Digital System Design - PA12

Image Convolution using VHDL

Deskripsi Proyek

Latar Belakang

- Image processing adalah sebuah operasi yang sangat intensif dalam hal komputasi-nya.
- Hardware acceleration dengan FPGA memberikan efisiensi yang lebih tinggi dibandingkan perangkat lunak beserta CPU biasa.

Tujuan Proyek

- Mengembangkan sistem *image convolution* berbasis FPGA.
- Mengintegrasikan ALU (Arithmetic Logic Unit) dan MMU (Matrix Multiplier Unit) untuk mendukung berbagai algoritma dalam melakukan *convolution*.
- Mendukung pemilihan kernel secara dinamis melalui instruksi yang berbeda.

Dasar Teori

Convolution

- Operasi matematika untuk memfilter atau memodifikasi gambar. Operasi ini berguna untuk menggabungkan hasil dari dua fungsi.
- Rumus dasar *convolution* sebagai berikut.

$$F(x,y) = \sum_{i}\sum_{j} f(x+i,\ y+j)\, h(i,j)$$

Kernel

- Matriks 3x3 atau 5x5 yang berfungsi untuk memodifikasi pixel sehingga menghasilkan *Feature Map*.
- Contoh: Box Blur, Sobel, Sharpening, Laplacian, dll.

Dasar Teori

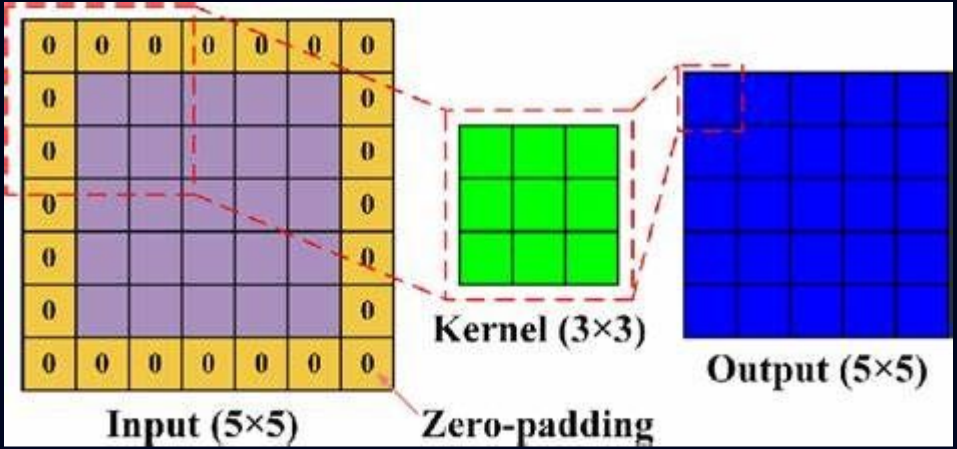
Jenis Algoritma Convolution

- **Direct Convolution** : Metode dasar untuk melakukan operasi *convolution*, di mana setiap elemen dalam filter (kernel) digeser melintasi elemen input (pixel pada gambar) dan dihitung satu per satu.
- **Im2col Convolution** : Metode ini mengubah input menjadi representasi matriks (disebut im2col, dari *image to column*) sebelum operasi *convolution* dilakukan. Lalu, menggunakan operasi matriks untuk menghitung hasil *convolution* secara bersamaan.

Dasar Teori

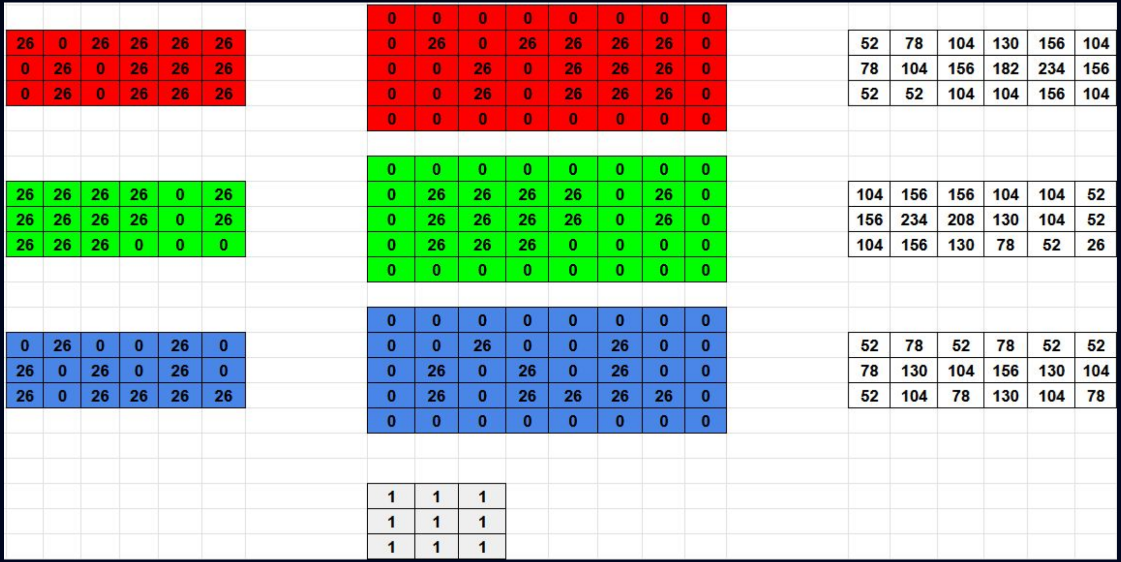
Zero Padding

Teknik dalam image processing di mana nilai nol ditambahkan di sekitar tepi data, seperti gambar. Teknik ini dilakukan untuk mengontrol dimensi output.



Dasar Teori

Direct Convolution



26	0	26	26	26	26
0	26	0	26	26	26
0	26	0	26	26	26

0	0	0	0	0	0	0	0
0	26	0	26	26	26	26	0
0	0	26	0	26	26	26	0
0	0	26	0	26	26	26	0
0	0	0	0	0	0	0	0

52	78	104	130	156	104
78	104	156	182	234	156
52	52	104	104	156	104

26	26	26	26	0	26
26	26	26	26	0	26
26	26	26	0	0	0

0	0	0	0	0	0	0	0
0	26	26	26	26	0	26	0
0	26	26	26	26	0	26	0
0	26	26	26	0	0	0	0
0	0	0	0	0	0	0	0

104	156	156	104	104	52
156	234	208	130	104	52
104	156	130	78	52	26

0	26	0	0	26	0
26	0	26	0	26	0
26	0	26	26	26	26

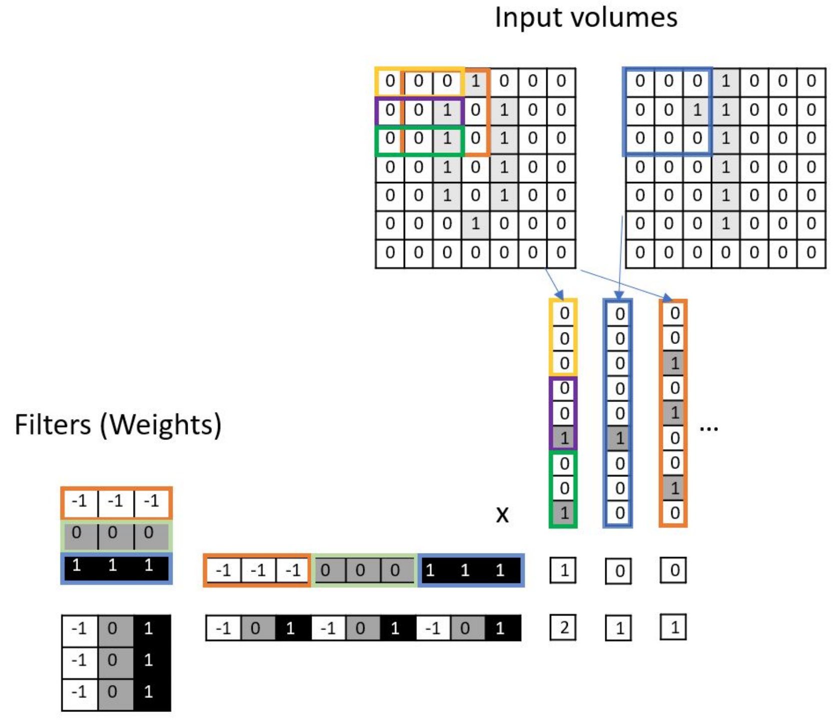
0	0	0	0	0	0	0	0
0	0	26	0	0	26	0	0
0	26	0	26	0	26	0	0
0	26	0	26	26	26	26	0
0	0	0	0	0	0	0	0

