usando ferramentas presentes no ISE. Uma vez que a ferramenta permitia a construção de um relógio com divisor de frequência, a frequência do relógio da placa, que nesse caso é de 200 MHz, foi reduzida para 20 MHz.

O módulo “Top” instancia os componentes descritos acima e faz a interface dos mesmos com a FPGA. O componente dinâmico precisa de uma declaração de protótipo para ser instanciado corretamente. Utilizou-se o código abaixo para esta finalidade.

```
component dynamic
    port ( clk  : in  std_logic;
           leds : out std_logic_vector (3 downto 0));
end component;
```

O módulo “Top” possui também um divisor de frequência para reduzir a frequência devolvida por “Clocks” para 1 e 2 Hz.

1.2.3.2 Síntese

Com o comportamento do projeto definido, o próximo passo segundo o fluxo de ferramentas é a síntese. Este passo é necessário uma vez que o próximo passo, referente ao PlanAhead, não aceita como entrada códigos-fonte. Os códigos-fonte precisam passar por uma etapa de síntese separada para poderem ser importados no PlanAhead. Este passo está ilustrado no fluxo de ferramentas da figura 1.2b como XST.

O comando XST recebe tipicamente um *script* contendo o endereço dos códigos-fonte, o nome do arquivo de saída, o tipo do arquivo de saida, o modelo da FPGA utilizada e uma indicação do código-fonte principal. O comando para iniciar o processo é o seguinte.

```
xst.exe -ifn Top.xst
```

O arquivo “Top.xst” contém os seguintes comandos.

```
run
-ifn Top.prj
-ofn Top
-ofmt NGC
-p xc7k325t-2-ffg900
-top top
```

O arquivo “Top.prj” contém os endereços dos arquivos, conforme a seguir.

```
vhdl work "../../Sources/static/top.vhd"
vhdl work "../../Sources/static/static.vhd"
vhdl work "../../Sources/static/clocks.vhd"
```

Note que estes comandos e arquivos indicados acima são para síntese dos componentes estáticos. Uma vez que não existe nenhuma restrição especial para tais componentes, eles podem ser síntetizados para um único arquivo de saída. O mesmo não pode ser dito para os elementos dinâmicos. Cada componente dinâmico precisa ser sintetizado em separado para depois ser incluído no projeto através do PlanAhead.

A síntese de componentes dinâmicos precisa ser realizada com um *script* “.xst” ligeiramente diferente. Como mostrado a seguir, faz-se necessária a inclusão do argumento “-iobuf NO”, que desabilita a inserção de componentes de Entrada/Saída (**????**).

```
run
-ifn DynFSM.prj
-ofn DynFSM
-ofmt NGC
-p xc7k325t-2-ffg900
-top dynamic
-iobuf NO
```

Note que o arquivo “DynFSM.prj” contém informações sobre o cógigo-fonte do componente dinâmico, como mostrado a seguir.

```
vhdl work "../../Sources/dynamic_fsm/dynamic.vhd"
```

Existe também a possibilidade de construção de um *script* para a síntese automática de todos os arquivos. Utilizou-se aqui uma adaptação do arquivo em linguagem TCL usado pela Xilinx em seus manuais (**??????**). A única coisa que se faz neste *script* é a construção dinâmica dos comandos com base em listas de arquivos pré-informados.

1.2.4 PlanAhead

Com os arquivos síntetisados, pode-se começar a etapa referente ao PlanAhead. Nela, importaremos os arquivos da etapa anterior, criaremos a partição reconfigurável, mapearemos esta partição no dispositivo, criaremos configurações alternativas, promoremos tais configurações e geraremos os *bitfiles* para a programação do dispositivo. Note que é preciso uma licença do ISE que permita o uso do PlanAhead e de reconfiguração parcial.

O primeiro passo necessário no PlanAhead é a criação do projeto. Para isso, após a abertura do programa, clica-se no ícone superior esquerdo mostrado na figura 1.4, onde se lê "Create New Project". Na janela que aparece, indicamos o nome do projeto e seu caminho, lembrando que foi criada uma pasta anteriormente especificamente para conter este projeto. Indicamos também que o projeto é do tipo "*Post-synthesis Project*" e que desejamos habilitar a reconfiguração parcial, indicamos quais *netlists* compõe o comportamento do projeto e qual corresponde ao arquivo principal, qual é o arquivo de restrições (*constrains*) e qual é o modelo da FPGA.

Figura 1.4: Imagem do PlanAhead logo que aberto.

