A fazer:

1. colocar imagens
2. escrever o nome dos registos correctamente
3. colocar maquina de estados da unidade de controlo de nível 0
4. explicar a comunicação com o computador
5. Colocar imagens utilizadas na demonstração
6. rever o texto e pô-lo mais bonito

talvez mais coisas mas agora não me lembro

Introdução

Este projeto tem como objetivo a criação de um circuito a ser implementado numa FPGA que faça algumas operações básicas de processamento de imagens. As imagens a serem processadas têm resolução máxima de 128x128 e são monocromáticas binárias (as únicas cores possíveis são branco ou preto). Pretende-se que este processamento seja o mais rápido possível, sem preocupação com eficiência energética.

Requisitos

O circuito deve ser capaz de realizar as seguintes operações morfológicas básicas sobre quaisquer imagens digitais binárias com dimensões de 128x128 pixéis:

* Erosão
* Dilatação
* Fecho
* Abertura
* Extração de Contornos

**Algoritmo**

Através da análise das cinco operações básicas a implementar, verifica-se que as operações de erosão e dilatação estão na base de todas as outras, por conseguinte o maior foco no desenvolvimento do dispositivo pretendido, centrou-se em volta do algoritmo capaz de resolver estas duas operações simples.

Estas duas operações distinguem-se pelo valor do pixel que têm em consideração, a dilatação considera os pixéis com valor 1 e a erosão tem em conta os pixéis com valor 0. Cada uma destas operações percorre todos os pixéis da imagem original avaliando uma sub-imagem com dimensões 3x3 bits com centro no pixel a ser processado. No caso de uma dilatação, o pixel central tem valor 1, na imagem resultante, caso um dos pixéis da sub-imagem tenha valor 1.

Desta forma, temos o seguinte algoritmo para a operação de dilatação:

**for** row **in** 0 **to** 127

**for** col **in** 0 **to** 127

**image(**row**,** col**)** **=** **image(**row**-**1**,** col**-**1**)** or i**mage(**row**-**1**,** col**)**

or **image(**row**-**1**,** col**+**1**)** or **image(**row**,** col**-**1**)**

or **image(**row**,** col**)** or **image(**row**,** col**+**1**)**

or **image(**row**+**1**,** col**-**1**)** or **image(**row**+**1**,** col**)**

or **image(**row**+**1**,** col**+**1**)**

**end** **for;**

**end** **for;**

A operação de erosão segue exatamente o mesmo algoritmo mas as portas lógicas OR utilizadas são substituídas por portas AND.

A imagem é processada linha a linha o que significa que, tendo em conta o algoritmo descrito anteriormente, para que essa mesma linha possa ser processada é necessário ter acesso à linha imediatamente acima e abaixo da mesma.

**Circuito**

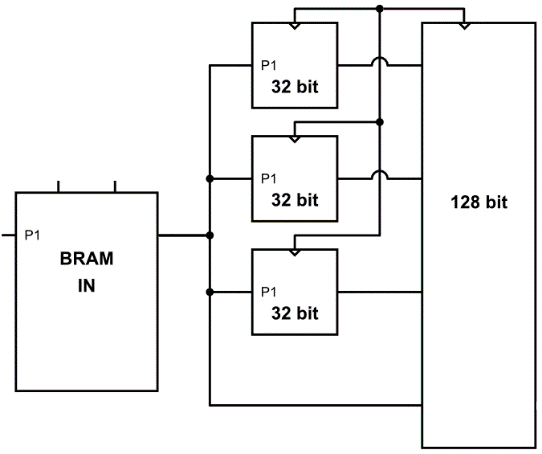
O circuito é composto por 3 partes distintas, a Unidade de Controlo, a *datapath* e as 2 memórias de leitura e escrita.

A imagem a ser processada encontra-se armazenada numa memória BRAM de leitura com um porto de comunicação com o dispositivo de 32 bits. Isto significa que para ler uma linha completa da imagem são necessários 4 ciclos de leitura da memória. Desta forma, na entrada da *datapath* encontra-se um bloco de 4 registos, utilizados para auxiliar a leitura de uma linha completa (128 bits) da seguinte forma:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Ciclos** | **R0 (32)** | **R1 (32)** | **R2 (32)** | **Rin (128)** | | | |
| **1** | 127-96 | - | - |  |  |  |  |
| **2** | 127-96 | 95-64 | - |  |  |  |  |
| **3** | 127-96 | 95-64 | 63-32 |  |  |  |  |
| **4** | 127-96 | 95-64 | 63-32 | 127-96 | 95-64 | 63-32 | 31-0 |

Os 3 primeiros blocos de 32 bits são armazenados nos registos R0, R1 e R2 pela ordem descrita na imagem acima. Após a leitura do 4º bloco, este é armazenado nos 32 bits menos significativos do registo Rin e os conteúdos dos 3 registos anteriores são carregados, pela mesma ordem, para os bits mais significativos do registo de 128 bits. Esta arquitetura permite que sejam feitas leituras concorrentes da memória sem que a linha armazenada em Rin seja alterada durante 3 ciclos de leitura.