52	78	52	78	52	52
78	130	104	156	130	104
52	104	78	130	104	78

1	1	1
1	1	1
1	1	1

Dasar Teori

Im2col Convolution



0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	0	1	0	1	0	0
0	0	1	0	1	0	0
0	0	1	0	1	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

0	0	0	1	0	0	0
0	0	1	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

0	0	0
0	0	0
0	0	1
0	0	0
0	0	1
1	1	0
0	0	0
0	0	1
1	0	0

-1	-1	-1
0	0	0
1	1	1

-1	-1	-1	0	0	0	1	1	1	1	0	0

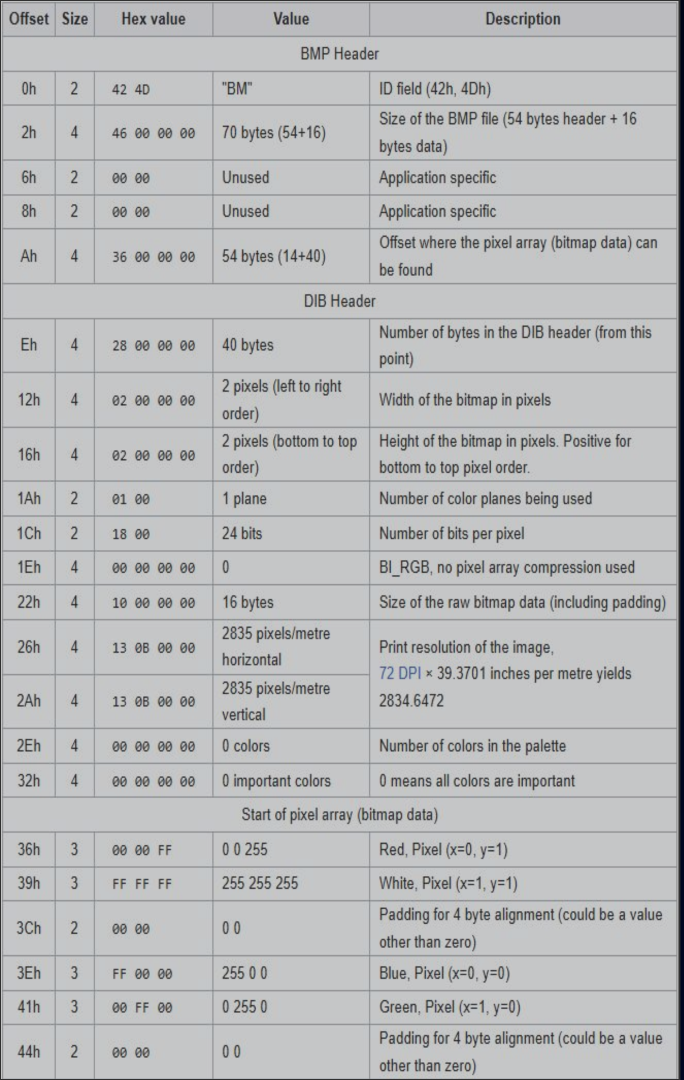
-1	0	1
-1	0	1
-1	0	1

-1	0	1	-1	0	1	-1	0	1	2	1	1

Dasar Teori

BITMAP (BMP) Image

Dalam proyek ini, gambar menggunakan format BMP karena tidak perlu ada langkah dekompresi dan data pixel dapat langsung diakses dengan mudah.



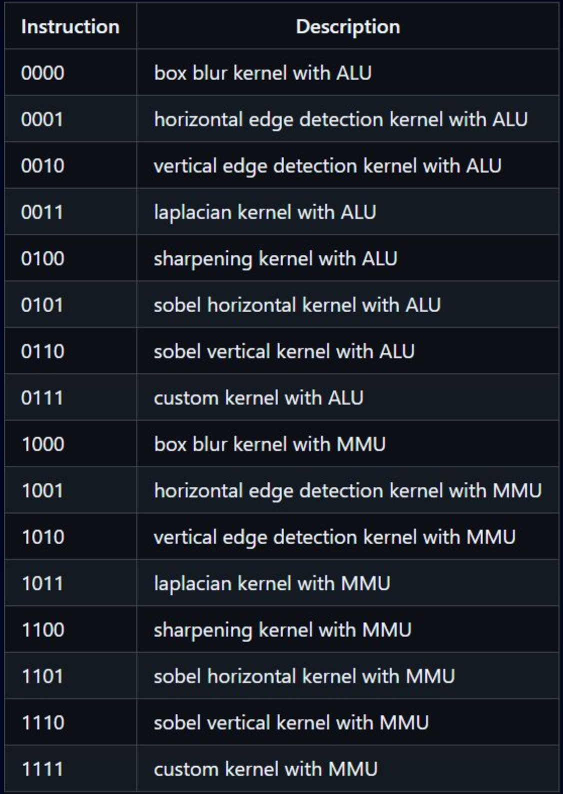
Offset	Size	Hex value	Value	Description
BMP Header				
0h	2	42 4D	"BM"	ID field (42h, 4Dh)
2h	4	46 00 00 00	70 bytes (54+16)	Size of the BMP file (54 bytes header + 16 bytes data)
6h	2	00 00	Unused	Application specific
8h	2	00 00	Unused	Application specific
Ah	4	36 00 00 00	54 bytes (14+40)	Offset where the pixel array (bitmap data) can be found
DIB Header				
Eh	4	28 00 00 00	40 bytes	Number of bytes in the DIB header (from this point)
12h	4	02 00 00 00	2 pixels (left to right order)	Width of the bitmap in pixels
16h	4	02 00 00 00	2 pixels (bottom to top order)	Height of the bitmap in pixels. Positive for bottom to top pixel order.
1Ah	2	01 00	1 plane	Number of color planes being used
1Ch	2	18 00	24 bits	Number of bits per pixel
1Eh	4	00 00 00 00	0	BI_RGB, no pixel array compression used
22h	4	10 00 00 00	16 bytes	Size of the raw bitmap data (including padding)
26h	4	13 0B 00 00	2835 pixels/metre horizontal	Print resolution of the image, 72 DPI × 39.3701 inches per metre yields 2834.6472
2Ah	4	13 0B 00 00	2835 pixels/metre vertical	
2Eh	4	00 00 00 00	0 colors	Number of colors in the palette
32h	4	00 00 00 00	0 important colors	0 means all colors are important
Start of pixel array (bitmap data)				
36h	3	00 00 FF	0 0 255	Red, Pixel (x=0, y=1)
39h	3	FF FF FF	255 255 255	White, Pixel (x=1, y=1)
3Ch	2	00 00	0 0	Padding for 4 byte alignment (could be a value other than zero)
3Eh	3	FF 00 00	255 0 0	Blue, Pixel (x=0, y=0)
41h	3	00 FF 00	0 255 0	Green, Pixel (x=1, y=0)
44h	2	00 00	0 0	Padding for 4 byte alignment (could be a value other than zero)

Desain Sistem

Terdapat empat komponen utama, yaitu:

- **CPU** : Mengontrol alur instruksi (IDLE, FETCH, DECODE, EXECUTE, dan COMPLETE). Terdapat RESET juga untuk mengembalikan semua nilai signal ke nilai awal.
- **Decoder** : Menerjemahkan instruksi ke opcode (jenis kernel mana yang dipakai) dan address_flag (jenis unit mana yang akan dieksekusi)
- **ALU** : Melakukan operasi convolution secara direct.
- **MMU** : Melakukan operasi convolution secara im2col.

