

UNIVERSIDADE DE SÃO PAULO
INSTITUTO DE CIÊNCIAS MATEMÁTICAS E DE COMPUTAÇÃO
SSC0740 – SISTEMAS EMBARCADOS

Trabalho 1 - Redução de Imagem

Leonardo Alves Paiva	10276911
Lucas Fernandes da Nobrega	9805320
João Pedro Doimo Torrezan	9806933

São Carlos
2020

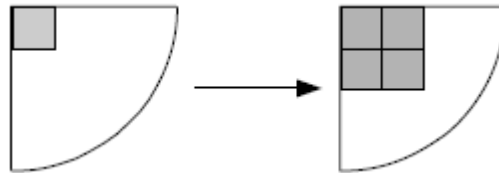
Sumário

Introdução	3
Objetivo	3
Desenvolvimento	4
Ambiente	4
Programas auxiliares	4
Simulação	6
Exemplos	7
Síntese	8
Minimização dos recursos de hardware	9
Maximização throughput	10
Referências	10

Introdução

O processo de ampliação e redução de imagens (também conhecidas como *zoom in* e *zoom out*, respectivamente) aumentam ou diminuem as dimensões de uma imagem para visualização. O jeito mais simples para ampliar uma imagem é duplicar os pixels nas direções horizontal e vertical da imagem, como mostrado na figura 1.

Figura 1 – Ampliação de imagem por fator 2



Fonte: MARQUES FILHO & VIEIRA NETO, 1999.

Para o processo de redução, basta aplicar o processo inverso da ampliação, isto é, reduzir quatro pixels para 1. Nesse caso, é necessário tirar a média dos quatro pixels originais, a fim de minimizar a perda de informação.

Objetivo

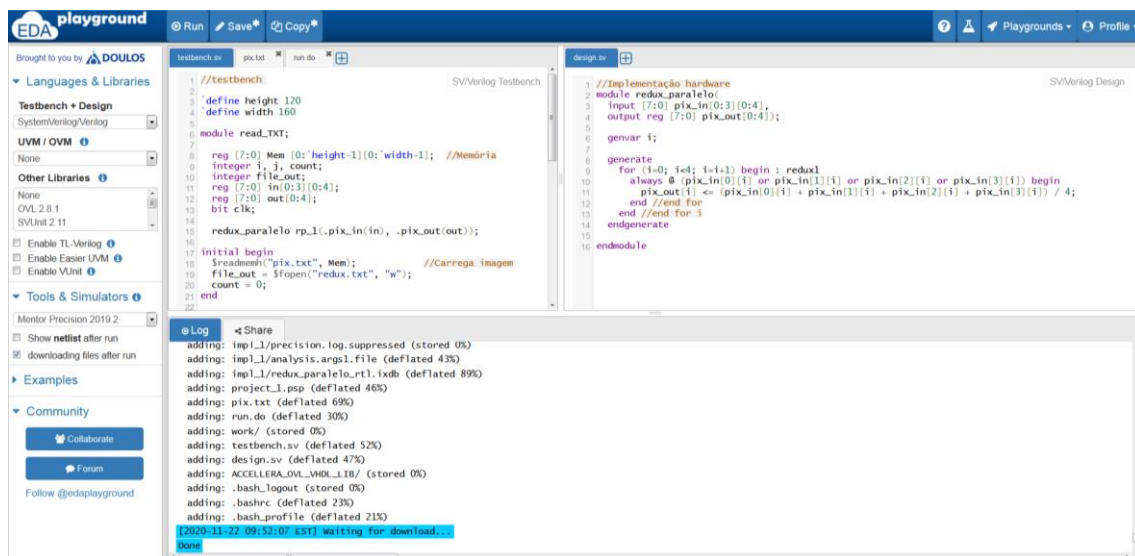
Este trabalho visa implementar o processo de redução de imagem em hardware através da linguagem de descrição *SystemVerilog/Verilog* de dois modos diferentes: um que minimize o uso de recursos de hardware e outro que maximize o *throughput*.

Desenvolvimento

Ambiente

Como ambiente de desenvolvimento foi usado o software online EDA Playground (<https://www.edaplayground.com>), exibido na figura 2. Como se pode ver, o ambiente é dividido em campos com funções específicas. A barra lateral esquerda serve para configurar o ambiente: linguagem, bibliotecas, simuladores, sintetizadores, entre outros. O campo inferior mostra as saídas, como *logs* da simulação/síntese. No centro, há dois campos para código: o esquerdo é reservado para os arquivos de *testbench*, e o direito para os arquivos de implementação do hardware.

