Documento de Arquitetura

Linguagem de Máquina e Microarquitetura - Problema 1

Universidade Estadual de Feira de Santana

Build 2.0a

Histórico de Revisões

Date	Descrição	Autor(s)
03/05/2018	Relatório Desenvolvido ao Final do Projeto	Nadine Cerqueira Marques Valmir Vinicius de Almeida Santos

SUMÁRIO

1 | Introdução

1. Propósito do Documento

Este documento tem por objetivo descrever a arquitetura do projeto Linguagem de Máquina e Microarquitetura - Problema 1, incluindo a apresentação da visão geral do processador utilizado, bem como foi instanciado. Além disso, descreve o algoritmo utilizado para o cálculo do CRC-32Q. Ademais também apresenta definições de entrada e saída do projeto final.

2. Acrônimos e Abreviações

Sigla	Descrição
FPGA	*Field Programmable Gate Array* (Arranjo de Portas Programáveis em Campo)
ASIC	*Application Specific Integrated Circuit* (Circuito Integrado de Aplicação Específica)
RISC	*Reduced Instruction Set Computer* (Computador com Conjunto Reduzido de Instruções)
LED	*Light Emitting Diode* (Diodo Emissor de Luz)
LE	*Logic Element* (Elemento Lógico)
LAB	*Logic Array Block* (Bloco de Matriz Lógico)
E/S	Entrada e Saída

3. Definições

Termo	Descrição
NIOS	Processador *softcore* RISC de 32 bits com arquitetura Harvard, desenvolvido pela Altera [1]
Processador *Softcore*	Implementação de um processador descrito em linguagem de hardware, que pode ser customizado e sintetizado em um *FPGA* ou *ASIC* [1]
CRC	Cógigo para detecção de erros em dados por meio da checagem por redundância cíclica
On Chip Memory	É o tipo de memória mais simples para uso em FPGA já que é implementada na própria placa [2]
Assembly	Representação simbólica das instruções de máquina [3]
bit	Dígito binário [3]
word	Unidade natural de acesso em um computador, geralmente em grupo de 32 *bits* [3]
Quartus	*Software* para criação de projetos com dispositivos programáveis, como o FPGA. Possui diversas ferramentas para análise de projeto, como *Chip Planner* e *TimeQuest Time Analyzer*

4. Objetivo do Projeto

O sistema desenvolvido tem por objetivo realizar o cálculo do CRC-32Q para uma sequência de dados de entrada cujo tamanho máximo é de 1KB. O cômputo em questão deverá ser efetuado por um processador NIOS II, implementado em um dispositivo FPGA Altera.

O resultado desse cálculo, uma sequência composta por 32 *bits* de dados, será apresentado a partir da interação do usuário com um *push button* e por meio de quatro *leds*. Além dos componentes da interface de entrada/saída anteriormente citados, é imprescindível que a unidade de processamento esteja associada a uma interface JTAG UART e a uma memória *on-chip*.

2 | Visão Geral da Arquitetura e Descrição do Projeto

A visão geral da arquitetura do processador NIOS II é apresentada nesta seção, bem como a arquitetura final do projeto. Além disso, as etapas de instanciamento e configuração do processador, implementação do algoritmo e componentes em *Assembly* utilizados para o cálculo do CRC-32Q são descritos.

1. Arquitetura do Processador

A arquitetura do processador NIOS II pode ser observada na figura 2.1. Nessa imagem, pode-se identificar os elementos essenciais para a implementação desse tipo de processador, como o banco de registradores e sua memória interna, com regiões de dados e de instruções.