Mapping Instruction



Instruction	Description
0000	box blur kernel with ALU
0001	horizontal edge detection kernel with ALU
0010	vertical edge detection kernel with ALU
0011	laplacian kernel with ALU
0100	sharpening kernel with ALU
0101	sobel horizontal kernel with ALU
0110	sobel vertical kernel with ALU
0111	custom kernel with ALU
1000	box blur kernel with MMU
1001	horizontal edge detection kernel with MMU
1010	vertical edge detection kernel with MMU
1011	laplacian kernel with MMU
1100	sharpening kernel with MMU
1101	sobel horizontal kernel with MMU
1110	sobel vertical kernel with MMU
1111	custom kernel with MMU

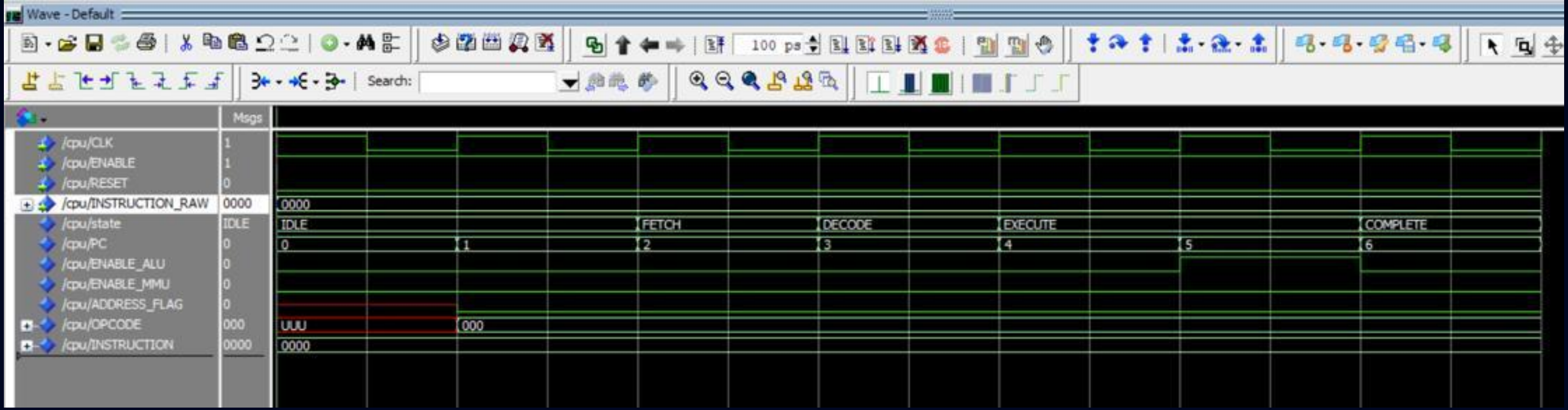
Hasil Pengujian

Box Blur

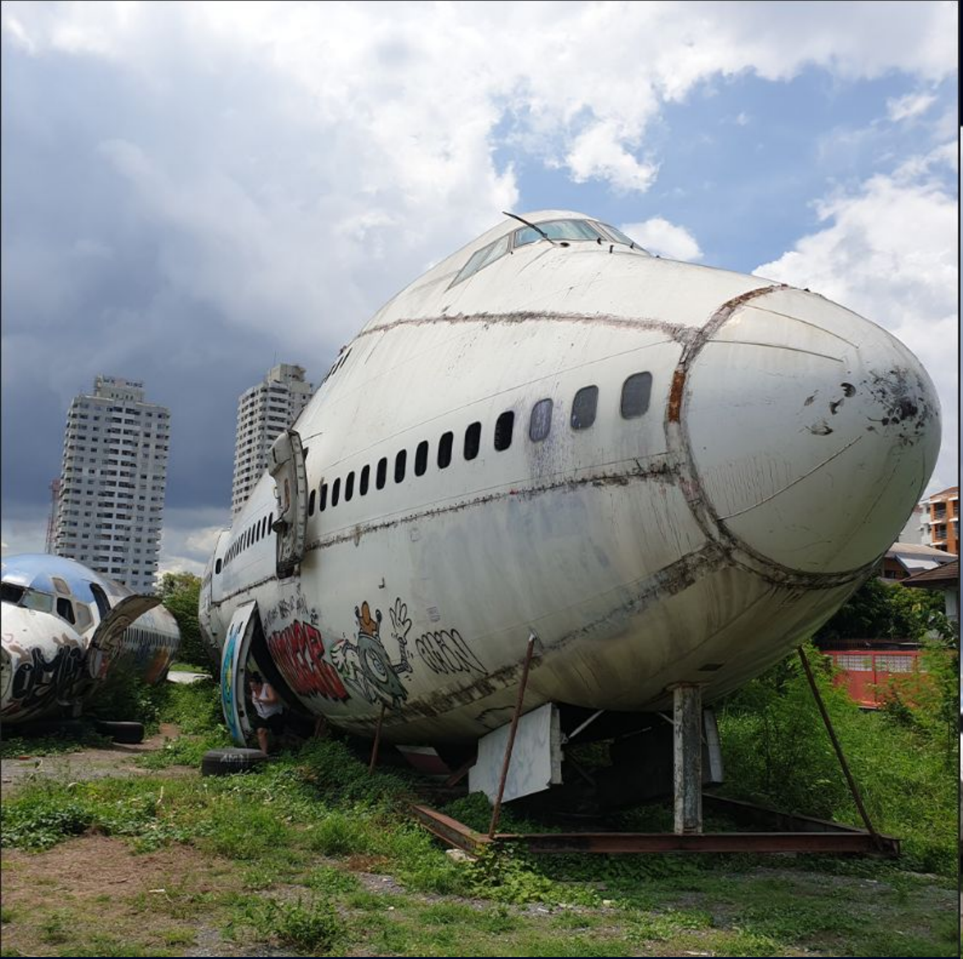
$$K = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