Figura 2 – Ambiente EDA Playground



Programas auxiliares

Como o ambiente EDA Playground possui limitações, foi necessário o uso de programas auxiliares para montagem e desmontagem das imagens. O programa **Pix** é responsável por ler uma imagem *bitmap* monocromática e criar o arquivo de texto *pix.txt* com os valores dos pixels em hexadecimal. Este arquivo servirá de entrada para o *testbench*. Depois de compilar, basta chamar por: **Pix.exe nome_bitmap.bmp**. O arquivo *nome_bitmap.bmp* é a imagem *bitmap* que se deseja extrair os valores.

Após simular os códigos HDL implementados, os valores dos pixels da imagem reduzida são escritos em um arquivo texto *redux.txt* em hexadecimal. Então para reconstruir a imagem, é necessário aplicar o programa **Repix**, cuja função é construir um *bitmap* com os novos valores. Depois de compilar o programa, basta chamar por: **Repix.exe img_original.bmp redux.txt fator**. O arquivo *img_original.bmp* é a imagem *bitmap* da qual se extraiu os valores com o programa **Pix**. O arquivo *redux.txt* contém os valores hexadecimais da imagem reduzida. E *fator* é o fator de redução, no caso será 0.5 sempre.

Na figura 3, está um exemplo de execução do programa *Pix.exe*, no qual são impressos alguns parâmetros da imagem.

Figura 3 – Execução *Pix.exe*

```
C:\Users\Leonardo\Downloads\Universidade\Sistemas Embarcados>Pix.exe u4.bmp
Tamanho: 20346
Offset dados bitmap: 1146
Comprimento: 160
Altura: 120
Size image: 19200
No. de cores usadas: 256
No. de bits por pixel: 8
Compressao: 0
```

Na figura 4, está um exemplo de execução do programa *Repix.exe*, no qual são impressos alguns parâmetros da imagem original e da imagem reduzida formada agora.

Figura 4 – Execução *Repix.exe*

```
C:\Users\Leonardo\Downloads\Universidade\Sistemas Embarcados>Repix.exe u4.bmp re
dux.txt 0.5
Tamanho: 20346
Offset dados bitmap: 1146
Comprimento: 160
Altura: 120
Size image: 19200
No. de cores usadas: 256
No. de bits por pixel: 8
Compressao: 0

Tamanho: 5946
Offset dados bitmap: 1146
Comprimento: 80
Altura: 60
Size image: 4800
No. de cores usadas: 256
No. de bits por pixel: 8
Compressao: 0
```

Simulação

Para a simulação, foi utilizada a ferramenta *Icarus Verilog 0.10.0 11/23/14*. Além de os arquivos de *testbench* e de *design* devidamente carregados no EDA Playground (figura 5), também é necessário carregar a entrada no formato texto como apresentado na figura 6. Também é importante marcar a opção “*download files after run*” no campo lateral esquerdo, para que o arquivo de saída *redux.txt* seja baixado ao final da simulação, como mostrado na figura 7.