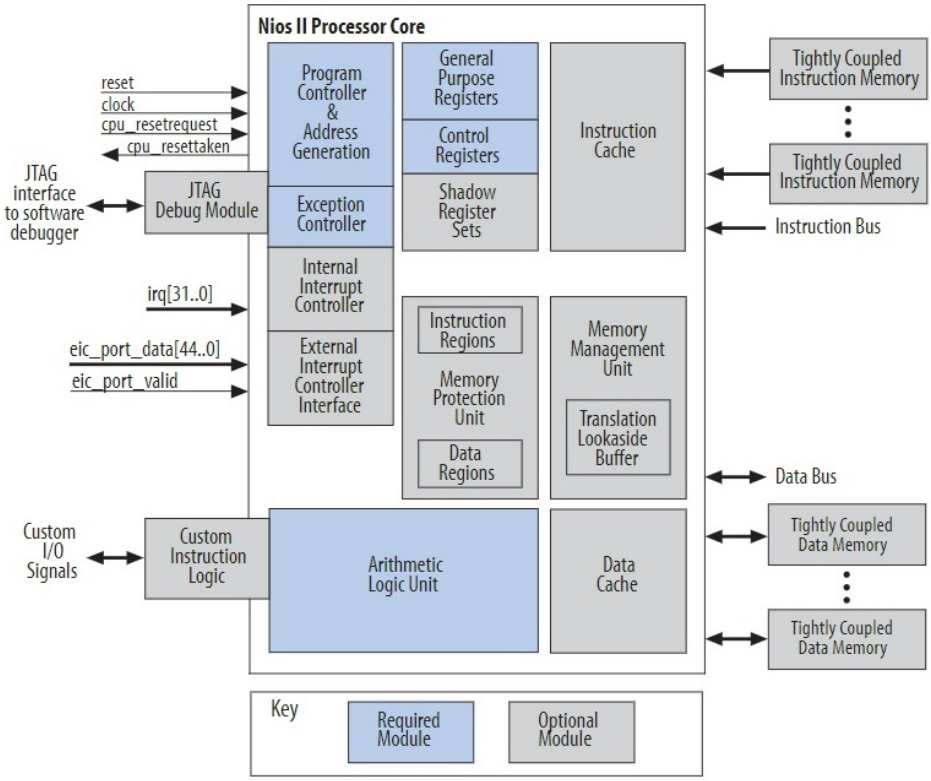

Figura 2.1: Diagrama de Blocos da Arquitetura do *Core* do NIOS II [4]

2. Configuração e Instanciamento do Processador

A fase inicial do processo de desenvolvimento do projeto teve como foco a preparação de ambiente, ou seja, instalação e configuração dos componentes fundamentais à implementação.

Nesse contexto, destaca-se o *software* QUARTUS II, na sua versão 13.0.1, utilizado para configuração e instanciamento do processador NIOS II. Dentro deste programa, adotou-se a ferramenta Qsys, a partir da qual foram definidos parâmetros importantes do *softcore*, bem como sua memória e elementos periféricos (*LEDs* e botões).

Além disso, tendo em vista as versões *fast*, *standart* e *economy* do processador NIOS II e suas respectivas especificações, selecionou-se, por meio da ferramenta Qsys, a versão a ser utilizada.

A versão do NIOS II adotada foi a *economy*, como pode observado na tabela de especificações do projeto. Isso porque, para o cálculo do CRC-32Q, não foi necessária, por exemplo, a realização de operações de multiplicação, presente nas outras versões do processador.

Especificações do Projeto

Processador	NIOS II *Economy*
Memória Associada	*On Chip Memory* com 4096 *bytes*
Periféricos E/S	*1 Push Button* e 4 *LEDs*
Placa	Cyclone IV E EP4CE6E22C8 com *clock* de 50MHZ

3. Arquitetura do Projeto

A arquitetura geral do projeto, representada pela figura 2.2, é composta pelo processador NIOS II/e, que se conecta aos periféricos de E/S presentes no FPGA em que foi instanciado e configurado.

Figura 2.2: **Arquitetura do Projeto**

4. Algoritmo

Existem, atualmente, duas abordagens algorítmicas para o cálculo do CRC-32. A primeira delas é mais tradicional e baseia-se no cômputo bit-a-bit da entrada. Por outro lado, o segundo método, conhecido como método baseado em tabela (*Lookup Table Based*), realiza operações byte-a-byte. Apesar de mais simples, a primeira técnica pode ser ineficiente quando a entrada possui tamanho significativo, já que cada bit deverá ser analisado. Levando em consideração essa característica, adotou-se o *Look Up Table* na implementação descrita neste documento.

