PROCESSADOR DIGITAL DE IMAGEM

Universidade Federal do Ceará - Campus Quixadá

Jonas Bezerra da Costa Máximo[1]

Resumo: Foi utilizado a linguagem de descrição de hardware VHDL para implementar o processador digital de imagem na qual seria executado na zibo

Palavras-chave: vhdl, processador e imagem.

1 BINARIZAÇÃO

O código referente a binarização foi 0001. Na binarização foi implementado apartir do filtro tons de cinza, na qual se o pixel vermelho($s\_shades\_of\_gray(23\ downto\ 16)$) tivesse Threshold maior que 100(x"64") e verde($s\_shades\_of\_gray(15\ downto\ 8)$) tivesse Threshold maior que100(x"64") e azul($s\_shades\_of\_gray(7\ downto\ 0)$) tivesse Threshold maior que 100(x"64") ele receberia preto(x"000000") se não ele receberia branco(x"FFFFFF")

2 TONS DE CINZA

O código referente a binarização foi 0010. Nos tons de cinza foi implementado aplicando multiplicando a cor vermelha($s\_mem(23\ downto\ 16)$) da imagem por 40(X"28") e deslocado 7 bits verde($s\_mem\_din(15\ downto\ 8)$) da imagem por 74(X"a4") e deslocado 7 bits azul($s\_mem\_din(7\ downto\ 0)$) da imagem por 14(X"0e") e deslocado 7 bits

3 ROTAÇÃO

O código referente a binarização foi 0100. Foram criados 4 sinais *rot*_0, *rot*_90, *rot*_180, *rot*_270, referentes a quais os graus de rotação da iamgem e um sinal *rot_type* para selecionar qual tipo de rotação fazer, *s_i* e *s_j* para controlar qaunto cada lado da imagem deve rotacionar, esses dados são enviados ao loop (s_exec_for) foi criado para mover os bits, foi criado um *clk_count* para girar a cada 1 segundo enquanto o *clk_count* for menor que 100000000 está girando a *rot*_90 quando estiver entre 100000000 e 100050000 estará na *rot*_180 e assim por diante.

4 NEGATIVO

O código referente ao negativo foi 0101. No filtro negativo foi implementado subtraindo da constante x"FF"a cor vermelha($s\_mem(23\ downto\ 16)$)da imagem e concatenada com a

[1] UFC, jonas.bezzerra@gmail.com

subtração da constante x"FF"a cor verde($s\_mem\_din(15\ downto\ 8)$) da imagem concatenada com a subtração da constante x"FF"a cor azul($s\_mem\_din(7\ downto\ 0)$) da imagem

5 AJUSTE DE BRILHO (FADING)

O código referente ao negativo foi 0111. Foi criado um contador para o brilho ser diminuído e aumentado gradativamente(*contador_de_pulsos_brilho*) com ciclos de 5000000, foi criado o sinal *s_brilho_op* que controla se o brolho deve aumentar ou diminuir o $brilho2$ que e uma constante que indica se o brilho deve ser diminuído $brilho2 > 0$ ou $brilho2 < 128$ deve ser aumentado, esses sinais são controlados pelas flags *s_done_aumenta_brilho* e *s_done_baixa_brilho*

6 SAL E PIMENTA

O código referente ao sal e pimenta foi 1000. foram criados 2 registradores um para armazenar o número do contador e outro para armazenar o número aleatório, foi criada a ($flag\_salt\_chili$) de 2 bits para enviar ao datapath os sinais 00 que servem para setar a imagem normal, o estado 01 para colocar o bit branco e o estado 10 para colocar o bit preto e foi criado o mux (*mux_salt_chili*) para chavear a entrada no (*s_mem_addr*), Foram criados 3 estados *salt_chili* no qual eu jogo a imagem original na ram o próximo estado será *state_salt* desabilito *s_exec_start* para não sobrepor os bits que serão colocados, gero uma constante aleatória $q_r andom * const_3 31$ e salvo na entrada do registrador *d_random* e ssp para o próximo estado que será *state_chi* faço as mesmas coisas que o estado anterior só que agora incremento o contador enquanto ele for menor que 100 volto para o estado de *state_saltse* não envai para o init.

7 REDIMENSIONAMENTO

O código referente ao redimensionamento foi 1001. No redimensionar foi criado um loop(s_exec_loop) para percorrer os pixels da imagem pulando de 2 em 2 os pixels da linha e da coluna e os redimensionando na memória ram dividindo o endereço deles por 2, foram criados 2 estados um para redimensionar(*remdimantion*) e ou outro para limpar o lixo deixado pela imagem original(*black*) que pinta os pixels de preto ele pinta de preto enviando o bit(op_red) em zero para o datapath chaveando com o *s_mem_din* que e o bit redimensionado

8 RGB

O código referente ao RGB foi 1010. Foi criado um contador(*contador*) para calcular o tempo estimado de números de clock para a imagem permanecer 1 segundo em cada cor, foi criado um sinal op_rgb que e um sinal que envia ao datapath a cor que deve ser colocado 00 vermelho, 01 verde, 10 azul, 11 imagem normal. foram criados 4 estado um para cada filtro de cor e um para a imagem normal, nos estados dos filtros habilita o contador(*habilitar_contador*)

seta o *op_rgb* trata a interrupção do botão caso aconteça para ir para a imagem normal, após o termino do contador que e de 100000000 ele vai pra o próximo estado de filtro.