Figura 5 – Tela de simulação

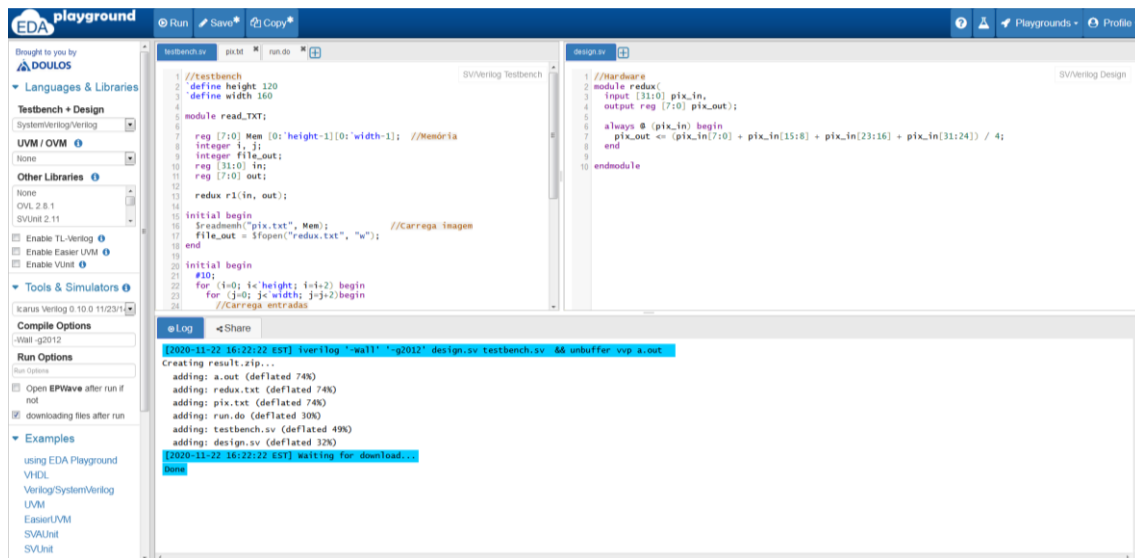


Figura 6 – Arquivo *pix.txt*

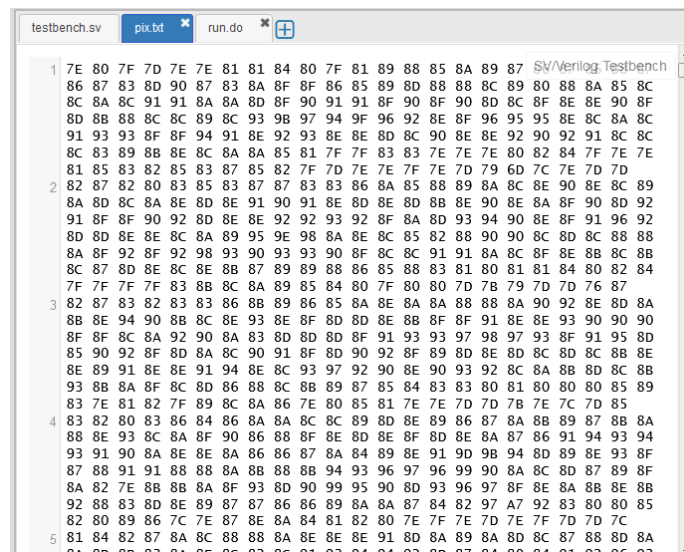
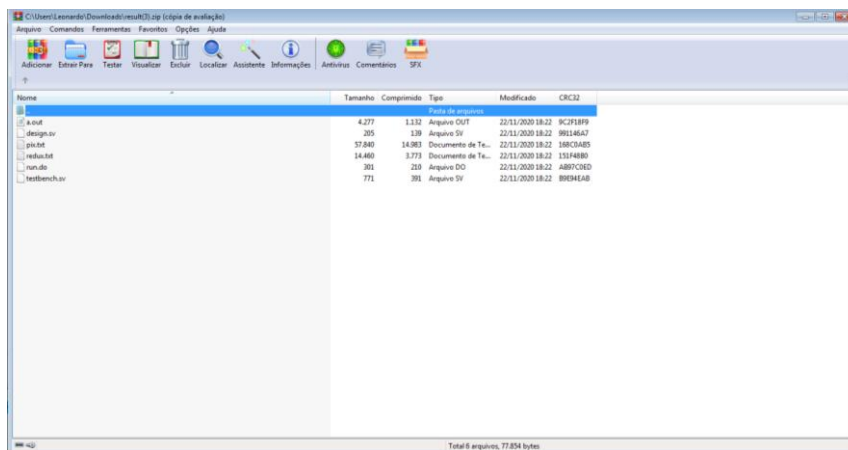


Figura 7 – Saída da simulação



Nome	Tamanho	Comprimido	Tipo	Modificado	CRC32
...	4.277	1.132	Arquivo ZIP	22/11/2020 18:22	9C2F38F9
...	205	139	Arquivo SV	22/11/2020 18:22	9B1146A7
...	57.840	14.983	Documento de Te...	22/11/2020 18:22	188C0A85
...	14.460	3.773	Documento de Te...	22/11/2020 18:22	121F4680
...	301	210	Arquivo DO	22/11/2020 18:22	A8B7C0ED
...	771	391	Arquivo SV	22/11/2020 18:22	B0E4E4AB

Total 6 arquivos, 77.854 bytes

Exemplos

Nas figuras 8, 9 e 10 são apresentados exemplos de imagens que sofreram redução. As imagens originais estão à esquerda, e as reduzidas, após a reconstrução feita pelo programa *Repix*, estão à direita.

