Pontifícia Universidade Católica do Rio Grande do Sul
Faculdade de Engenharia
Faculdade de Informática
Curso de Engenharia da Computação

Contribuições ao Desenvolvimento de Sistemas Digitais com Reconfiguração Parcial Dinâmica

Autores
Sérgio Damo de Lemos
Luís Felipe Auad Guedes

Trabalho de Conclusão do Curso de
Engenharia de Computação na Pontifícia
Universidade Católica do Rio Grande do Sul

Orientador
Prof. Dr. Fernando Gehm Moraes

Porto Alegre
2012

RESUMO

No presente Trabalho de Conclusão de Curso é abordada a técnica de Reconfiguração Parcial e Dinâmica (RPD) em dispositivos lógicos programáveis do tipo FPGAs. Estes dispositivos estão sendo cada vez mais empregados na indústria de sistemas eletrônicos, de sistemas embarcados e de tempo real, devido à facilidade de configuração e alto nível de desempenho. Também apresentam longo ciclo de vida e baixa dissipação de potência, característica antes presentes apenas em dispositivos do tipo ASIC. Além de vários dispositivos e periféricos integrados aos FPGAs, outro ponto atrativo destes é a grande versatilidade no uso de seus recursos, podendo até mesmo ter parte de sua funcionalidade alterada em funcionamento para atender demandas específicas, como iremos demonstrar utilizando RPD. Tendo em vista esta característica de reconfigurabilidade, que agrega a flexibilidade do software ao hardware, apresenta-se no decorrer desta proposta uma solução de fluxo de projeto utilizando RPD, para aumentar a flexibilidade dos projetos e para diminuir o consumo de área em projetos de lógica programável que usam como plataforma FPGAs com essa funcionalidade. Este fluxo de projeto será desenvolvido utilizando as ferramentas disponibilizadas pela Xilinx que é uma das principais fabricantes de FPGAs atualmente. Como contribuição deste trabalho, documenta-se de forma didática o fluxo de desenvolvimento completo voltado para arquiteturas de Lógica Programável que utilizam RPD, tornando mais simples e amigável a ambientação a este fluxo de projeto, que provêm uma grande flexibilidade para projetos implementados em FPGA.

Palavras chave: reconfiguração parcial dinâmica, FPGAs, fluxo de projeto reconfigurável.

ABSTRACT

In this end-of-term-work (TCC) the Dynamic and Partial Reconfiguration (DPR) technique in FPGAs is studied. These devices are being increasingly used in electronics industry, embedded systems and real-time systems, due to their ease of configuration and high performance. They have also a long life cycle and a low power dissipation, which are characteristics present before only in ASIC devices. In addition to several devices and peripherals integrated into FPGAs, another attractive point of these devices is the great versatility in the use of its resources, allowing even the modification of part of their functionality at run time to meet specific demands, as we will demonstrate using DPR. Given this characteristic of reconfigurability, which provides the flexibility of software to hardware, it is presented in the course of this proposed solution a design flow using DPR, to increase the flexibility of the projects and to reduce the area consumption in the design of programmable logic that uses FPGAs as a platform. This design flow is developed using the tools provided by Xilinx that is one of the major manufacturers of FPGAs. As a contribution to architectures this work documents in a didactic way a complete flow of development oriented programmable logic using DPR, making it simple and user-friendly with this design flow, that provides great flexibility for projects implemented in FPGA.

Key words: dynamic partial reconfiguration, FPGAs, reconfigurable design flow.

SUMÁRIO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

ÍNDICE DE LISTAGENS

LISTA DE SIGLAS

ASIC – *Application Specific Integrated Circuit*
BMM – *Block RAM Memory Map*
BRAM – *Block RAM*
CAD – *Computer Aided Design*
CLB – *Configurable Logic Block*
CMT – *Clock Management Tile*
CRC – *Cyclic Redundancy Check*
DCM – *Digital Clock Manager*
DRC – *Design Rules Check*
DSP – *Digital Signal Processor*
EDIF – *Electronic Design Interchange Format*
FIFO – *First in first out*
GPP – *General Purpose Processors*
HDL – *Hardware Description Language*
ICAP – *Internal Configuration Access Port*
IDE – *Integrated Development Environment*
IOB – *Input/Output Block*
ISE – *Integrated Software Environment*
IP – *Intellectual Property*
JTAG – *Joint Test Action Group*
LUT – *Look-Up Table*
MAC – *Medium Access Controller*
NGD – *Native Generic Database*
PLB – *Processor Local Bus*
PLL – *Phase Locked Loop*
PR – Partição Reconfigurável
RAM – *Random Access Memory*
ROM – *Read-Only Memory*
RPD – Reconfiguração Parcial Dinâmica
RTL – *Register Transfer Level*
SDK – *Software Development Kit*
SDR – *Software Defined Radio*
SoC – *System on Chip*
SRAM – *Static RAM*
TCC – Trabalho de Conclusão de Curso
UCF – *User Constraints File*
VHDL – VHSIC HDL
VHSIC – *Very High Speed Integrated Circuit*
XPS – *Xilinx Platform Studio*
XST – *Xilinx Synthesis Tool*

1 INTRODUÇÃO

Atualmente qualquer dispositivo eletrônico possui algum tipo de inteligência embarcada. Esta capacidade de análise e resposta à estímulos pode ser implementada utilizando uma série de tecnologias.

Uma das opções disponíveis atualmente são os Circuitos Integrados de Aplicação Específica ou ASIC (*Application Specific Integrated Circuit*). ASICs representam o máximo que podemos obter em termos de desempenho e eficiência energética, porém seu desenvolvimento é complexo, demandando tempo, ferramentas sofisticadas e profissionais altamente treinados. A fabricação de ASICs também é cara, geralmente variando de centenas de milhares de dólares a milhões de dólares. Além disto, dada sua natureza estática, uma vez que um projeto seja fabricado e o mesmo contiver alguma falha, esta não pode ser corrigida após o produto ter sido lançado no mercado, introduzindo grandes fatores de risco em seu desenvolvimento.

Como alternativa aos ASICs cita-se por exemplo os dispositivos programáveis, como os Processadores de Propósito Geral (*General Purpose Processors* - GPPs), processadores para tratamento digital de sinais e micro-controladores. Todos estes dispositivos possuem o mesmo princípio de funcionamento, executando uma série de instruções previamente armazenadas em uma memória externa aos mesmos. A capacidade que estes dispositivos apresentam de ter sua programação alterada, praticamente em qualquer lugar e a qualquer momento, introduz uma grande flexibilidade aos produtos desenvolvidos com este tipo de tecnologia. O revés introduzido pelo uso de dispositivos programáveis está no maior consumo de energia e desempenho inferior aos ASICs, pois sua operação é limitada à execução sequencial de instruções.

Uma categoria intermediária entre ASICs e dispositivos programáveis são os dispositivos configuráveis, como os FPGAs (*Field Programmable Gate Arrays*). A diferença entre os termos empregados no presente trabalho, programáveis e configuráveis, reside na granularidade da computação que pode ser modificada em tempo de execução. Os GPPs modificam a função realizada em uma ULA a cada nova instrução – granularidade no nível de palavra, enquanto em FPGAs a cada arquivo de configuração alteram-se as funções executadas no hardware no nível da função lógica – granularidade no nível de bit. FPGAs podem oferecer um desempenho próximo ao de circuitos dedicados (ASICs) com a flexibilidade dos circuitos programáveis. Estima-se que, em média, projetos implementados em FPGA sofrem uma perda de 5 a 25% de eficiência em termos de área ou velocidade quando comparado com uma implementação em ASIC [DEH08].

FPGAs são os dispositivos que serão objeto de estudo neste Trabalho de Conclusão de Curso (TCC), devido às vantagens de sua utilização quando comparados a ASICs e dispositivos programáveis.

1.1 Field Programmable Gate Array

FPGAs representam uma classe de dispositivos que possui a capacidade de implementar diferentes funções lógicas em hardware, permitindo a obtenção de desempenho similar ou próximo ao de ASICs, ao possibilitar a implementação de sistemas com um alto grau de paralelismo. Também oferecem a flexibilidade de sistemas programáveis, podendo ter seu comportamento configurado de maneira fácil e rápida [DEH08].

Além destes benefícios, cita-se a vantagem financeira do uso de FPGAs, pois estes são

produzidos em altíssimas escalas, chegando no mercado com um preço muito competitivo. Ressalta-se que o custo de produtos eletrônicos desenvolvidos com dispositivos FPGAs é competitivo em relação aos ASICs somente para volumes pequenos e médios de produção. Porém dispositivos FPGAs possuem a limitação de implementar apenas circuitos digitais.

Esta classe de circuitos integrados também representa o estado da arte em termos de processos de fabricação. Para efeito de comparação, no momento em que este trabalho está sedo desenvolvido, a série mais recente de FPGAs da Xilinx, Virtex-7, é fabricada utilizando um processo de 28 nm, enquanto o último processador lançado pela Intel utiliza tecnologia de 32 nm. [XIL12b]

Juntamente com as funcionalidades básicas de um FPGA é comum que estes dispositivos também incluam módulos auxiliares de hardware, como por exemplo memórias RAM, gerenciadores de relógio, multiplicadores, controladores de rede (MAC Ethernet), e inclusive processadores, fazendo com que se tornem verdadeiras plataformas para o desenvolvimento de SoCs (*Systems on a Chip*).

1.1.1 Estrutura e Funcionamento dos dispositivos FPGAs

Esta seção apresenta a estrutura de um FPGA de uma forma genérica, visando apresentar o seu funcionamento básico. Porém vale lembrar que sua implementação pode variar entre fabricantes diferentes ou até mesmo entre modelos diferentes de um mesmo fabricante.

Para compreender seu funcionamento, podemos pensar no FPGA como um dispositivo composto por duas camadas. Uma delas, a Camada de Lógica, a qual abriga estruturas lógicas genéricas, cujo comportamento é definido pelo estado dos *bits* armazenados na Camada de Memória de Configuração. A Figura 1 ilustra esta configuração.

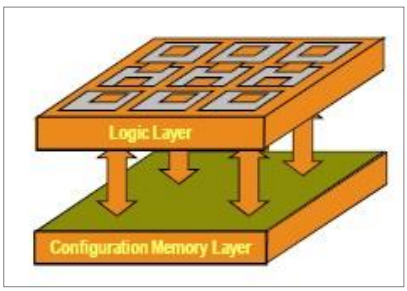


Figura 1 - FPGA visto como duas camadas: Lógica e Memória de Configuração [DEH08].

Basicamente, na Camada Lógica encontraremos três tipos de estruturas configuráveis: Blocos Lógicos (*Configurable Logic Block* - CLB), interconexões, e blocos de entrada/saída. A Figura 2 ilustra esta estrutura conceitual.

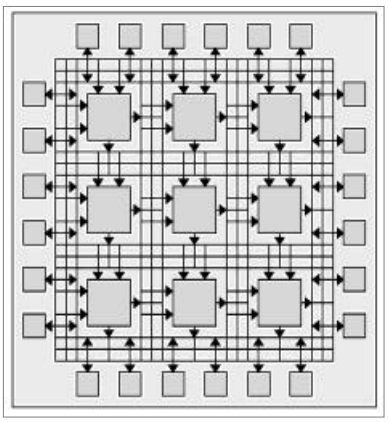


Figura 2 - Estrutura básica de um FPGA formado por Blocos Lógicos interconectados [DEH08].

Cada CLB é composto por circuitos semelhantes ao representado na Figura 3. O multiplexador M1 implementa uma tabela, denominada *Lookup Table* (LUT), responsável por implementar tabelas verdade de funções booleanas através da seleção das saídas dos registradores R, permitindo a criação de qualquer lógica combinacional, como pode ser visto na Figura 4. O registrador R1 é utilizado para registrar a saída em circuitos lógicos que necessitem de manutenção de estado. O multiplexador M2 seleciona se a saída do circuito será a saída da LUT ou o estado armazenado em R1.

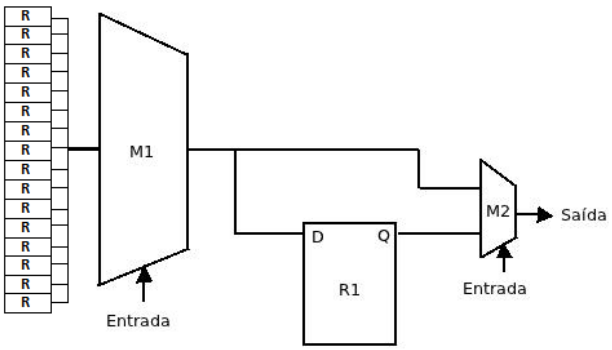


Figura 3 - Estrutura básica de um Bloco Lógico com LUT de 4 entradas.

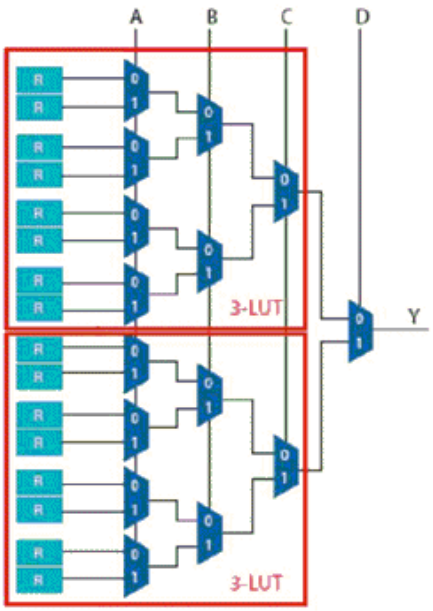


Figura 4 - Estrutura básica de uma LUT de 4 entradas [VER07]

Os valores armazenados nas LUTs, o estado inicial de R1 e a entrada de M2 ficam associados a bits de memória presentes na Camada de Memória de Configuração.

A quantidade de entradas de uma LUT é tema de muita pesquisa, sendo que geralmente os fabricantes de FPGAs adotam LUTs de 4 ou 6 entradas [DEH08]. Também é comum que um Bloco Lógico possua a estrutura apresentada na Figura 3, replicada mais de uma vez, juntamente com outros elementos que auxiliam na implementação de circuitos específicos, como somadores.

Juntamente com as CLBs encontra-se na Camada Lógica as interconexões e as estruturas responsáveis por rotear os sinais que conectam um CLB à outro. Estas estruturas são chamadas de Blocos de Chaveamento, ou *Switching Blocks*. Atualmente técnicas de roteamento são um tema muito pesquisado e de grande importância, visto que interconexões podem representar até

80% da área de um FPGA moderno [DEH08].

Além das estruturas já mencionadas, encontramos no FPGA blocos responsáveis por realizar a interface do interior do FPGA com o mundo exterior, isto é, os pinos de entrada e saída. Estes blocos de I/O (*Input/output*, entrada/saída) fornecem a versatilidade de configurar os pinos do FPGA, que geralmente podem ser utilizados com entradas digitais, saídas digitais ou ambos. Também é comum encontrarmos blocos de I/O que permitam o uso de lógica *tri-state*, além de contar com buffers para fornecer ganhos de corrente.

Conforme já comentado, a Camada de Memória de Configuração armazena os *bits* responsáveis por configurar as estruturas do FPGA. Geralmente a memória utilizada é do tipo RAM estática. Desta forma, o dispositivo necessita ser configurado cada vez que a alimentação é fornecida. Comumente os dados para configuração ficam armazenados em uma memória *flash* externa e são carregados no FPGA por meio de outro dispositivo, como um microcontrolador. Maiores detalhes a respeito da implementação de FPGAs podem ser encontrados em [DEH08].

Para implementação deste trabalho será utilizada a plataforma de prototipação ML505 [XIL11c], a qual contém um FPGA Virtex 5, produzido pela empresa Xilinx. A escolha desta plataforma se deve a dois fatores, sendo que o principal deles é de que a Xilinx é companhia que, atualmente, possui um suporte bastante desenvolvido na área de Reconfiguração Parcial e Dinâmica (RPD). Além disto, a escolha da plataforma Xilinx também se deu em virtude da disponibilidade destes equipamentos em nossos laboratórios.

1.2 Reconfiguração Parcial e Dinâmica

Podemos definir Reconfiguração Parcial e Dinâmica (RPD) como a técnica que permite que, após um FPGA tenha sido configurado com uma determinada implementação, seja possível reconfigurar partes deste mesmo FPGA sem comprometer a integridade ou interromper o funcionamento das suas demais partes [XIL11]. Esta reconfiguração pode ser realizada de maneira autônoma pelo próprio FPGA.

1.2.1 Vantagens e desvantagens de RPD

Dentre as vantagens da aplicação da técnica de RPD estão a redução do espaço físico necessário dentro do FPGA para a implementação de uma determinada função, aumento da flexibilidade das aplicações implementadas, diminuição no tempo de configuração ao iniciar o FPGA e o aumento na tolerância à falhas da aplicação [XIL11a]. Outra vantagem é a redução do consumo de energia do FPGA [XIL10a].

A aplicação de RPD também traz algumas desvantagens, como por exemplo uma degradação de aproximadamente 10% na frequência de relógio [XIL11a], devido parcialmente a restrição que ela impõem às otimizações no circuitos reconfiguráveis.Outra desvantagem é o acrescimo de novas etapas no fluxo de projeto.

1.2.2 Exemplos de Aplicações de RPD

Em [XIL11a] encontramos descritos três exemplos de aplicações de RPD, os quais incluem:

- **Interface de rede multiporta**. Podemos considerar uma *switch* de rede como um típico exemplo de possível aplicação de RPD. Muitas vezes *switches* implementam em suas portas diferentes protocolos de rede, porém apenas um é utilizado. Neste caso o uso de técnicas de RPD pode evitar o desperdício de manter o *hardware* dedicado à implementar os demais

protocolos não utilizados.

- **Dispositivo periférico utilizando barramento PCI Express**. Podemos encontrar FPGAs em dispositivos conectados a um barramento PCI-Express. A especificação deste barramento estipula um tempo máximo para todos os dispositivos responderem ao controlador. Muitas vezes este tempo não é suficiente para que o processo de configuração de todo o FPGA seja concluído. É possível utilizar RPD para configurar no FPGA inicialmente apenas o módulo PCI-Express, para então depois configurar os demais módulos do sistema.
- **Processador de pacotes reconfigurável dinamicamente**. Em uma aplicação que necessite processar uma grande quantidade de pacotes de rede, é possível aplicar a técnica de RPD para que o hardware implementado no FPGA possa ser alterado em tempo real dependendo do tipo de pacote recebido.

McDonald [MCD08] propõe a utilização da tecnologia de RPD para aplicações de comunicação via rádio. Atualmente existe uma quantidade muito grande de tecnologias sendo utilizadas para a comunicação por radiofrequência. Assim, é muito comum que estes dispositivos utilizem processos implementados em *software* para realizarem suas funções, formando assim uma classe de dispositivos conhecidos como *Software Defined Radios* (SDRs). A natureza configurável e paralela dos FPGAs permite a implementação de SDRs com elevado desempenho e a aplicação de RPD neste contexto permite projetos mais eficientes, pois os blocos digitais que compõem os rádios podem ser configurados sob demanda.

1.3 Objetivos do Trabalho

Os objetivos do presente TCC incluem:

- Compreender os princípios e o funcionamento das técnicas direcionadas para o projeto de *hardware* baseado em FPGA utilizando RPD, bem como investigar as atuais tecnologias disponíveis no mercado, com foco na tecnologia Xilinx.
- Desenvolver, com sucesso, uma aplicação que empregue a técnica de RPD em um sistema embarcado.
- Desenvolver um trabalho que sirva como referência para estudantes, pesquisadores e engenheiros que no futuro desejem investir na área RPD, fornecendo uma aplicação de exemplo que empregue o que, no presente momento, são as tecnologias mais atuais para este fim.
- Aplicar conceitos, metodologias e tecnologias aprendidas durante o curso de graduação em Engenharia de Computação.

1.4 Estrutura do Documento

Após a introdução e contextualização realizada pelo Capítulo 1, apresentamos no Capítulo 2 uma descrição detalhada da tecnologia do FPGA que será utilizada no desenvolvimento do trabalho.

No Capítulo 3 é apresentado uma introdução ao fluxo de projeto com FPGA. O Capítulo 4 apresenta o a estrutura do projeto, bem como uma descrição detalhada do fluxo de projeto adotado para seu desenvolvimento.

O Capítulo 5 resume os resultados obtidos após a implementação do projeto. No Capítulo 6 encontra-se a conclusão do trabalho.

2 ARQUITETURA VIRTEX COM SUPORTE A RPD

Para o desenvolvimento deste trabalho será utilizado um FPGA da família Virtex 5, produzido pela empresa Xilinx. A escolha desta família deve-se principalmente à disponibilidade de plataformas de prototipação com este dispositivo em nossos laboratórios (ver Seção 1.1.1).

O estudo da arquitetura Virtex é dividido em duas partes: recursos de hardware, onde serão descritos os principais módulos existentes no FPGA, e os recursos que hardware que viabilizam a implementação de RPD.

2.1 Arquitetura Virtex 5

2.1.1 Lookup Tables (LUTS) e CLBs

No caso da arquitetura do Virtex 5, as LUTs são implementadas utilizando RAMs de 64 bits cada. Cada LUT conta com seis entradas e duas saídas independentes, assim cada LUT é capaz de implementar uma função booleana qualquer com 6 entradas, ou duas funções booleanas com 5 entradas, desde que essas funções compartilhem algumas das entradas.

Na arquitetura Xilinx, cada CLB é dividido em estruturas chamadas de *Slices.* No caso da Virtex 5 cada CLB contém dois *Slices*, sendo que cada um deles está conectado à uma matriz de chaveamento (*Switching Matrix*), porém os *Slices* dentro do CLB não estão conectados diretamente entre si. Esta arquitetura é ilustrada na Figura 5.

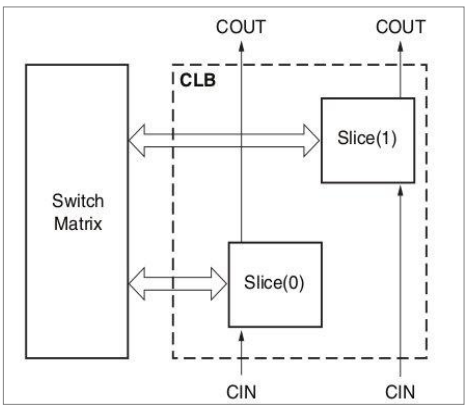


Figura 5 - CLB contendo Slices conectados a uma matriz de chaveamento [XIL12a].

Cada *Slice* contém 4 LUTs, quatro registradores para manutenção de estado, além de circuitos que auxiliam na implementação da lógica de vai-um (*carry)* em operações aritméticas. Estes *Slices* são chamados de SLICEL. A Figura 6 ilustra o SLICEL com maiores detalhes.

Além do SLICEL também existe na arquitetura Virtex uma variação de *Slice* chamada de SLICEM, que além dos recursos já citados, permite também ter suas LUTs configuradas como um registrador de deslocamento de 32 *Bits* e a capacidade de armazenar dados, implementando uma RAM Distribuída.

Slices também contam com outras estruturas para auxiliar na implementação de circuitos digitais comuns na maioria dos projetos. No caso dos SLICEMs, as LUTs podem ser configuradas para, juntamente com os registradores disponíveis no *Slice*, formar um registrador de deslocamento de 32 *bits*. Além disto, os *Slices* também contam com uma lógica dedicada para o tratamento do bit de carry em operações aritméticas (*fast carry chain*), permitindo que somas, subtrações, multiplicações e divisões sejam realizadas mais rapidamente [XIL12a].

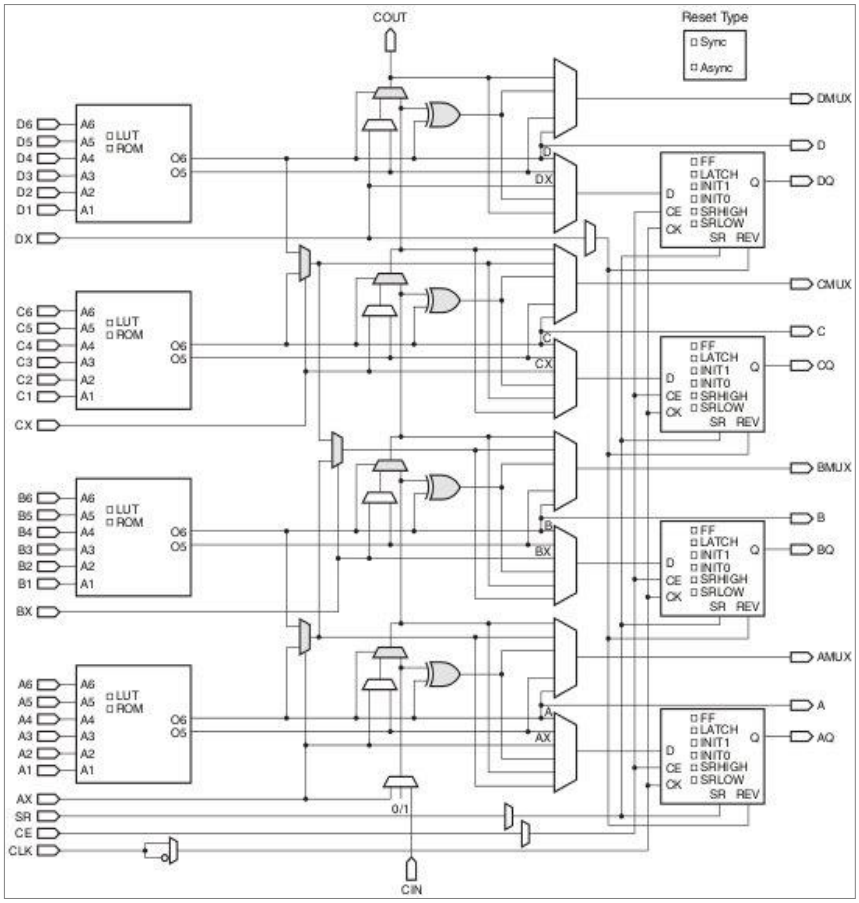


Figura 6 - Estrutura de um SLICEL, observa-se 4 LUTs de 6 entradas, além de 4 registradores [XIL12a].

Vale ressaltar que as LUTs apresentam um atraso constante, ou seja, o atraso na propagação dos sinais é o mesmo, seja qual for a função booleana implementada pela LUT [XIL12a].

2.1.2 Recursos de memória RAM interna

A arquitetura de FPGA Virtex 5 provê duas alternativas de implementação de memórias RAM intrachip, o uso de memórias RAM internas denominadas Block RAMs ou BRAMs e o uso de memória RAM distribuída utilizando LUTs disponíveis dentro de cada CLB [XIL11a].

As memórias RAM dedicadas da arquitetura de FPGAs da Xilinx são denominadas BRAMs. Estas são blocos de memória posicionados ao longo do dispositivo para utilização como memórias de acesso rápido para uma lógica dedicada ou também como FIFOs.

Cada BRAM é uma memória RAM dupla porta com 36kbits de capacidade. É possível definir domínios de clock diferentes para cada porta a fim de gerar uma FIFO assíncrona. Para evitar problemas de acesso simultâneo é possível definir a sequência de prioridades de operação em cada porta através de primitivas Xilinx. Ambas as portas de cada BRAM tem primitivas de prioridade do tipo *Read First* ou então *Write First*.

Além dos módulos dedicados de RAM, é possível utilizar as LUTs dos SLICEM para implementar memórias RAM síncronas com uma, duas ou quatro portas. Utilizando as quatro LUTs disponíveis no *Slice* é possível implementar até 64bits x 4 de RAM (utilizando configuração de porta única) [XIL12a].

2.1.3 Recursos Aritméticos

Além da lógica de Fast Carry Chain já mencionada em 2.1.1, FPGAs da família Virtex 5 também possuem blocos de DSP (Digital Signal Processor, Processador Digital de Sinais). Estes blocos possuem estruturas otimizadas para a realização de diversas operações aritméticas como multiplicação, soma, e comparação.

Estes blocos são chamados de DSP48E. Cada FPGA Virtex 5 pode conter de 24 a 1056 destes blocos [XIL12b].

2.1.4 Arquitetura de hierarquia de *clock*

Em um dispositivo Virtex 5 os sinais de *clock* estão divididos em sinais globais e em sinais locais, sendo que cada FPGA pode conter de 8 a 24 regiões de *clock*, dependendo de seu modelo Figura 7.

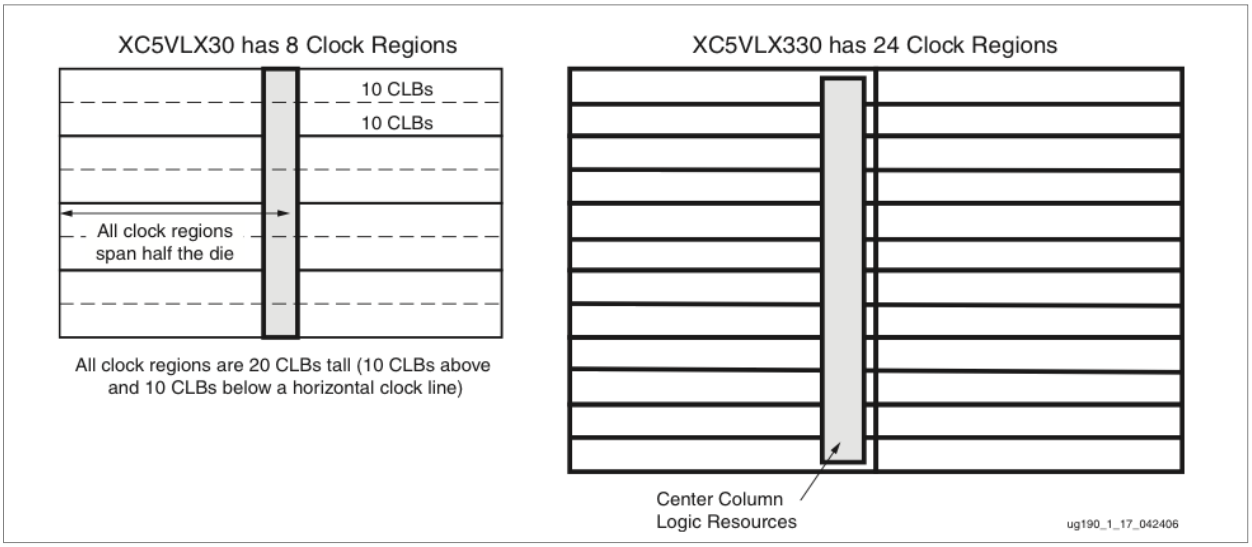


Figura 7 - Regiões de clock em uma Virtex 5 [XIL12a].

