Documento de Arquitetura

Calculadora CRC

Universidade Estadual de Feira de Santana

Versão 1.0

Histórico de Revisões

Date	Descrição	Autor(s)
03/05/2018	Versão Inicial do Sistema	Guilherme Wanderley, Manuella Vieira

Acrônimos e Abreviações

Sigla	Descrição
CPU	*Central Processing Unit* ou Unidade de Processamento Central
CRC	*Cyclic Redundancy Check* ou Conferência Redundante Cíclica
FPGA	*Field Programmable Gate-Array* ou Vetor de Portas Programáveis por Campo
ISA	*Instruction Set Architecture* ou Arquitetura do Conjunto de Instruções
JTAG	*Joint Test Action Group*. Padrão industrial para programar FPGAs e verificar circuitos
RISC	*Reduced Instruction Set Computer* ou Computador com Conjunto de Instruções Reduzido
UART	*Universal asynchronous receiver-transmitter* ou Transmissor-Receptor Assíncrono Universal
ULA	Unidade de Lógica Aritmética. Onde as operações são realizadas

SUMÁRIO

1 | Introdução

1. Propósito do Documento

Este documento objetiva a descrição do processo de desenvolvimento de um sistema para cálculo do CRC-32, também chamado de Calculadora CRC. Para tanto, o documento descreve os requisitos do projeto, arquitetura e conjunto de instruções do processador utilizado e análise do circuito e algoritmo construído.

2. Visão Geral do Documento

O presente documento está organizado da seguinte maneira:

- **Capítulo 2** – Explicita os requisitos funcionais e não-funcionais do projeto, assim como requisitos de entrada e saída.
- **Capítulo 3** – Apresenta a arquitetura do processador e seu conjunto de instruções. Em seguida, mostra o relatório de uso da FPGA pelo circuito construído.
- **Capítulo 4** – Descreve e analisa o algoritmo utilizado para cálculo do CRC.

2 | Requisitos

1. Requisitos Funcionais

O sistema deve ser capaz de calcular o CRC-32 de uma sequência de dados com tamanho de 1 KB. O CRC-32 é um código de tamanho 32 bits para conferência da validade da mensagem. A mensagem possui 1 KByte, ou seja, 8Kbits.

2. Requisitos Não-Funcionais

Em sistemas digitais é sempre desejável que o sistema seja mais rápido e mais eficiente energeticamente possível. No caso deste projeto o fator energético é fixo, restando apenas o tempo de execução, que deve ser o menor possível.

3. Requisitos de Entrada e Saída

Também é estabelecido que a saída para exibição do CRC-32 será feita através de 4 LEDs, presentes no kit da FPGA. Um botão é responsável por alternar os bits do código sendo exibidos.

3 | Descrição da Arquitetura

1. Arquitetura do Processador

O Nios II é um processador softcore RISC de 32 bits, distribuido pela Altera. Existem três implementações distintas, Nios II *Economy*, *Standart* e *Fast*. Neste projeto é utilizada a implementação *Economy*: menor, mais lenta e gratis.

1.1. Geral

Por ser a versão econômica, nem todas as estruturas estão presentes em sua implementação, como visto nas seções seguintes. Com relação a meta-arquitetura, o *pipeline* é de apenas um estágio, não havendo *pipeline* nos barramentos de instrução e de dados. Também não há cache para estes barramentos.

1.2. Registradores

O Nios II possui um banco de 32 registradores de uso genérico, cada um com 32 bits. Apesar de não terem sido utilizados neste projeto, o Nios II disponibiliza também a possibilidade de 32 registradores de controle e, para a versão *Fast*, até 63 bancos de registradores sombra, para troca de contexto.

1.3. ULA

A ULA do Nios II recebe um ou dois registradores como operandos e escreve o resultado em um registrador. A ULA é responsável pelas operações aritméticas, relacionais, lógicas e de deslocamento de bits num registrador.

As operações aritméticas incluem soma, subtração e, nas versões mais potentes, opção para multiplicação e divisão.

As operações relacionais são: igualdade, diferença, maior ou igual a e menor que. Tanto operações ariméticas como relacionais possuem a opção de números com ou sem sinal.