Colocar aqui uma imagem melhor sem ligacoes de clock



***Datapath***

A datapath tem como entrada o registo Rin e é responsável pelo processamento propriamente dito da imagem, sendo que este é feito linha a linha da imagem. Tendo isso em conta e uma vez que para processar uma linha são necessárias: a linha acima e a linha abaixo da mesma é necessário ter a três linhas disponíveis em cada ciclo de processamento. Para resolver este problema utilizou-se 3 registos de 128 bits ligados à entrada da *datapath*, responsáveis por armazenar, respetivamente, as linhas anterior, atual e a linha a seguir à linha a ser processada. A cada atualização do registo Rin a nova linha é carregada para o registo Ri+1 e o conteúdo atual de cada um dos 3 registos é transferido para o registo seguinte.

Imagem com os registos de shift antes da logica

Durante a leitura da primeira linha da imagem os registos Ri-1, Ri e Ri+1 são carregados com o valor neutro da operação correspondente, ou seja, no caso da dilatação são carregados com o valor 0 e no caso da erosão com o valor 1, de forma a tornar possível o processamento dos extremos da imagem.

Abaixo encontra-se uma demonstração do armazenamento das linhas nos 3 registos de 128 bits, de uma imagem com 4 linhas de 128 bits dividia em blocos de 32 bits.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | Imagem | | | | | | Linha 0 | 32bits |  |  |  | | Linha 1 |  |  |  |  | | Linha 2 |  |  |  |  | | Linha 3 |  |  |  |  | | | | |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Processamento Linha 0** | | | | | | Ri-1 | neutro | | | | | Ri |  |  |  |  | | Ri+1 |  |  |  |  | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Processamento Linha 1** | | | | | | Ri-1 |  |  |  |  | | Ri |  |  |  |  | | Ri+1 |  |  |  |  | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Processamento Linha 2** | | | | | | Ri-1 |  |  |  |  | | Ri |  |  |  |  | | Ri+1 |  |  |  |  | | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **Processamento Linha 3** | | | | | | Ri-1 |  |  |  |  | | Ri |  |  |  |  | | Ri+1 | neutro | | | | |

As operações são realizadas pelo bloco de lógica à saída dos 3 registos anteriores. Este bloco é composto pela lógica necessária para obter a linha (128 bits) resultante para cada uma das duas operações simples, sendo que estas diferem apenas na porta lógica utilizada: para obter o resultado de uma erosão utilizam-se portas lógicas AND e para obter o resultado de uma dilatação utilizam-se portas lógicas OR. À saída deste bloco um multiplexer permite selecionar qual dos resultados se pretende colocar no registo de saída.

imagem com o bloco de logica, o multiplexer de seleção da saida e o registo de resultado

O resultado de cada linha, armazenado no registo de saída, é carregado para a memória de escrita em blocos de 32 bits começando pelos blocos mais significativos. Esta seleção é feita com recurso a um multiplexer de 4-1 cujos bits de seleção são dados pelos 2 bits menos significativos do sinal de endereçamento da memória de escrita no ciclo correspondente.

Abaixo encontra-se o circuito completo que permite executar as duas operações binárias simples:

1. Imagem com o circuito sem a memoria adicional e sem xor

**Unidade de Controlo**

A outra componente do circuito é a Unidade de Controlo, responsável pela gestão de todos os recursos utilizados no circuito. Esta define quais os registos ativados em cada ciclo, os sinais de seleção dos multiplexers, o sinal *write enable* da memória de escrita e o endereçamento das memórias de leitura e de escrita.

O endereçamento das memórias é realizado através de um único contador, cuja saída endereça diretamente a memória de leitura, o endereçamento da memória de escrita é realizado com recurso a X *shift-registers* que produzem um atraso de X ciclos no endereço do contador permitindo assim, endereçar as duas memórias com apenas um contador.

Desta forma, tem-se os seguintes bits de estado:

|  |  |
| --- | --- |
| **Sinais de Estado** | **Descrição** |
| Addr | Endereço das memorias |
| Rread\_en | Enables dos registos de leitura |
| Mux1\_select | Select do multiplexer que permite carregar os registos do valor neutro |
| Rline\_en | Enables dos registos que armazenam as linhas |
| Oper | Sinal que identifica a operação |
| Mux2\_select | Select do multiplexer que seleciona o resultado a armazenar |
| Rout\_en | Enable do registo de saída |
| Write\_en | Write enable da memória de escrita |

Entre a Unidade de Controlo e a *datapath* colocou-se um registo de controlo que permite sincronizar a leitura da memória e as operações realizadas pela *datapath* bem como melhorar a frequência de execução do dispositivo.