Nos dispositivos Virtex 5 existem 32 linhas de *clock* globais, que estão disponíveis para qualquer região. Estas linhas contam com recursos especiais, como *buffers* e linhas de interconexão especialmente projetadas, utilizados para reduzir os atrasos sofridos pelo sinal de *clock* ao longo das linhas.

Além das linhas de *clock* globais, cada região possui suas próprias linhas de distribuição de *clock* local. Cada região possui dois *buffers* de *clock*.

A existência dos *buffers*, tanto globais como locais, é de grande importância pois estes são os dispositivos que garantem a integridade do sinal de clock, evitando problemas comuns como o c*lock skew* (atraso na fase do sinal de *clock* em regiões diferentes do dispositivo) além de garantir o *fan out* necessário para a árvore de *clock* do circuito [XIL12a].

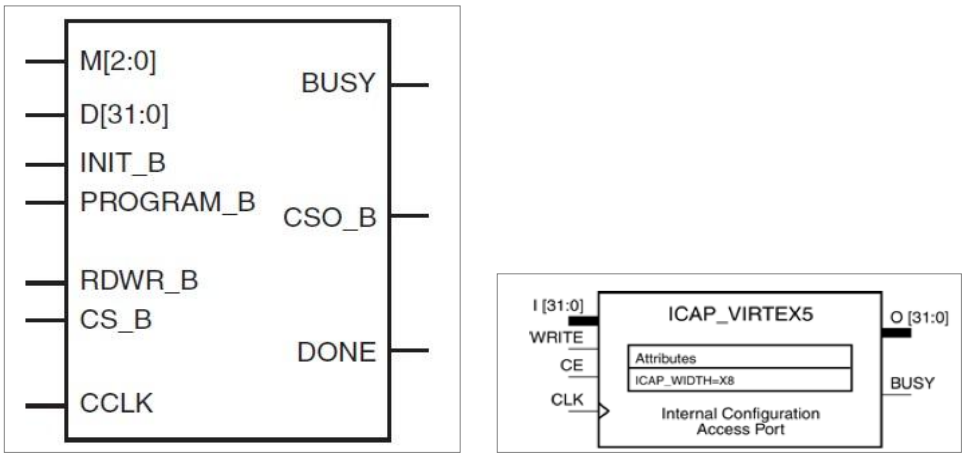
Explorar os recursos de genrenciamento de *clock* do dispositivo Virtex 5 foge ao escopo do presente trabalho, Para maiores detalhes pode-se consultar [XIL12a].

2.2 Recursos de Hardware para RPD

O elemento chave para a implementação de RPD em um FPGA está em seus mecanismos de programação. No caso da tecnologia Xilinx, mais especificamente um dispositivo embarcado no FPGA, chamado de ICAP (*Internal Configuration Access Port)*. A compreensão do

funcionamento do ICAP se torna simples após o estudo de outra interface de programação Xilinx, chamada de *SelectMap*.

A interface SelectMap (Figura 8(a)) consiste em uma interface através da qual é possível tanto configurar o FPGA quanto ler sua configuração através de um barramento cuja largura de bits pode ser configurada para operar com 8, 16 ou 32 bits. [XIL11b]. Através do barramento D da interface *SelectMap* é possível enviar ao FPGA instruções de programação. Estas instruções junto com os dados que estas manipulam (os dados de configuração) compõem o que é chamado de *bitstream*.



(a) (b)

Figura 8 – Interfaces (a) SelectMAP e (b) ICAP.

A interface ICAP (Figura 8(b)) é semelhante à SelectMap, porém está disponível para ser apenas acessada internamente ao FPGA, diferentemente da *SelectMap* que é acessada externamente, através dos pinos do FPGA. Outra diferença é que a ICAP possui dois barramentos separados para leitura e escrita, enquanto a SelectMap possui as duas funcionalidades em um único barramento. Estes dois barramentos, entretanto, não podem ser acessados simultaneamente [XIL11b] .

Internamente a memória de configuração do FPGA está organizada em uma matriz contendo linhas e colunas. A menor unidade endereçável desta memória se chama *quadro*, ou *frame*. Um quadro não necessariamente corresponde a um CLB, geralmente correspondendo a varios CLBs.

Desta forma, apesar do ICAP ser capaz de acessar e reconfigurar os *frames* do FPGA tanto em linhas como em colunas, não é possível reconfigurar um CLB em particular. A granularidade de reconfiguração parcial nos dispositivos Virtex 5 compreende:

- Regiões de Slice: 20 CLBs de altura por 11 CLBs de largura
- Regiões de BRAM: 4 RAMB36
- Regiões de DSP: 8 DSP48
- Regiões de IOB: 40 IOB (um banco)

Sob o ponto de vista prático, a reconfiguração parcial pode ser realizada de forma 2D, ou seja, módulos retangulares, que não necessariamente cobrem toda a altura do dispositivo.

A Figura 9 ilustra a utilização das portas de configuração de um dispositivo Xilinx.

Inicialmente, carrega-se um *bitstream* completo através de uma porta de configuração (como SelectMAP), e durante a execução da aplicação, pode-se carregar diferentes funções, através de arquivos de configuração parciais, através da porta ICAP.

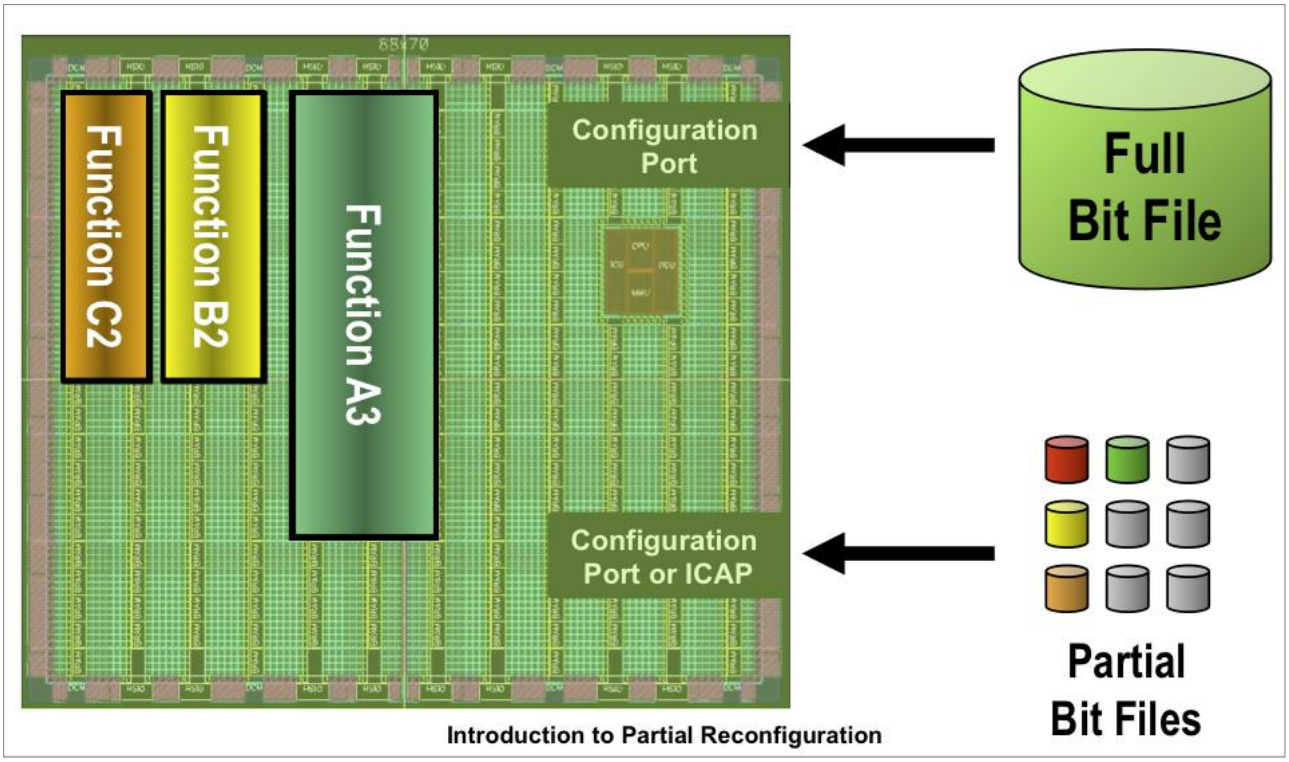


Figura 9 - Utilização das portas de configuração de um dispositivo Xilinx.

2.3 Lógica de interface entre lógica estática e lógica reconfigurável

A interface entre a parte estática e a parte reconfigurável do projeto é feita pelos pinos virtuais atribuídos à partição reconfigurável. Nesses pontos é inserido na periferia da lógica estática um componente de ancoragem para a lógica reconfigurável. Esse elemento de ancoragem é chamado de *Proxy Logic*, normalmente é utilizado para esse fim uma LUT que terá apenas uma entrada utilizada, denominada então como LUT1 [XIL11a]. A existência desses elementos é o que possibilita e garante a correta ligação física entre o roteamento da lógica estática e os módulos reconfiguráveis.

A atribuição desses elementos é feita automaticamente pela ferramenta, mas caso seja necessário ou desejável, é possível também atribuir a localização desses pinos virtuais utilizando restrições específicas no arquivo UCF do projeto.

2.4 Arquitetura dos bitstreams parciais

Bitstreams parciais possuem um cabeçalho com informações sobre a sua implementação, essas informações são acrescentadas pela própria ferramenta *bitgen,* que é a responsável pela geração de *bitstreams* no ambiente Xilinx. A Figura 10 ilustra a composição deste cabeçalho, que contém informações tais como nome do módulo, data de implementação e dispositivo alvo da implementação. Após este o restante da composição do *bitstream* segue a estrutura básica utilizada pelos dispositivos Virtex. A estrutura completa de um *bitstream* de um device Virtex está disponível em [XIL11b].

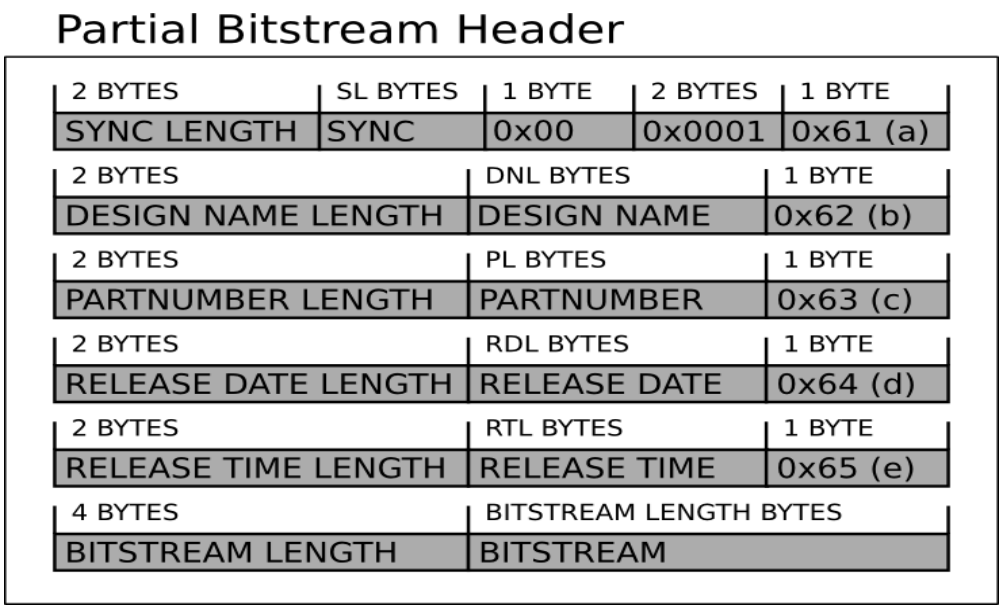


Figura 10 - Cabeçalho padrão de bitstreams parciais.

3 INTRODUÇÃO AO FLUXO DE PROJETO PARA RPD

Neste Capítulo são abordados os fluxos de projeto utilizados para dispositivos configuráveis, que em parte assemelham-se ao fluxo utilizado em dispositivos ASIC, sendo mais simples no que se refere às etapas de verificação.

3.1 Fluxo Genérico de Projeto

No fluxo de projeto para FPGAs é comum se utilizar estratégias *top-down* de desenvolvimento, ou seja, após definida uma arquitetura para o projeto a ser implementado, a implementação das primeiras etapas não influencia o resultado final que será mapeado no FPGA. É recomendado segmentar o projeto em níveis hierárquicos, mas isto não é obrigatório para uma estratégia descendente (*top-down)* convencional de projeto.

Vamos abordar abaixo as etapas de um fluxo convencional de projeto para FPGAs. Indiferentemente do fabricante do dispositivo a ser utilizado, todos seguem fluxos similares. O fluxo geral de projetos para FPGAs segue as seguintes etapas: síntese lógica, mapeamento, posicionamento e roteamento, e geração do *bitstream*.

3.1.1 Síntese Lógica

No processo de síntese lógica utiliza-se como entrada o código RTL que pode ser descrito em VHDL ou Verilog. Há também a possibilidade de ser utilizado como entrada um *netlist* pré-sintetizado. Neste processo é gerado um *netlist* completo para o projeto, que é utilizado como entrada para o próximo passo no fluxo.

As ferramentas de síntese procuram identificar estruturas que foram descritas no código fonte HDL e gerar o seu equivalente em componentes lógicos ainda genéricos. Após esta etapa é comum ser disponibilizada pelas ferramentas de CAD a visualização do diagrama lógico do projeto sintetizado.

3.1.2 Mapeamento

Etapa onde a *netlist* gerada pela síntese lógica é mapeada em blocos lógicos disponíveis no dispositivo utilizado, isto é, a descrição RTL que foi sintetizada em *netlist* agora é transcrita para *flip-flops*, LUTs, etc, que estão disponíveis em cada CLB do FPGA.

As ferramentas procuram nesta etapa otimizar a implementação da arquitetura definida pela descrição HDL, podendo por exemplo definir as estruturas que armazenam informação como um conjunto de *flip-flops,* ou então, como um bloco específico de memória disponibilizada pelo dispositivo utilizado.

Para esta etapa é obrigatória a definição da arquitetura e o membro da família de FPGA que será utilizado, para que a ferramenta de CAD possa inferir corretamente os componentes de lógica a serem utilizados.

3.1.3 Posicionamento e Roteamento

Esta etapa corresponde à síntese física. Nesta etapa é feito o posicionamento dos blocos lógicos que foram gerados na etapa anterior, e a conexão entre os mesmos. É comum que ao efetuar o posicionamento algumas operações lógicas compartilhem recursos do FPGA. As ferramentas disponibilizadas pelos fabricantes tentam aproveitar ao máximo os recursos de cada dispositivo. Portanto, essa etapa pode ser muito demorada para dispositivos de grande

capacidade.

3.1.4 Geração de bitstream

Nesta etapa do fluxo, a lógica a ser configurada no FPGA é escrita em um arquivo, em um formato proprietário, contendo informações adicionais como CRC e definições das características elétricas do dispositivo, como padrões das portas de entrada/saída. Este arquivo é denominado de *bitstream.*

3.1.5 Fluxo de projeto em FPGA utilizando a ferramenta PlanAhead

Como descrito anteriormente, neste trabalho iremos utilizar ferramentas e dispositivo da fabricante Xilinx. Dado que atualmente a ferramenta de projeto utilizada pelo grupo de pesquisa que sedia este trabalho é o *PlanAhead*, iremos fazer uma introdução ao uso dessa ferramenta para o fluxo de projeto FPGA.

O fluxo de projeto é guiado pelo menu de acesso rápido a ferramentas e opções específicas ao ponto atual na implementação. Este menu fica a esquerda da interface principal, como pode ser visto na Figura 11.

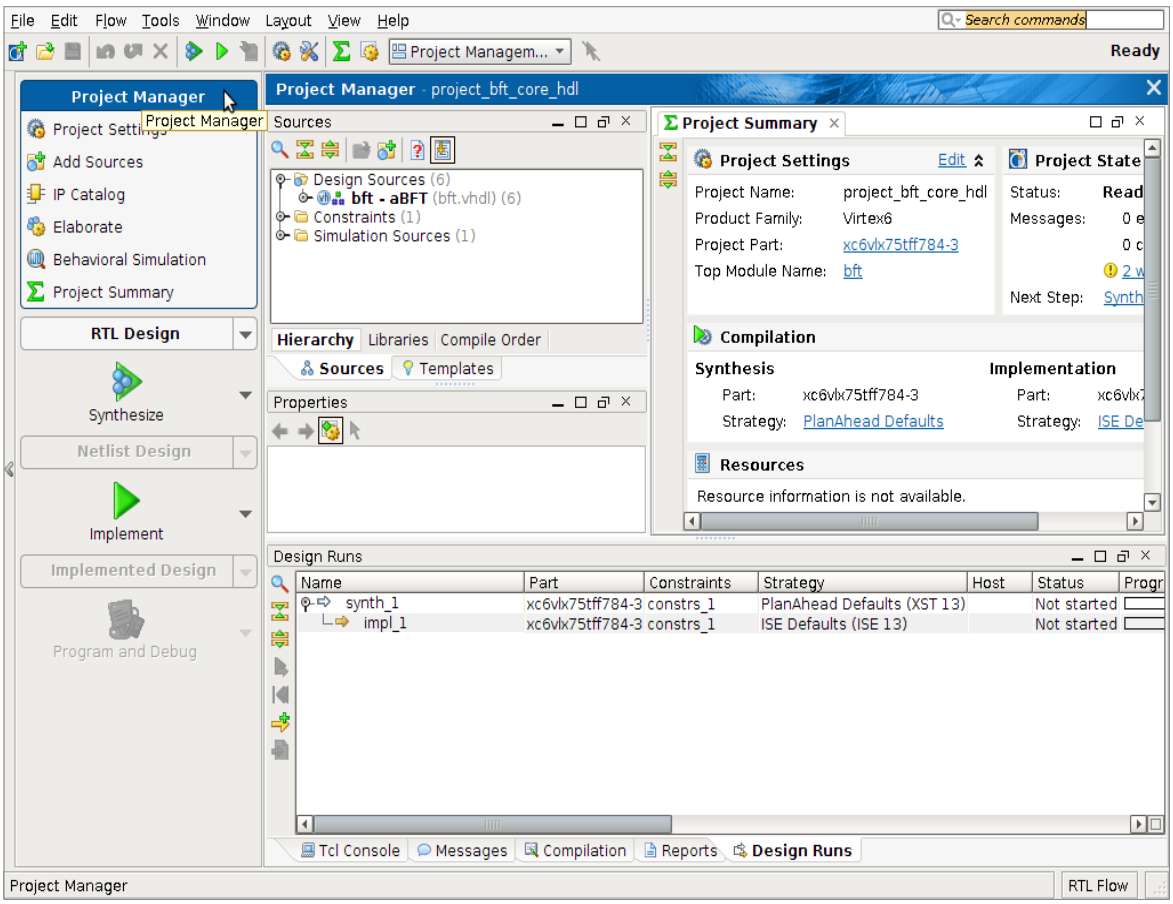


Figura 11 - Interface de navegação do fluxo de projeto da ferramenta PlanAhead.

A primeira etapa no fluxo utilizando a ferramenta PlanAhead é criar um projeto e nele especificar o diretório destino de todos os arquivos gerados no processo. Após especificar o nome e o local do projeto é necessário especificar quais serão os arquivos de entrada, isto é, definir se o projeto utilizará arquivos RTL, arquivos já sintetizados ou se será feita a importação de um projeto

completo da ferramenta ISE. A interface do IDE (*Integrated Development Environment*) disponibiliza um assistente com dicas e informações para a criação do projeto como pode ser visto na Figura 12.

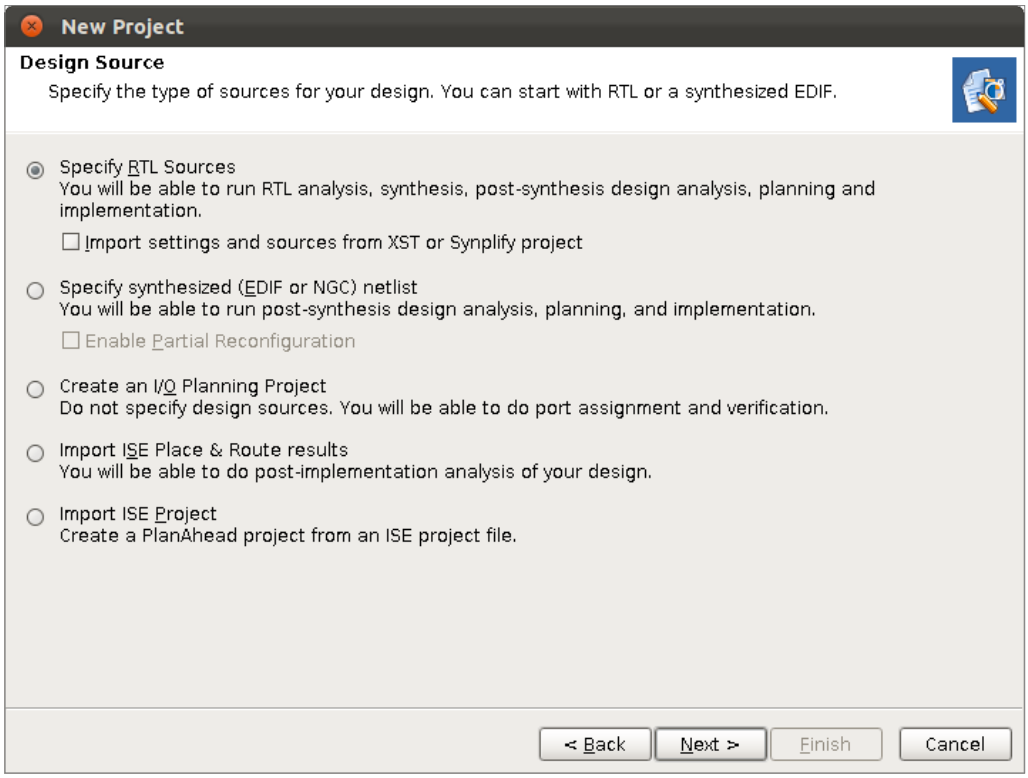


Figura 12 - Seleção de arquivos fonte do projeto.

Após a criação do projeto e a inclusão dos arquivos fonte, é apresentado um conjunto de ferramentas e análises que já estão disponíveis nesta etapa de projeto, tais como: adição de IPs, elaboração do projeto e simulação comportamental do código RTL.

3.1.5.1 Etapa de síntese RTL

A sequência do fluxo corresponde à síntese lógica. Para esta síntese é necessário executar o comando "*Synthesize*" que se encontra na barra de navegação do fluxo de projeto. É possível alterar alguns parâmetros de síntese, tais como definições de uso de recursos, caso seja necessário. A ferramenta disponibiliza opções de otimização de estrutura nessa etapa, tais como: balanceamento de atraso inserindo-se registradores para diminuir o caminho entre pontos registrados por *clock*, além da opção de otimização de estrutura, inferindo funções nos blocos específicos (DSP e memória) disponíveis no FPGA.

Após a síntese, ao selecionar-se *"Netlist Design",* é apresentado um conjunto de ferramentas e análises que já estão disponíveis nesta etapa de projeto, tais como: estimação de uso de recursos utilizados pelo projeto, verificação de boas práticas de codificação através da ferramenta DRC e análise de *timing* (apenas preliminar nesta etapa).

3.1.5.2 Etapa de Implementação

Esta operação compreende as etapas de mapeamento, de posicionamento e de roteamento, sendo a entrada o *netlist* produzido pela síntese RTL. Assim como as etapas anteriores, também é possível mudar alguns parâmetros da implementação. Um dos parâmetros de implementação que podem ser mudados é o esforço no posicionamento e roteamento, que

podem ser realizados com objetivo de: menor consumo de área, maior desempenho possível ou outras abordagens que podem ser inclusive especificadas pelo usuário.

3.1.5.3 Configuração

Chegando nesta etapa é possível gerar o *bitstream* e utilizar a ferramenta *Impact* para realizar a gravação da imagem gerada em memórias *flash* ou no próprio dispositivo alvo através do barramento JTAG.

3.2 Fluxo com suporte à RPD

O fluxo de projeto voltado a RPD é tratado com a metodologia de projeto ascendente (*bottom-up)*, ou seja, todas as etapas tem de antever o uso de partições reconfiguráveis em tempo de execução e tomar as devidas precauções para seu correto funcionamento.

3.2.1 Preparação da parte estática do projeto

Para a implementação de um sistema com capacidade de reconfiguração parcial dinâmica é necessário definir em um sistema estático regiões capazes de serem reconfiguradas.

É necessário então um ambiente estático com conexões padrão a todos os módulos reconfiguráveis que serão desenvolvidos para aquele ambiente. Após a síntese desse sistema estático define-se as regiões onde será possível realizar a RPD. Isto é feito através de um processo chamado de *Floorplan*, onde são criados regiões para alocação de recursos. Cada módulo terá seu conjunto de restrições (*constraints*),. Para a definição de *constraints* de área utiliza-se um novo conceito de especificação introduzido na ferramenta PlanAhead, chamado de Pblock. A Figura 13 mostra a interface gráfica utilizada para definir os *Pblocks.*

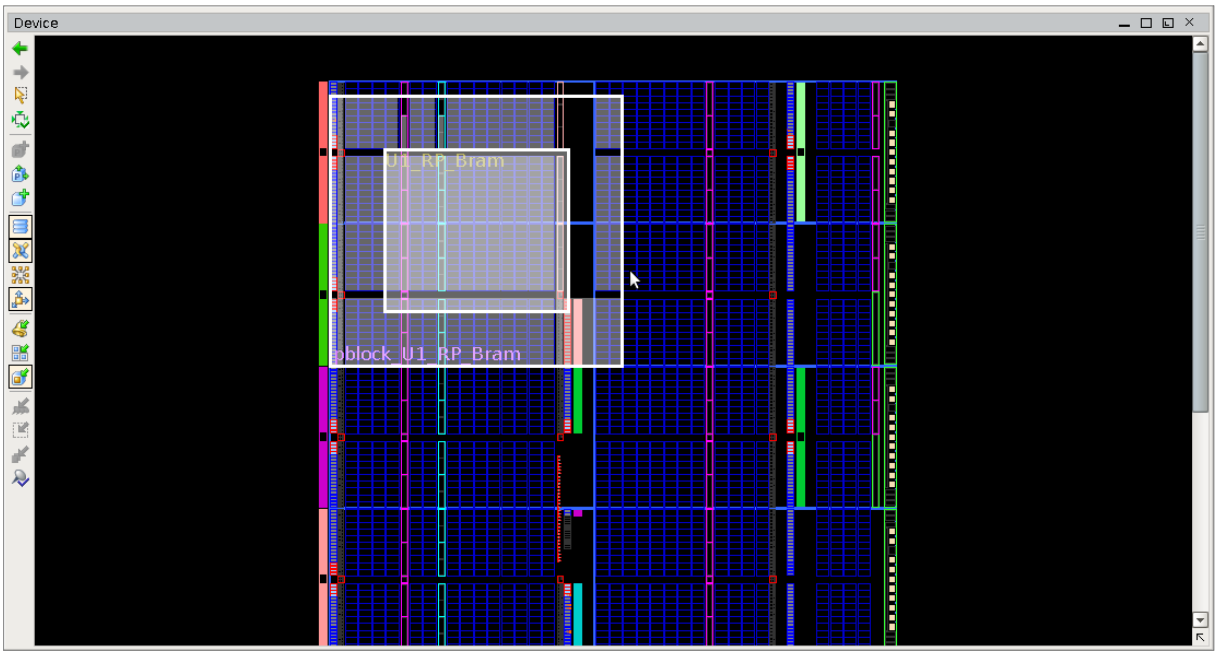


Figura 13 - ferramenta gráfica para geração de constraints de área.

Algumas das estruturas que são obrigatoriamente integradas à parte estática do projeto são: DCMs, linhas globais de clock além da interface de gerência e gravação ICAP. Por outro lado, estruturas como linhas de clock locais, blocos de memória e blocos de DSP, são utilizáveis em módulos reconfiguráveis.

3.2.2 Criação de módulos reconfiguráveis

Após a criação das partições reconfiguráveis, qualquer lógica que seja atribuída àquela partição será considerada como um módulo reconfigurável e após o comando "*Promote Partitions*" (vide 4.3.12) estará disponível para ser incluída em outros projetos, desde que seja refeita a etapa de síntese física. Após a síntese física dessa etapa, usando a informação de PR destino de cada módulo reconfigurável, serão inseridos os pontos de ligação dos pinos da partição à lógica estática. Esta ligação é realizada por meio de uma lógica de *Proxy* que usualmente é uma LUT1 que serve para ancoragem, conforme dito em 2.3.

É preciso cuidado no desenvolvimento dos módulos a serem definidas como módulos reconfiguráveis, pois ao contrário de um fluxo de projeto *top-down*, aqui não será realizada otimização de estruturas entre módulo reconfigurável e ambiente estático.

Após definir os módulos reconfiguráveis a serem utilizados no projeto é necessário fazer a verificação da integridade da estrutura de RPD implementada. Para esta verificação utiliza-se o comando "PR_Verify", cuja interface gráfica pode ser vista naFigura 14. Nesta etapa serão verificadas as boas práticas de projeto de RPD como não incluir buffers de IO no módulo, assim como o correto posicionamento e roteamento de pinos que ligam os módulos reconfiguráveis à lógica estática.