As operações lógicas são: AND, OR, NOR e XOR. As operações de deslocamento incluem deslocamento e rotação de 0 a 31 bits.

Qualquer operação diferente destas é tratada por *software*, sendo realizada em *hardware* pelas operações destacadas anteriormente. O Nios, no entanto, também permite ao usuário a criação de instruções personalizadas, bem como instruções de ponto flutuante de precisão simples nas versões mais completas.

1.4. Sinais e Interrupções

A arquitetura do processador também fornece o uso de sinais para *reset* e depuração do processador, bem como interrupções. As interrupções podem ser tratadas diretamente pelo processador ou estruturadas em um módulo de vetor de interrupções. Estes sinais não foram diretamente usados neste projeto.

1.5. Memória

O Nios II utiliza memória mapeada *Avalon-MM*. Tanto a memória de dados quanto periféricos de entrada e saída são endereçados pela porta mestre de dados. A memória é organizada no formato *little-endian*, onde os bytes mais significativos de uma palavra são gravados em endereços de memória maiores. Nas versões maiores do Nios II, a arquitetura também oferece cache e opções de desvio do cache para os barramentos de instrução e dados, assim como Unidade de Mapeamento de Memória (para virtualização), Unidade de Proteção de Memória e *Tightly-Coupled Memory* (para aumentar performance de memórias fora do chip).

1.6. JTAG

O módulo do JTAG é utilizado para programar e depurar o programa.

1.7. Configuração

Para configurar o Nios, foi utilizado o *software* Qsys, embarcado no *software* Quartus II, ambos distribuidos pela Altera. No arranjo deste projeto, foram utilizados cinco módulos, o *core* do Nios II, *On-chip RAM memory*, JTAG UART e dois periféricos de entrada e saída: botão e LEDs.

2. Arquitetura do Conjunto de Instruções

Todas as instruções do processador Nios II possuem um tamanho de 32 bits. As estruturas de algumas instruções se diferenciam das outras de acordo com os campos que são reservados por elas para determinados dados. O processador aqui especificado possui três tipos distintos de formato de instrução: tipo I, tipo R e tipo J. A estrutura específica a cada um dos formatos será explicitada nas subseções a seguir.

2.1. Tipo I

Instruções com formato do tipo I contêm um valor imediato embutido na palavra - para o Nios II, uma palavra é uma sequência de 32 bits - da instrução. Instruções do tipo I possuem:

- Um campo de 6 bits para código de operação (OP)
- Dois campos de 5 bits para registradores (A e B)
- Um campo de 16 bits para valor imediato (IMM16)

Na maioria dos casos, A e IMM16 representam operandos e B é o registrador destino, onde o resultado da operação será armazenado. Instruções do tipo I incluem operações aritméticas (addi, subi, muli[...]), lógicas (andi, andhi, xori, ori, orhi[...]), de desvio de execução (beq, bge, blt[...]), leitura e escrita em memória (ldb, stb [...]) e gerenciamento de cache (flushda, initda [...]).

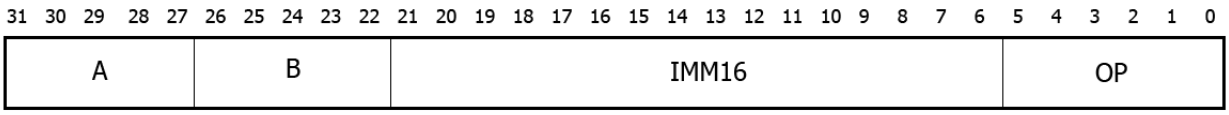

Figura 3.1: Representação gráfica da organização de campos em instruções do tipo I

2.2. Tipo R

Instruções com formato do tipo R possuem todos os argumentos e resultados armazenados em registradores. Instruções do tipo R contêm:

- Um campo de 6 bits para código de operação (OP)
- Três campos de 5 bits para registradores (A, B e C)
- Um campo de 11 bits para extensão de código de operação (OPX)

Algumas instruções do tipo R possuem um valor imediato embutido nos 5 bits de menor ordem de OPX. Bits não utilizados de OPX são sempre 0. Em geral, os registradores A e B são operandos e o resultado da operação é armazenado em C. O *opcode* para todas as instruções do tipo R é 0x3A, desse modo, as instruções se diferenciam umas das outras através do OPX.