Kernel yang dipakai

Hasil Modelsim

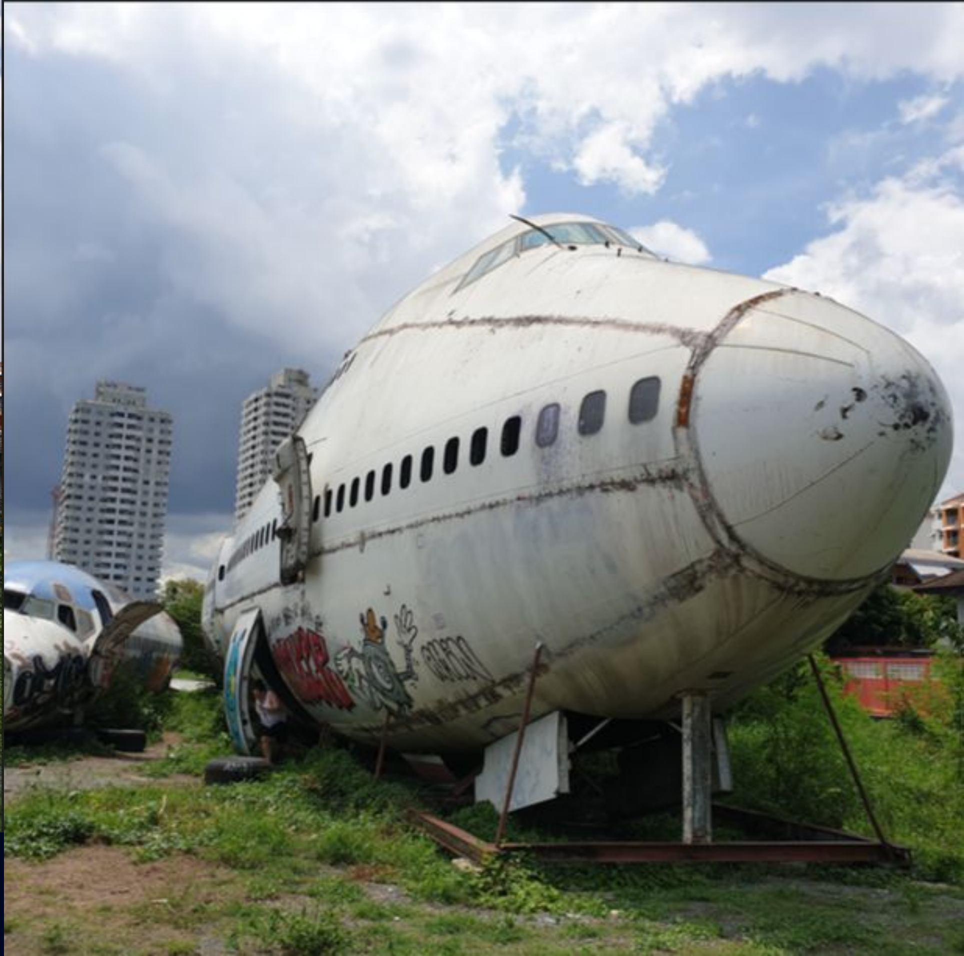


	Msgs
/cpu/CLK	1
/cpu/ENABLE	1
/cpu/RESET	0
/cpu/INSTRUCTION_RAW	0000
/cpu/state	IDLE
/cpu/PC	0
/cpu/ENABLE_ALU	0
/cpu/ENABLE_MMU	0
/cpu/ADDRESS_FLAG	0
/cpu/OPCODE	000
/cpu/INSTRUCTION	0000



Original

Box Blur

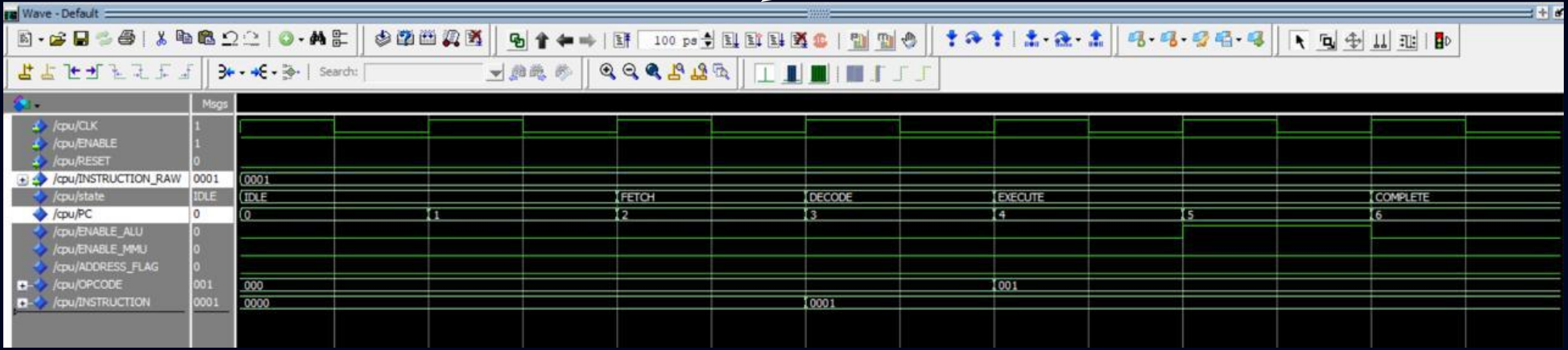


Horizontal Edge

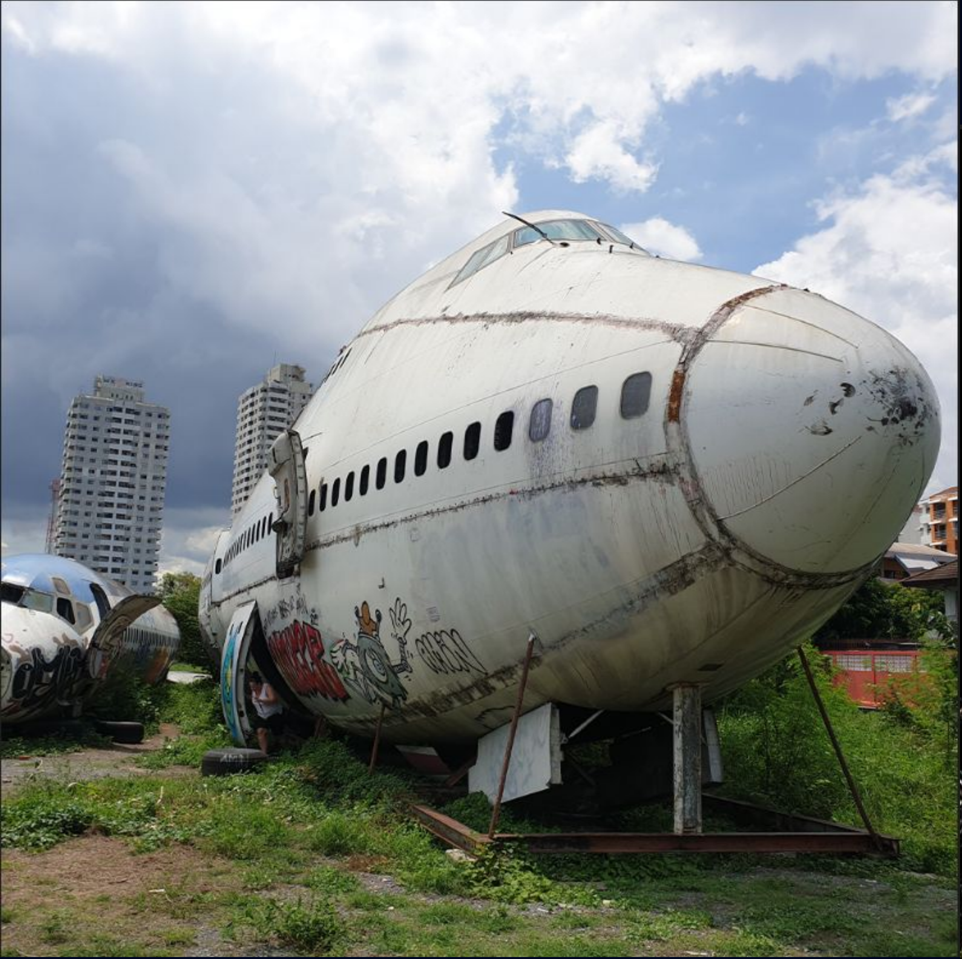
$$K = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Kernel yang dipakai