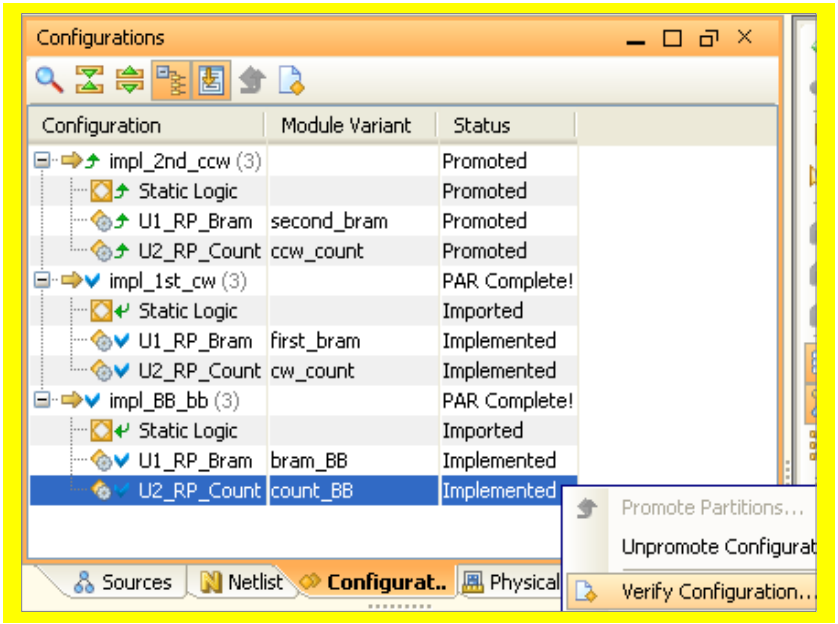


Figura 14 - assistente para verificação da estrutura de RPD implementada.

3.2.3 Geração de *bitstreams* completos e parciais

Após a implementação dos módulos reconfiguráveis é possível gerar os *bitstreams* parciais de cada módulo e então carregá-los para uma memória, que será acessada via processador ou via uma lógica dedicada que fará a comunicação com o ICAP.

Aqui também é válida a utilização de ferramentas como o *Impact* para a gravação dos dos *bitstreams* diretamente no dispositivo ou em memórias flash, podendo ser empregada uma cadeia JTAG.

4 DESENVOLVIMENTO DO PROJETO RECONFIGURÁVEL

O projeto desenvolvido para demonstrar o fluxo de projeto com característica de reconfiguração parcial dinâmica (RPD) é um sistema composto por um processador embarcado no FPGA e periféricos, sendo um dos periféricos um módulo aritmético reconfigurável projetado como aplicação exemplo. Este fluxo teve por base a referencia [XIL12c].

Esse projeto será gerado através de ferramentas de apoio contando com um processador MicroBlaze e periféricos, como a interface com a memória onde serão armazenados os *bitstreams* parciais e a interface RS232 que servirá como interface para a aplicação que estará sendo executada. A Figura 14Figura 15 apresenta um diagrama ilustrando a arquitetura do projeto alvo.

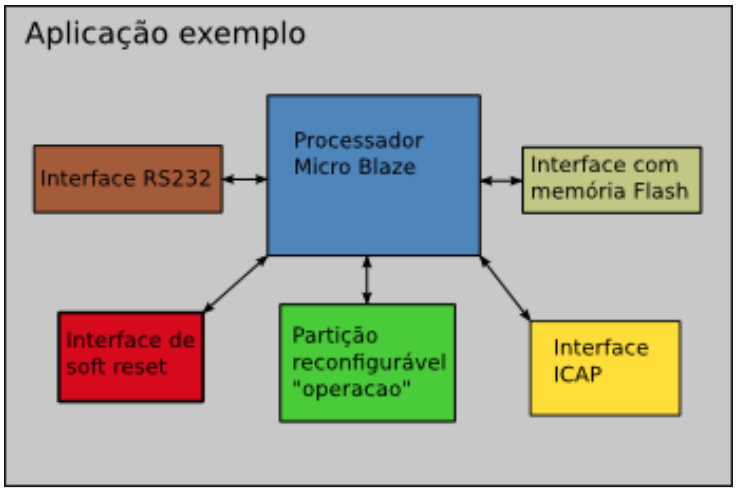


Figura 15 – Arquitetura do projeto com partição reconfigurável.

4.1 Projeto de Hardware

O módulo reconfigurável desenvolvido é uma unidade aritmética que tem como entrada 2 operandos, e após receber uma sinalização de ativação, através do pino inEN, retorna o valor calculado, sinalizando com o status de operação concluída através do pino outDone. A interface do módulo é apresentada na Figura 16.

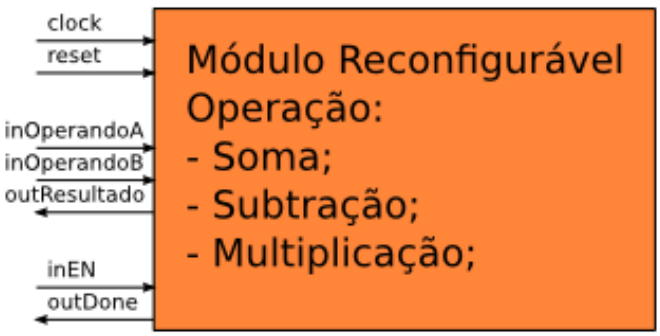


Figura 16 - Interface dos módulos reconfiguráveis do projeto desenvolvido.

Serão implementadas três variações do módulo, uma realizará a soma dos operandos, uma a subtração e uma a multiplicação.

O módulo reconfigurável possui um atraso constante para o processamento da operação requerida. O diagrama de tempo da comunicação entre o processador e o módulo reconfigurável é apresentado na Figura 17.

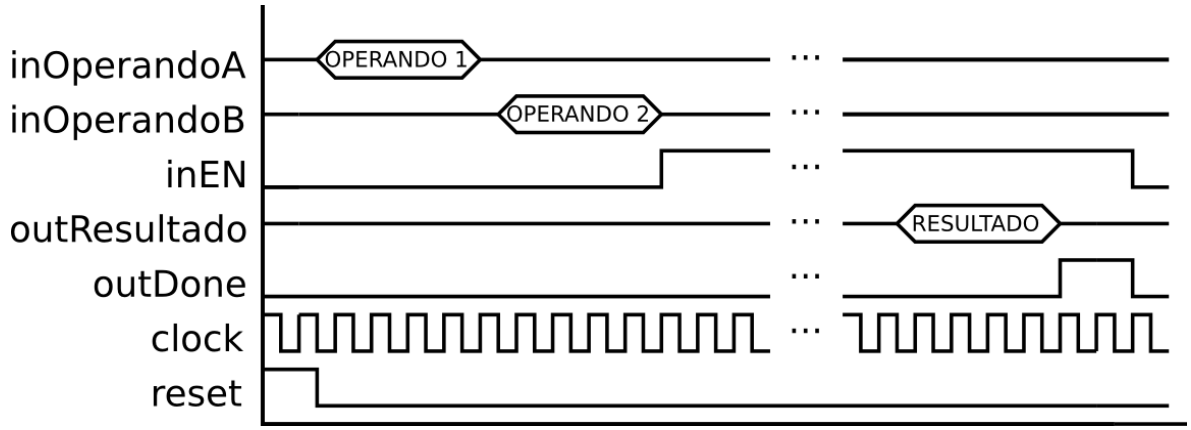


Figura 17 - Diagrama de temporização da comunicação entre o processador e módulo reconfigurável.

Para a correta operação o módulo deve receber os dois operandos pelas portas inOperandoA e inOperandoB, após é necessário levar o pino inEN ao nível alto por pelo menos um ciclo de *clock*.

Quando o módulo concluir o processamento dos operandos o pino outDone será levado ao nível alto, sinalizando ao processador que o resultado encontra-se na porta outResultado.

4.2 Projeto de Software

O *software* desenvolvido deverá implementar a interface com o usuário através da porta serial, bem como as rotinas de comunicação com os periféricos.

Dependendo da operação necessária no Módulo Reconfigurável, o *software* necessitará se comunicar com o ICAP e enviar para este o *bitstream* adequado que deverá ser carregado da memória *flash*. Para realizar a interface do *software* com o ICAP é utilizada a biblioteca fornecida pela Xilinx [XIL10b].

Figura 18 ilustra o fluxo de execução implementado pelo *software* escrito em linguagem C para a plataforma MicroBlaze.

4.3 Fluxo de Projeto

O fluxo de projeto utilizando partições reconfiguráveis é apresentado na Figura 19. São necessárias ferramentas específicas para executar cada passo do fluxo. As ferramentas que são utilizadas incluem o XST, o XPS, o SDK, o PlanAhead e o Impact.

As seções seguintes abordam detalhadamente casa etapa deste fluxo.

4.3.1 Criação dos Módulos Reconfiguráveis

O primeiro passo a ser realizado no fluxo de projeto é a codificação dos módulos reconfiguráveis. Para isto pode-se utilizar qualquer editor de textos, preferencialmente algum que ofereça recursos amigáveis para o uso da linguagem VHDL, como EMACS [EMA12]. Deve-se observar que todos os módulos devem possuir a mesma interface externa, ou seja, a mesma *Entity*, devendo apenas possuir implementações diferentes. A entidade do módulo reconfigurável é apresentada na Listagem 1.

```
entity executor is
    port
    (
        inOperandoA : in std_logic_vector(0 to 31);
        inOperandoB : in std_logic_vector(0 to 31);
        outResultado    : out std_logic_vector(0 to 31);

        inEN        : in std_logic;
        outDone     : out std_logic;

        clock       : in std_logic;
        reset       : in std_logic
    );
end entity executor;
```

Listagem 1 - Interface padrão adotada no presente projeto para o módulo reconfigurável.

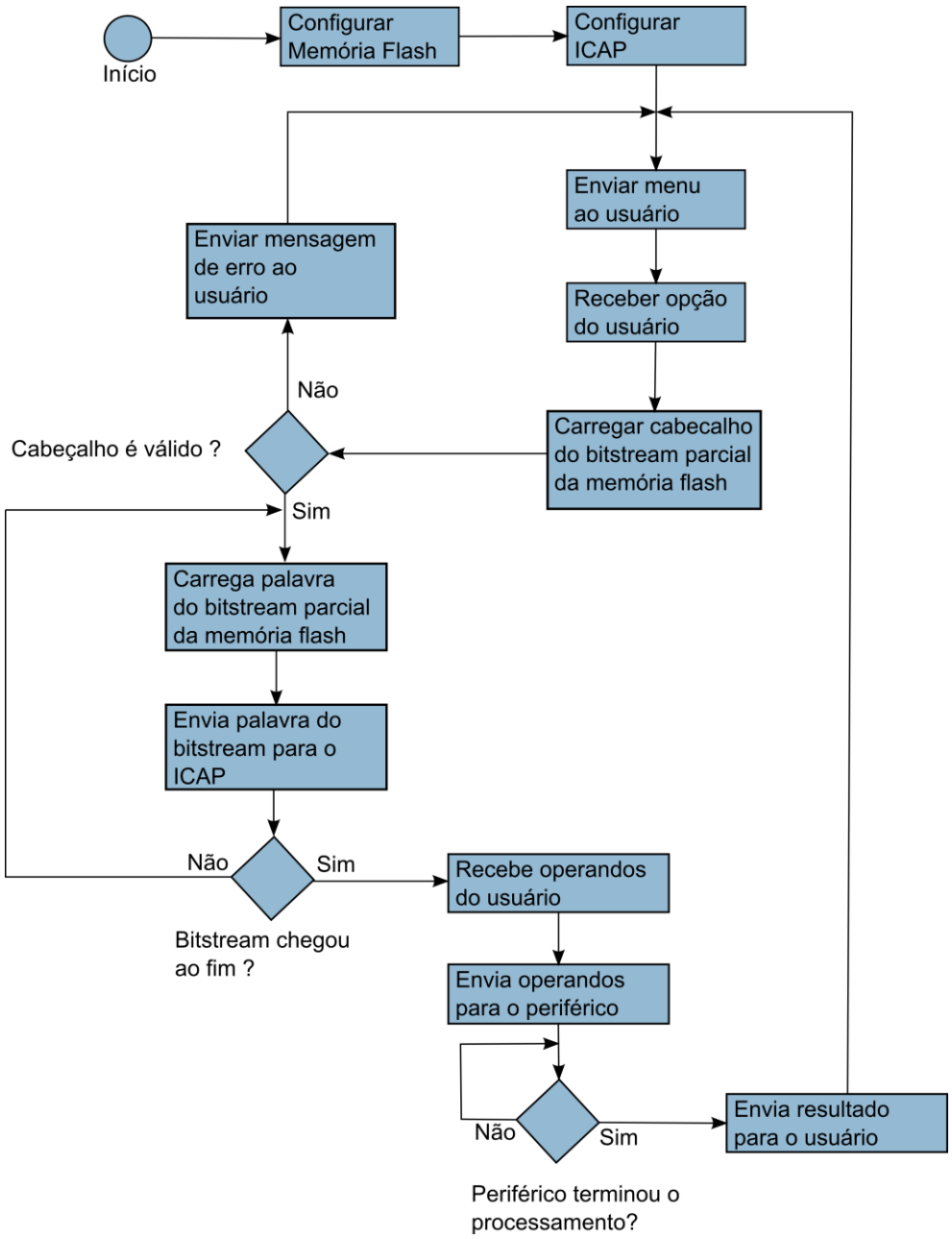


Figura 18 - Fluxo de execução do software

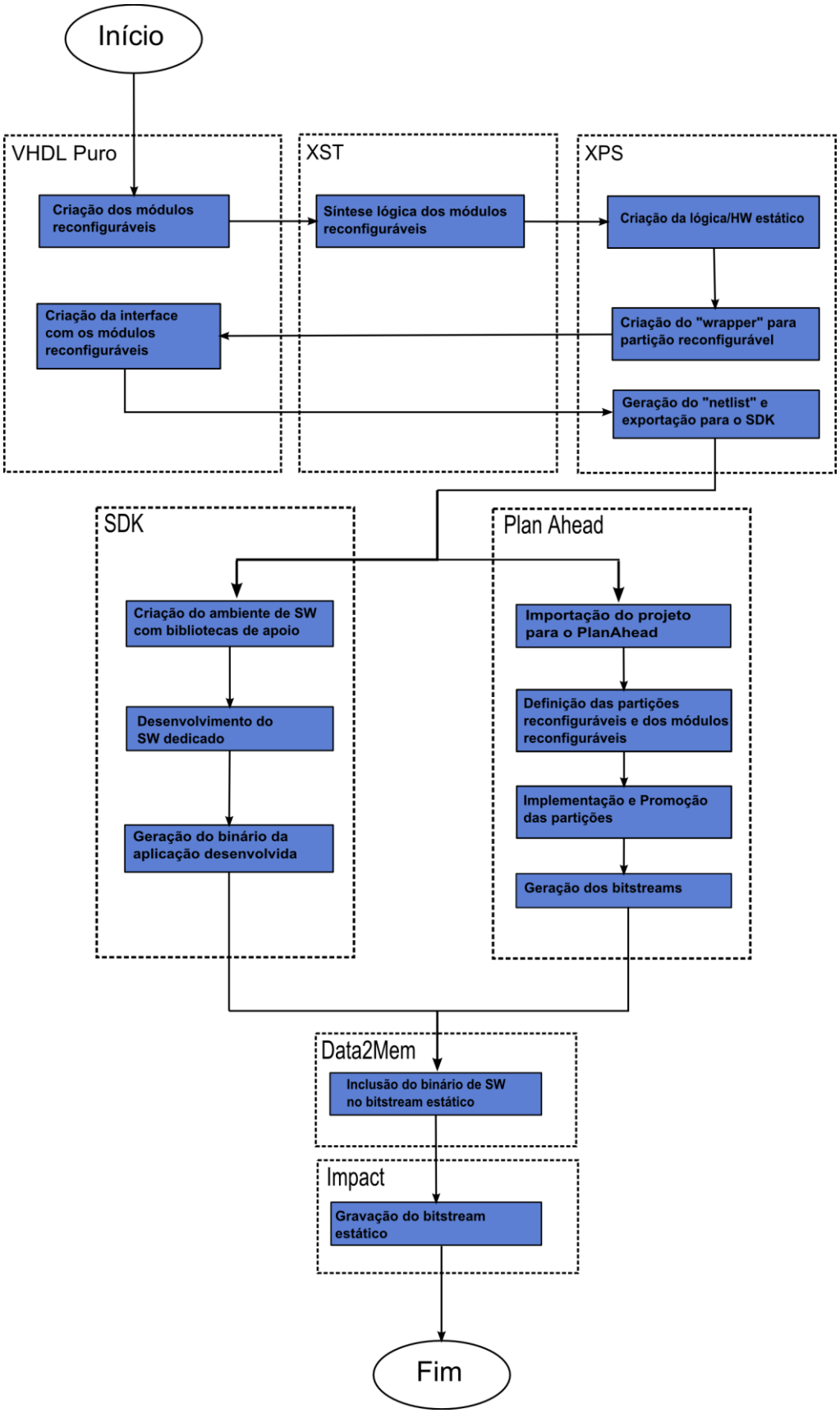


Figura 19 - Fluxo de projeto com partições reconfiguráveis.

4.3.2 Síntese Lógica dos Módulos Renconfiguráveis

Uma vez descritos os módulos reconfiguráveis, o passo seguinte consiste em realizar a síntese lógica de cada módulo. Para isto deve ser utilizada a ferramenta XST (*Xilinx Synthesis Tool*). Como resultado deste processo, obtém-se um ou mais arquivos de *netlist*, cada um contendo a descrição de cada módulo reconfigurável.

O uso do XST requer que seja desenvolvido um *script* de síntese. Neste *script* devem estar contidas informações como os arquivos de entrada, arquivos de saída e parâmetros para a síntese. O *script* utilizado para síntese dos módulos do projeto é apresentado na Listagem 2, onde é possível observar que existe um comando *Run* seguido de uma lista de parâmetros. Este conjunto se repete para cada módulo que deve ser sintetizado, neste caso soma.vhd, subtração.vhd e multiplicação.vhd. A descrição detalhada de cada parâmetro é apresentada na L**Erro! Fonte de referência não encontrada.**. Para executar o XST deve-se usar o comando "*xst –ifn scrip.run*", onde *script.run* corresponde ao nome do *script* de síntese.

```
Run
-ifn ./src/soma.vhd
-iobuf NO
-bufg 0
-iob false
-p virtex5
-ifmt VHDL
-ofn ./ngc/soma.ngc

run
-ifn ./src/subtracao.vhd
-iobuf NO
-bufg 0
-iob false
-p virtex5
-ifmt VHDL
-ofn ./ngc/subtracao.ngc

run
-ifn ./src/multiplicacao.vhd
-iobuf NO
-bufg 0
-iob false
-p virtex5
-ifmt VHDL
-ofn ./ngc/multiplicacao.ngc
```

Listagem 2 - Script para síntese lógica dos módulos reconfiguráveis no XST.

Tabela 1 - Parâmetros para uso do XST.

Parâmetro	Descrição
-ifn	Arquivo(s) de entrada.
-iobuf NO, -bufg 0 e –iob false	Informam ao XST que não devem ser incluídos buffers nos pinos de IO. Isto é necessário para que na etapa de síntese física seja possível utilizar este módulo como um módulo reconfigurável.
-p virtex5	Dispositivo alvo, pode ser um *part number* específico ou o nome de uma família.
-ifmt	Linguagem em que o módulo foi codificado.
-ofn	Nome do arquivo em formato NGC a ser gerado pelo XST.

Como resultado deste processo obtém-se um ou mais arquivos no formato *netlist* (arquivo

contendo as portas lógicas resultantes da síntese) cara um contendo a descrição de cada módulo reconfigurável.

4.3.3 Criação da Lógica/Hardware estáticos

Neste passo é criado o sistema contendo o processador MicroBlaze, a interface com a partição reconfigurável e demais periféricos utilizados no projeto, como a interface com o cartão *compact flash*. Para isto deve ser utilizada a ferramenta XPS, conforme os passos que se seguem.

Inicialmente deve-se abrir a ferramenta XPS e na primeira janela selecionar "*Base System Builder Wizard*" para criar um novo projeto, conforme Figura 20.

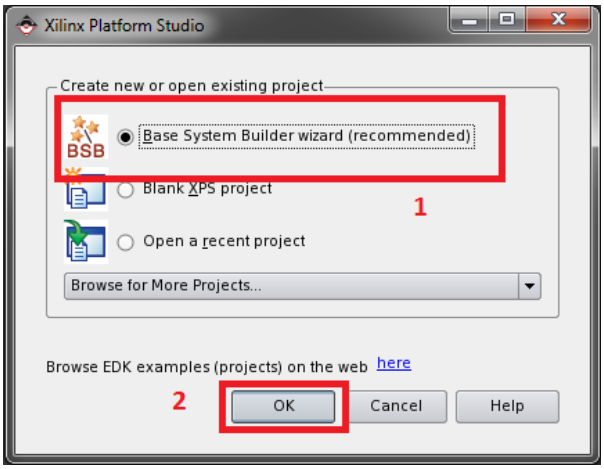


Figura 20 - Criação de um novo projeto no XPS.

Na tela seguinte, Figura 21, seleciona-se o diretório em que os arquivos do projeto devem ser salvos. Nesta tela também deve ser definido o padrão do barramento que será utilizado para conectar o processador MicroBlaze com seus periféricos. Neste projeto será utilizado o padrão PLB, que é um padrão de barramento disponibilizado pela Xilinx. Na próxima tela não é necessário realizar nenhuma alteração, pois a opção de criar um novo projeto já está selecionado por padrão, bastando avançar para a tela seguinte.

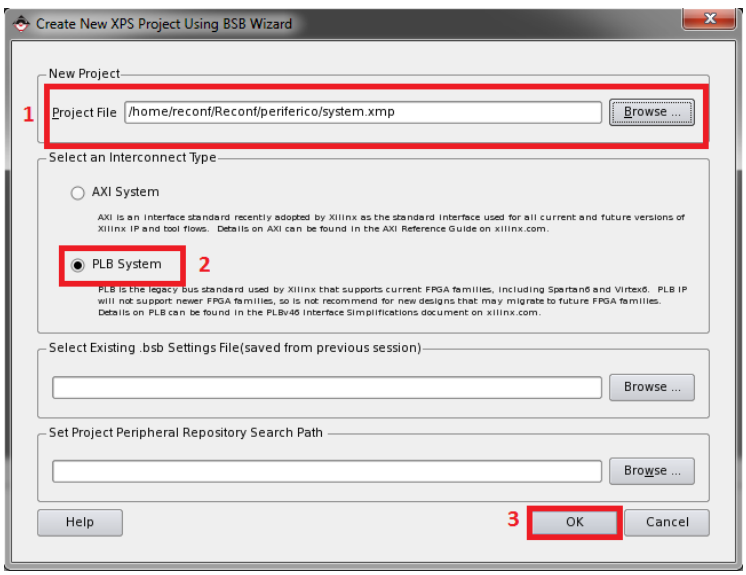


Figura 21 - Seleção do diretório do projeto e padrão do barramento.

Na tela seguinte seleciona-se o dispositivo alvo do projeto que está sendo criado. No presente caso utiliza-se a placa ML505, devendo-se selecionar "*Virtex 5 ML505 Evaluation Platform*" em *Board Name* e "1" em *Board Revision*, conforme a Figura 22.

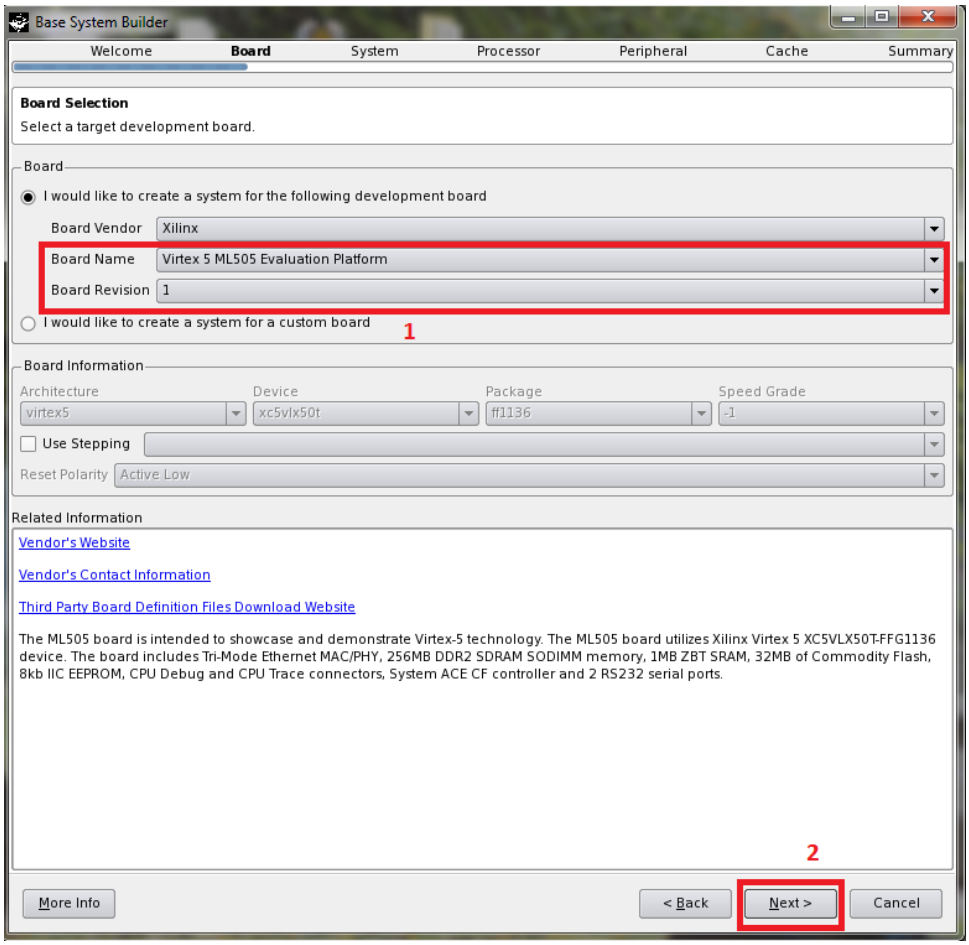


Figura 22 - Seleção da placa ML505.

Na tela seguinte não é necessário realizar nenhuma alteração, pois a opção selecionada por padrão, criação de um sistema com apenas um processador, é adequada para o projeto realizado, bastando avançar para a próxima tela.

Na tela seguinte deve-se informar o tipo de processador, sua frequência de relógio e sua memória local. No caso do projeto desenvolvido o tipo deve ser MicroBlaze, frequência de operação igual a 100 Mhz e 64 KBytes como quantidade de memória local. É importante selecionar a frequência de 100 Mhz, pois esta coincide com a frequência de operação do módulo ICAP interno do FPGA. A Figura 23 ilustra as opções necessárias.

Na sequência, seleciona-se os periféricos que farão parte do sistema. Para o desenvolvimento do projeto são necessários: F*lash* e *RS232_Uart_1*, sendo que este último deve ter seus parâmetros ajustados conforme se observa na Figura 24.

As próximas duas telas não precisam ser alteradas, bastando-se avançar clicando nos botões *Next* e *Finish*, o que fará com que a ferramenta retorne para sua tela principal exibindo as características do projeto que foi criado.

4.3.4 Criação do “Wrapper” para partição reconfigurável

Após a criação do sistema de base, é necessário criar um *wrapper*, ou seja, uma camada de hardware que realiza a interface entre o módulo reconfigurável e o barramento do processador. Nesta sessão esta camada será criada como um periférico comum. Em 4.3.5 este *wrapper* será modificado para se comunicar com o Módulo Reconfigurável.

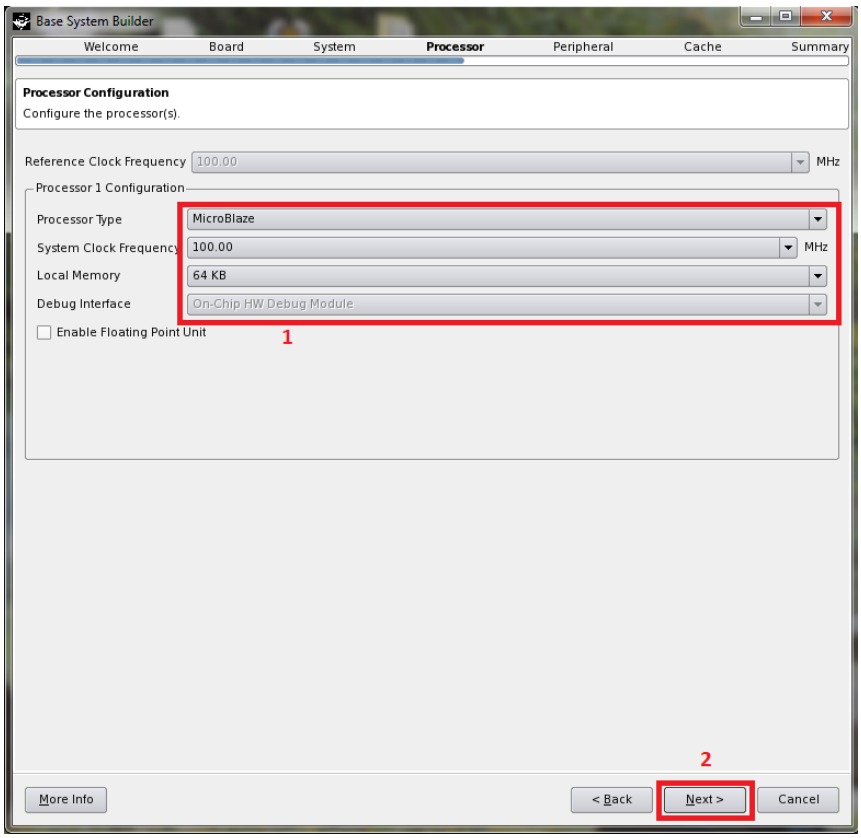


Figura 23 - Seleção de parâmetros do processador.