Outro aspecto a ser considerado são as diversas modalidades de CRC-32 existentes. Esses tipos diferem entre si por conta de alguns parâmetros que influenciam no valor final. Tais parâmetros são: inversão da entrada, valor inicial do resultado, XOR final do resultado e inversão da saída. O produto implementado foi desenvolvido com base no CRC-32Q, que não realiza inversão de entrada e saída, utiliza 0 como valor inicial do resultado e não realiza operação XOR ao final do cômputo.

Além do cálculo do CRC, foi necessário também desenvolver um algoritmo para possibilitar a exibição do resultado de 32 bits em sequências de 4 bits por meio de 4 *LEDs* e um *push button*. Para esse caso aderiu-se a técnica de espera por eventos externos denominada *polling*.

4.1. Construção da Tabela (Lookup Table)

O cálculo do CRC-32 baseado em tabelas efetua o processamento de cada *byte* da entrada por iteração. Cada um desses *bytes* pode assumir apenas 256 valores diferentes. Com base nessas observações, surge a possibilidade de pré computar a divisão de cada possível *byte* pelo polinômio, que é fixo, e armazenar os resultados em uma tabela. A *lookup table* é indexada por cada valor diferente que um byte pode assumir, indo desde 0 até o valor máximo de 255. O algoritmo completo para construção da tabela pode ser visualizado no fluxograma da figura 2.3.

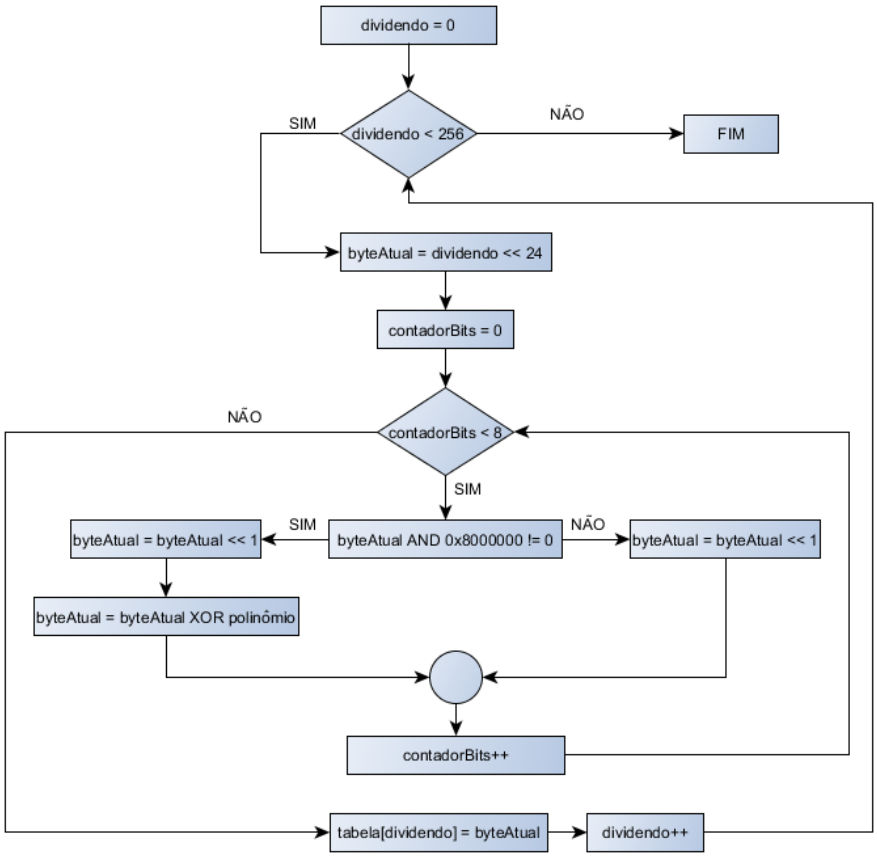

Figura 2.3: **Fluxograma - Construção da Tabela**

4.2. Cálculo do CRC-32

Tendo sido construída a tabela, o próximo passo consiste em obter cada *byte* da entrada e utilizá-lo como índice para obtenção do valor pré-computado na tabela. A entrada do sistema possui tamanho de 1 KB e é obtida da memória em sequências com tamanho máximo de 32 bits. Assim, são necessárias 250 leituras à memória para obtenção de toda a entrada e 1000 iterações para o cálculo do CRC, já que para cada sequência obtida são realizadas 4 iterações, uma para cada *byte*.