Figura 8 – Exemplo 1



Figura 9 – Exemplo 2



Figura 10 – Exemplo 3



Síntese

Para a síntese, foi utilizada a ferramenta *Mentor Precision 2019.2*. O arquivo de *design* deve estar devidamente carregado no EDA Playground (figura 11), sendo também necessário carregar o *script run.do* como apresentado na figura 12. Os *logs* do processo de síntese são exibidos no campo inferior do EDA Playground. Ao marcar a opção “*download files after run*” no campo esquerdo, os arquivos de saída da síntese podem ser baixados, como mostrado na figura 13.

Figura 11 – Tela de síntese

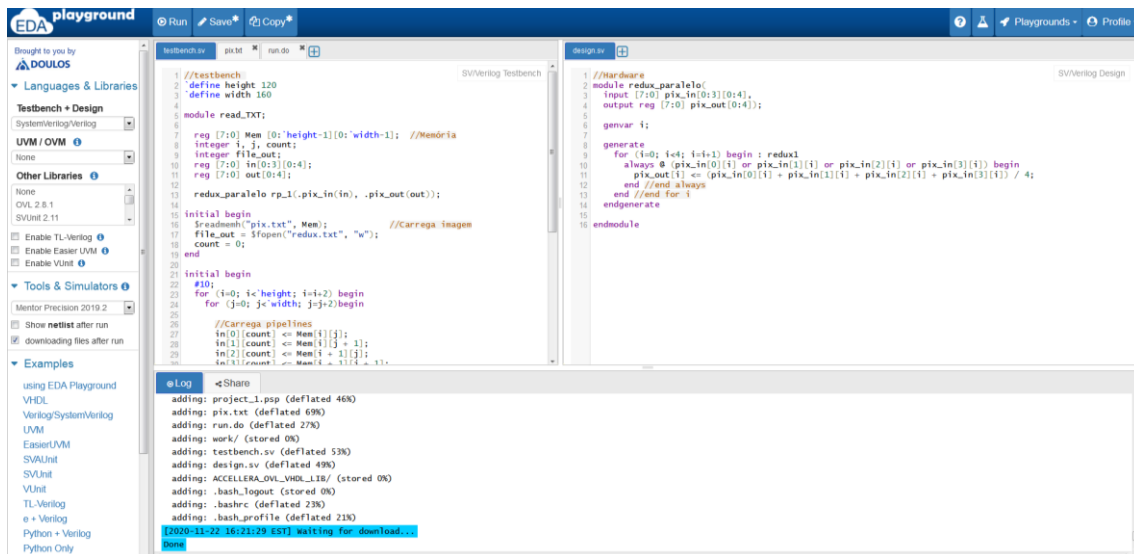


Figura 12 – Arquivo *run.do*

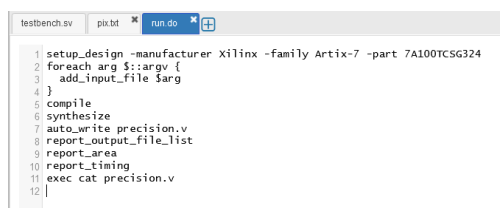
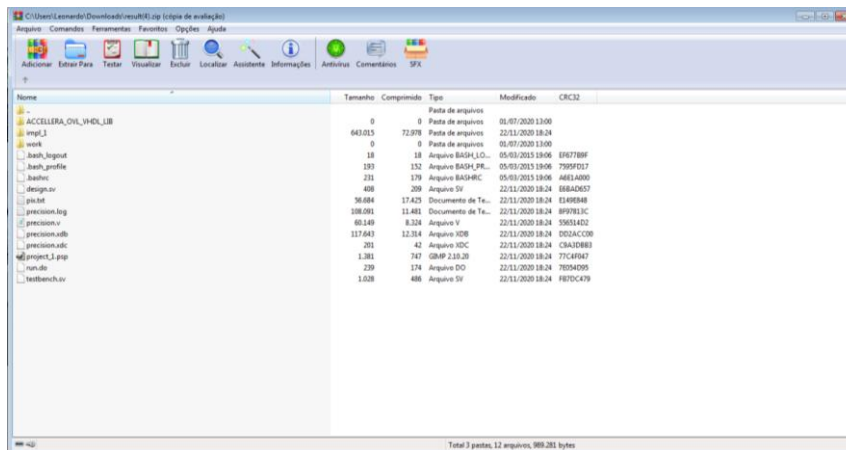