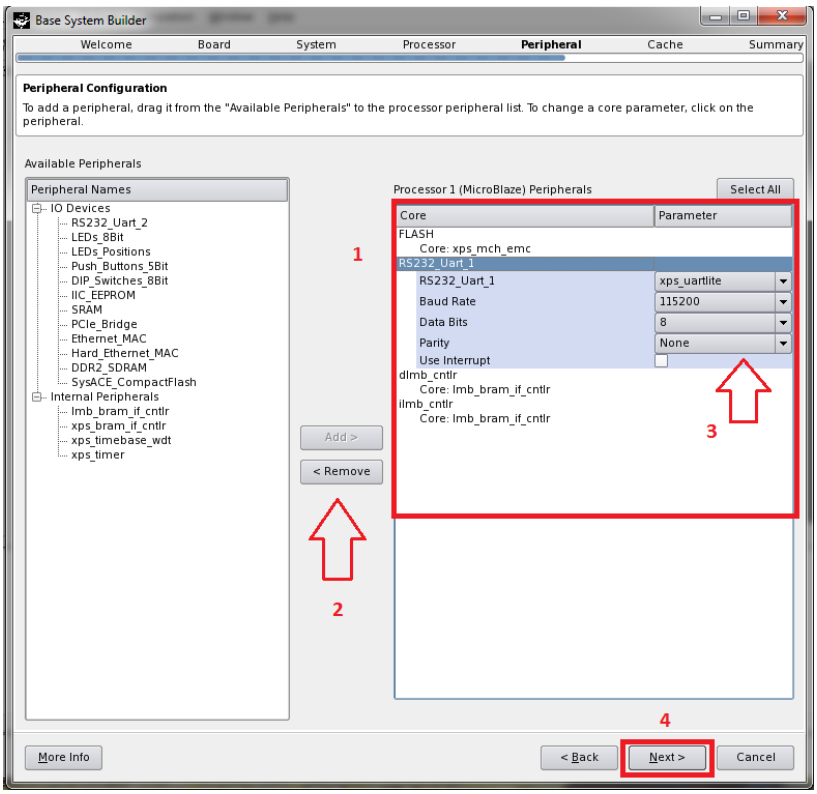


Figura 24 - Seleção e configuração dos periféricos.

Para criar um novo periférico na ferramenta XPS, deve-se selecionar a opção *Create or Import Peripheral*, que se encontra no menu *Hardware*, conforme ilustra a Figura 25. Após selecionado o item no menu, clique em *Next* na tela de apresentação que deverá ser exibida.

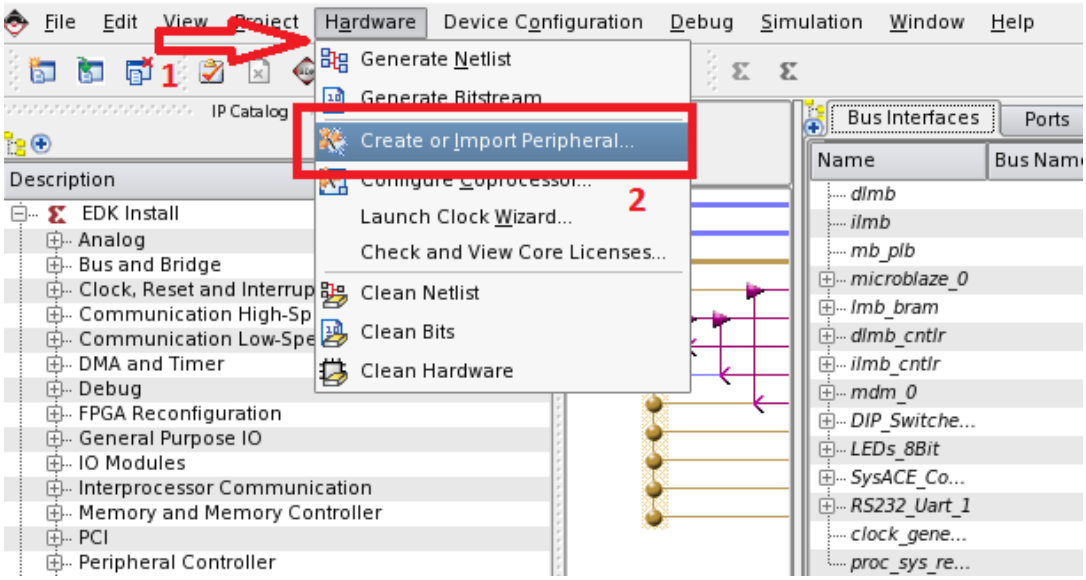


Figura 25 – Passo de criação de um novo periférico no XPS.

Na próxima tela selecione *Create Template for new Peripheral*, caso esta já não esteja selecionada, e clique em *Next*, conforme ilustra a Figura 26.

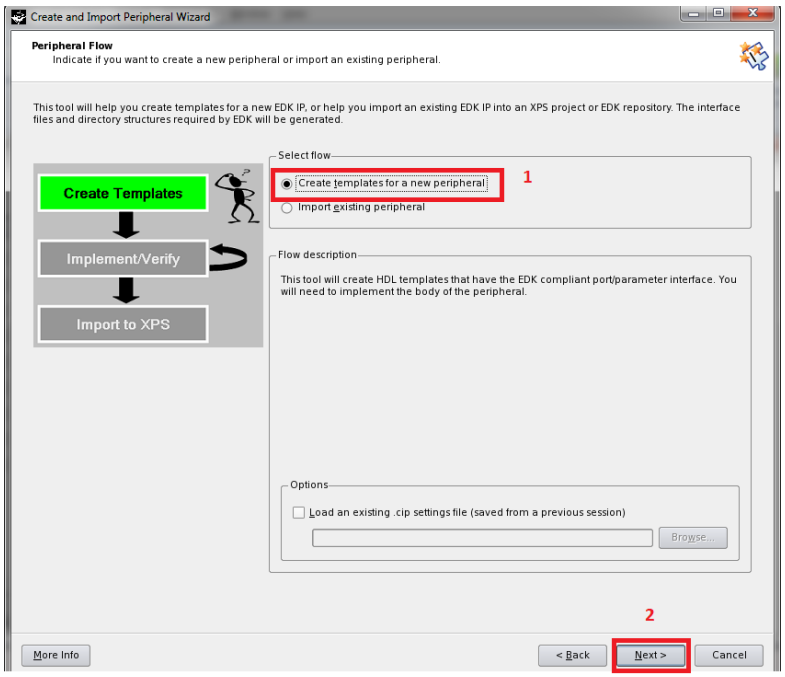


Figura 26 - Diálogo para criação de um novo periférico.

Na tela seguinte deve-se informar que o novo periférico deve ser armazenado no próprio projeto do XPS, selecionando-se a opção *To an XPS Project* e clicando-se em *Next,* conforme a Figura 27.

A seguir deve-se definir o nome e a versão do periférico. Para este projeto o periférico será denominado de "*operacao*", enquanto a versão não deve ser alterada, ficando como 1.00.a, conforme Figura 28.

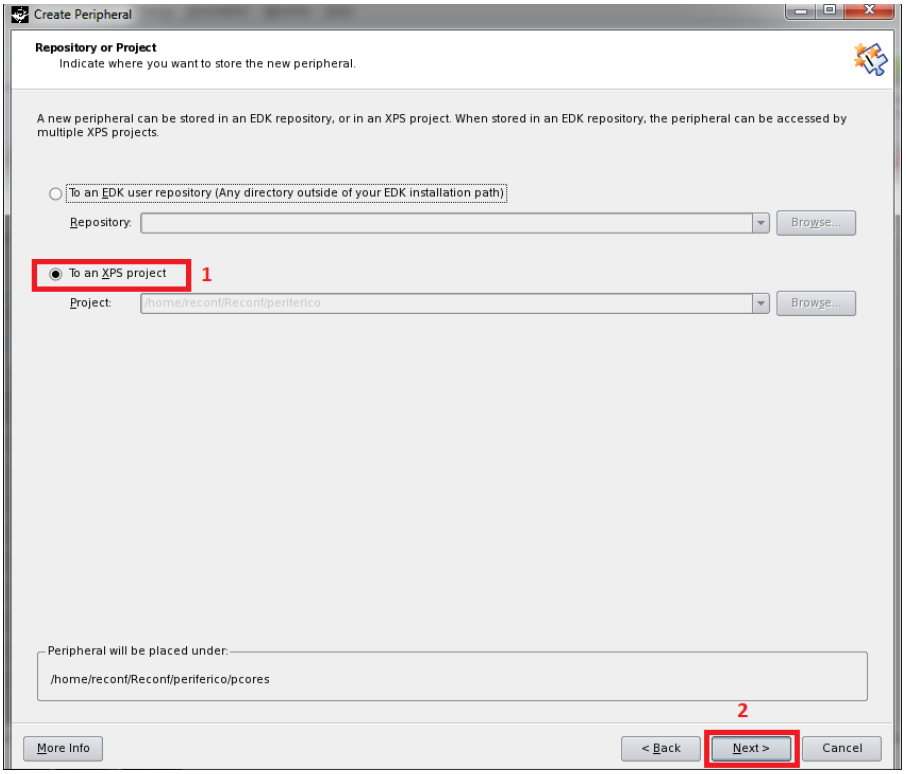


Figura 27 - Salvar periférico juntamente com o projeto do XPS.

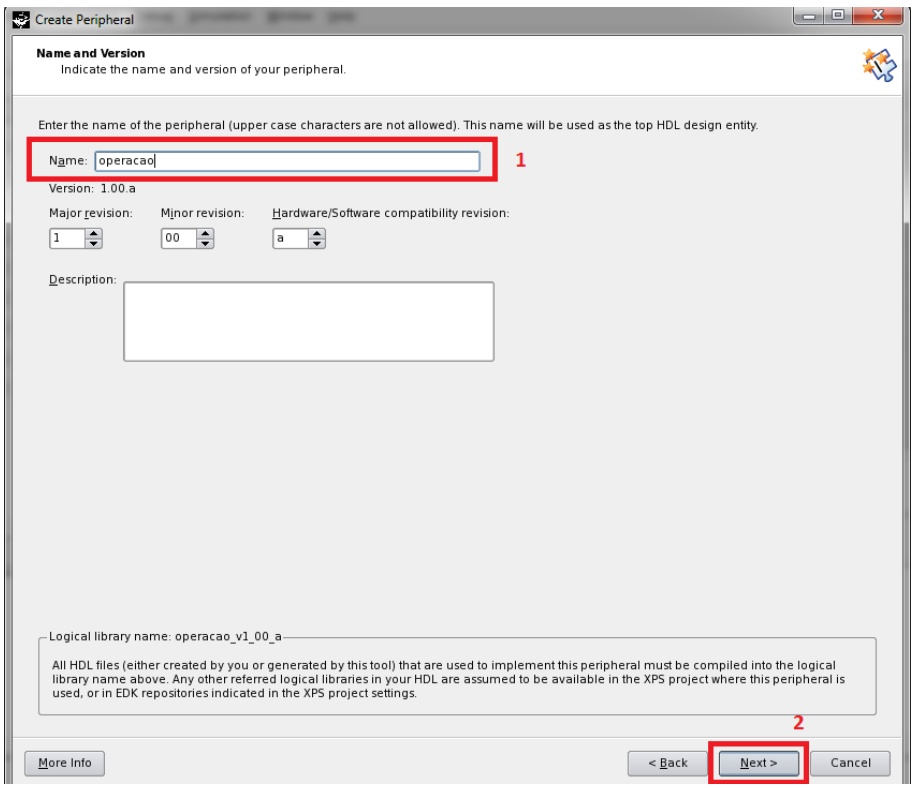


Figura 28 – Definição do nome e versão do periférico.

Na próxima tela deve-se especificar qual o padrão de barramento o periférico utilizará. Conforme definido na sessão anterior, o barramento utilizado é o PLB, logo esta deve ser a opção informada, conforme Figura 29. As próximas duas telas não necessitam de alterações, podendo serem avançadas clicando-se no botão *Next.*

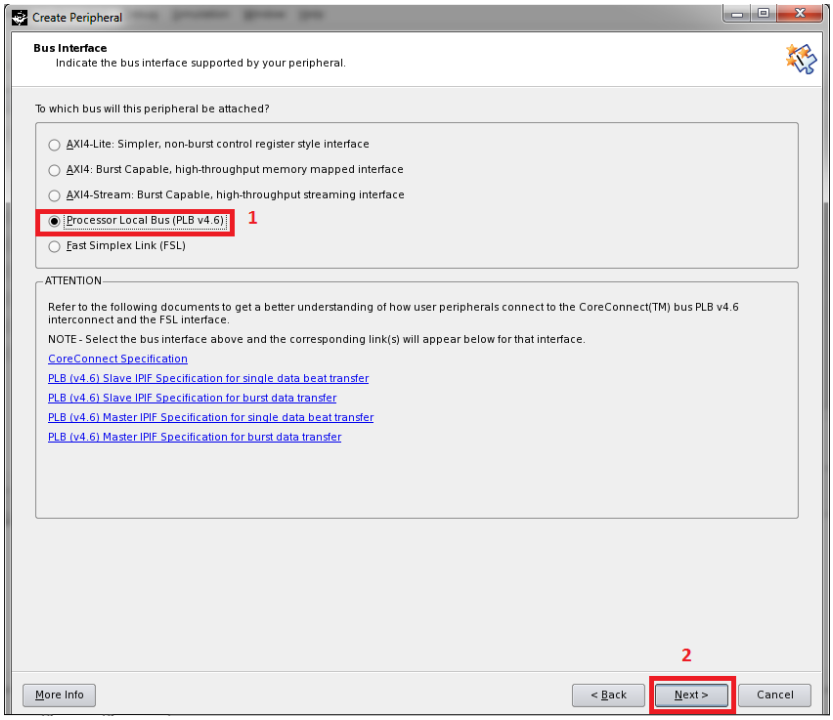


Figura 29 - Seleção do tipo de barramento utilizado pelo periférico.

A seguir define-se o número de registradores mapeados em memória que o periférico pode acessar. Para o presente projeto são utilizados 4 registradores, que devem ser informados em *Number of software acessible registers*, conforme ilustra Figura 30.

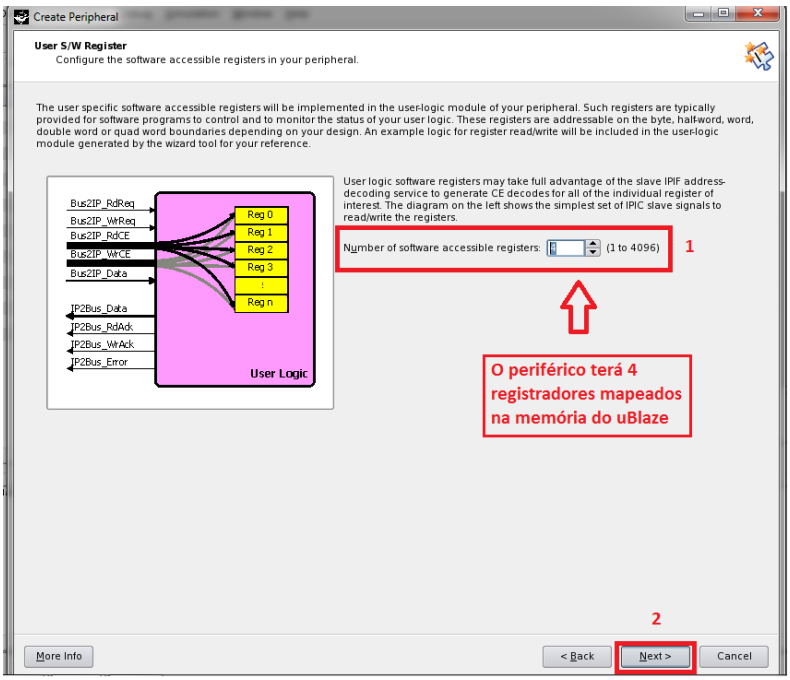


Figura 30 - Quantidade de registradores do periférico mapeados na memória do processador.

Nas próximas telas não são necessárias alterações, bastando clicar no botão *next* e por fim no botão *finish* para ter o periférico criado e retornar à tela principal do XPS. Para que o XPS reconheça adequadamente o novo periférico é necessário recarregar o repositório de periféricos do usuário. Para isto utilizar a opção *Rescan User Repositories* no menu *Project*, conforme ilustra a Figura 31.

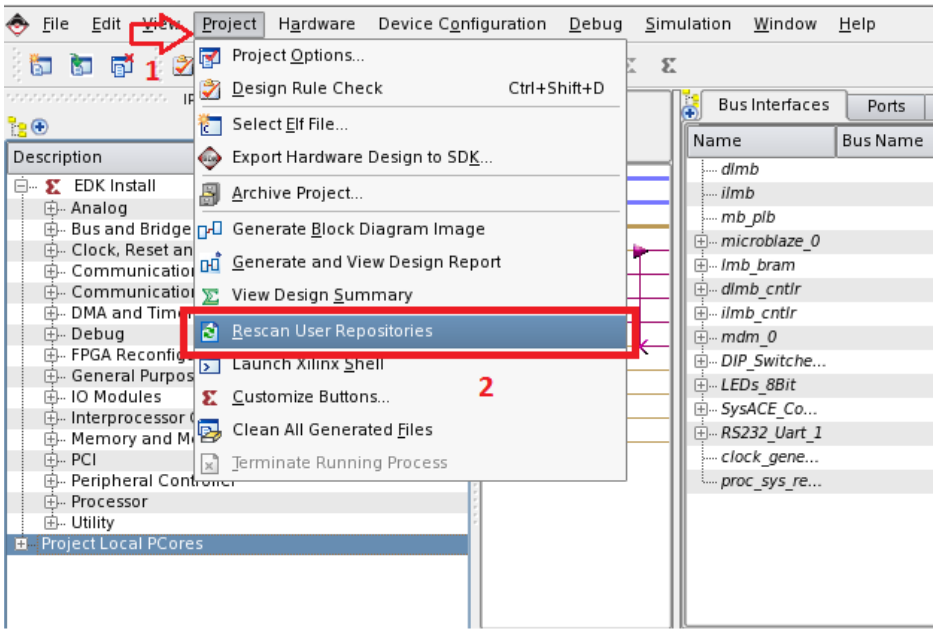


Figura 31 – Atualização do repositório de periféricos do usuário.

Após este passo ter sido executado, o novo periférico passa a estar disponível na janela *IP Catalog*, dentro do grupo *Project Local Pcores*. Para adicionar o novo periférico ao projeto deve-se clicar com o botão direito do *mouse* sobre o nome do periférico e então selecionar a opção *Add IP*, conforme mostrado na Figura 32. No diálogo exibido em seguida basta clicar em OK.

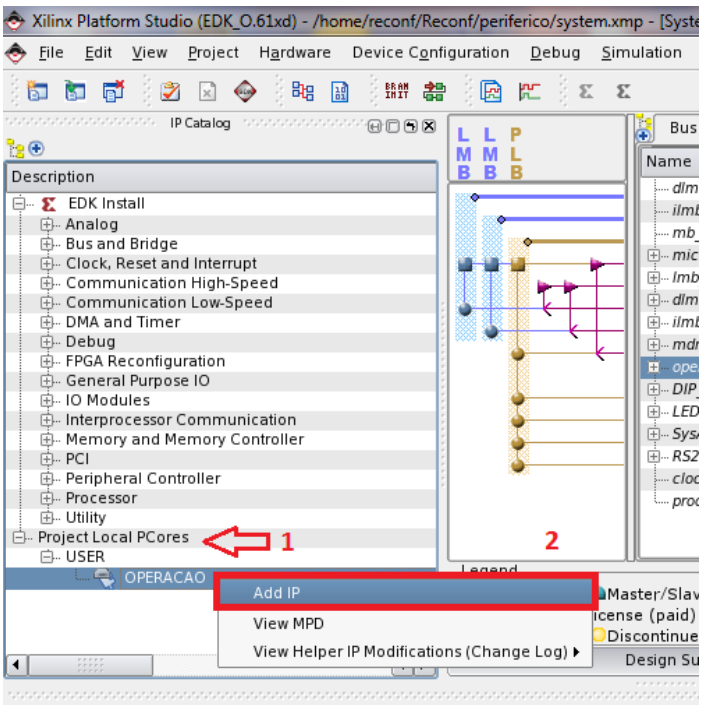


Figura 32 – Inserção do periférico OPERACAO no projeto.

Para que seja possível utilizar o recurso de reconfiguração parcial do FPGA é necessário também adicionar o periférico responsável pela interface com o ICAP do dispositivo. Este periférico é encontrado na janela *IP Catalog*, no grupo *FPGA Reconfiguration* com o nome de *FPGA Internal Configuration Access Port*. Ele deve ser adicionado ao projeto da mesma forma que o periférico criado anteriormente, conforme mostrado em Figura 33.

Neste projeto será utilizada a implementação de periférico para interface com o ICAP fornecido pela própria Xilinx. Em [DUH12] é apresentada uma implementação alternativa.

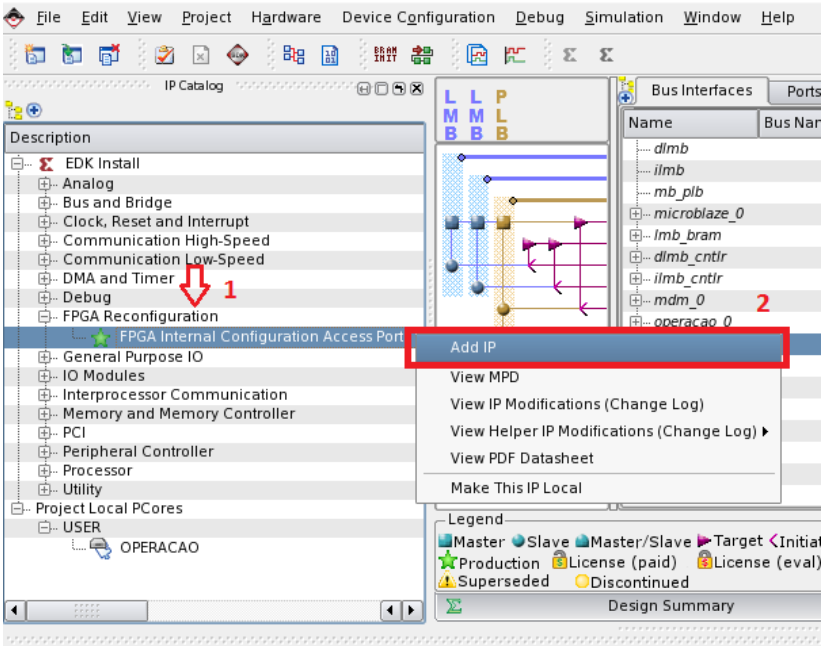


Figura 33 – Inserção do periférico ICAP no projeto.

Além de adicionar os dispositivos periféricos que desejamos utilizar no projeto, também é necessário configurar a forma como estes periféricos serão conectados ao barramento do processador. Na janela principal do XPS, na aba *Bus Interface*, encontram-se todos os periféricos que compõem o sistema que esta sendo criado. Expandindo o grupo correspondente ao periférico criado é possível ver que este possui uma interface chamada SPLB que possui *No Connection*. Esta opção deve ser alterada para "*mb_plb*", de forma que o periférico seja conectado ao barramento do processador. Esta ação é ilustrada na Figura 34.

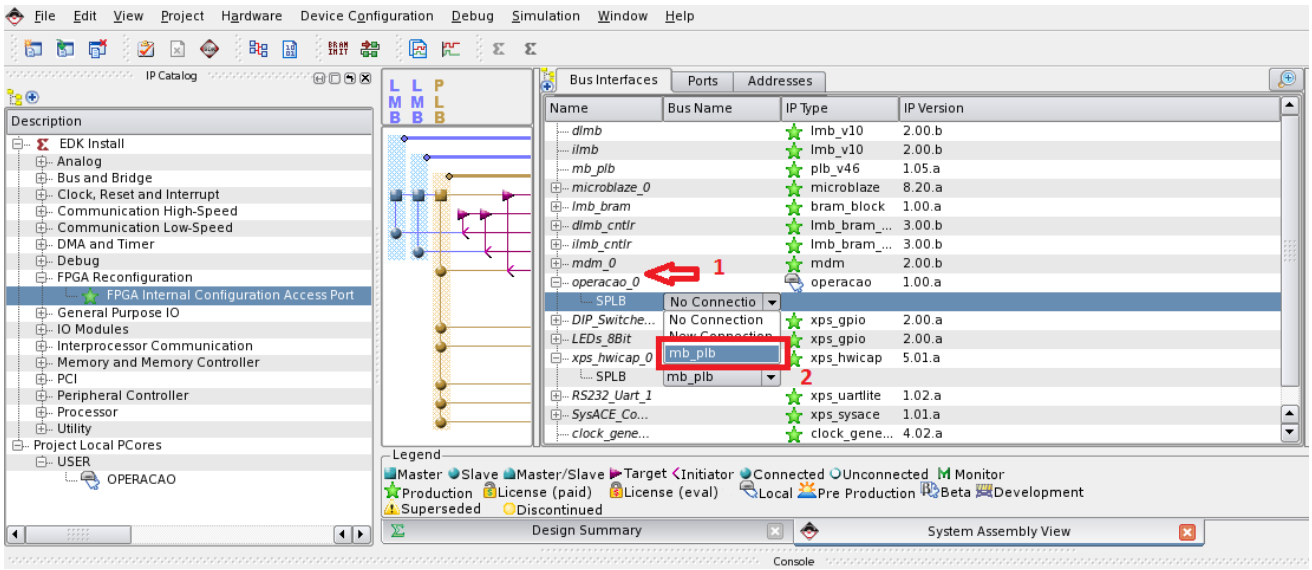


Figura 34 - Conexão do periférico criado ao barramento do processador.

O mesmo procedimento deve ser realizado para conectar o periférico ICAP ao processador, conforme a Figura 35.

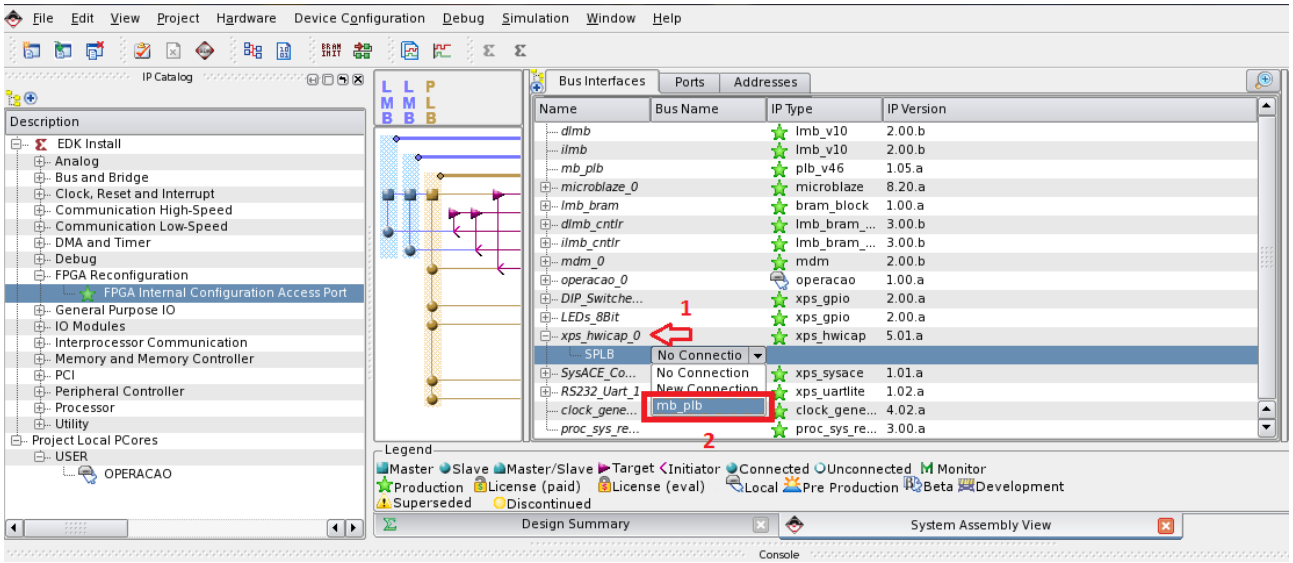


Figura 35 – Conexão do periférico ICAP ao processador.

Quando adicionamos novos periféricos ao sistema, o XPS não gera automaticamente os endereços na memória do processador onde os registradores dos periféricos serão mapeados. Este procedimento deve ser realizado manualmente, clicando-se na aba *Address* e então no botão *Generate Address*, conforme ilustrado na Figura 36.

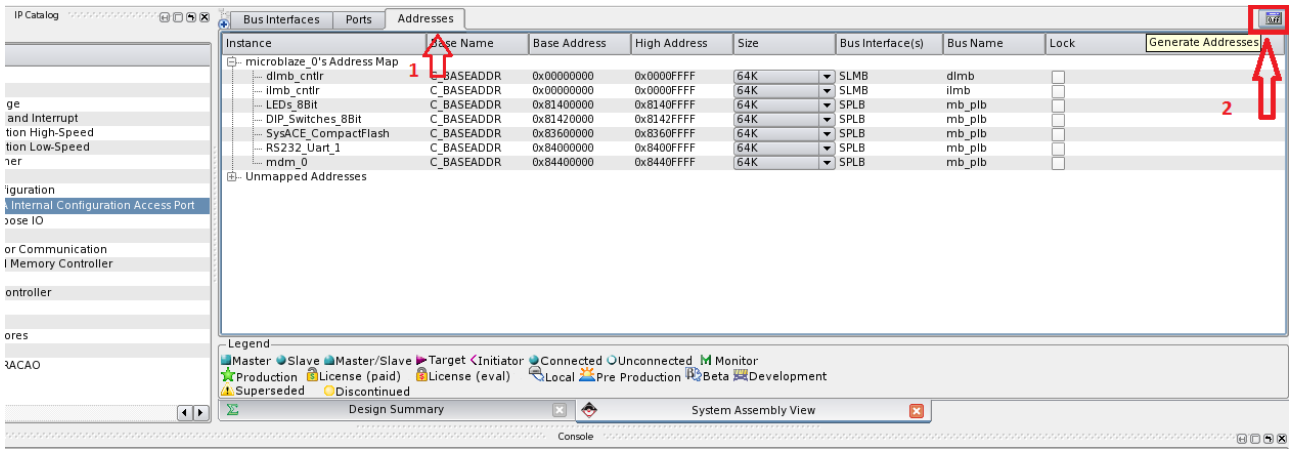


Figura 36 – Geração do endereços dos periféricos mapeados em memória.