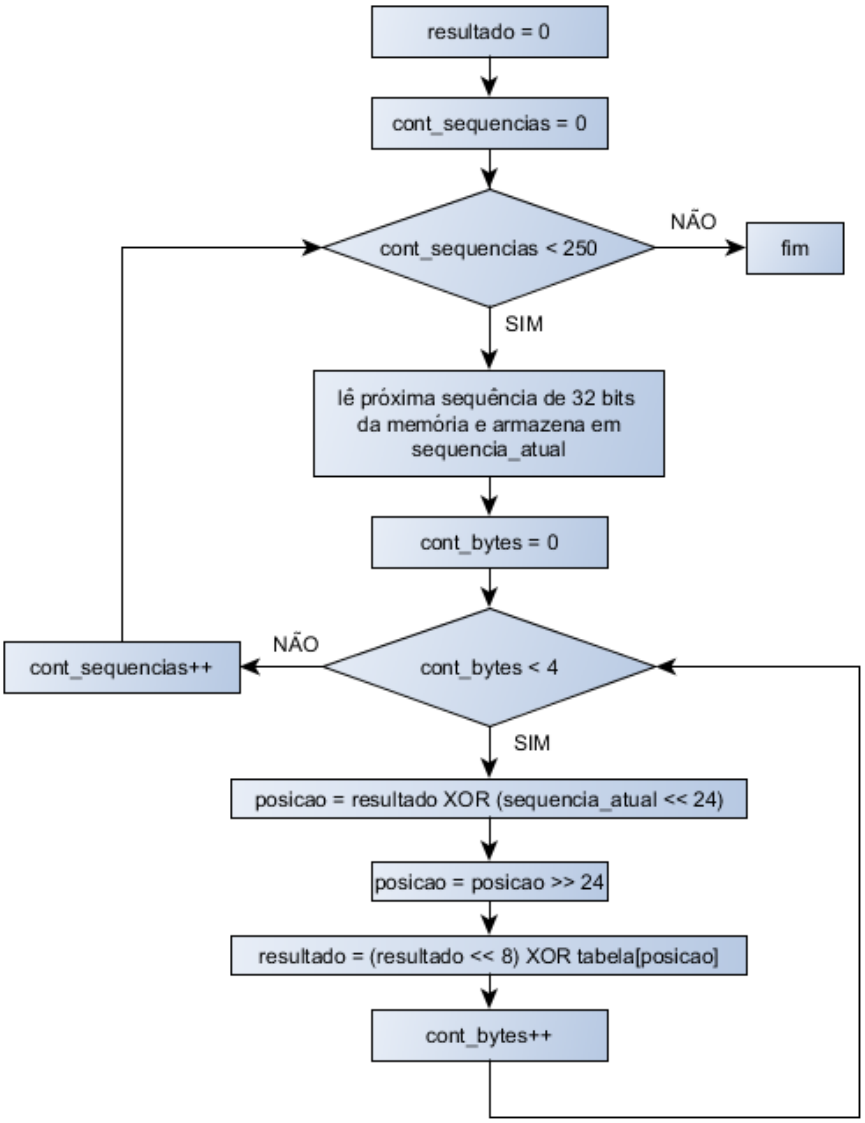

Figura 2.4: Fluxograma - Cálculo do CRC-32

4.3. Exibição do Resultado

A etapa final do algoritmo implementado consiste na exibição do resultado obtido. O CRC calculado, que é composto por 32 bits, é apresentado a partir de 4 *LEDs*. Dessa forma, foi fundamental utilizar um botão do tipo *push button*, de forma que cada toque no botão carrega os 4 bits menos significativos do resultado nos *LEDs*.