Hasil Modelsim

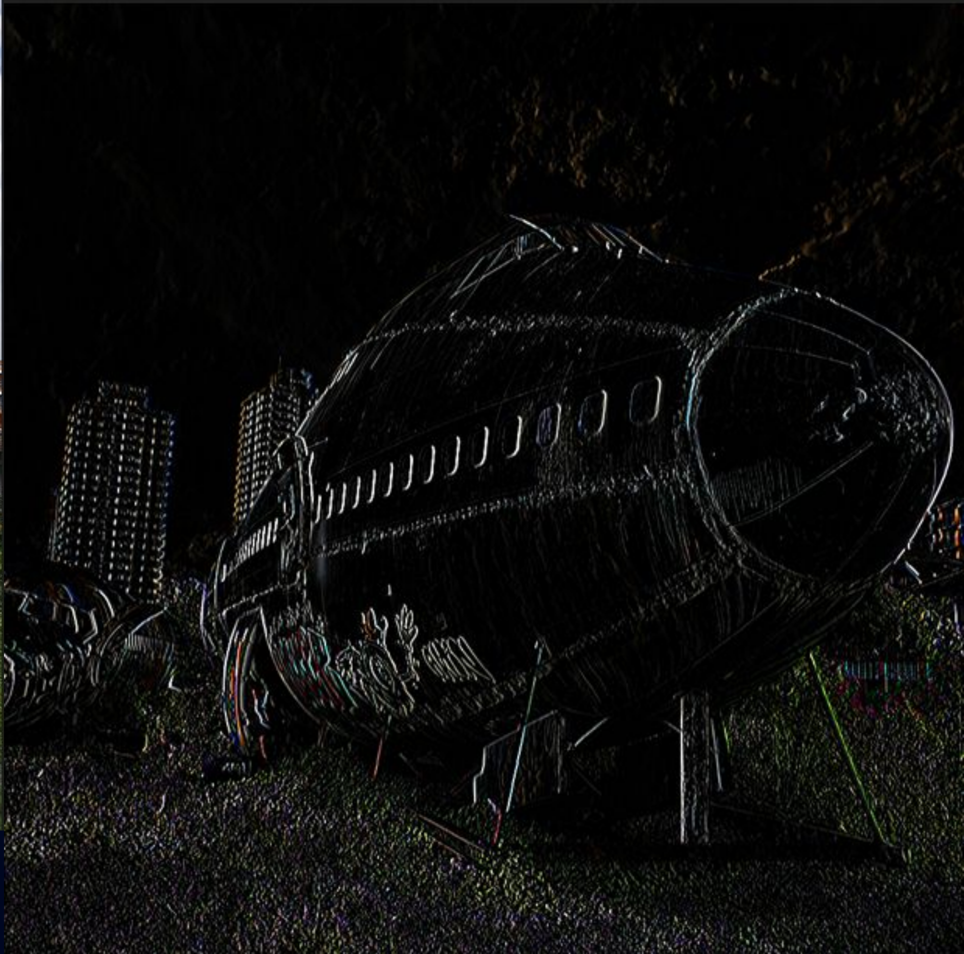


/cpu/CLK 1
/cpu/ENABLE 1
/cpu/RESET 0
/cpu/INSTRUCTION_RAW 0001
/cpu/state IDLE
/cpu/PC 0
/cpu/ENABLE_ALU 0
/cpu/ENABLE_MMU 0
/cpu/ADDRESS_FLAG 0
/cpu/OPCODE 001
/cpu/INSTRUCTION 0001



Original

Horizontal Edge

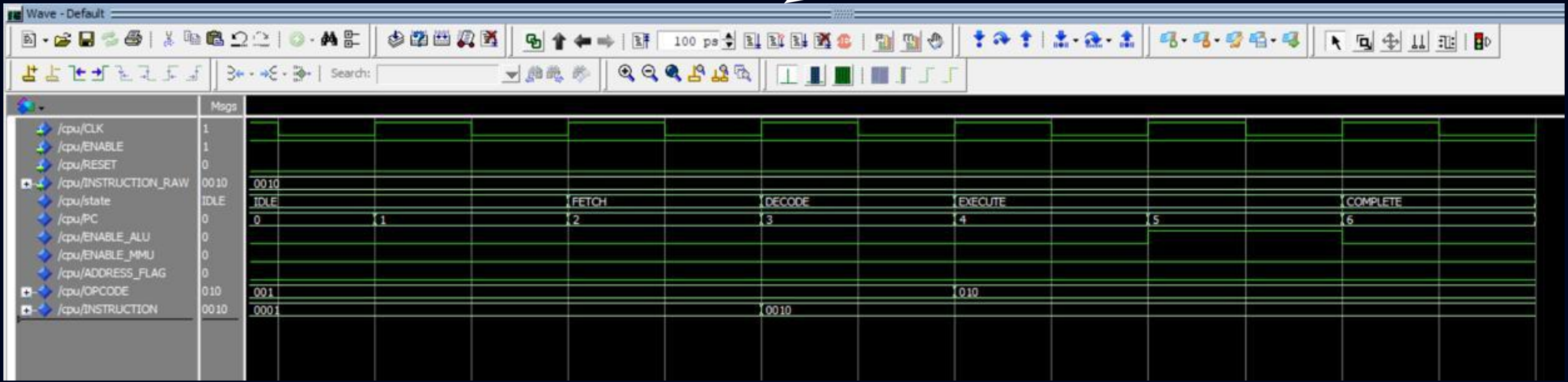


Vertical Edge

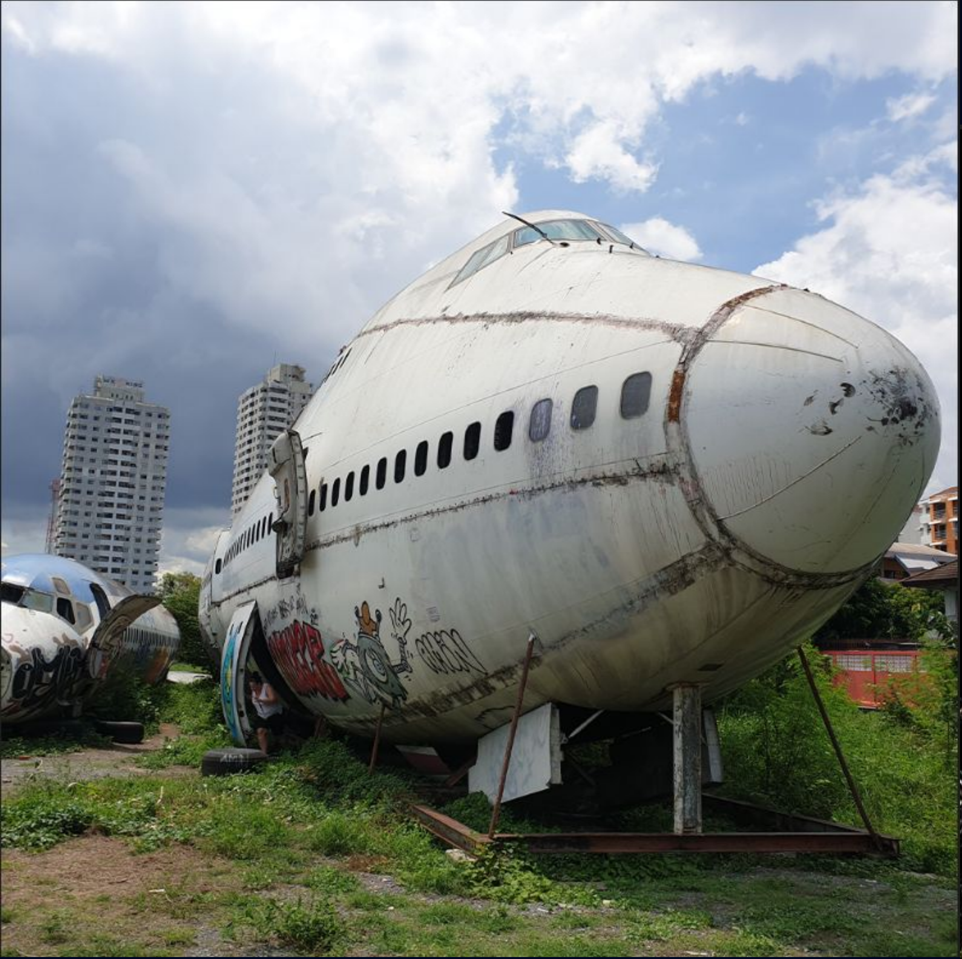
$$K = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Kernel yang dipakai