Como último passo para a conexão dos periféricos, o periférico ICAP necessita de uma ligação do sinal de relógio em uma porta específica. Esta ligação pode ser feita na aba *Ports*, onde após expandir o grupo correspondente ao ICAP deve-se associar a porta ICAP_Clk ao valor “clk_100_0000Mhz”, conforme ilustrado pela Figura 37.

Neste ponto o projeto já possui todos os periféricos e suas interfaces configuradas. É importante ressaltar que o periférico reconfigurável ainda constitui apenas uma “casca vazia”. Sua conclusão, que inclui a interface com o Módulo Reconfigurável, será descrita na próxima Sessão.

Como último passo é recomendável realizar uma validação das regras de projeto, ou *Design Rules Check* (DRC). Isto é possível através da opção *Design Rule Check* presente no menu *Project*, conforme visto na Figura 38. Algumas mensagens de aviso podem ser geradas. O importante é que o XPS não acuse nenhuma mensagem de erro.

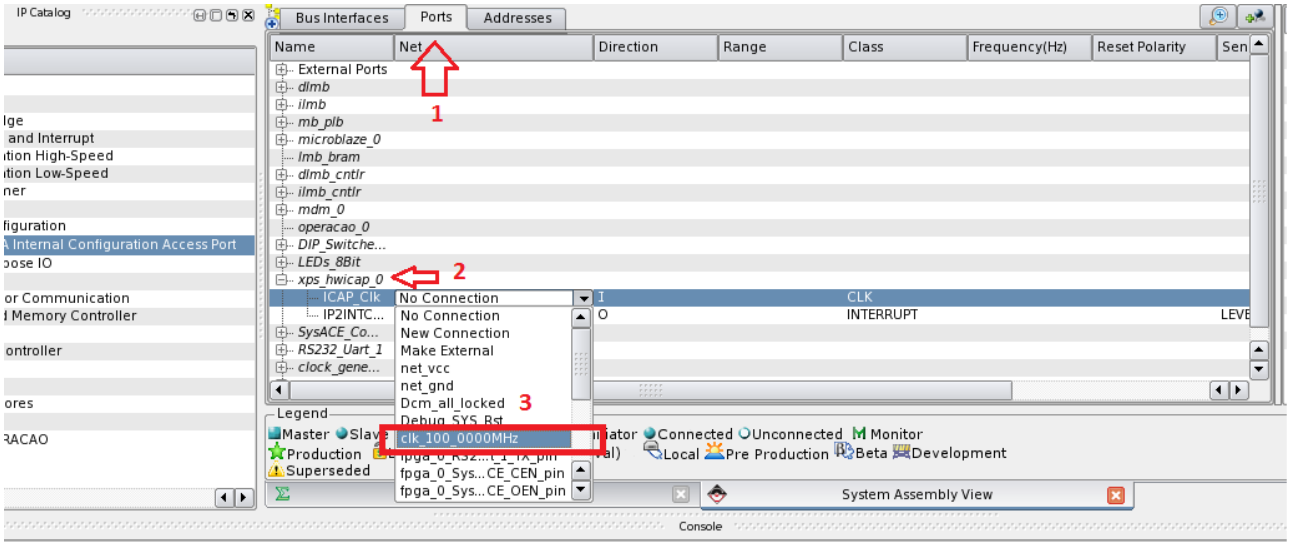


Figura 37 - Ligação do sinal de relógio do periférico do ICAP.

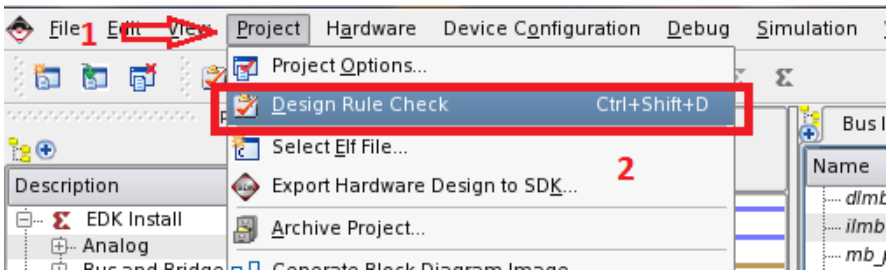


Figura 38 - Validação de regras de projeto.

4.3.5 Criação da interface com Módulos Reconfiguráveis

Uma vez construído o *wrapper* do módulo reconfigurável, é necessário que seja implementada a interface entre o periférico reconfigurável e o módulo reconfigurável em si. Este processo consiste em conectar os sinais vindos do barramento do processador com as portas do módulo reconfigurável. Neste passo apenas arquivos VHDL são editados, não necessitando do uso de uma ferramenta específica.

O arquivo que deve ser editado já foi gerado pelo XPS na sessão anterior e encontra-se em *pcores/operacao_v1_00_a/hdl/vhdl/user_logic.vhd*, dentro do diretório informado como raiz do projeto do XPS. O passo anterior apenas criou o *wrapper*. A inclusão do módulo reconfigurável é feita declarando-se o *component* do mesmo no arquivo *user_logic.vhd*. Esta declaração é apresentada na Listagem 3.

```
architecture IMP of user_logic is
  --USER signal declarations added here, as needed for user logic
   component executor
    port (
          inOperandoA     : in    std_logic_vector(0 to C_SLV_DWIDTH-1);
          inOperandob     : in    std_logic_vector(0 to C_SLV_DWIDTH-1);
          outResultado    : out std_logic_vector(0 to C_SLV_DWIDTH-1);
          inEN              : in    std_logic;
          outDone         : out   std_logic;
          clock             : in    std_logic;
          reset             : in    std_logic
          );
  end component executor;
```

Listagem 3 - Declaração do *component* do módulo reconfigurável.

Após a declaração do *component* instancia-se o módulo reconfigurável dentro da arquitetura do periférico, conectando seus registradores às portas do módulo. Este procedimento é exemplificado na Listagem 4. Os sinais *slv_reg* correspondem aos registradores mapeados em memória, sendo os valores dos mesmos lidos/escritos pelo processador Microblaze.

```
--USER logic implementation added here

executor_inst: executor
   port map (
      inOperandoA     => slv_reg0,
      inOperandoB     => slv_reg1,
      outResultado    => slv_reg2,

      inEN            => slv_reg3(0),
      outDone         => slv_reg3(1),

      clock           => Bus2IP_Clk,
      reset           => Bus2IP_Reset
   );
```

Listagem 4 - Módulo reconfigurável instanciado dentro do periférico.

4.3.6 Geração da Netlist e exportação para o SDK

Uma vez criado o projeto base com o XPS, e concluída a codificação do periférico, é necessário gerar o *netlist* do projeto. Para isso deve ser utilizada a opção *Generate Netlist*, que pode ser encontrada no menu *Hardware*, conforme mostrado na Figura 39.

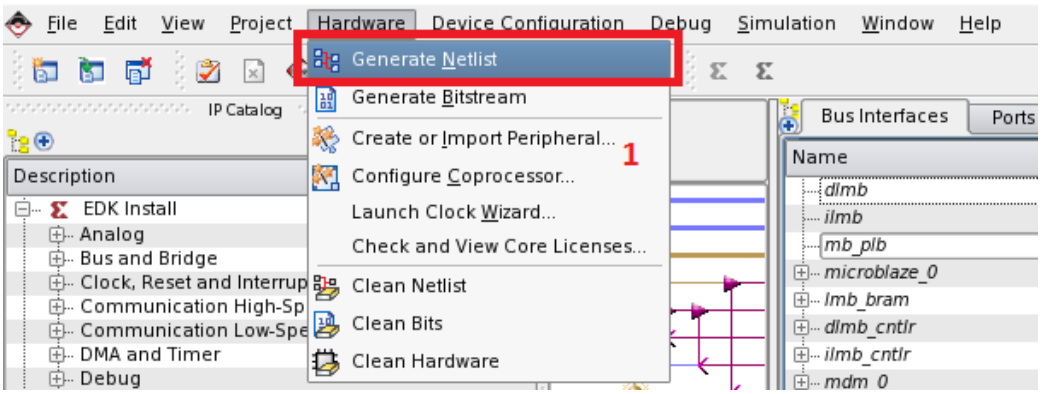


Figura 39 - Criação do netlist do projeto do XPS.

Concluído este processo, o próximo passo no fluxo consiste em exportar o projeto criado no XPS para o SDK, onde o *software* embarcado no sistema poderá ser desenvolvido. Para isto deve ser utilizada a opção *Export Hardware Design to SDK*, que pode ser encontrada no menu *Project*, conforme ilustrado pela Figura 40 .

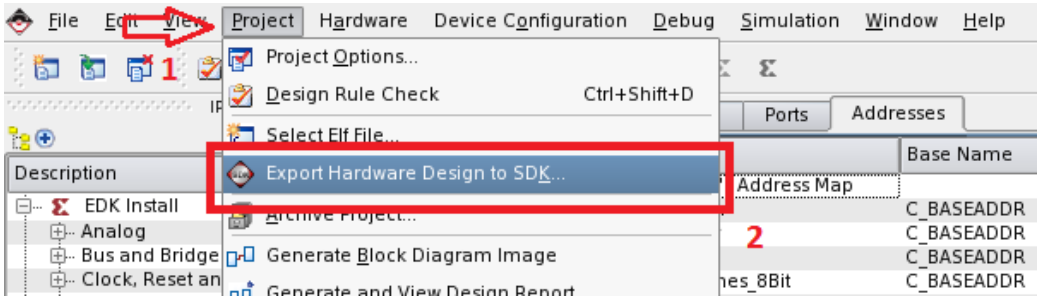


Figura 40 - Exportando o projeto do XPS para o SDK.

Na tela seguinte deve-se desmarcar a opção *Include bitstream and BMM file*, pois o *bitstream* será gerado mais adiante no *Plan Ahead*. Deve-se então clicar em *Export & Launch SDK*, conforme ilustrado na Figura 41. Feito isto, a ferramenta SDK deve é iniciada automaticamente.

Figura 41 – Opções para exportar o projeto do XPS para o SDK.

A SDK ao iniciar requer que se indique qual o diretório desejado para servir como espaço de trabalho (*workspace*), isto é, o diretório onde os projetos serão salvos. Deve-se definir um diretório e clicar em OK, conforme exemplificado na Figura 42.

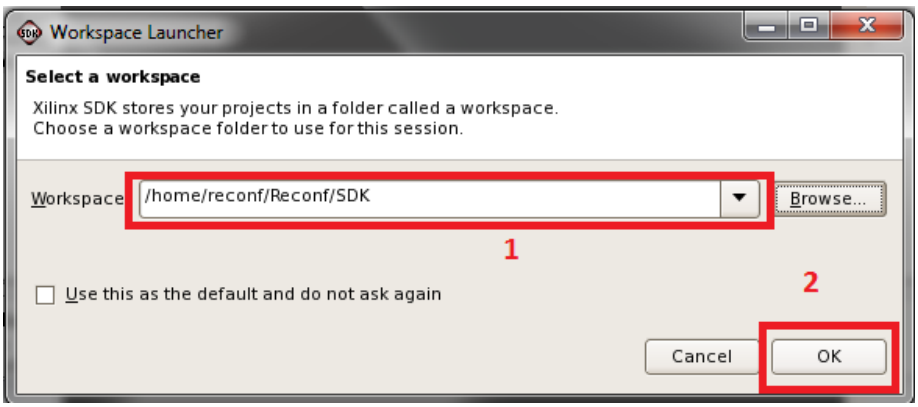


Figura 42 – Definição do diretório de trabalho no SDK.

Após isto o SDK apresentará sua janela principal exibindo um resumo do projeto criado no XPS, com todos os periféricos e seus respectivos endereços mapeados na memória do processador. Esta tela pode ser observada na Figura 43.

4.3.7 Criação do ambiente de software com bibliotecas de apoio

Para que seja possível desenvolver o software necessário no SDK antes é necessário criar um ambiente de trabalho, importando as bibliotecas adequadas para a programação do sistema criado através do XPS. Isto envolve a criação de um projeto de suporte dentro do SDK, conforme descrito a seguir.

Inicialmente, deve-se criar um projeto do tipo *Xilinx Board Support Package*, podendo-se realizar esta etapa através dos menus *File* e *New,* conforme Figura 44.

Na tela seguinte deve-se selecionar *standalone* em *Board Support Package OS* e clicar em *Finish*, conforme a Figura 45.

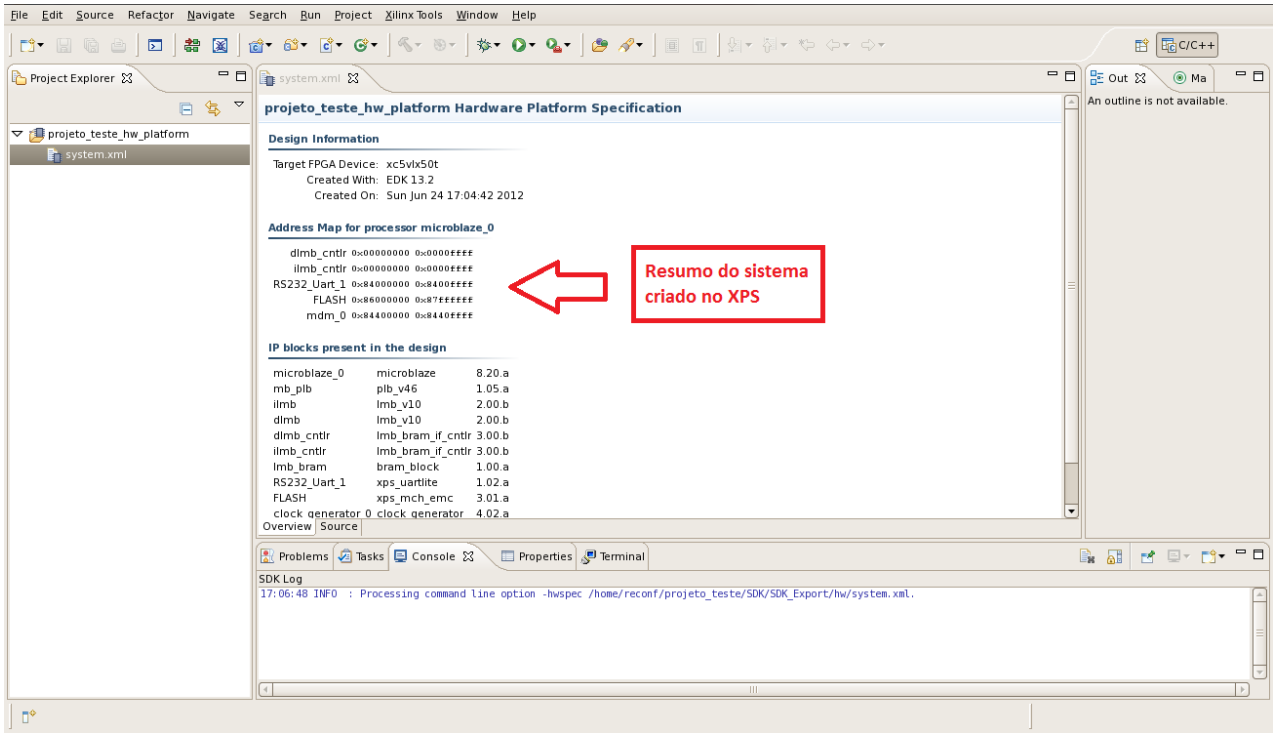


Figura 43 - Resumo do projeto exportado para o SDK.

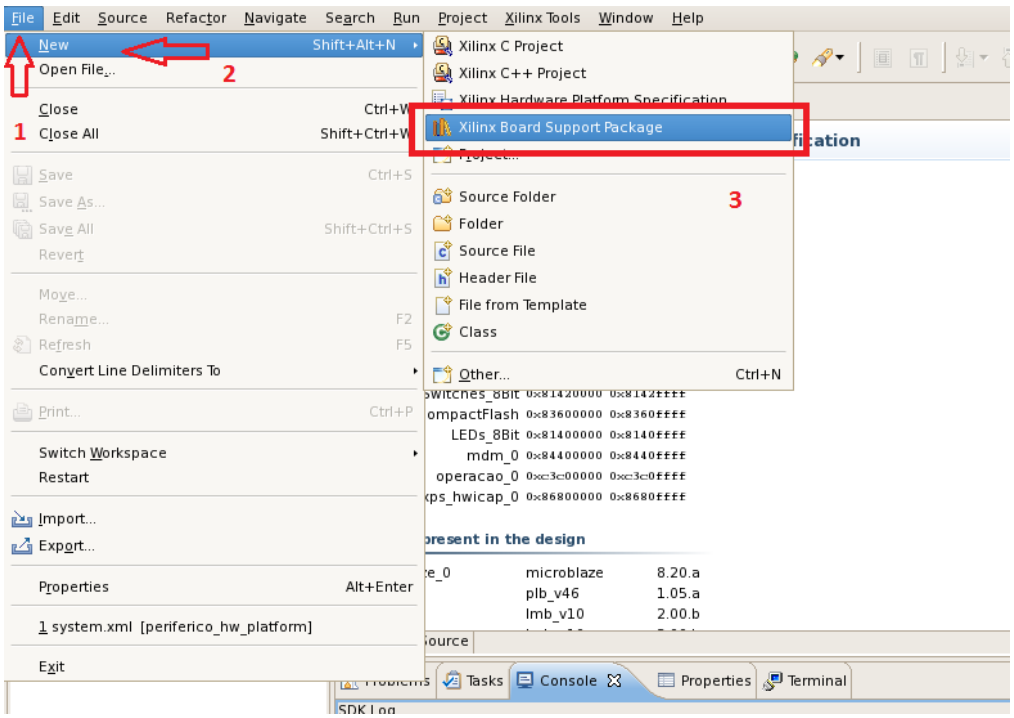


Figura 44 - Criação de um projeto de suporte no SDK.

Na próxima tela deve-se marcar a opção "*xilflash*" em *Supported Libraries*, conforme a Figura 46. Esta ação inclui no projeto um biblioteca que implementa o acesso ao sistema de arquivos da *Linear Flash* presente na placa.

Após estes passos o ambiente de desenvolvimento está pronto para receber a codificação do *software* embarcado que executará no sistema.

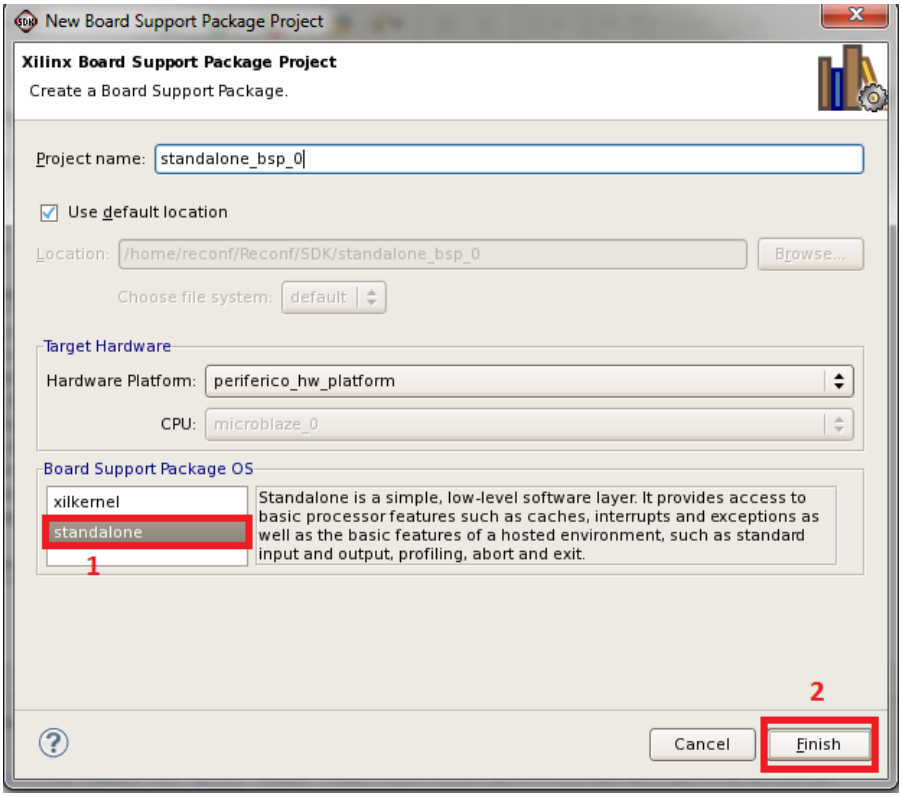


Figura 45 - Configuração do projeto de suporte no SDK.

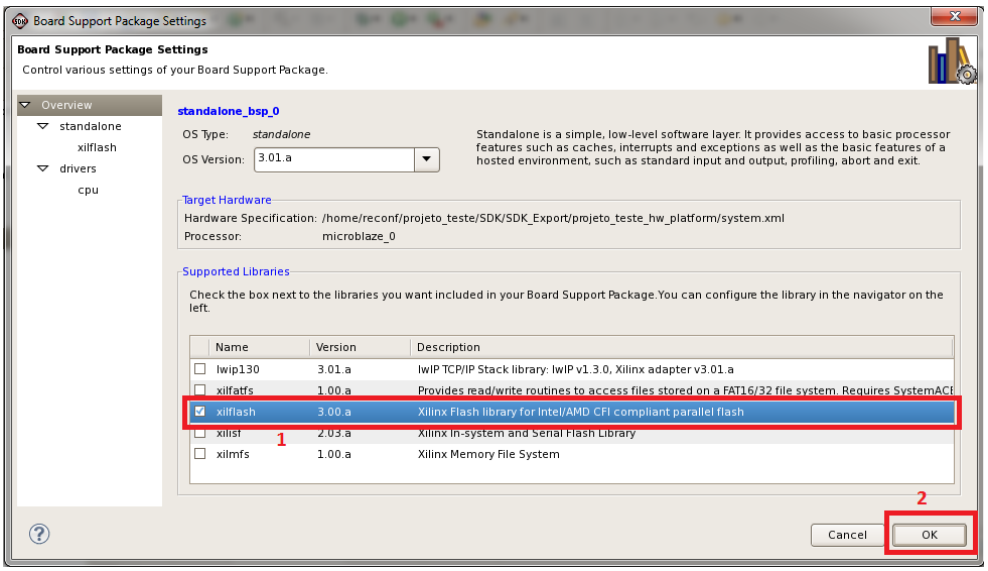


Figura 46 - Selecionar bibliotecas adicionais ao projeto de suporte.

4.3.8 Desenvolvimento do Software

Para iniciar o desenvolvimento do *software* deve-se criar um novo projeto no SDK. Como a aplicação que será desenvolvida utiliza a linguagem C, este deve ser o tipo de projeto a ser criado. Para isto deve-se selecionar a opção *Xilinx C Project*, nos menus *File* e *New*. Este procedimento é ilustrado na Figura 47.

Em seguida deve-se informar o nome do projeto e, se desejado, selecionar um *template*, ou seja, um modelo, para o mesmo. No caso deste projeto nenhum *template* será utilizado, por isso deve-se selecionar a opção *Empty Application*, conforme Figura 48.

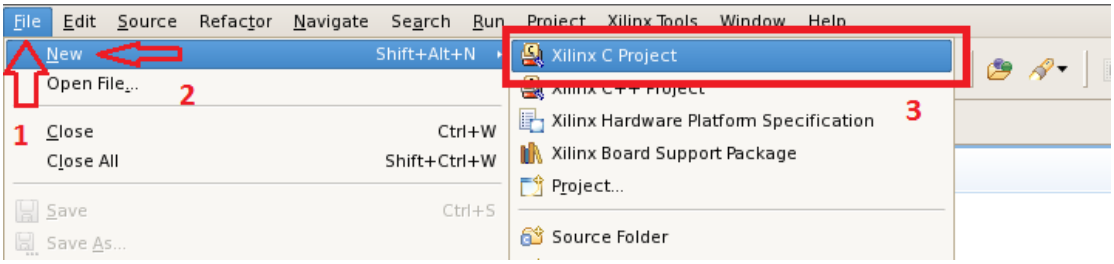


Figura 47 - Criação de um projeto de software utilizando a linguagem C.

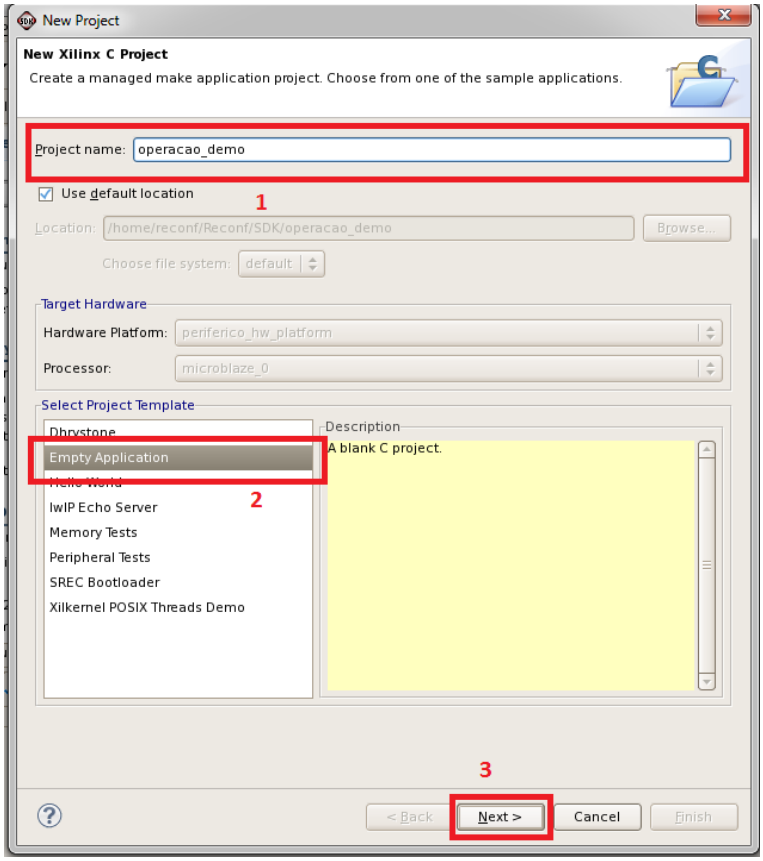


Figura 48 - Seleção do nome do projeto e do template utilizado.

Na tela seguinte é necessário selecionar para qual sistema o projeto de *software* será desenvolvido. Isto é feito indicando o projeto de suporte que foi criado anteriormente. Para isto deve-se selecionar *Target an existing Board Support Package* e o nome do projeto de suporte criado, conforme Figura 49.

Neste ponto o SDK está pronto para ser utilizado para codificar o *software* que será embarcado no processador MicroBlaze da aplicação. Para adicionar um novo arquivo de código fonte deve-se clicar com o botão direito sobre o nome do projeto C recém criado e selecionar as opções *New* e *Source File*, conforme ilustrado na Figura 50.

4.3.9 Geração do binário da aplicação desenvolvida

A ferramenta SDK realiza a compilação do projeto automaticamente conforme este vai sendo editado. Para forçar a compilação podemos utilizar o comando *Clean Project*, clicando-se com o botão direito do *mouse* sobre o nome do projeto, conforme a Figura 51. O arquivo gerado pelo processo de compilação possui a extensão .elf e representa um binário executável.

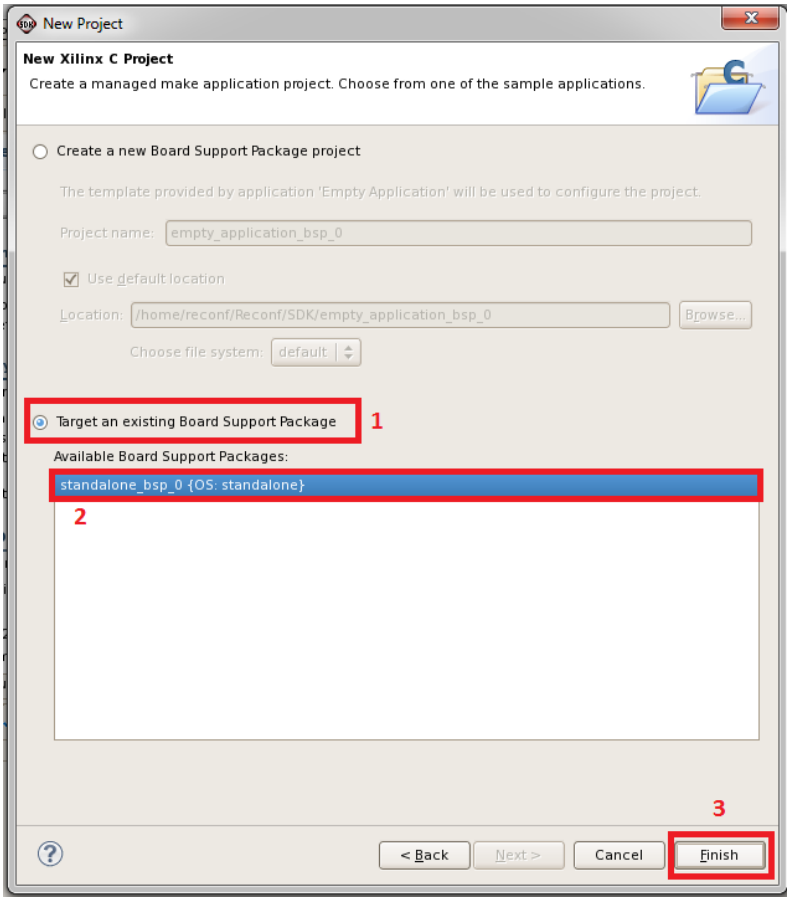


Figura 49 - Seleção do sistema para o qual o software será desenvolvido.