Essa funcionalidade foi implementada por meio da técnica *polling*, que consiste em fazer diversas verificações, até que a condição do botão ser pressionado seja atendida.

Vale ressaltar que, após a exibição de 4 bits, é necessário inserir um atraso, pois o tempo em que o dedo humano permanece pressionando o botão é consideravelmente

maior que o tempo de execução do código. O *delay* foi implementado através da utilização de um decrementador, cujo valor inicial é 500000.

O algoritmo da rotina de exibição do resultado está descrito no fluxograma da figura 2.5.

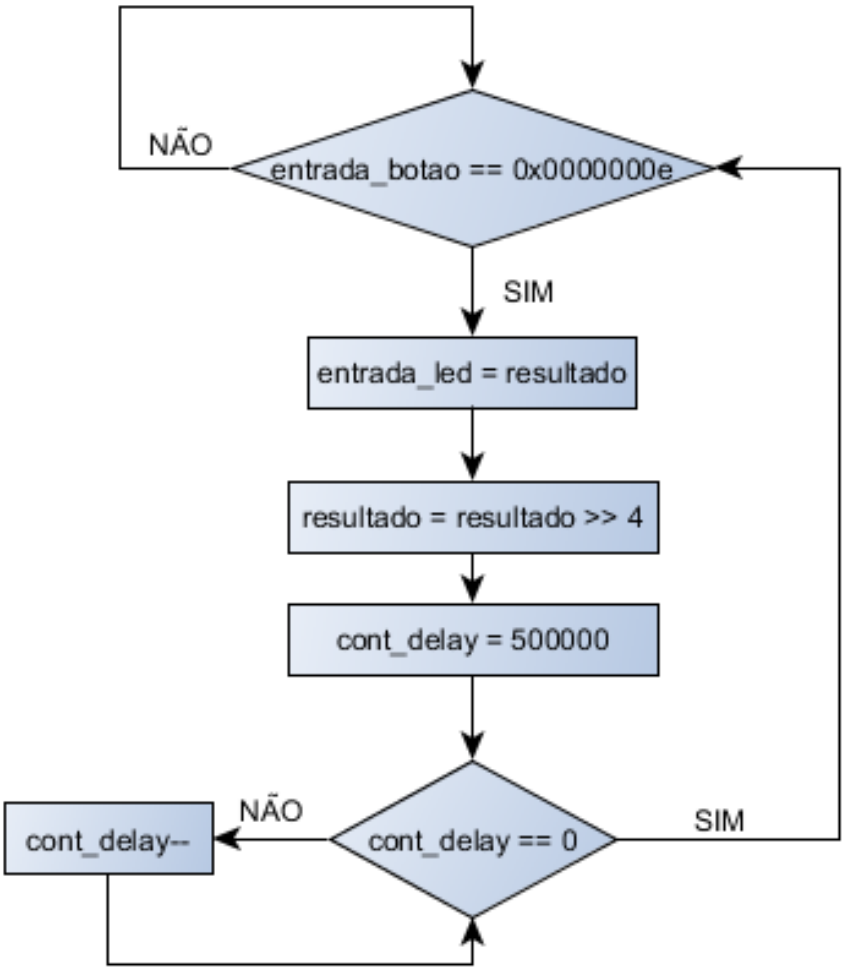

Figura 2.5: Fluxograma - Exibição do Resultado

5. Assembly

Esta seção visa apresentar os componentes em *Assembly* utilizados para a implementação do algoritmo para o cálculo do CRC-32Q e para exibição do resultado desse cálculo.

5.1. Instruções e Diretivas

Instruções Utilizadas

Segue abaixo a tabela composta pelas instruções do NIOS II utilizadas para o desenvolvimento do projeto.