Hasil Modelsim

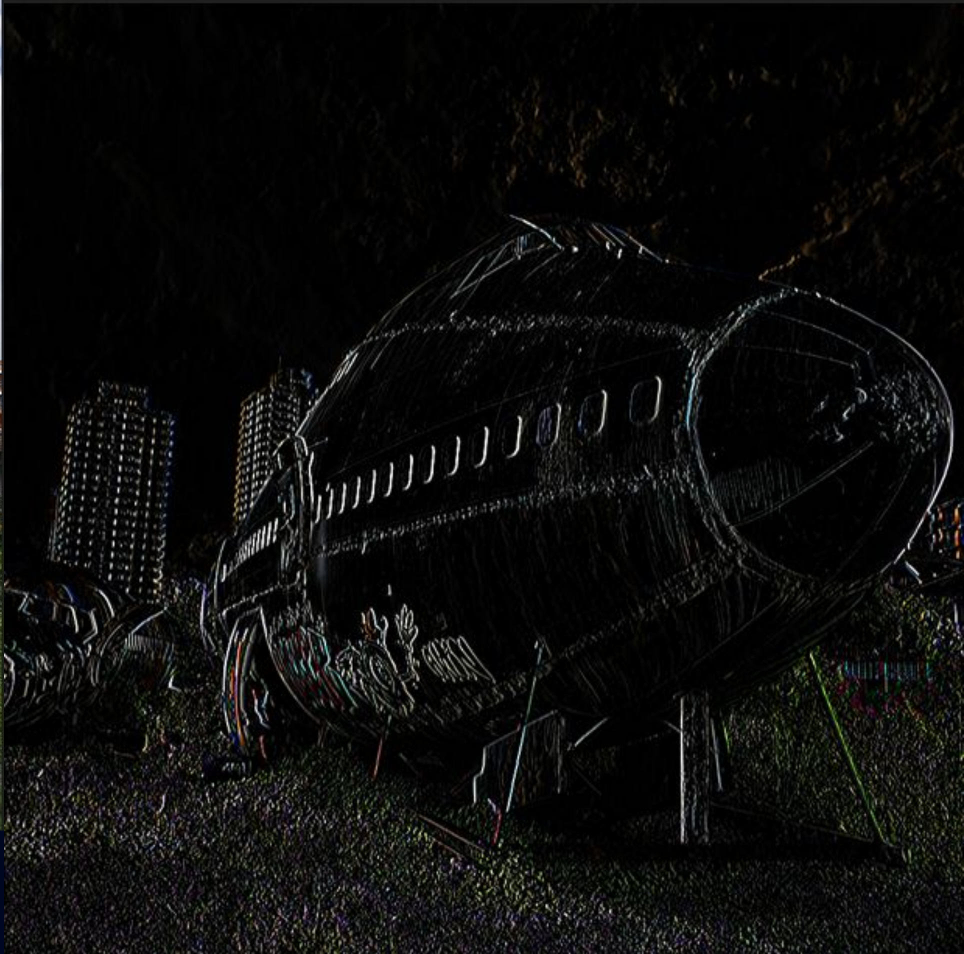


	Msgs
/cpu/CLK	1
/cpu/ENABLE	1
/cpu/RESET	0
/cpu/INSTRUCTION_RAW	0010
/cpu/state	IDLE
/cpu/PC	0
/cpu/ENABLE_ALU	0
/cpu/ENABLE_MMU	0
/cpu/ADDRESS_FLAG	0
/cpu/OPCODE	010
/cpu/INSTRUCTION	0010