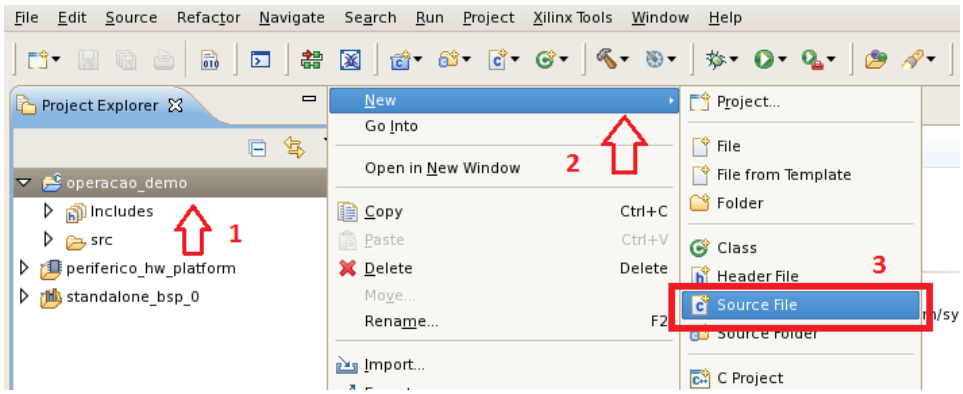


Figura 50 - Inserir arquivo de código fonte do projeto.

Uma vez que o projeto de *software* esta concluído, deve-se gera um script responsável por realizar o processo de *linking*, onde o binário compilado é combinado com suas bibliotecas. Para isto utiliza-se a opção *Generate Linker Script* localizada no menu *Xilinx Tools*, conforme Figura 52. Na tela seguinte basta confirmar as opções clicando no botão *Generate*, conforme a Figura 53.

Este processo conclui a etapa de desenvolvimento do *software* embarcado.

4.3.10 Importação do Projeto para o PlanAhead

Neste estágio do fluxo de projeto já foram criados o *nestlist* do sistema (no ambiente XPS) e o executável do *software* embarcado (no ambiente SDK). O próximo passo consiste em realizar a configuração da partição reconfigurável, *floorplaning* e a síntese física do projeto. Para isto é utilizada a ferramenta *PlanAhead*.

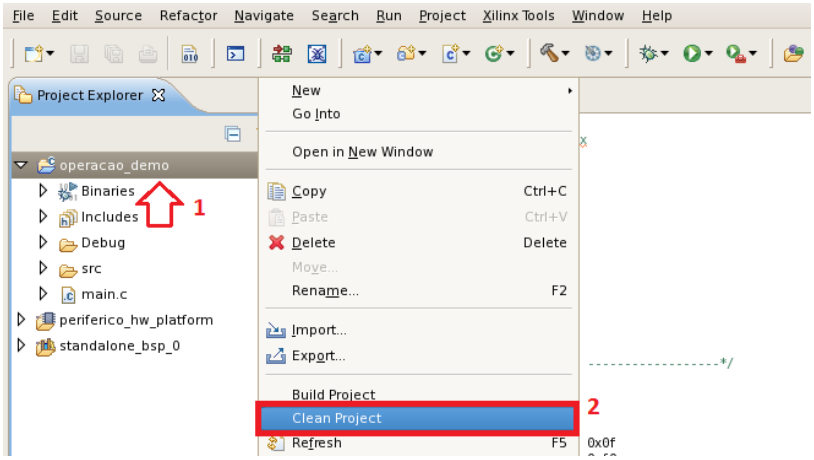


Figura 51 - Forçar a compilação do projeto.

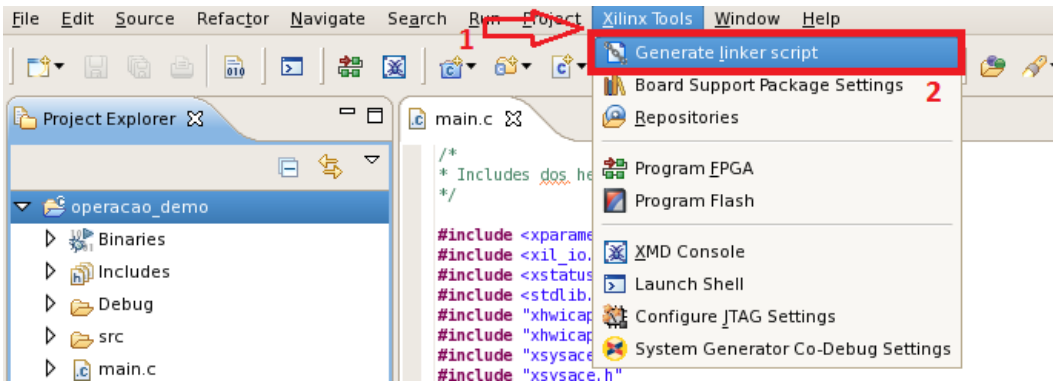


Figura 52 - Criação do script de *linking*.

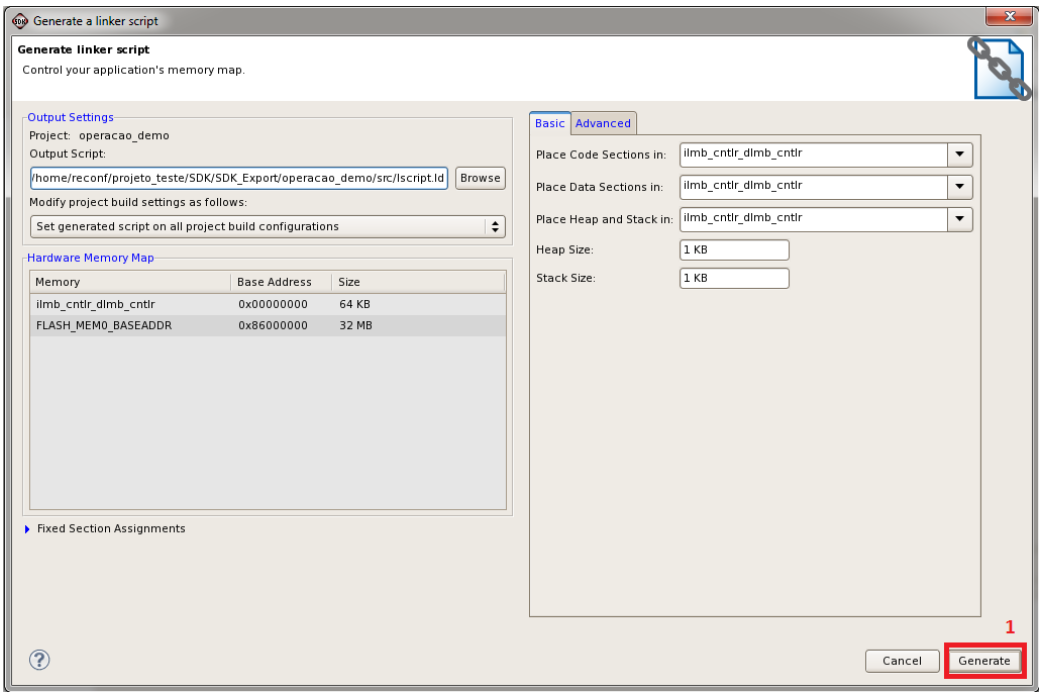


Figura 53 - Opções para criação do script de *linking*.

Após abrir a ferramenta *PlanAhead*, deve-se selecionar a opção *Create New Project*, que dará inicio ao procedimento para criação de um novo projeto. Na tela inicial deve-se clicar no botão *Next* que então levará à tela apresentada na Figura 54 onde deve-se configurar o nome do projeto e o diretório onde este deve ser salvo.

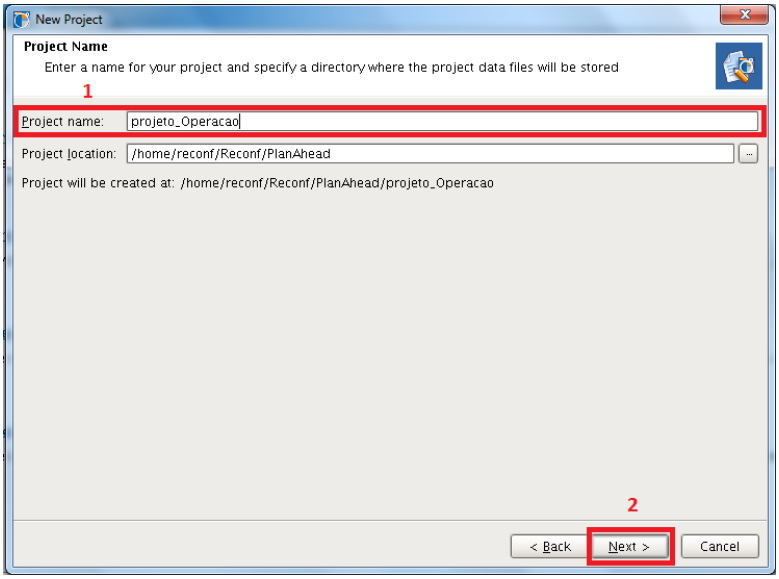


Figura 54 - Criação de um novo projeto no PlanAhead.

Na tela seguinte deve-se selecionar a opção *Specify systhesized (EDIF or NGC) netlist*, e então selecionar a opção *Set PR Project* de forma a permitir o uso de reconfiguração parcial, conforme Figura 55 . Esta opção pode não estar disponível para todas as formas de licenças da ferramenta.

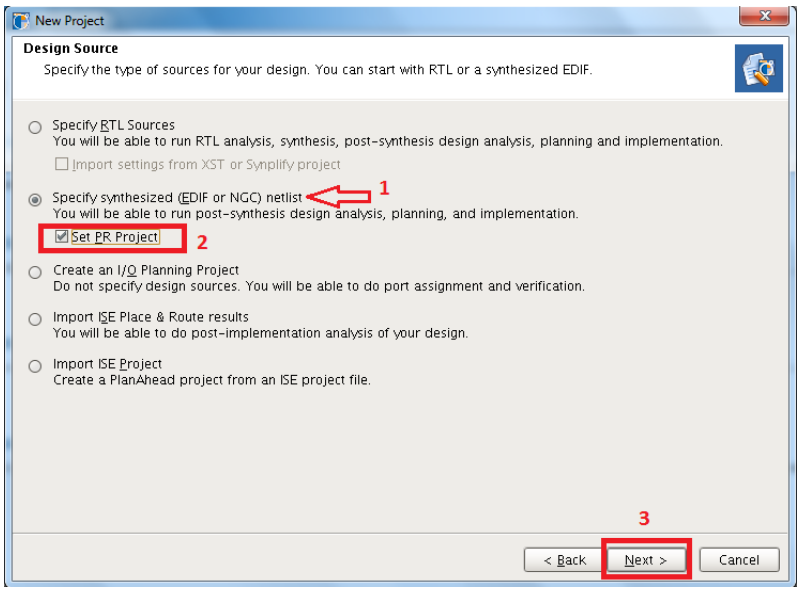


Figura 55 - Seleção do tipo de projeto no PlanAhead.

Na próxima etapa, Figura 56, deve-se selecionar como *Top Netlist File* a *netlist* criada na etapa de projeto com o XPS, que consiste em um arquivo NGC, que por padrão se chama *system.ngc* e encontra-se do diretório *implementation*, no diretório raiz do projeto do XPS.

Na próxima tela existe a possibilidade de inclusão de novos arquivos com definições de *constraints*. É necessário adicionar o arquivo de *constraints* do projeto criado pelo XPS, presente na pasta Data, dentro da pasta raíz do projeto do XPS. Conforme Figura 57.

Na tela seguinte é necessário selecionar o FPGA para qual o projeto deve ser sintetizado. No caso do projeto desenvolvido este é o xc5vlx50tff1136-1, conforme pode ser observado na Figura 58.

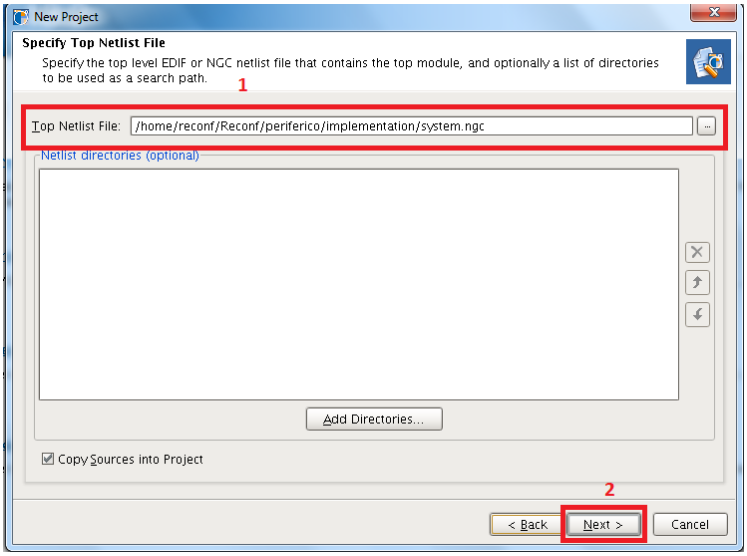


Figura 56 - Importação da netlist criada no XPS para o PlanAhead.

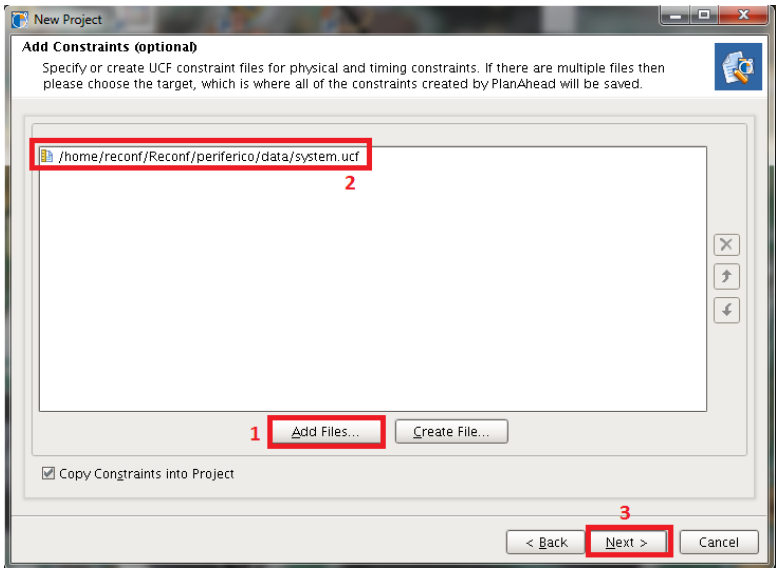


Figura 57 - Inserir constraints (restrições) do projeto.

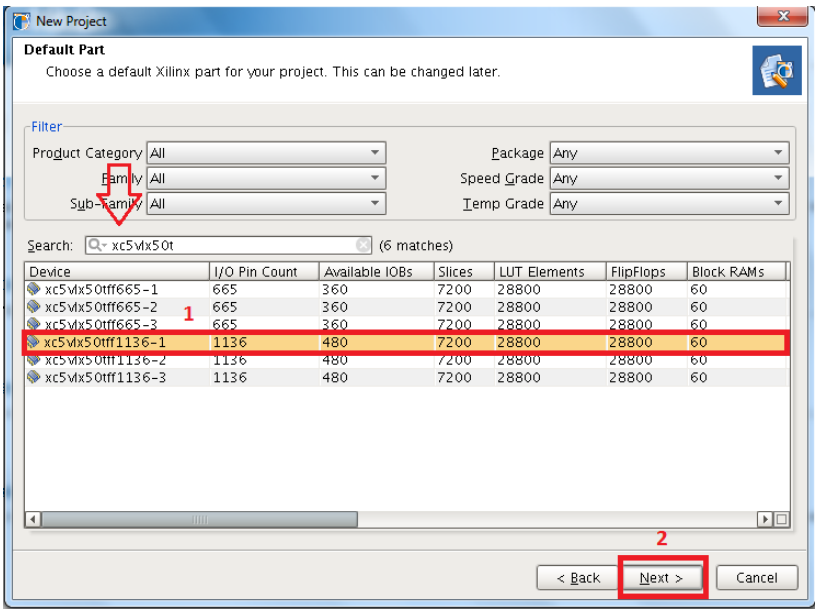


Figura 58 - Selecionar o modelo de FPGA para qual o projeto deve ser sintetizado.

A próxima tela apresenta um resumo das opções selecionadas, bastando clicar em *Finish* para retornar à tela principal do Plan Ahead. Uma vez que o projeto tenha sido criado, o próximo passo é gerar a sua *netlist*. Para isto deve-se selecionar a opção *Netlist Design*, que encontra-se no menu a esquerda, conforme visto na Figura 59.

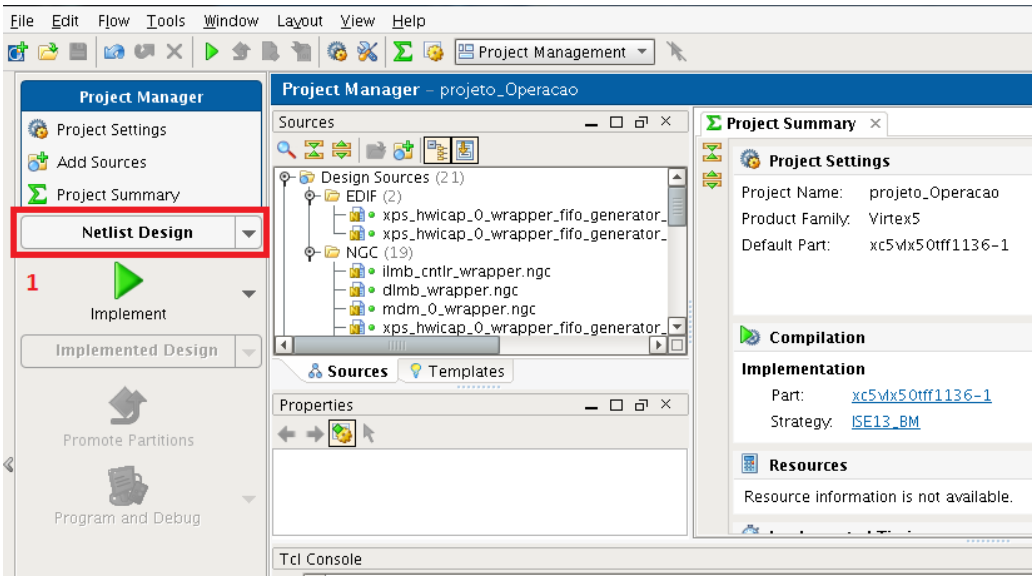


Figura 59 - Geração da netlist do projeto no PlanAhead.

4.3.11 Definição das Partições e dos Módulos Reconfiguráveis

O próximo passo a ser executado no PlanAhead é configurar quais módulos do sistema serão partições reconfiguráveis. No caso do projeto desenvolvido existe apenas uma, que corresponde ao módulo que está dentro do periférico criado no XPS. Esta partição pode ser encontrada na janela *Netlist*, expandindo o grupo *operacao_0* e clicando-se com o botão direito do *mouse* sobre operacao_0/USER_LOGIC_1/executor_inst. No menu que será exibido deve-se selecionar a opção *Set Partition* , conforme ilustra a Figura 60. Será exibida uma tela de apresentação, onde basta clicar em *Next.*

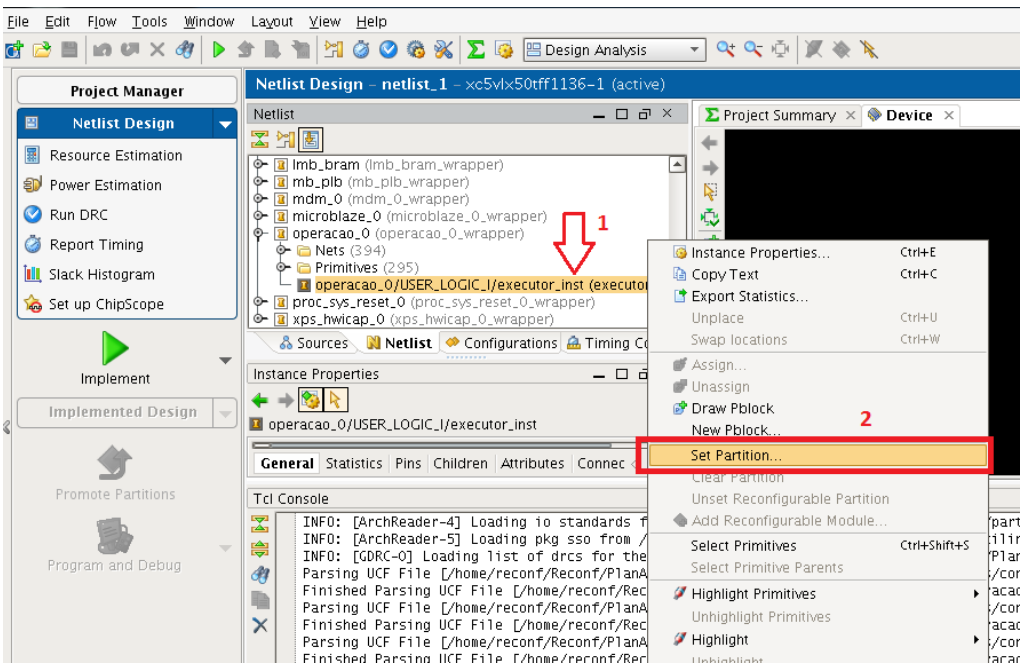


Figura 60 - Criação de uma partição.

Na tela seguinte é necessário informar se a partição deve ser considerada como uma partição reconfigurável. Para tal deve-se selecionar a opção *is a reconfigurable partition*, conforme a Figura 61.

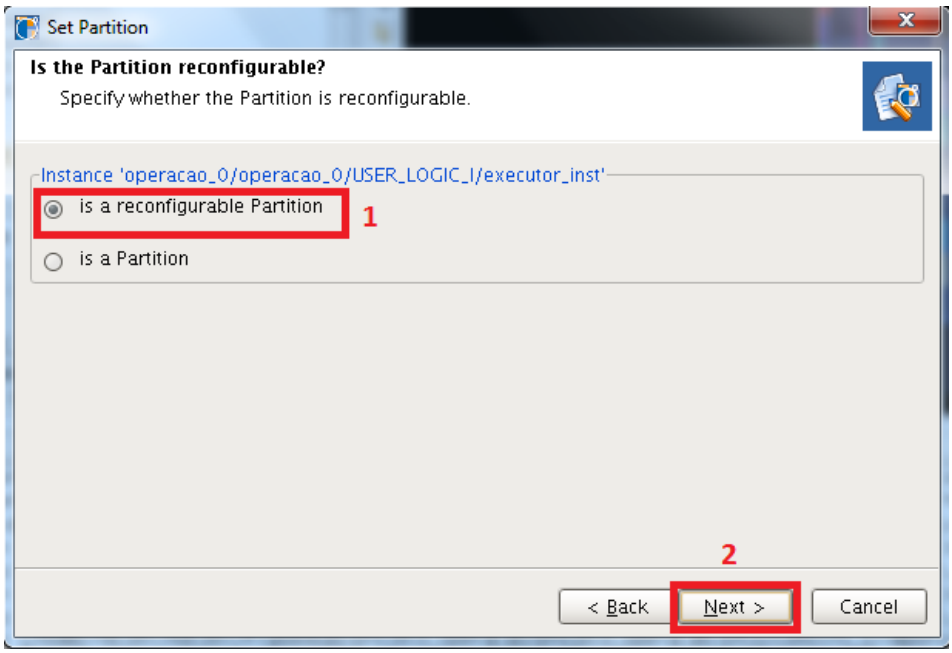


Figura 61 - Indicação que a partição deve ser tratada como uma partição reconfigurável

A seguir deve-se informar um nome para o primeiro módulo reconfigurável. No caso do projeto desenvolvido o primeiro módulo será o somador, logo se chamará *module_somador*. Também deve-se marcar a opção *Netlist already avaliable for this reconfigurable module*, indicando que a *netlist* do módulo já foi criada anteriormente. Observe a Figura 62 .

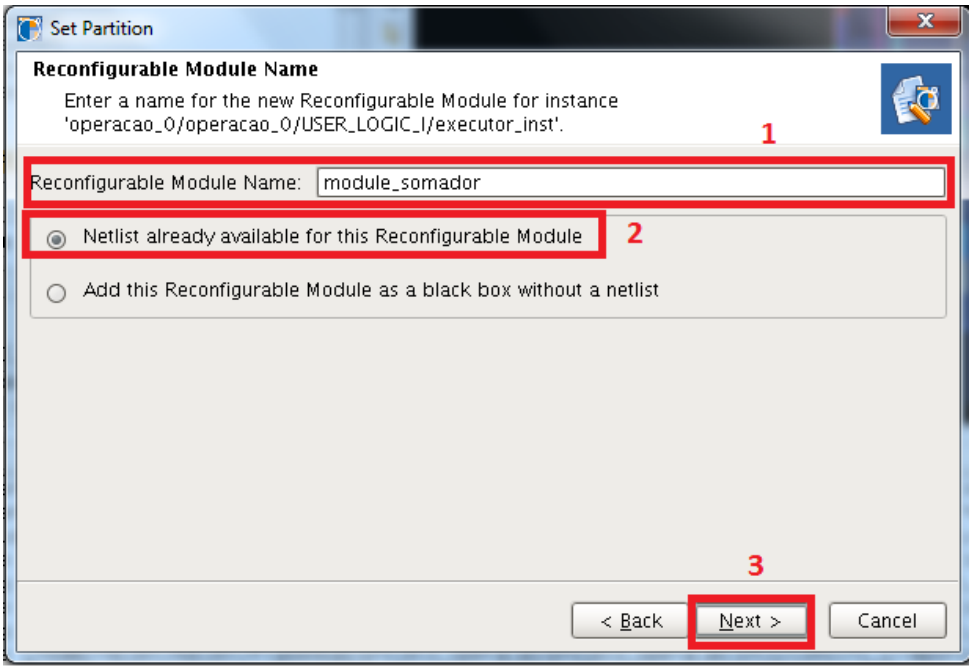


Figura 62 – Seleção do primeiro módulo reconfigurável.

A seguir é necessário informar o caminho para o arquivo NGC do módulo que está sendo criado, conforme Figura 63. Este arquivo foi gerado pelo XST na sessão 4.3.2. O campo *Netlist Directories* não necessita ser alterado.

A próxima tela apresenta a possibilidade de adicionar um arquivo de restrições (*constraints*) associado ao módulo. Este recurso não foi utilizado no projeto, bastando clicar em *Next* e após em *Finish* para retornar à tela principal do PlanAhead.

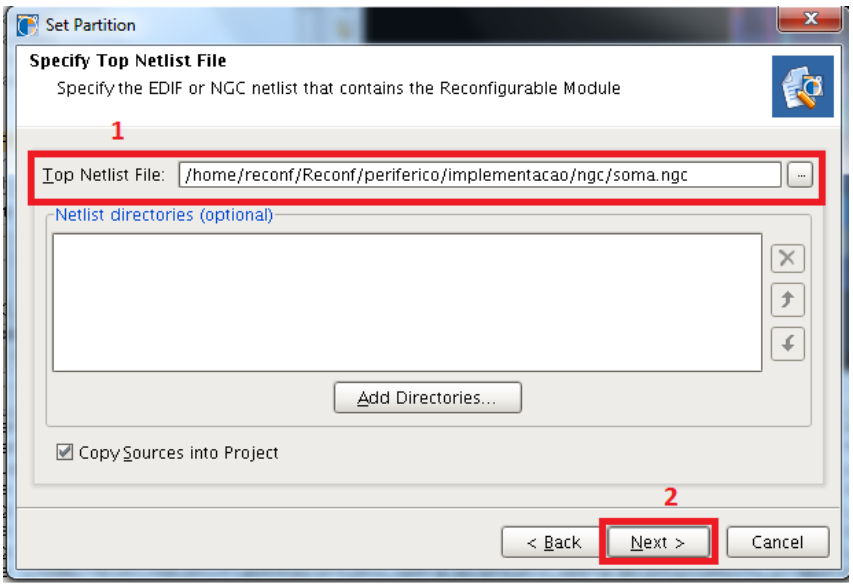


Figura 63 – Adição da netlist do módulo reconfigurável.

O mesmo procedimento deve ser repetido mais duas vezes, adicionando os módulos de subtração e multiplicação com suas respectivas *netlists*. Neste projeto estes módulos foram chamados de *module_subtracao* e *module_multiplicacao*. Para este projeto nenhum módulo reconfigurável será incialmente incluído no *bitstream* gravado no FPGA. Para isto é preciso também adicionar um módulo "Caixa Preta", ou *Black Box*, que não contém nenhuma implementação. Para adicionar uma *Black Box* deve-se clicar com o botão direito do *mouse* sobre a partição na janela *Netlist* e selecionar a opção *Add Reconfigurable Module* como normalmente feito para adicionar um novo módulo. Este módulo será chamado de *Module*_BB, também é necessário selecionar a opção *Add this Reconfigurable Module as a Black Box Without a netlist*, conforme a Figura 64.

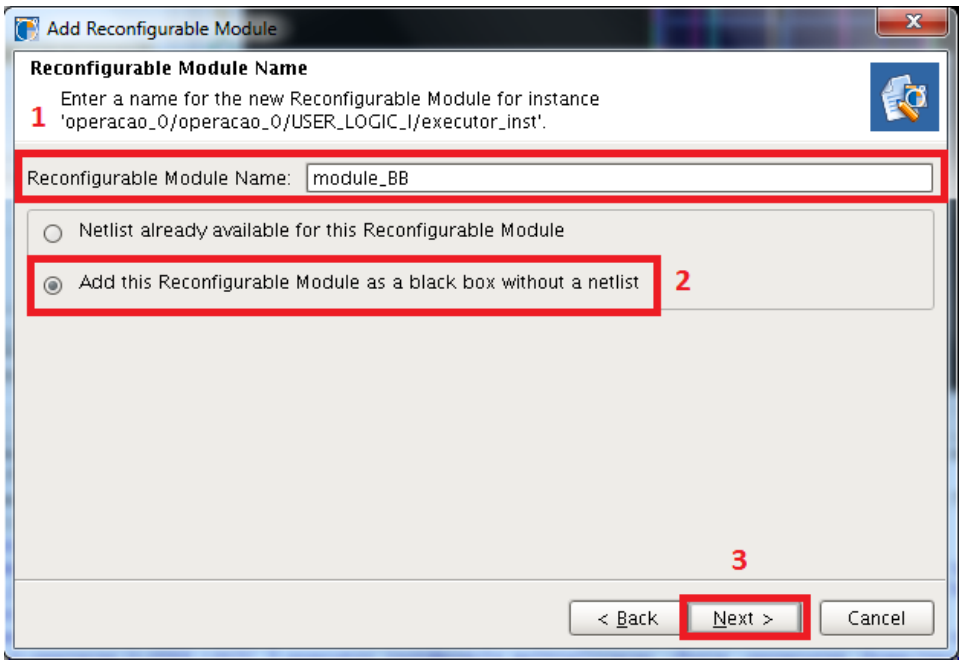


Figura 64 – Inserção de um módulo Black Box.