Instrução	Descrição
AND	Lógico AND bit a bit
XOR	Lógico OR exclusivo bit a bit
ADD	Soma conteúdo de um registrador com outro
ADDI	Adição com valor imediato
SUBI	Subtração por valor imediato
MOVIA	Move endereço imediato para a *word*
MOVI	Move imediato com sinal para a *word*
MOV	Mover conteúdo de um registrador para outro
STW	Armazena *word* para memória ou periférico de E/S
LDW	Carrega *word* da memória ou periférico de E /S
STB	Armazena *byte* para memória ou periférico de E/S
LDB	Carrega *byte* da memória ou periférico de E/S
SRLI	Deslocamento lógico para a direita em quantidade de vezes imediata
SLLI	Deslocamento lógico para a esquerda em quantidade de vezes imediata
ROR	Rotação para direita
BEQ	Desvia o fluxo de execução caso a condição de igualdade seja verdadeira
BNE	Desvia o fluxo de execução caso a condição de desigualdade seja verdadeira
CALL	Chama subrotina
BR	Desvio incondicional

Diretivas Utilizadas

A tabela a seguir é composta pelas diretivas do NIOS II utilizadas para o desenvolvimento do projeto.

Diretiva	Descrição
.data	Identifica os dados que devem ser colocados na seção de dados da memória
.word	Expressões separadas por vírgula seguintes a essa diretiva são montadas em um número binário de 32 *bits*
.equ	Seta o valor de um símbolo para uma expressão
.text	Identifica o código que deve ser colocado na seção de código da memória
.global	Faz com que símbolo após a diretiva seja visível externamento ao arquivo objeto montado
.end	Identifica o final do arquivo do código fonte

3 | Resultados

1. Testes

Durante o processo de implementação da solução foram realizados diversos testes, buscando, assim, garantir o funcionamento esperado do algoritmo em linguagem *Assembly*, assim como do processador NIOS II.

A primeira fase de testes foi baseada na realização de simulações do código fonte por meio do *software* JNIOSEmu. Por outro lado, a segunda fase aferições baseou-se carregamento dos arquivos do processador (gerados pela ferramenta Qsys) e do código fonte na placa Cyclone IV EP4CE6E22C8, por meio da aplicação Altera Monitor.

Os resultados obtidos em tempo de execução foram comparados os valores fornecidos por ferramenta *online*, exibida na figura 3.1, para geração de sequências aleatórias de 1 KB de dados e de valores de CRC-32 com base nessas sequências.

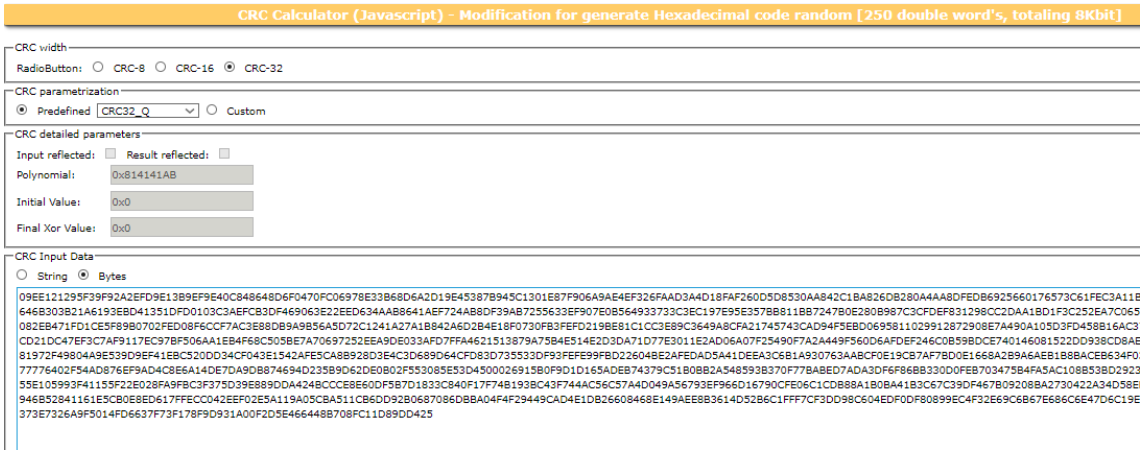

Figura 3.1: **Calculadora *Online* CRC-32Q**

Os primeiros testes da segunda fase revelaram problemas ocasionados por conta do baixo valor (1000) utilizado inicialmente parar gerar *delay* através de decrementos. Dessa forma, tornou-se fundamental alterar esse número para 50000. Após essa modificação, o sistema passou a apresentar o comportamento esperado, atendendo, assim, aos requisitos propostos.