Abaixo encontra-se o esquema da máquina de estados utilizada na Unidade de Controlo:

Imagem da maquina de estados de nível 0

**Operações Compostas**

* Fecho
* Abertura

As operações de fecho e abertura são operações compostas das duas anteriores. Uma operação de fecho executa uma dilatação seguida de uma erosão e uma abertura executa as mesmas duas operações pela ordem inversa.

Para que seja possível executar duas operações simples é necessário que o resultado da primeira operação seja armazenado para que posteriormente possa ser consultado durante a operação seguinte. Para esse efeito recorreu-se a uma memória auxiliar, onde o resultado intermédio é guardado e de onde esse mesmo resultado é lido durante a segunda operação. Por esta razão é necessário introduzir-se um multiplexer na entrada dos registo de leitura que permite selecionar se a imagem a ser processada tem origem na memória de leitura ou na memória auxiliar. Utiliza-se assim 3 memórias BRAM, a memória de leitura, a memória de escrita e a memória auxiliar, o que está dentro dos recursos disponíveis na FPGA utilizada no laboratório (basys2 / spartan 3e?).

Uma solução para não utilizar uma memória auxiliar para fazer este processamento seria ter o seguinte circuito:

Imagem com o circuito com dois blocos de registos shift

Este circuito aplica exatamente o mesmo conceito utilizado anteriormente, tendo agora dois níveis de processamento, em que o primeiro processa a primeira operação e o segundo processa a segunda operação com base no resultado da operação anterior. Esta solução requer aproximadamente o dobro das portas lógicas da solução implementada, mas apresenta melhor performance uma vez que não é necessário ler e escrever numa memória auxiliar. Não tenho a certeza se a performance é assim tao melhor. Por outro lado, o uso de uma memória auxiliar torna a implementação mais simples, reduz a área utilizada e permite uma maior flexibilidade e expansibilidade do circuito, pois permite adicionar novas operações com alterações mínimas na arquitetura do dispositivo e sem ter de duplicar a área utilizada para executar as operações.

Para controlar a ordem das operações e determinar a partir de qual das duas memórias, memória de leitura ou memória auxiliar, é lida a imagem a ser processada pelo dispositivo em cada ciclo de execução utilizou-se uma Unidade de Controlo adicional permitindo assim fazer uso do circuito anteriormente utilizado para as operações simples.

Obteve-se então o seguinte circuito que permite tanto obter resultados das duas operações simples bem como das duas operações compostas.

Colocar imagem do circuito com as 3 memorias mas ainda sem a xor

**Extração de Contornos**

Esta operação resume-se a executar uma operação de erosão sobre a imagem e no final comparar esse resultado com a imagem original, extraindo a diferença entre as duas.

A diferença entre dois bits em que o *carry* é sempre 0 é dada pela função lógica XOR. Fazendo uso desta propriedade colocou-se uma porta XOR à saída do bloco de processamento cujas entradas são o sinal da linha original e o sinal linha processada. De forma integrar o esquema anterior no circuito já implementado recorreu-se a um multiplexer que permite selecionar se o resultado que se pretende transferir para a saída, corresponde ao resultado de uma operação ou a diferença entre a linha processada e a linha original. Atenção que não é necessário recorrer à memória auxiliar para executar esta operação, logo pode ser executa em apenas um ciclo de execução.

Obtém-se assim, o seguinte circuito que permite obter a diferença entre cada bit da imagem original e da imagem resultante:

Imagem do circuito com foco apenas na parte da xor

Abaixo está demonstrado o circuito final com todas as operações implementadas:

Imagem do circuito final

**Comunicação Entre Computador e Dispositivo**

As imagens que se pretende processar são carregadas para o dispositivo através do porto de escrita A da memória de leitura e após serem processadas são exportadas pelo porto de leitura A da memória de escrita. Ambos estes portos são portos de 8 bits o que significa que são escritos 8 bits de cada vez na memória de leitura e lidos 8 bits de cada vez da memória de escrita. Por outro lado a comunicação entre o dispositivo e as memórias é feito por portos de 32 bits …………

explicar a comunicação com o computador

**Interface de interação com o dispositivo**

O dispositivo é controlado através de 2 botões de pressão e dois interruptores. Os botões de pressão servem para reiniciar (botão 0) o dispositivo e iniciar o processamento (botão 1), cada vez que uma imagem acaba de ser processada o circuito deve ser reinicializado antes de se iniciar o processamento seguinte. Os interruptores permitem selecionar a operação a realizar segundo a seguinte tabela:

|  |  |  |  |
| --- | --- | --- | --- |
| **Interruptor 2** | **Interruptor 1** | **Interruptor 0** | **Operação** |
| 0 | 0 | 0 | Erosão |
| 0 | 0 | 1 | Dilatação |
| 0 | 1 | 0 | Abertura |
| 0 | 1 | 1 | Fecho |
| 1 | 0 | 0 | Contornos |
| Outros | | | Não definido |

**Falar da frequência de operação**

**Resultados**

**Colocar imagens utilizadas na demonstração**