Original

Vertical Edge

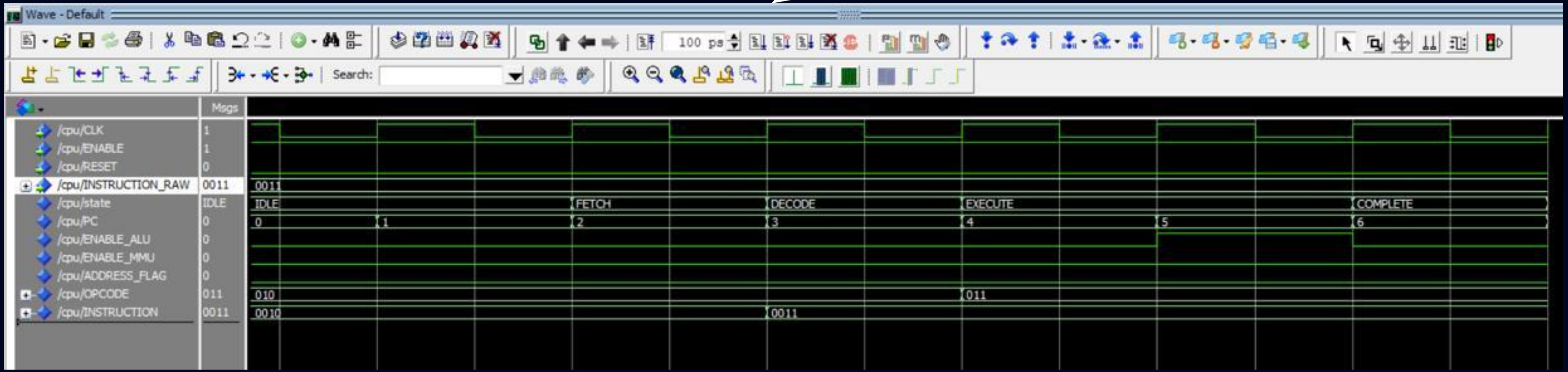


Laplacian

$$K = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

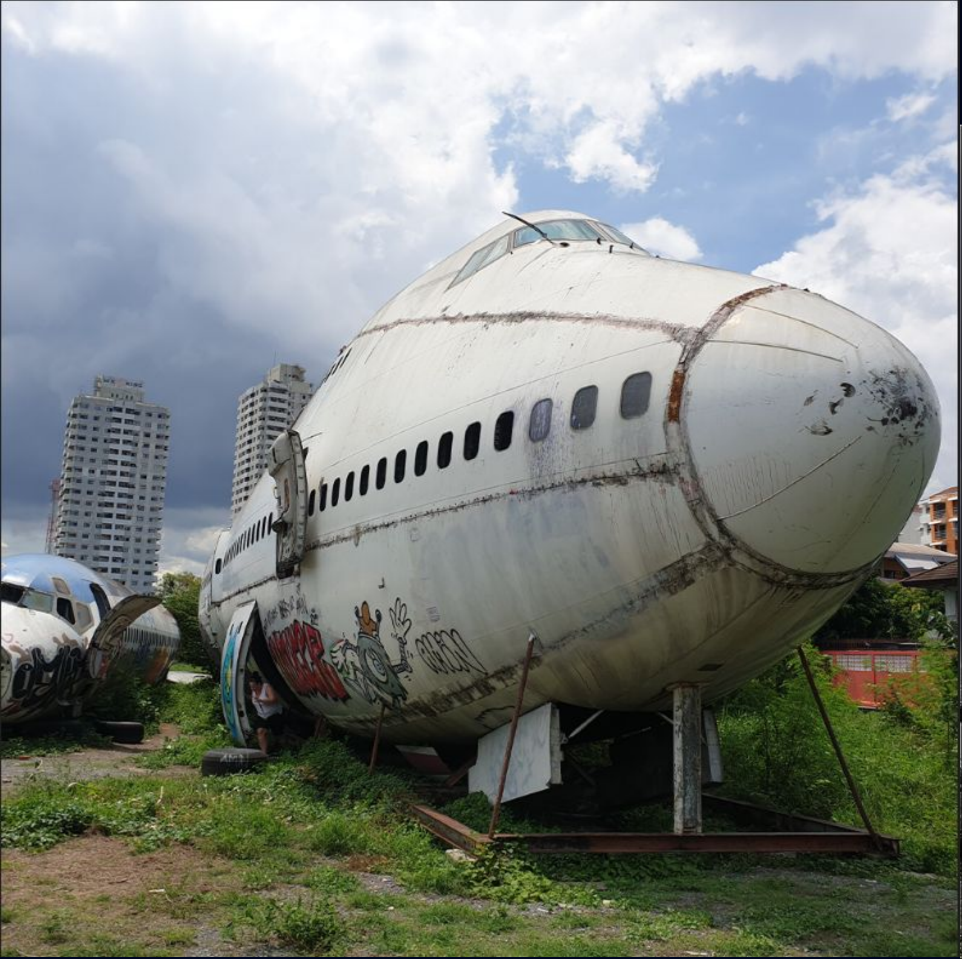
Kernel yang dipakai

Hasil Modelsim



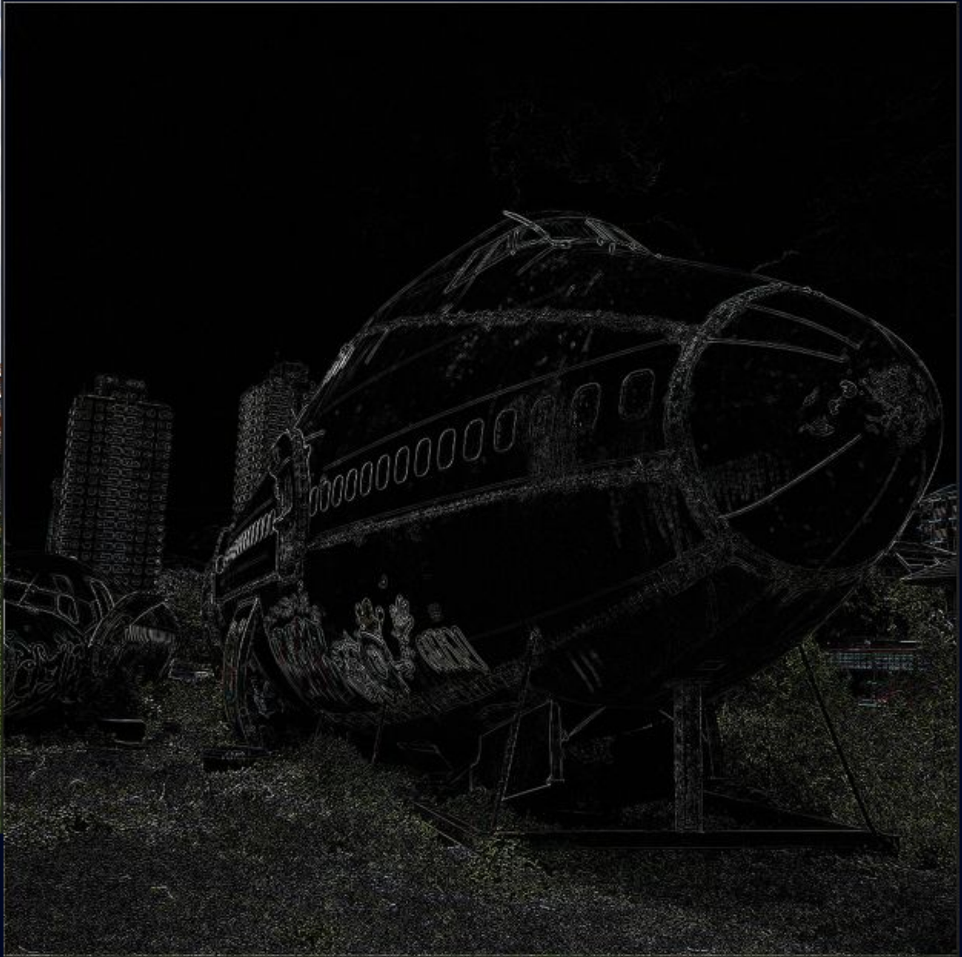
Wave - Default

```
                         Msgs
/cpu/CLK                 1
/cpu/ENABLE              1
/cpu/RESET               0
/cpu/INSTRUCTION_RAW     0011    0011
/cpu/state               IDLE    IDLE         FETCH      DECODE     EXECUTE               COMPLETE
/cpu/PC                  0       0       1    2          3          4          5          6
/cpu/ENABLE_ALU          0
/cpu/ENABLE_MMU          0
/cpu/ADDRESS_FLAG        0
/cpu/OPCODE              011     010                                011
/cpu/INSTRUCTION         0011    0010                    0011
```



Original

Laplacian

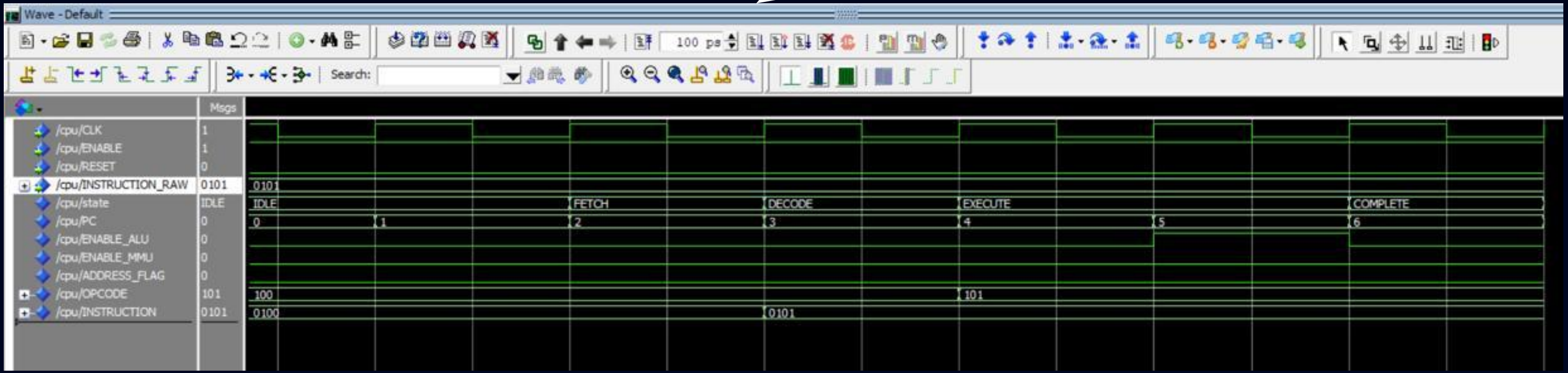


Sobel Horizontal

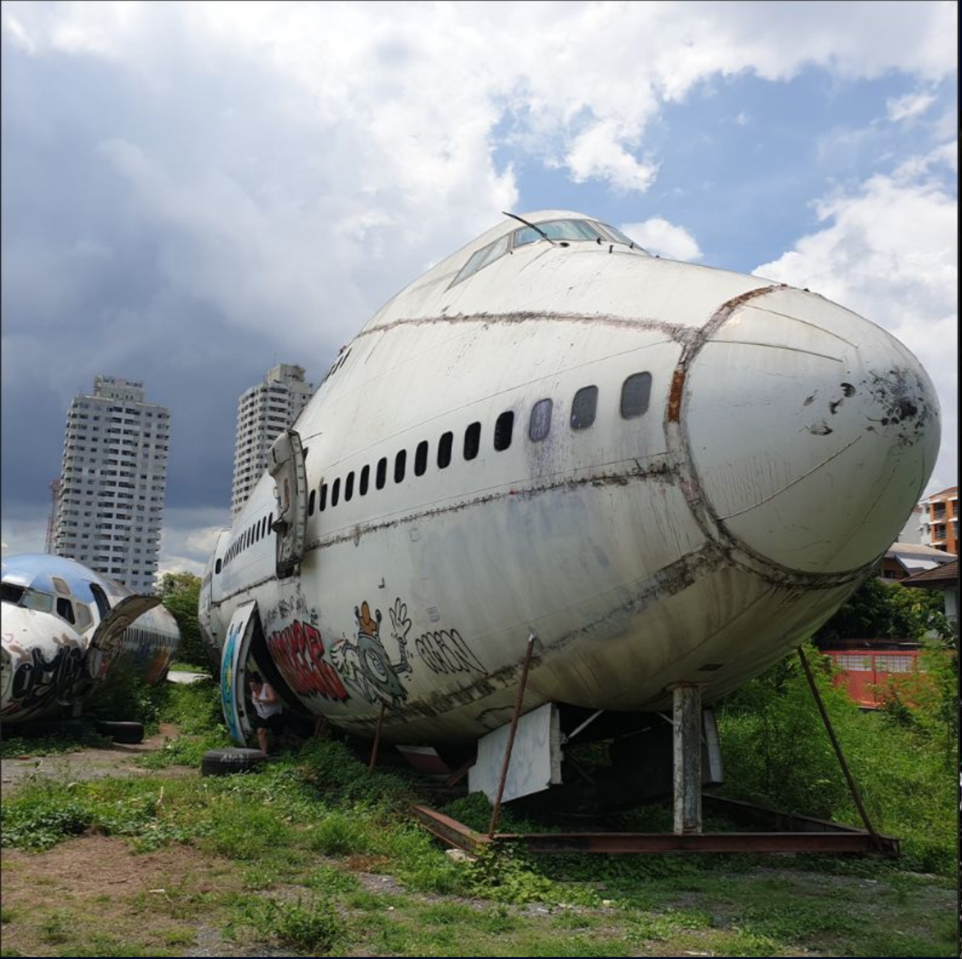
$$K = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Kernel yang dipakai