2. Área Total Ocupada pelo Circuito

A partir de relatório gerados pelo *software* Quartus foi possível obter dados sobre a área que o projeto implementado consome em função dos elementos que o compõem.

Área Ocupada em Função dos Elementos Internos do Dispositivo

A tabela a seguir é composta pelas por elementos internos do dispositivo e suas respectivas quantidades ao final do projeto.

Elemento	**Quantidade**
LE	1.695
LAB	136
Registradores	920
Memória	44.032 *bits*

Área Ocupada por Elementos Lógicos por Modo

Além disso, esse relatórios ainda apresentam a quantidade de elementos lógicos utilizados em cada modo de operação, como mostra a tabela a seguir.

Modo	**Quantidade**
Normal	1.464
Aritmético	137

Representação da Área Total Ocupada Pelo Circuito

A figura 3.3 a seguir, gerada pela ferramenta *Chip Planner* do *software Quartus*, apresenta a área total ocupada pelo circuito do projeto, simbolizada pelos blocos em azul escuro.

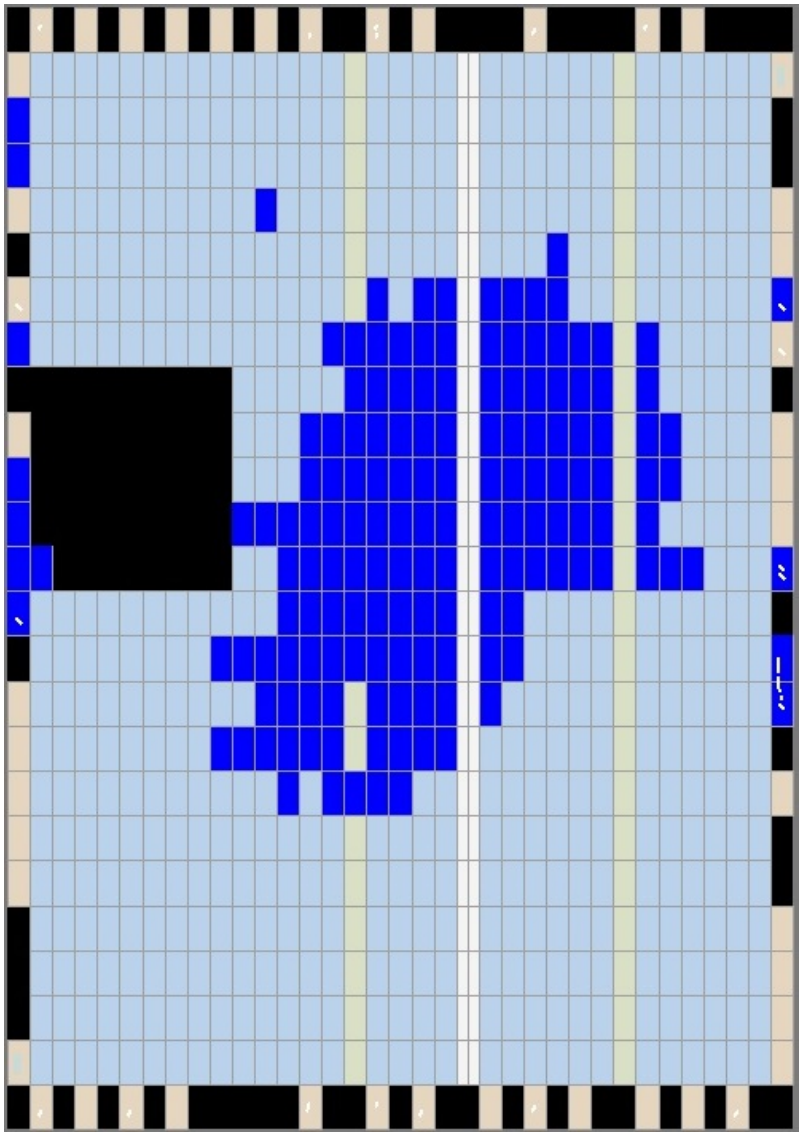

Figura 3.2: **Área Total Ocupada pelo Circuito**

3. Caminho Crítico do Circuito

A partir da geração de relatórios com tabelas por meio da ferramenta *TimeQuest Timing Analyzer*, presente no *software* Quartus, foi possível observar qual caminho do sistema com maior custo em termos de tempo. Assim, verificou-se que esse caminho crítico se dá entre registradores presentes na interface JTAG e que consome 99.209ns.