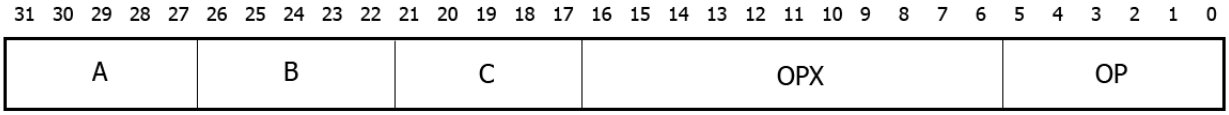

Figura 3.2: Representação gráfica da organização de campos em instruções do tipo R

2.3. Tipo J

As instruções com formato do tipo J (jmpi e call) transferem a execução do código para qualquer ponto dentro do alcance de 256MB. Instruções do tipo J possuem:

- Um campo de 6 bits para código de operação (OP)
- Um campo de 26 bits para valor imediato (IMM26)

3. Análise do Circuito

Após compilado no modo normal no software Altera Quartus II, foi gerado o seguinte relatório referente ao uso dos elementos da FPGA. O primeiro número indica a quantidade utilizada e o segundo número a quantidade disponível no dispositivo.

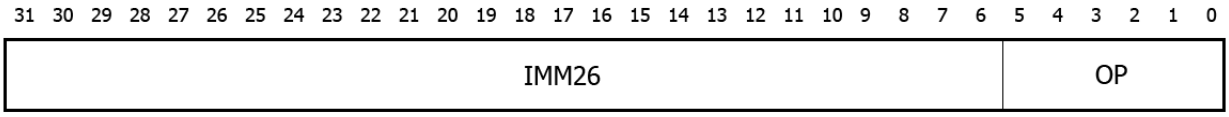

Figura 3.3: Representação gráfica da organização de campos em instruções do tipo J

Elemento	Uso
LEs	1.690/22.320 (8%) 1.457 modo normal. 138 modo aritmético.
LABs	124/1.395 (9%)
Registradores Lógicos	914 / 23.018 (4%)
M9Ks	9/66 (14%)
Total de blocos de memória implementados	82.944 / 608.256 (14 %)

O caminho crítico do circuito é 99,144 nano segundos.

4 | Algoritmo do Cálculo

Como mencionado anteriormente, o CRC-32 é um método de detecção de erros para identificar mudanças indesejadas em uma sequência de dados. Descrevendo o funcionamento deste método de forma breve, uma mensagem que passa por uma verificação de redundância cíclica CRC-32 é acrescida de um anexo de 32 bits com um valor de verificação baseado no resto da divisão polinomial da mensagem por um determinado polinômio. Na recepção da mensagem, o cálculo é refeito e o resultado comparado ao valor gerado anteriormente. Se estes não coincidirem, houve erro na transmissão dos dados.

Para implementar o algoritmo necessário ao cálculo do CRC, existem dois métodos distintos que podem ser utilizados: um deles faz uso de uma tabela de consulta que contém resultados pré-computados de operações de divisão polinomial de todos os possíveis conjuntos de bits em um byte (ou ainda em espaços maiores, como palavras). Para cada byte da mensagem, um valor correspondente é carregado da tabela, economizando tempo de cômputo, mas, por outro lado, ocupando uma maior porção da memória do dispositivo no qual o CRC será calculado. A outra possibilidade implica na realização do cálculo das divisões polinomiais em tempo de execução, realizando n divisões para uma mensagem de n bits. Para uma sequência de bits muito longa, este pode se tornar um processo custoso.

1. Algoritmo Implementado

A escolha feita na implementação concernente a este relatório foi pelo método da tabela de consulta, optando pela rapidez no processamento em detrimento da economia de espaço de memória. O código para geração da tabela e cálculo do CRC da mensagem de 1KB foi feito na linguagem Assembly para o processador Nios II.