Tendo sido criado o módulo *Black Box* este deve ser selecionado como o módulo reconfigurável ativo da partição, isto é, o módulo que será incluído no *bitstream* completo do sistema, e que será configurado na inicialização do FPGA . Para isto deve-se clicar com o botão direito do *mouse* sobre *module_BB* e então selecionando a opção *Set as Active Reconfigurable Module*. A Figura 65 ilustra este procedimento.

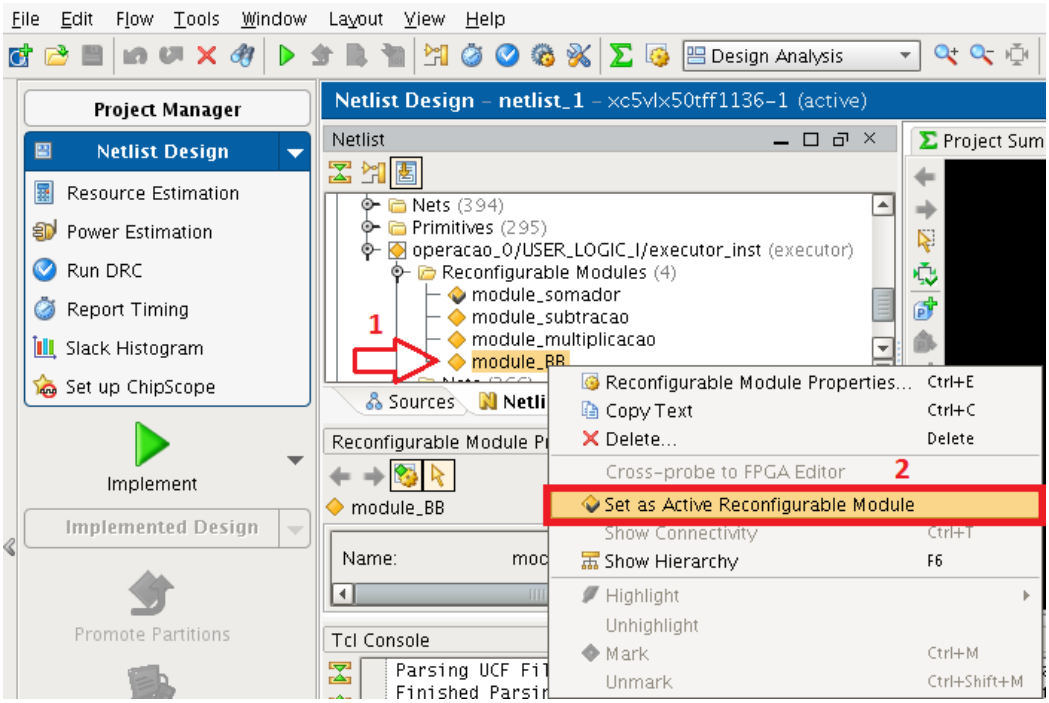


Figura 65 - Definição da Black Box como módulo ativo.

Outro procedimento necessário para a configuração da partição reconfigurável é a definição de suas restrições de área. Isto deve ser feito manualmente. Para isto deve-se primeiramente selecionar a opção *Physical Constraints*, presente no menu *Window* para exibir a janela de restrições físicas, conforme a Figura 66.

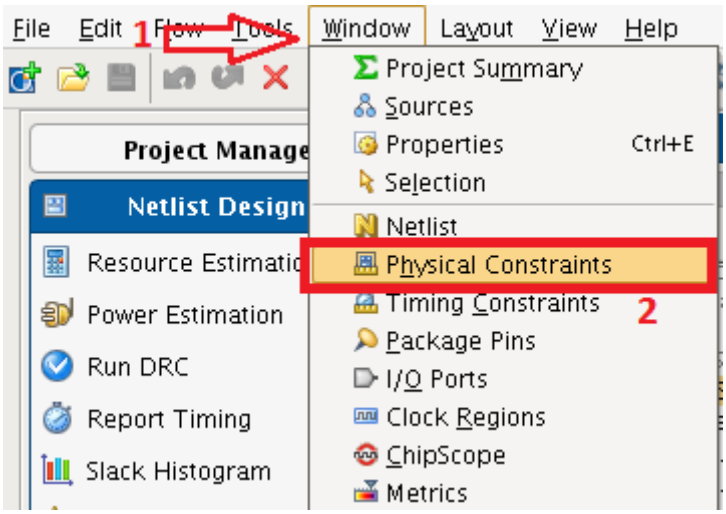


Figura 66 - Janela de restrições físicas.

Em seguida, na janela *Physical Constraints*, deve-se expandir o grupo *ROOT* e clicar como o botão direito do *mouse* sobre *pblock_operacao_0*. No menu que será exibido deve-se selecionar a opção *Set Pblock size*, conforme ilustra a Figura 67.

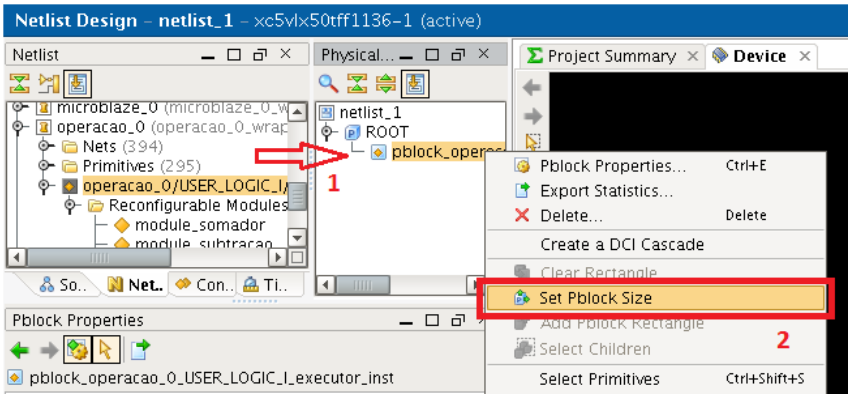


Figura 67 - Definição da área destinada à partição reconfigurável.

Quando selecionada esta opção o *PlanAhead* habilitará o desenho de um retângulo sobre o mapa do FPGA. Deve-se desenhar a área que deve ser destinada à partição, como ilustrado na Figura 68. Quando o retângulo for definido, será exibida uma janela indicando os recursos de *hardware* disponíveis na área marcada. É importante que estes recursos sejam suficientes para a implementação de todos os Módulos Reconfiguráveis associados à partição, caso contrário um erro será gerado no passo 4.3.12. Para verificar se a área selecionada é adequada para todas as partições, pode-se executar a verificação de regras de projeto. Para isto deve ser utilizada a opção *Run DRC* localizada no menu *Tools*. No diálogo que será exibido é conveniente deixar marcada apenas a opção *Partial Reconfiguration*, conforme exibido na Figura 69, assim será verificado se os Módulos Reconfiguráveis e a área definida para a Partição Reconfigurável são compatíveis em recursos.

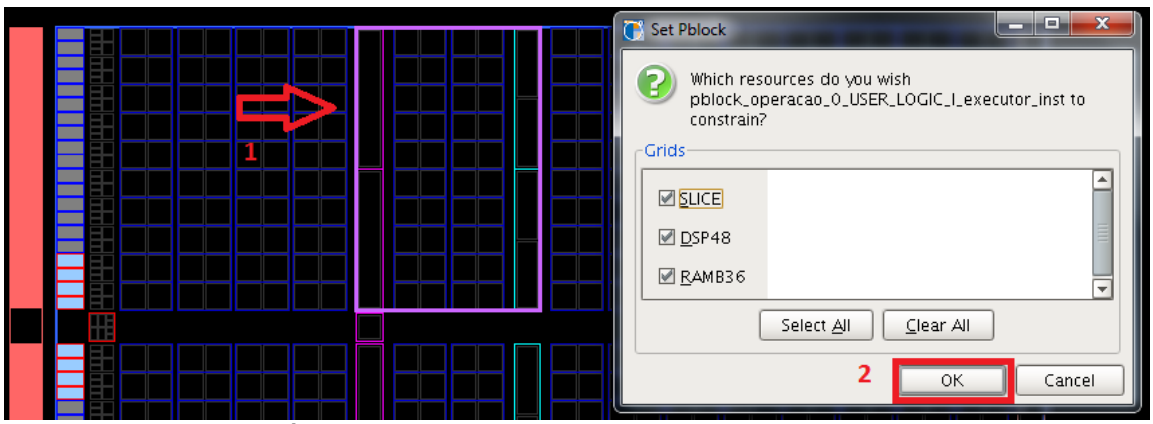


Figura 68 - Retângulo definindo a área reservada à partição reconfigurável.

Caso a ferramenta não exiba nenhuma mensagem de erro, é possível avançar para o próximo passo.

4.3.12 Implementação e Promoção das Partições

Para realizar a síntese física do projeto é necessário executar o processo de implementação do *hardware* estático e dos módulos reconfiguráveis. Antes disto é preciso criar uma estratégia de implementação adequada ao projeto.

4.3.12.1 Definição de Estratégias de Síntese

Como no fluxo adotado o *netlist* já foi gerado pela ferramenta XPS, que definiu parâmetros específicos para a memória do processador MicroBlaze, se faz necessário existir um cuidado especial durante o processo de síntese física. Para que as definições de área de memória que serão mapeadas em BRAMs sejam respeitadas.

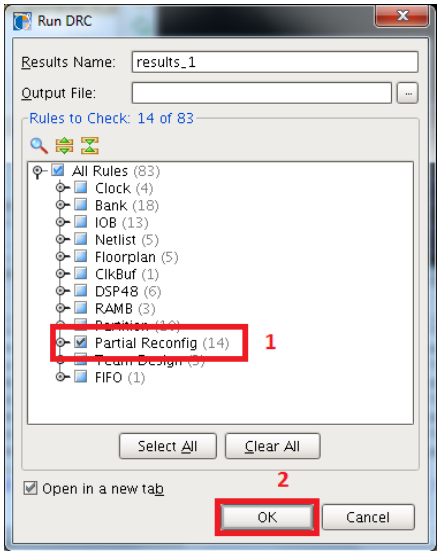


Figura 69 – Execução do DRC para verificação dos recursos alocados à partição reconfigurável.

Para criar um estratégia de implementação deve-se selecionar a opção *Options* no menu *Tools*. Na tela que será apresentada deve-se selecionar a opção *Strategies* (Figura 70). Na opção *Flow* (‘2’ no destaque da Figura 70) deve ser alterada de XST 13 para ISE 3. A nova estratégia é criada clicando-se no botão com um sinal de "mais" (Figura 71), sendo esta nova estratégia denominda de "ISE13_Reconf".

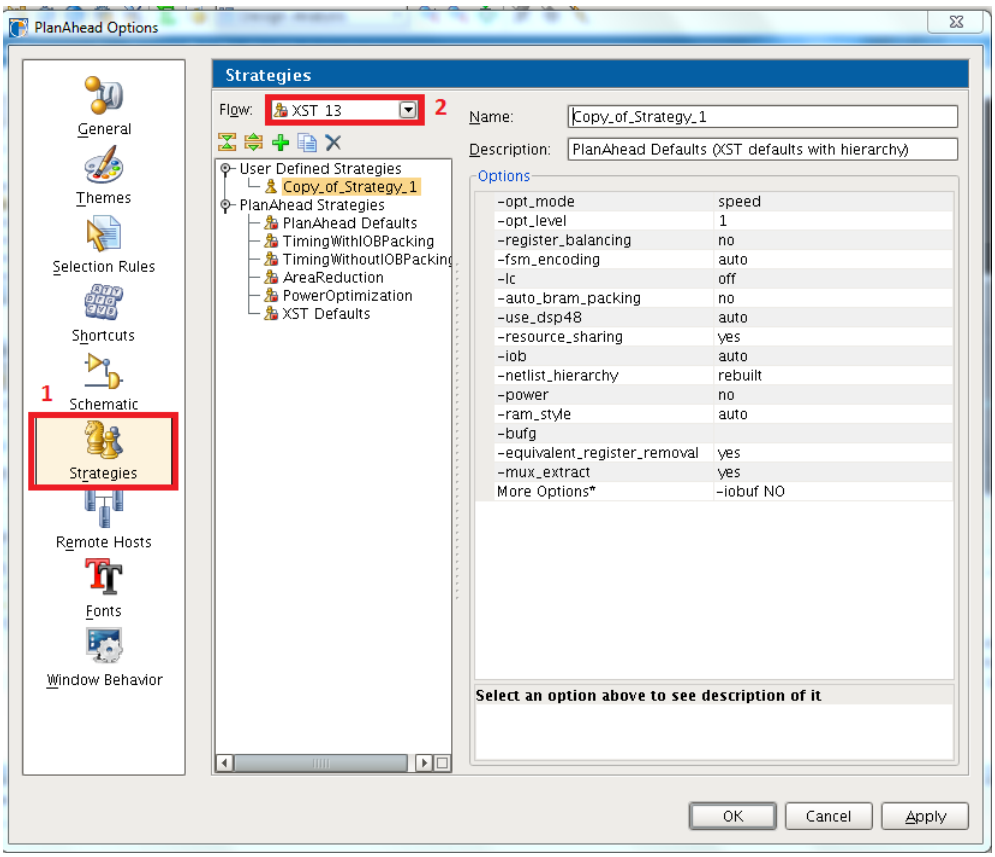


Figura 70 – Criação de uma nova estratégia de síntese física.

Na nova estratégia criada é preciso configurar o parâmetro *More Options* com o valor "-bm" seguido pelo caminho para o arquivo .bmm (Block RAM Memory Map) gerado pelo XPS, conforme se observa na Figura 72.

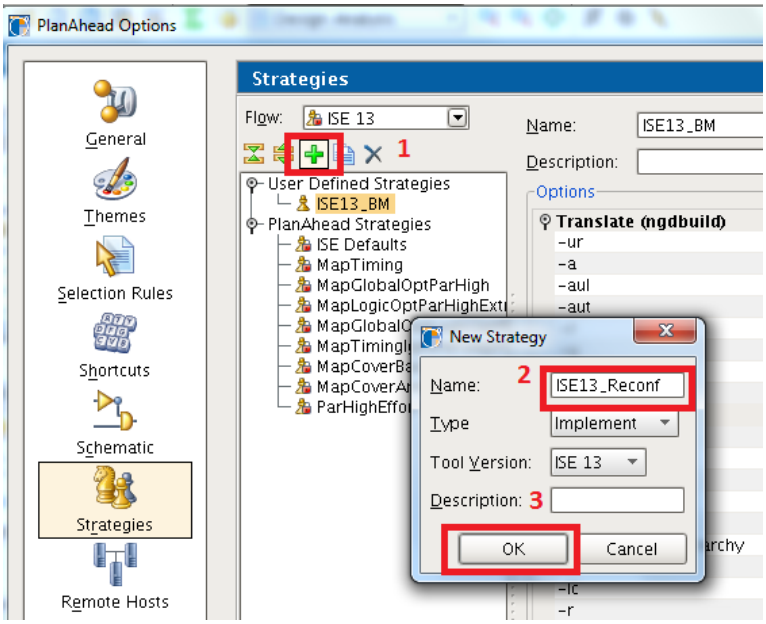


Figura 71 – Criação da estratégia de síntes ISE13_Reconf.

Este arquivo se chamará system.bmm e estará localizado no diretório *Implementation*, que se encontra na pasta raíz do projeto do XPS.

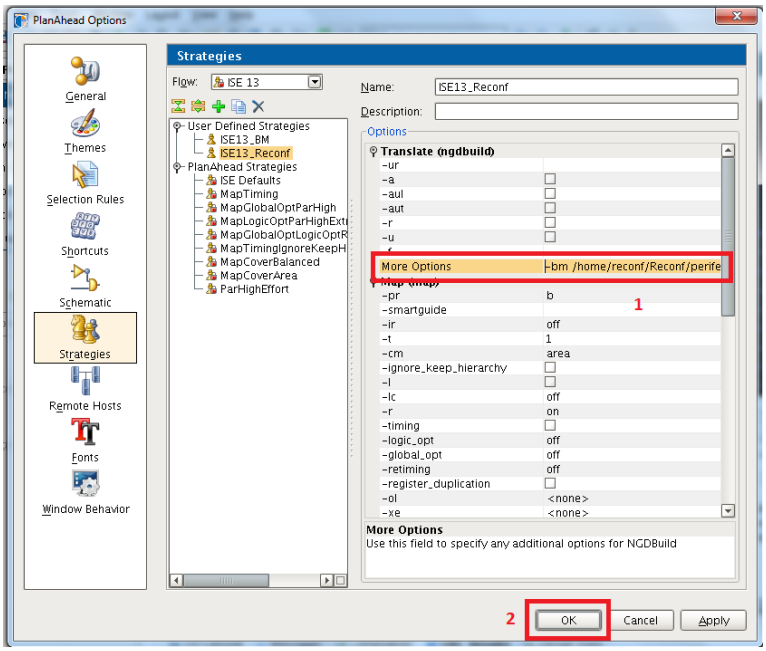


Figura 72 - Configurar a estratégia criada.

4.3.12.2 Implementação das partições

Após a estratégia ter sido criada é possível realizar as implementações, que são executadas através de *Design Runs*. Será necessário criar uma *Design Run* para a parte estática do sistema e um de seus módulos reconfiguráveis e depois uma *Design Run* para cada módulo reconfigurável restante. Na janela *Design Runs* já existirá uma opção criada por padrão pelo *PlanAhead* chamada *config_1*, esta deverá ser modificada clicando-se com o botão direito do *mouse* sobre ela e então na opção *Implementation Run Properties*, conforme ilustrado na Figura 73 .

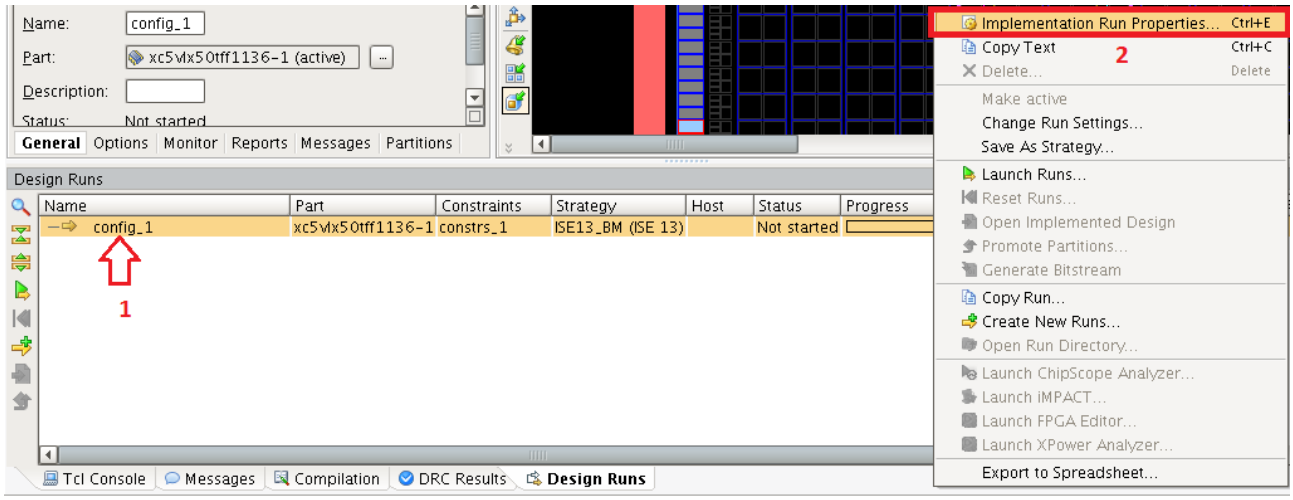


Figura 73 - Alterar as propriedades da Design Run.

A seguir, na janela *Implementation Run Properties*, o campo *Name* deverá ser alterado para um nome mais sugestivo, no exemplo da Figura 74 o nome escolhido foi *config_somador*, pois irá se realizar a implementação do *hardware* estático e do Módulo Somador.

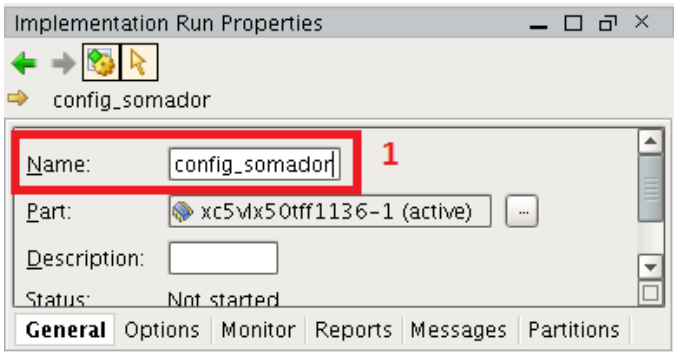


Figura 74 – Definição do nome da implementação física.

Em seguida deve-se selecionar a aba *Options* e alterar o campo *Strategy* para a estratégia criada em 4.3.12.1, no caso do projeto desenvolvido para ISE13_Reconf (ISE 13). Observar a Figura 75 . Para confirmar a alteração deve-se clicar em *Apply*.

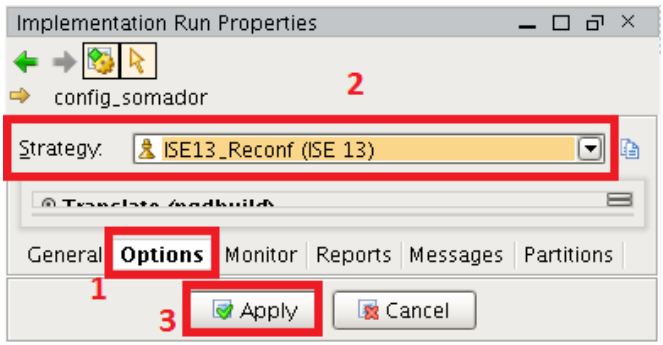


Figura 75 – Alteração da estratégia da implementação.

Na aba *Partitions* é necessário informar uma variação para o módulo reconfigurável, como no caso do exemplo da Figura 76 onde é selecionado o Módulo de Subtração como variante.

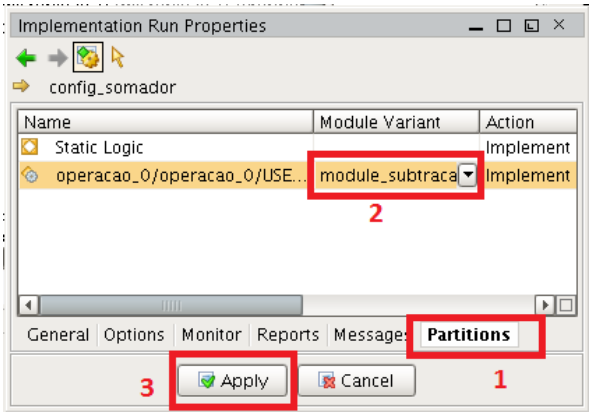


Figura 76 – Seleção de novo módulo reconfigurável.

Após a *Design Run* ter sido configurada é possível iniciar a primeira implementação. Para isso deve-se clicar com o botão direito do *mouse* sobre o nome da *Design Run* na janela *Design Runs* e então selecionar a opção *Launch Runs* conforme ilustrado na Figura 77.

O processo de implementação é computacionalmente custoso e pode levar vários minutos para ser executado. No diálogo que será exibido é possível definir o número de processos que serão responsáveis por executar a implementação. É recomendável configurar mais de um processo. No exemplo da Figura 78 4 processos serão utilizados.

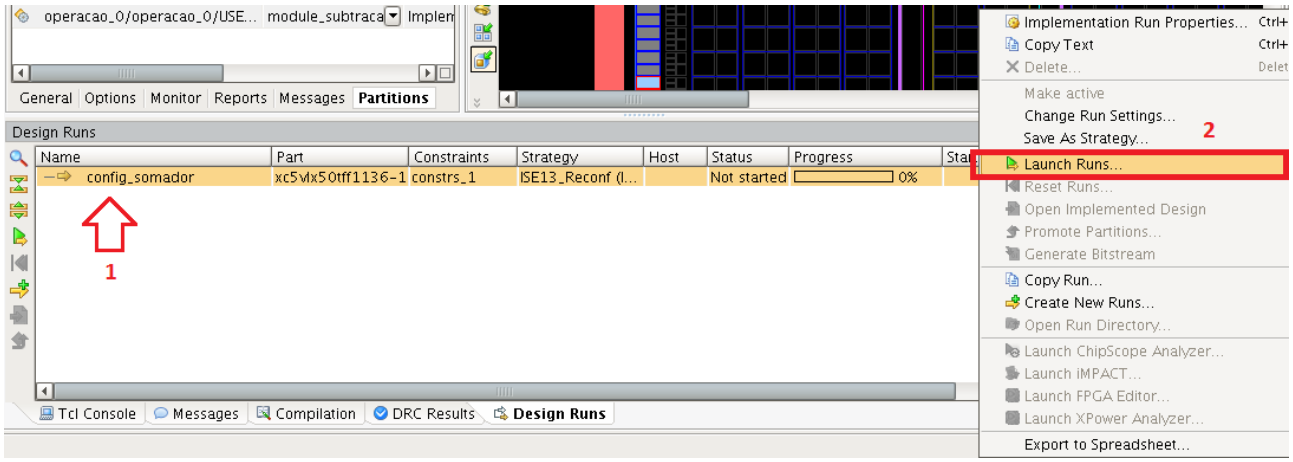


Figura 77 - Iniciar Design Run.

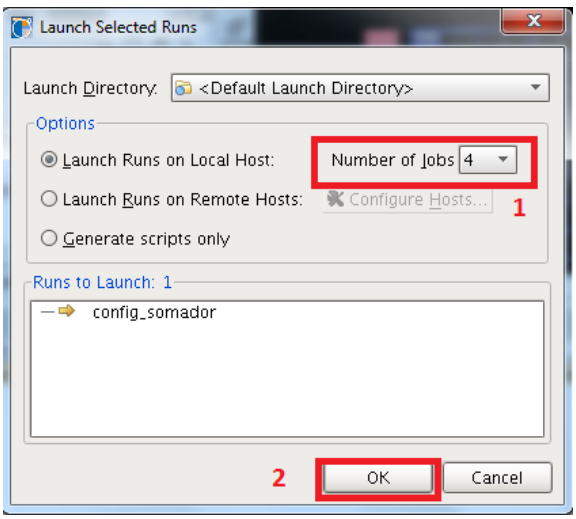


Figura 78 - Implementação será executada em 4 processos para melhor desempenho.

Após a implementação ter sido concluída, é necessário “promover” as partições implementadas. Este processo consiste em armazenar o resultado da síntese física. Desta forma, informações de síntese física destas partições estarão disponíveis para as próximas implementações. Isto garante que a interface física entre lógica estática e lógica reconfigurável ocorrerá de forma adequada.

É importante lembrar que para realizar a implementação dos Módulos Reconfiguráveis é necessário antes ter concluído a promoção da parte estática do projeto, pois esta será importada durante a síntese física dos Módulos Reconfiguráveis.

Para realizar a promoção de uma partição deve-se clicar sobre seu nome com o botão direto dou *mouse* e selecionar a opção *Promote Partitions*, conforme ilustrado na Figura 79 . Na tela seguinte a partição já estará selecionada por padrão, bastando clicar no botão OK para executar a promoção.

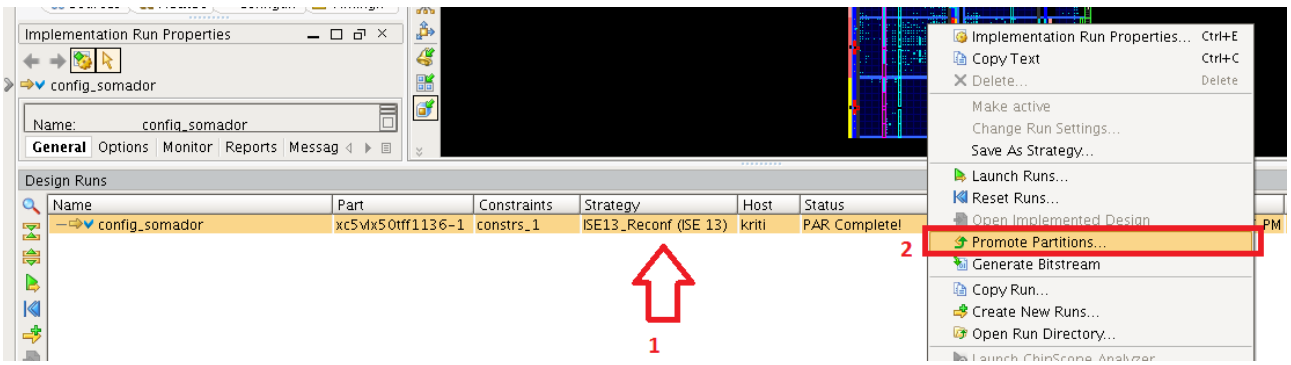


Figura 79 – Promoção da partição.

É necessário implementar e promover também os demais módulos reconfiguráveis. Para isto deve-se criar um nova *Design Run*. Para isto deve-se clicar com o botão direito do *mouse* sobre a área livre da janela *Design Runs* e selecionar a opção *Create New Runs*. Observe a Figura 80.