Esse caminho foi identificado no circuito por meio da ferramenta *Chip Planner* e é apresentado na figura a seguir, extraída da própria ferramenta:

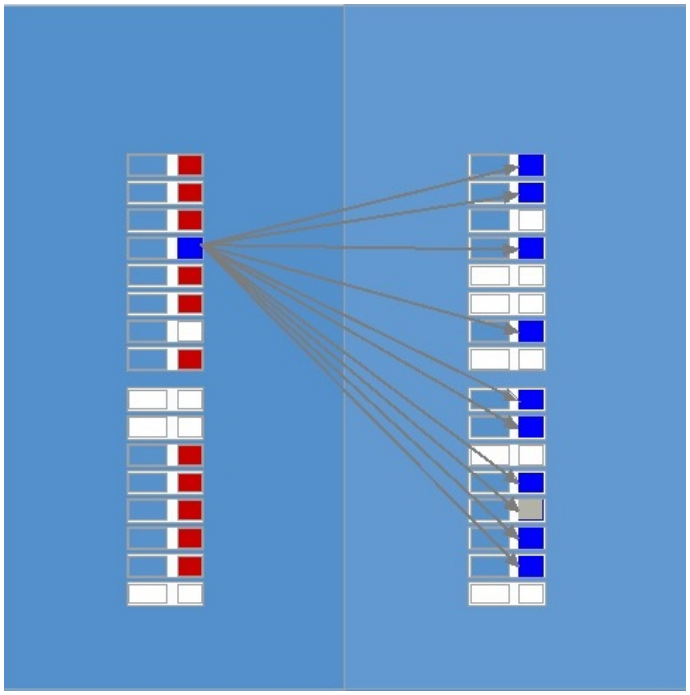

Figura 3.3: **Caminho Crítico do Circuito**

4 | Conclusão

A concepção sistema para cálculo de CRC-32, utilizando o *sofcore* NIOS II, exigiu a aplicação prática de conceitos teóricos relacionados ao funcionamento de processadores e à programação em baixo nível por meio da linguagem *Assembly*. Vale destacar, ainda, que o processo de desenvolvimento proporcionou o ganho de experiência na utilização dos componentes de *software* (Quartus II, Altera Monitor e JNIOSEmu) e *hardware* (placa Cyclone IV) adotados.

Entende-se que o produto obtido ao final do projeto foi satisfatório, pois atende aos requisitos funcionais e não funcionais propostos. Entretanto, existe ainda abertura para futuros trabalhos direcionados à promoção, por exemplo, de melhorias relacionadas ao desempenho (tempo de execução). Ademais, outro possível aperfeiçoamento é promover uma flexibilização no código fonte, de maneira a proporcionar o cálculo de outras versões do CRC-32.

1. Referências

[1] R. P. M. da Silva, *Processador softcore Altera Nios II*. 2014. Disponível em: <https://www.embarcados.com.br/altera-nios-ii/>. Acesso em: 3 maio 2018.

[2] Altera, *Memory System Design*. Disponível em: <https://www.altera.com/en_US/pdfs/literature /hb/nios2/edh_ed51008.pdf>. Acesso em: 3 maio 2018.

[3] J. L. H. David A. P, *Computer Organization and Design: The Hardware/Software Interface ARM Edition*. Morgan Kaufman, 2017.

[4] Altera, *Nios II Processor Reference Guide*. Disponível em: <https://www.altera.com/documentation/iga1420498949526.html#iga1409258110065>. Acesso em: 3 maio 2018.