Após a criação do projeto, carregam-se os arquivos sintetizados abrindo "*Open Synthesized Design*". Duas janelas de avisos aparecem, informando que existe uma partição não implementada e aviso sobre

alguns pinos.

Pode-se agora definir a partição que armazenará o componente dinâmico. Isso é feito através do painel "*Netlist*", selecionando a opção "*Set Partition*" do menu que aparece ao se clicar em "dynamic_i" com o botão direito. Na janela que aparece, selecionamos a opção referente à partição reconfigurável, nomeamos o módulo reconfigurável de acordo com o componente que será carregado e indicamos o arquivo que corresponde ao arquivo sintetizado do componente reconfigurável. Não é necessário informar restrições, visto que o componente, seguindo recomendações, não acessa os pinos de entrada e saída diretamente. Note ainda que é recomendado que o primeiro módulo a ser incluído seja o mais complexo e sucetível a falhas, permitindo que erros e falhas na definição da região a seguir sejam identificados mais cedo.

Precisa-se definir também uma região para a partição recém definida. Isto pode ser feito pelo mesmo painel "*Netlist*", selecionando a opção "*Set Pblock Size*" do menu que aparece ao se clicar em "dynamic_i" com o botão direito. Nesse momento, precisa-se selecionar na aba "*Device*" uma região do dispositivo que contenha uma quantidade de recursos maior que a requerida pelo projeto. Note que o processo de escolha dessa região não é bem definido, o que abre espaço para diversos erros de alocação. Para testar se a região foi bem alocada, abre-se a aba "*Design Runs*" do painel inferior, clica-se com o botão direito na única configuração disponível no momento e seleciona-se "*Run Launch*". Este processo pode demorar. Em tentativas subsequentes, seleciona-se a opção de recomeçar todo o processo do zero, evitando erros gerados nos estágios iniciais. Uma região possível, que foi utilizada nesse projeto, foi a que contém as células (*slice*) de X80Y244 a X139Y205. Os nomes das células ficam visíveis através da ampliação do dispositivo.

Uma vez terminado o "*Design Run*", adiciona-se duas novas possibilidades de módulos reconfiguráveis para esta partição: um com comportamento definido e outro vazia. Para isso, clica-se em "*Synthesized Design*" novamente e seleciona-se a opção "*Add reconfigurable module*" do menu que aparece ao se clicar em "dynamic_i" no painel "*Netlist*" com o botão direito. O processo é o mesmo da definição da partição, sendo a única mudança na seleção do arquivo sintetizado e do nome do módulo. Um módulo vazio pode ser criado usando esta mesma opção, mas selecionando a opção que indica a inclusão de uma *blackbox* sem o uso de uma *netlist*.

Com as possibilidades de módulos reconfiguráveis definidas, pode-se construir várias configurações. Isso é feito através do clique com botão direito na "*Design Runs*" do painel inferior e selecionando-se a opção "*Create Runs...*". A janela que abre possui um painel chamado "*Create Implementation Runs*". Nesse painel, na coluna "*Partition Action*", define-se, na coluna "*Module Variant*" da nova janela que aparece, o módulo desejado. Voltando para a primeira janela, clica-se em "*More*" para adicionar a última configuração desejada. Em seguida, as duas novas configurações serão criadas através de "*Design Runs*". É necessário promover neste momento a primeira configuração implementada. O processo de promoção será comentado a seguir. Note que os avisos que aparecerem podem ser ignorados.

Ao final dos "*Design Runs*", recomenda-se fazer a verificação das configurações através do painel "*Configurations*". Clicando-se com o botão direito, encontra-se a opção "*Verify Configuration...*", que faz com que uma janela seja aberta. Selecionado-se todos os itens e clicando em "*OK*", a verificação se inicia. Nenhum erro deve ser encontrado.

Promoção de partições explicado em ug748, p. 50.

1.2.5 iMPACT

Explicar como fui para o próximo experimento

1.3 Experimento 2 - Teste de Acesso a Memória

Descrever o experimento 2

Descrever os tipos de memória disponíveis e o porque de escolher a DDR3

Explicar porque não deu certo e como fui para o próximo experimento

1.4 Experimento 3 - Teste do *Bootloader*

Descrever o experimento 3

Explicar como fui para o próximo experimento

1.5 Experimento 4 - Teste da Autoreconfiguração com *Bootloader* Dedicado

Descrever o experimento 4

Explicar como fui para o próximo experimento

1.6 Experimento 5 - Teste da Autoreconfiguração

Descrever o experimento 5

Explicar como fui para o próximo experimento