Figura 13 – Saída da síntese



Minimização dos recursos de hardware

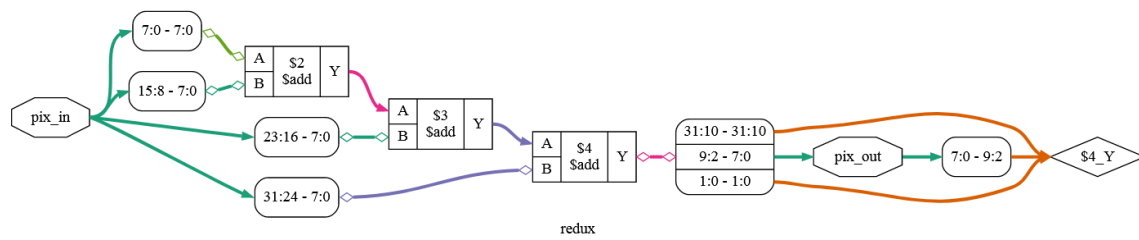
Da saída da síntese vale destacar a tabela com os recursos utilizados. Na figura 14, são exibidos os recursos usados do dispositivo 7A100TCSG324 da família Artix-7 da Xilinx, no qual se procurou minimizar o *hardware*.

Figura 14 – Recursos utilizados visando sua minimização

```
# Info: *****
# Info: Device Utilization for 7A100TCSG324
# Info: *****
# Info: Resource                Used      Avail    Utilization
# Info: -----
# Info: IOs                     40        210      19.05%
# Info: Global Buffers           0         32        0.00%
# Info: LUTs                     27      63400      0.04%
# Info: CLB Slices               0      15850      0.00%
# Info: Dffs or Latches          0     126800      0.00%
# Info: Block RAMs               0        135      0.00%
# Info: DSP48E1s                 0         240      0.00%
# Info: -----
# Info: *****
# Info: Library: work    Cell: redux    View: INTERFACE
# Info: *****
# Info: Number of ports :                      40
# Info: Number of nets :                      136
# Info: Number of instances :                  104
# Info: Number of references to this view :      0
# Info: Total accumulated area :
# Info: Number of LUTs :                      27
# Info: Number of LUTs with LUTNM/HLUTNM :    16
# Info: Number of MUX CARRYs :                17
# Info: Number of accumulated instances :      104
# Info: *****
```

Um esquemático do *hardware* também pode ser impresso com uso da ferramenta *Yosys 0.9.0*, como se pode ver na figura 15.

Figura 15 – Esquemático do hardware



Maximização throughput

Procurando maximizar o *throughput* e levando em consideração o dispositivo anterior, foi criado um paralelismo aplicando 5 elementos do hardware mínimo. Esta quantidade foi escolhida devido à existência de portas IO limitadas (210 no total). Os recursos utilizados podem ser vistos na figura 16.

Figura 16 – Recursos utilizados visando o máximo throughput

```
# Info: *****
# Info: Device Utilization for 7A100TCSG324
# Info: *****
# Info: Resource                Used    Avail    Utilization
# Info: -----
# Info: I0s                     200     210     95.24%
# Info: Global Buffers          0        32      0.00%
# Info: LUTs                    108    63400    0.17%
# Info: CLB Slices              0    15850    0.00%
# Info: Dffs or Latches         0   126800    0.00%
# Info: Block RAMs              0       135    0.00%
# Info: DSP48E1s                0       240    0.00%
# Info: -----
# Info: *****
# Info: Library: work    Cell: redux_paralelo    View: INTERFACE
# Info: *****
# Info: Number of ports :                200
# Info: Number of nets :                 546
# Info: Number of instances :            418
# Info: Number of references to this view :      0
# Info: Total accumulated area :
# Info: Number of LUTs :                108
# Info: Number of LUTs with LUTNM/HLUTNM :      64
# Info: Number of MUX CARRYs :          68
# Info: Number of accumulated instances :      418
# Info: *****
```

Referências

MARQUES FILHO, Ogê; VIEIRA NETO, Hugo. Processamento Digital de Imagens, Rio de Janeiro: Brasport, 1999. ISBN 8574520098.