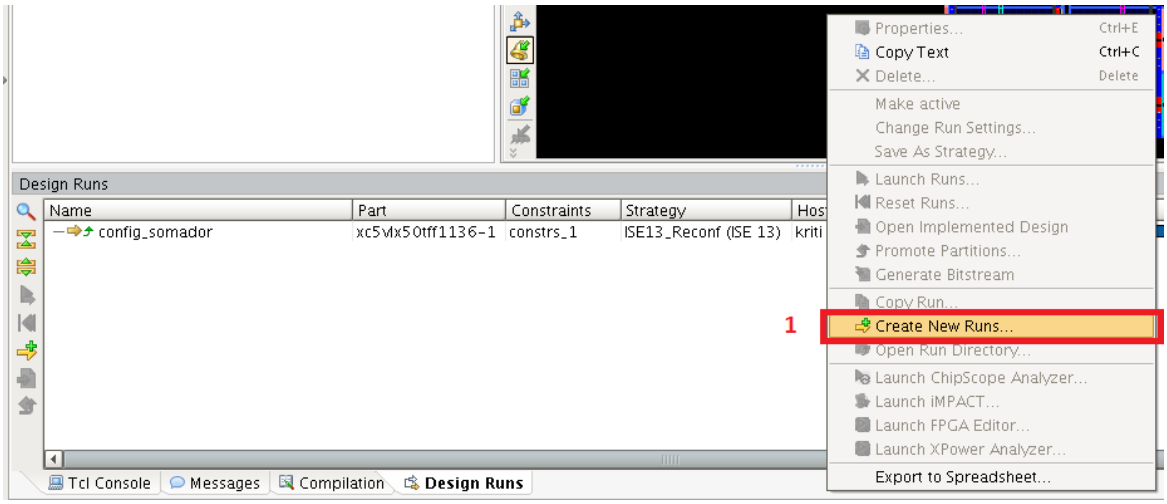


Figura 80 - Criar nova Design Run.

Na tela que será exibida é necessário configurar um nome, no caso do exemplo este é *config_subtrator*. A coluna *Strategy* também deve estar configurada com a mesma estratégia criada anteriormente, que já deve estar selecionada por padrão (Figura 81).

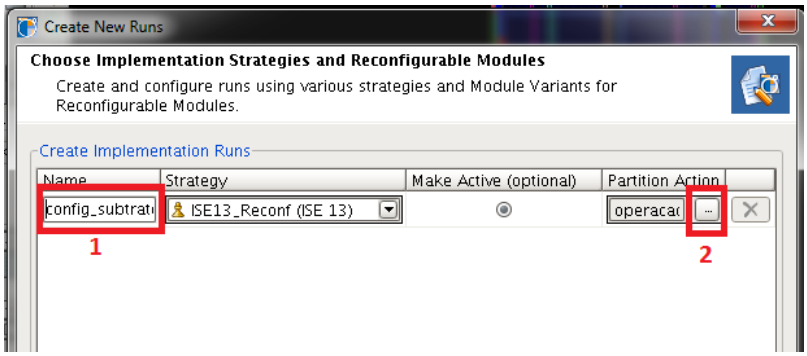


Figura 81 - Configurando uma nova Design Run.

Ao configurar a nova implementação é preciso selecionar o módulo reconfigurável que será implementado, também é necessário informar o *PlanAhead* que a parte estática do sistema não deve ser implementada novamente. Para isto deve-se clicar no botão contendo o sinal de três pontos na coluna *Partition Action.* Este botão deve exibir a tela apresentada na Figura 82, onde deve-se configurar o *Module Variant* como o próximo módulo reconfigurável que deseja-se implementar. Observa-se que a lógica estática por padrão não será implementada novamente, mas apenas importada.

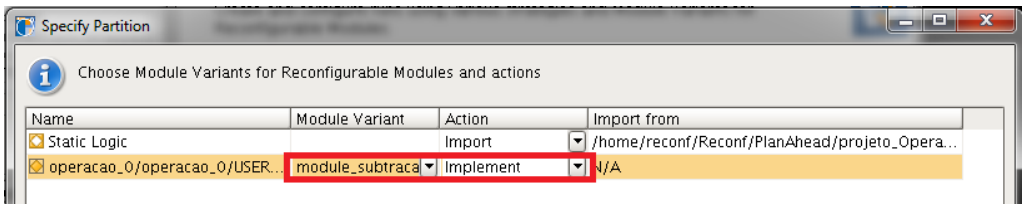


Figura 82 - Configuração de nova implementação.

Ao se clicar no botão OK iniciará o processo de implementação do Módulo Reconfigurável selecionado. Após a implementação ter sido concluída a módulo reconfigurável deve também ser promovido. Para isto deve-se clicar com o botão direito do *mouse* sobre o nome da Design Run e selecionar a opção *Promote Partitions*. Este processo de criação da Desing Run, implementação e promoção deve ser repetido para cada Módulo Reconfigurável que será utilizado no sistema.

Por fim, após serem realizadas as implementações e promoções de todas as partições é recomendável rodar um comando de verificação. Para isto deve-se selecionar a opção *Verify Configuration* que se encontra no menu *Flow*, conforme exemplo da Figura 83.

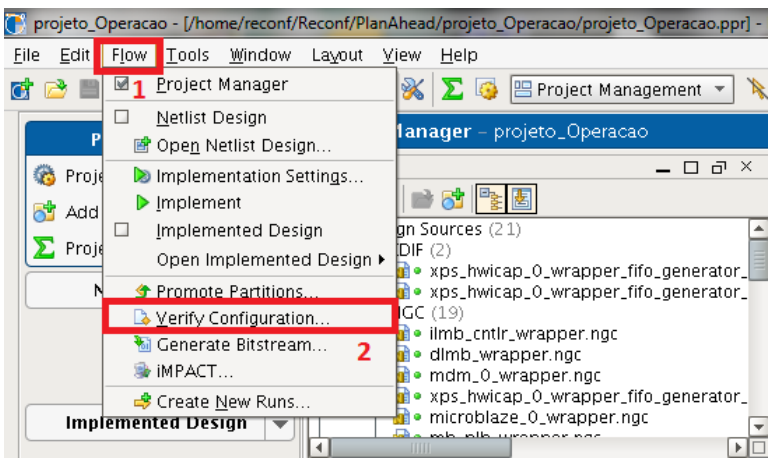


Figura 83 - Verificação Módulos Reconfiguráveis gerados.

Na tela seguinte deve-se selecionar todos os módulos cuja compatibilidade deva ser verificada, conforme Figura 84.

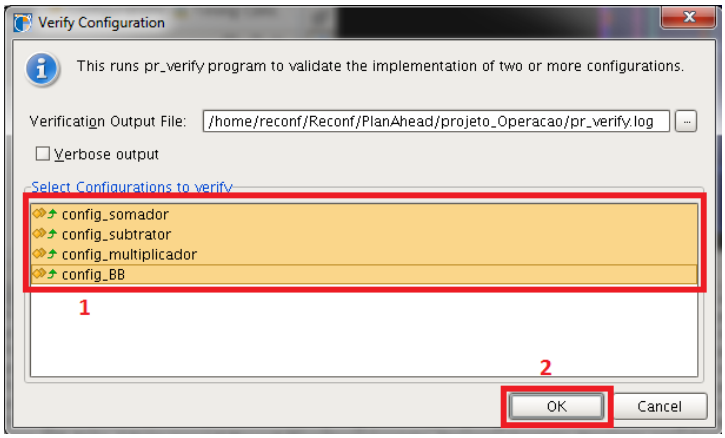


Figura 84 – Seleção dos módulos para serem verificados.

4.3.13 Geração dos Bitstreams

O último passo a ser realizado na ferramenta *PlanAhead* é a geração dos arquivos de *bitstream* para o *hardware* estático e para os Módulos Reconfiguráveis. Para gerar os *bitstreams* deve-se clicar com o botão direito do *mouse* sobre o nome das *Design Runs* e selecionar a opção *Generate Bitstream*, conforme ilustrado na Figura 85.

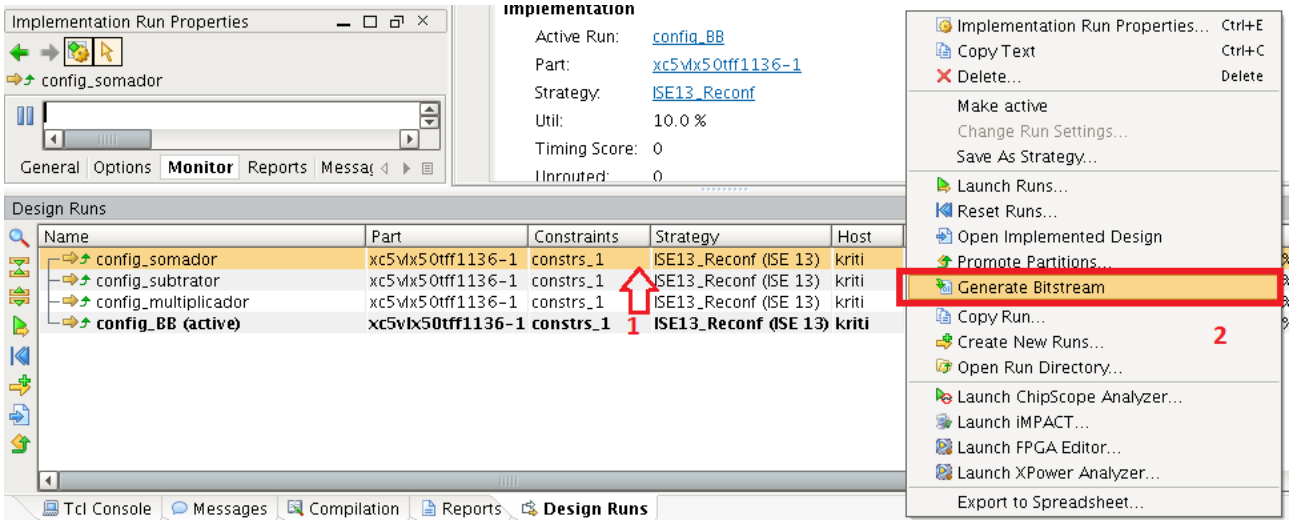


Figura 85 – Geração dos bitstreams.

Este processo deve ser repetido para cada *Design Run*, que ao fim deverá exibir uma tela como a da Figura 86 .

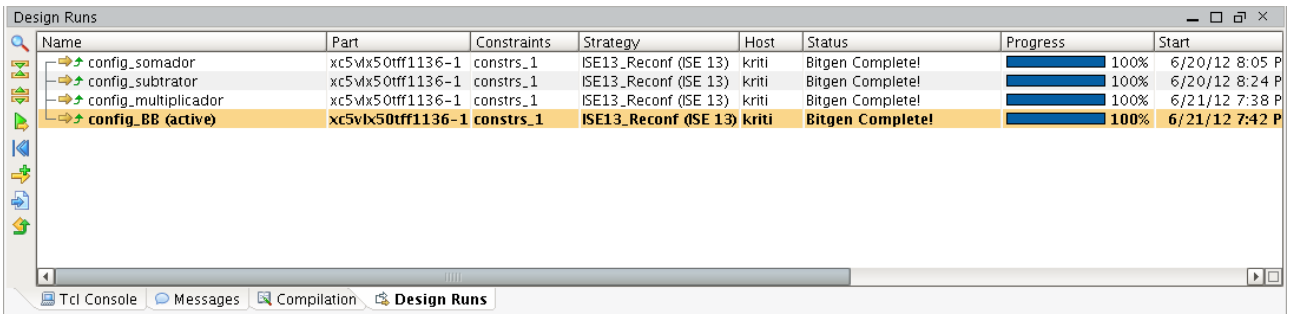


Figura 86 – Conclusão da geração dos bitstreams.

4.3.14 Inclusão do binário de software no Bitstream estático

Antes de realizar a gravação do *bitstream* no FPGA é preciso embarcar neste o binário compilado do *software* desenvolvido para o sistema. Isto é feito através da ferramenta *data2mem*. Esta ferramenta tem por entradas: (*i*) o arquivo .bmm gerado pelo XPS contendo informações sobre a configuração do sistema; (*ii*) o arquivo.elf gerado pelo SDK contendo o executável do *software*; (*iii*) arquivo .bit gerado pelo PlanAhead contendo o *bitstream* do *hardware* estático. A ferramenta deverá produzir um arquivo .bit contendo o *bitstream* do *hardware* estático combinado com as instruções do *software* carregadas na memória de programa do MicroBlaze. A Figura 87 ilustra este processo.

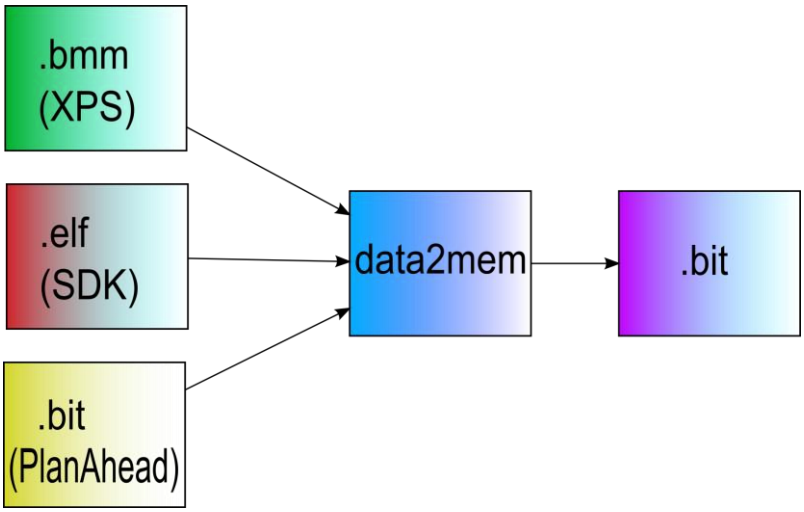


Figura 87 - Ferramenta data2mem

O data2mem deverá ser executado via linha de comando, recebendo os arquivos acima citados por parâmetros. Um exemplo de utilização da ferramenta pode ser visto na Listagem 5.

```
data2mem -bm periferico/implementation/system_bd -bt
PlanAhead/projeto_Operacao/projeto_Operacao.runs/config_somador_2/config_somador.bit -bd
SDK/operacao_demo/Debug/operacao_demo.elf tag microblaze_0 -o b download.bit
```

Listagem 5 - Chamada para a ferramenta data2mem.

4.3.15 Gravação do Bitstream estático

Antes de realizar a gravação a aplicação na memória *flash* da placa é necessário criar um arquivo em formato *.mcs* contendo tanto o *bitstream* estático, que será utilizado para configurar o FPGA a cada boot deste, quanto os *bitstreams* dos Módulos Reconfiguráveis.

Para isto será utilizada a ferramenta "promgen", que deve ser executada via linha de comando e receberá como parâmetros a largura do barramento de endereços da memória além dos arquivos contendo os *bitstreams* e seus respectivos descolamentos dentro da memória *flash*.

Um exemplo de utilização do promgen pode ser visto na Listagem 6 .

```
promgen -w -p mcs -data_width 16 -o bitstreams_flash -u 0x0000000 bitstream_full_test.bit -
data_file up 0x0200000 bitstream_somador_partial.bit -data_file up 0x0210000
bitstream_subtrator_partial.bit -data_file up 0x00220000 bitstream_multiplicador_partial.bit
```

Listagem 6 - Uso do promgen para criar um arquivo .mcs.

Após gerada a imagem a ser gravada na memória *flash*, será utilizada a ferramenta Impact para realizar o processo de gravação da mesma. Ao abrir a ferramenta será apresentada a tela da Figura 88 , onde deve-se selecionar a opção *Configure devices using Boundary-Scan (JTAG)* e clicar em OK.

Na tela seguinte não são necessárias alterações, bastando clicar novamente em OK.

A seguir será apresentada a tela principal da ferramenta, exibindo todos os dispositivos reconhecidos na cadeia JTAG. Deve-se então clicar com o botão direito do *mouse* sobre o rótulo "SPI/BPI", logo acima do dispositivo correspondente ao FPGA a ser programado e selecionar a opção *Add SPI/BPI flash*, conforme o exemplo da Figura 89 .

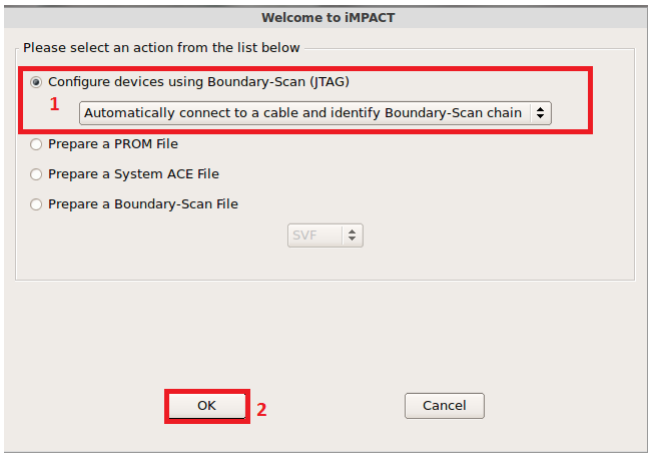


Figura 88 - Iniciar cadeia JTAG

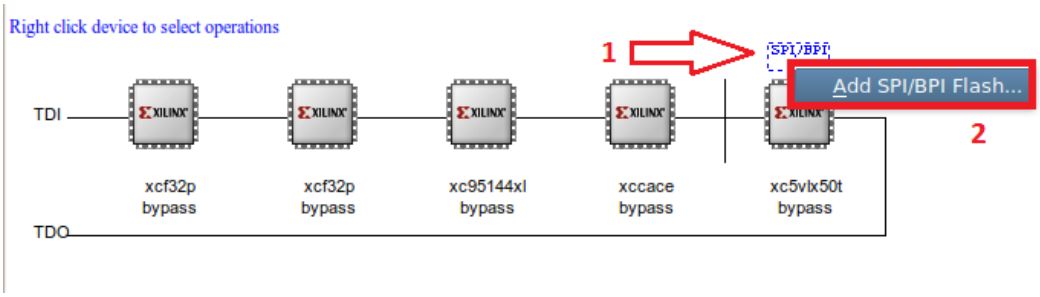


Figura 89 - Adicionar memória flash de gravação

Será exibida uma tela onde deverá ser selecionado o arquivo .mcs criado anteriormente e que será gravado na memória *flash*.

Na tela seguinte é necessário especificar qual o modelo de memória *flash* utilizada, no caso da placa ML505 esta é 28F256P30. Também deve-se informar quais são pinos para seleção da revisão de *bitstream* para ser utilizado na inicialização do FPGA. A memória especificada possui 4 endereços possíveis e os pinos são 23 e 22. Um exemplo de configuração pode ser visto na Figura 90.

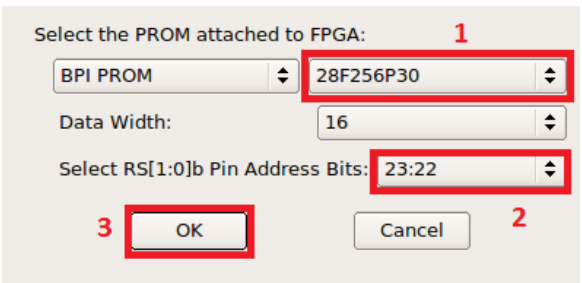


Figura 90 - Configuração da gravação da memória flash

No caso da placa ML505, antes de realizar a gravação da memória *flash* é necessário configurar a *DIP Switch* 3 para o valor "00001001". Esta configuração fará com que a placa inicialize a configuração do FPGA a partir da *Linear Flash*.

Após a configuração da gravação da memória *Flash* e da *DIP* Switch da placa pode-se então iniciar o processo de gravação efetivamente. Para isto deve-se clicar com o botão direito do *mouse* sobre a representação gráfica criada para a memória *flash* e selecionar a opção *Program*, conforme exemplo da Figura 91 .

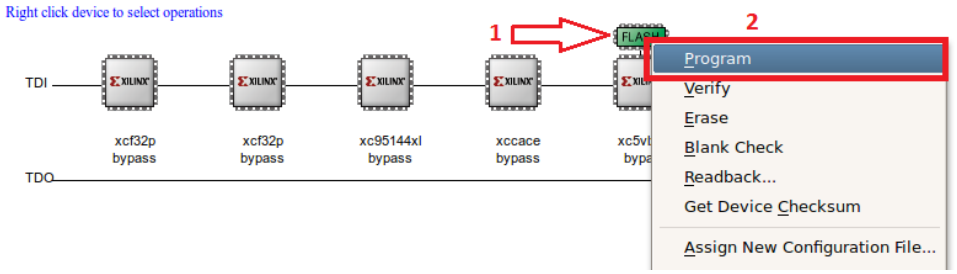


Figura 91 - Iniciar processo de gravação da memória flash

5 RESULTADOS

Após os *bitstreams* estarem devidamente gravados na memória *flash* é possível iniciar a execução da aplicação. Para isto deve-se conectar a placa ML505 à porta serial e iniciar um *software* de emulação de terminal devidamente configurado para utilizar uma taxa de *bits* de 115.200 *bits* por segundo.

A tela principal da aplicação deverá ser apresentada, conforme visto na Figura 92, onde é possível observar o uso do Módulo Reconfigurável com a implementação da operação de soma.

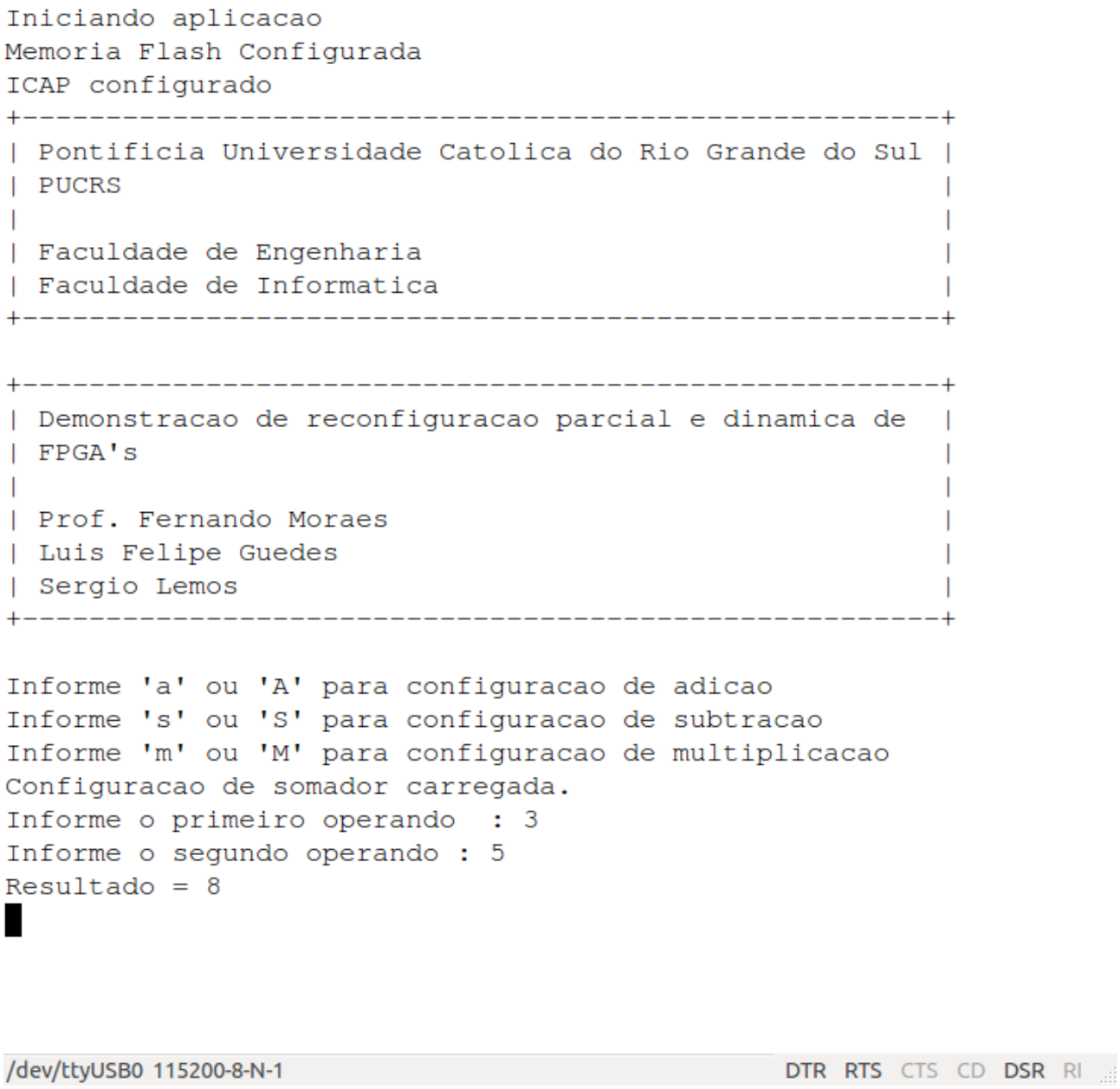


Figura 92 - Tela principal da aplicação

6 CONCLUSÃO

O fluxo de projeto para aplicações que utilizam a Reconfiguração Parcial Dinâmica está estável e muito mais acessível e automatizado que fora outrora. Porém a concepção de um projeto inteiro que necessitasse de um processador embarcado é demasiadamente extensa o que pode confundir ou dificultar fluxo de desenvolvimento do projeto.

Este trabalho procurou suprir a necessidade de um fluxo de projeto bem definido para a implementação de um projeto de sistema embarcado com a capacidade de RPD. Ele foi desenvolvido utilizando ferramentas presentes no ISE Design Suite 13, que é disponibilizado pelo fabricante Xilinx.

Foi desenvolvida uma aplicação exemplo para ilustrar o fluxo proposto como uma forma de tutorial. A aplicação consiste em um sistema embarcado contando com um processador Micro Blaze, um módulo de comunicação com a memória externa, um módulo para comunicação com a porta ICAP, que é responsável pela capacidade de reconfiguração parcial dinâmica, além de um módulo do usuário que possui a capacidade de RPD.

O ambiente de prototipação para a aplicação exemplo foi o kit de desenvolvimento ML505 da fabricante Xilinx. O kit possui diversas interfaces de IO e diversas memórias de estado sólido acessíveis pelo FPGA Virtex 5 presente no kit.

Todo o processo de criação de um sistema embarcado com capacidade de RPD foi documentado nesse trabalho e disponibilizado para uso como base para projetos posteriores.

REFERÊNCIAS

[CHA09] Chauhan, A.; Rajawat, A.; Patel, R. “Reconfiguration of FPGA for Domain Specific Applications using Embedded System Approach”. In: International Conference on Signal Processing Systems, 2009, pp. 438-442.

[DEH08] Dehon, A.; Hauck, S. “Reconfigurable Computing The Theory and Practice of FPGA-Based Computing”. Elsevier, 2008, 945 p.

[DUH12] Duhem, F. Muller, F. Lorenzini, P. "Reconfiguration time overhead on field programmable gate arrays: reduction and cost model". IET Computer & Digital Techniques, v.6(2), 2012, pp 105-113.

[DYE11] Dye, D. “*Partial Reconfiguration Of Xilinx FPGAs Using ISE Design Suite*”. Capturado em: http://www.xilinx.com/support/documentation/white_papers/wp374_Partial_Reconfig_Xilinx_FPGAs.pdf. Março 2012

[EMA12] GNU Emacs. Capturado em: http://www.gnu.org/software/emacs/. Junho 2012

[INT12] Intel. Capturado em: www.intel.com. Março 2012

[MCD08] McDonald, E. J. “*Runtime FPGA Partial Reconfiguration*”. In: IEEE Aerospace Conference, 2008, pp. 1-7.

[MES01] Mesquita, D. "*Contribuições para Reconfiguração Parcial, Remota e Dinâmica de FPGAs*". Dissertação de Mestrado, Programa de Pós-Graduação em Ciência da Computação, PUCRS, 2001, 103 p.

[VER07] Verma, S. "*How to design an FPGA architecture tailored for efficiency and performance*". Capturado em: http://www.eetimes.com/design/programmable-logic/4015097/How-to-design-an-FPGA-architecture-tailored-for-efficiency-and-performance. Junho 2012.

[WUK05] Wu, K.; Madsen, J. “Run-time Dynamic Reconfiguration: A Reality Check Based on FPGA Architectures from Xilinx”. In: NORCHIP Conference, 2005, pp 192-195.

[XIL10a] “Partial Reconfiguration Flow Presentation Manual”. Capturado em: http://www.xilinx.com/member/xup/workshops/partial-reconfiguration-flow/materials/13x/ml605/docs_pdf.zip. Março 2012

[XIL10b] “LogiCORE IP XPS HWICAP datasheet”. Capturado em: http://www.xilinx.com/support/documentation/ip_documentation/xps_hwicap.pdf . Junho 2012

[XIL10c] "*Indirect Programming of BPI PROMs with Virtex-5 FPGAs*". Capturado em: http://www.xilinx.com/support/documentation/application_notes/xapp973.pdf .Junho 2012

[XIL11a] “Partial Reconfiguration User Guide”. Capturado em: http://www.xilinx.com/support/documentation/sw_manuals/xilinx12_4/ug702.pdf. Março 2012

[XIL11b] “*Virtex-5 FPGA Configuration User Guide*”. Capturado em: http://www.xilinx.com/support/documentation/user_guides/ug191.pdf. Março 2012.

[XIL11c] “*ML505/ML506/ML507 Evaluation Platform User guide*”. Capturado em: http://www.xilinx.com/support/documentation/boards_and_kits/ug347.pdf. Abril 2012;

[XIL12a] “*Virtex-5 FPGA User Guide*”. Capturado em: http://www.xilinx.com/support/documentation/user_guides/ug190.pdf. Março 2012.

[XIL12b] “*Virtex 5 FPGA XtremeDSP Design Considerations*”. Capturado em: http://www.xilinx.com/support/documentation/user_guides/ug193.pdf. Abril 2012

[XIL12c] “Partial Reconfiguration Flow Workshop and Teaching Materials”. Capturado em: http://www.xilinx.com/university/workshops/partial-reconfiguration-flow/index.htm. Março 2012.