Os parâmetros do CRC implementado seguem o padrão CRC32-MPEG2, com as seguintes características:

- Polinômio: 0x4C11DB7
- Valor Inicial: 0xFFFFFFFF
- Xor Final: 0x00000000
- Entrada e Saída não-refletidos

A mensagem a ser calculada foi gravada na memória do processador através do código do programa, com as diretivas de alocação de memória do Assembly .data e .byte. A tabela de consulta gerada possui 256 resultados, visto que esta é a quantidade

de combinações possíveis de bits em um byte. O resultado de uma operação xor do byte atual com o polinômio resulta em um valor de 32 bits, para cada resultado, uma palavra é gravada na memória.

Para computar o CRC da mensagem completa, um byte da mensagem é recuperado da memória por vez para um registrador e uma operação de xor é realizada com o primeiro byte do resto da divisão anterior (na figura 4.1, o resto da divisão é representado pelo registrador R2), o resultado desta operação permite encontrar o índice da tabela correspondente ao resultado da divisão do byte atual pelo polinômio. O valor é, então, carregado da tabela para um registrador. É realizada uma operação de xor entre o valor recuperado da tabela e o valor do resto da divisão anterior. Este loop se repete 1.000 vezes, já que a mensagem possui 1KB e as divisões serão realizadas de byte em byte.

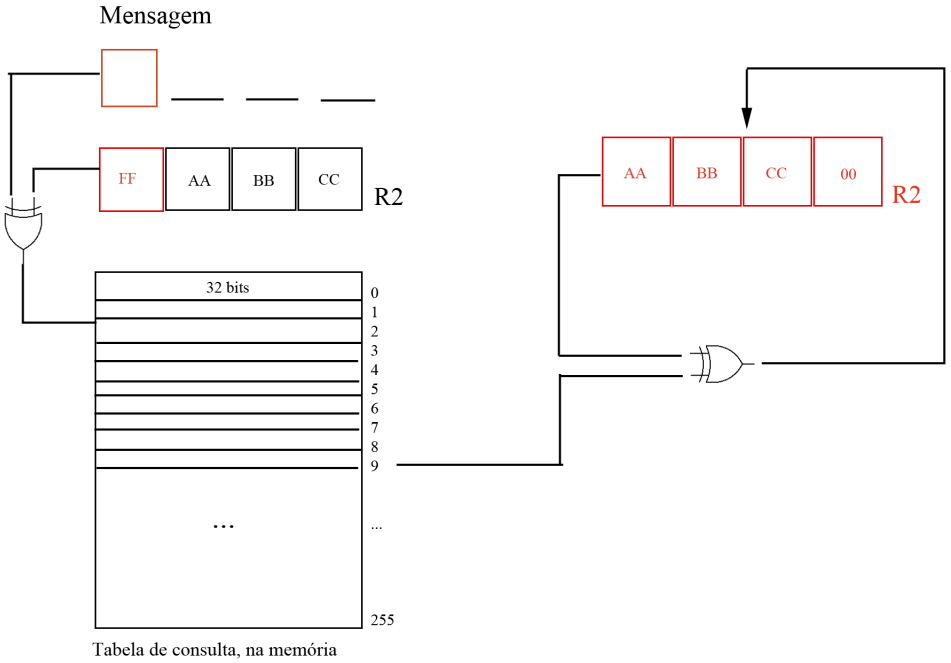

Figura 4.1: Diagrama representativo do cálculo do CRC-32 pela implementação da tabela de consulta

O throughput (quantidade de dados processados em um espaço de tempo) para o algoritmo implementado é de aproximadamente 411.384 bytes por segundo.

2. Testes

Para testes, foram utilizadas calculadoras online que contém o resultado de diversos CRCs. Inicialmente, foram inseridas mensagens identicas em calculadoras diversas para comparação e validação das calculadoras. Foram utilizadas as calculadoras presentes na subseção recursos.

5 | Recursos

Calculadoras acessadas em 03/05/2018 às 19:00.

http://www.sunshine2k.de/coding/javascript/crc/crc_js.html

http://crccalc.com/

Instruction Set Reference:

https://www.altera.com/en_US/pdfs/literature/hb/nios2/n2cpu_nii51017.pdf

Nios II Core Implementation Details:

https://www.altera.com/en_US/pdfs/literature/hb/nios2/n2cpu_nii51015.pdf

Nios II Classic Processor Reference Guide:

https://www.altera.com/en_US/pdfs/literature/hb/nios2/n2cpu_nii5v1.pdf