Hasil Modelsim

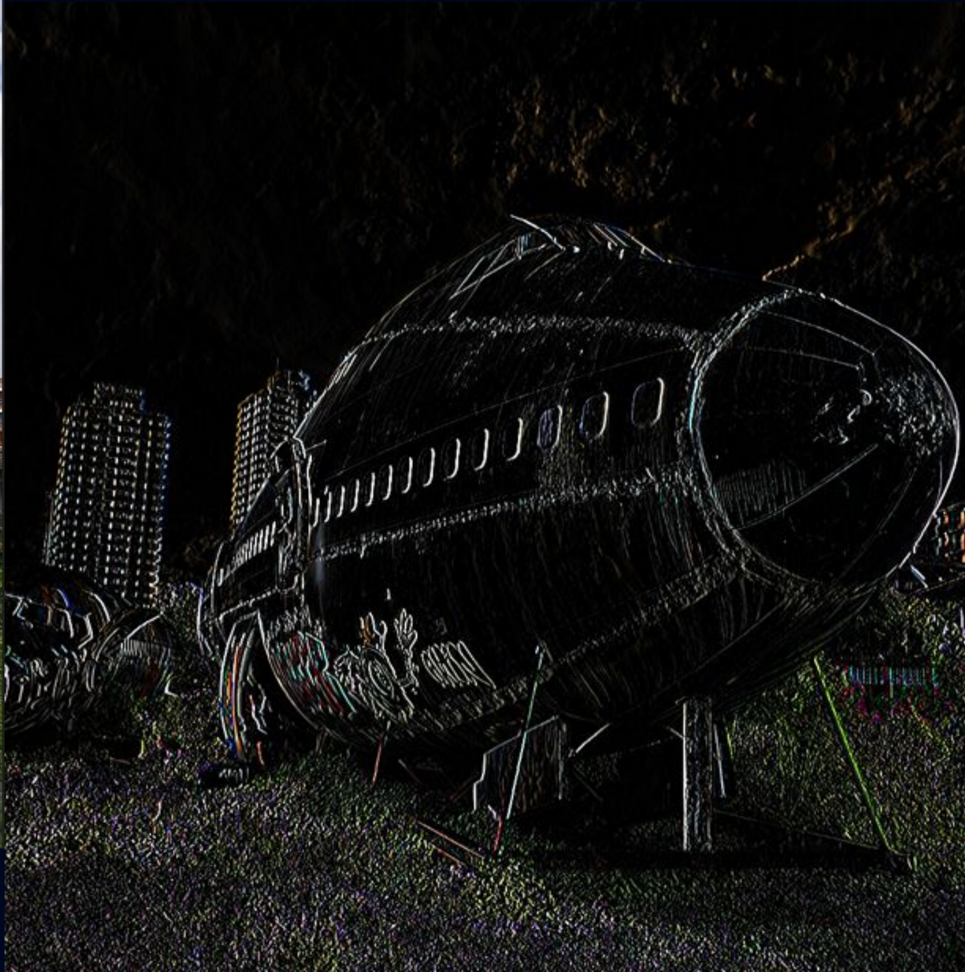


	Msgs
/cpu/CLK	1
/cpu/ENABLE	1
/cpu/RESET	0
/cpu/INSTRUCTION_RAW	0101
/cpu/state	IDLE
/cpu/PC	0
/cpu/ENABLE_ALU	0
/cpu/ENABLE_MMU	0
/cpu/ADDRESS_FLAG	0
/cpu/OPCODE	101
/cpu/INSTRUCTION	0101



Original

Sobel Horizontal

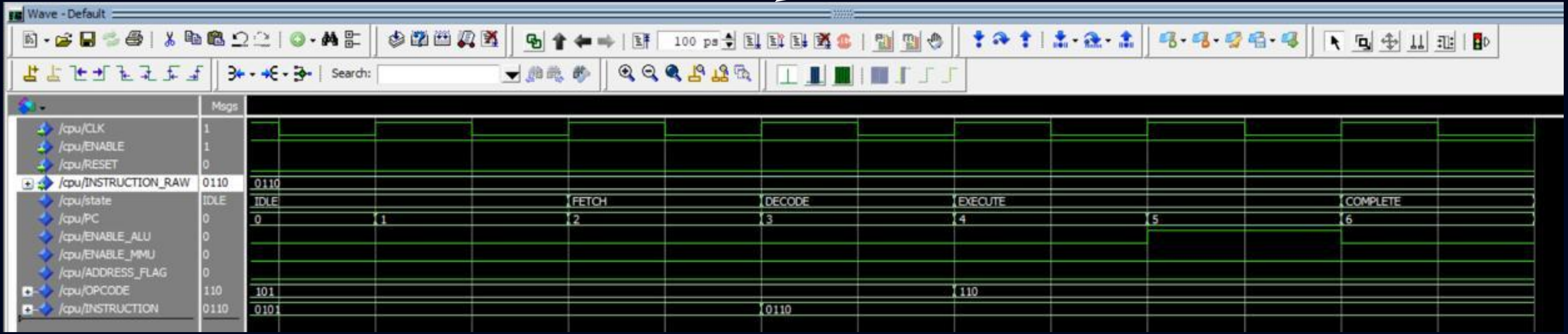


Sobel Vertical

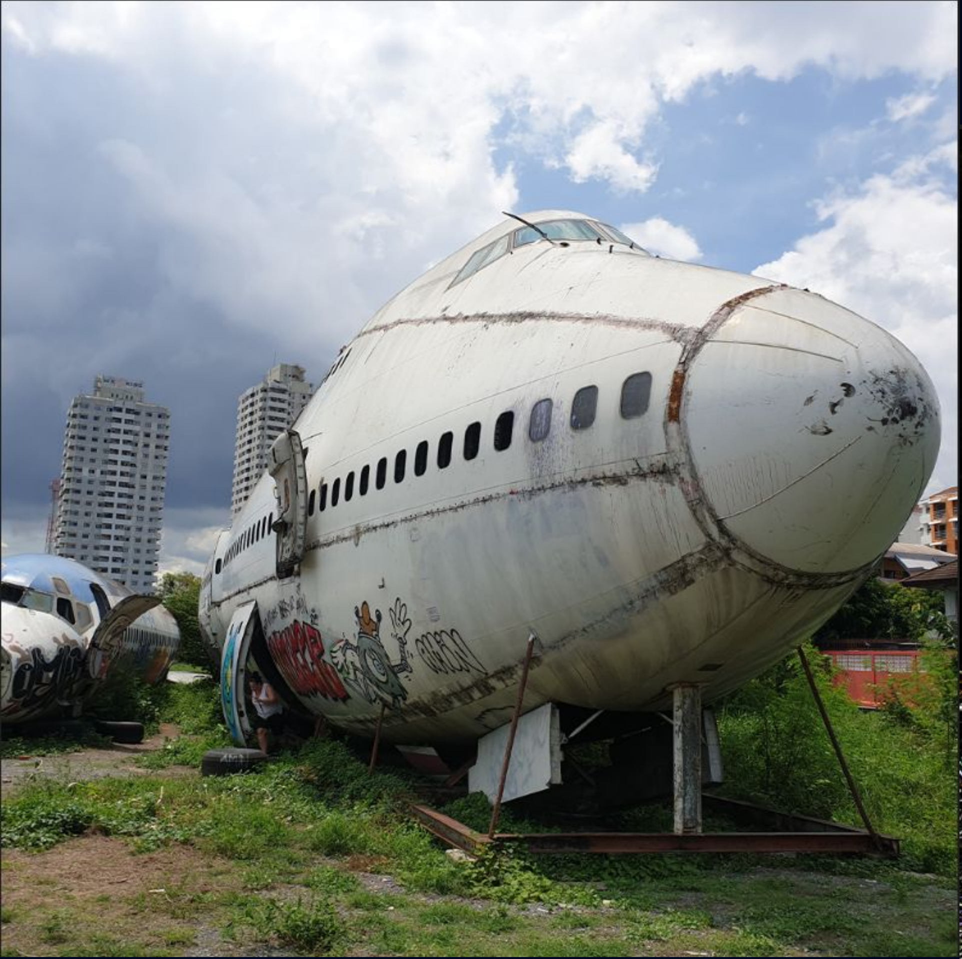
$$K = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Kernel yang dipakai

Hasil Modelsim

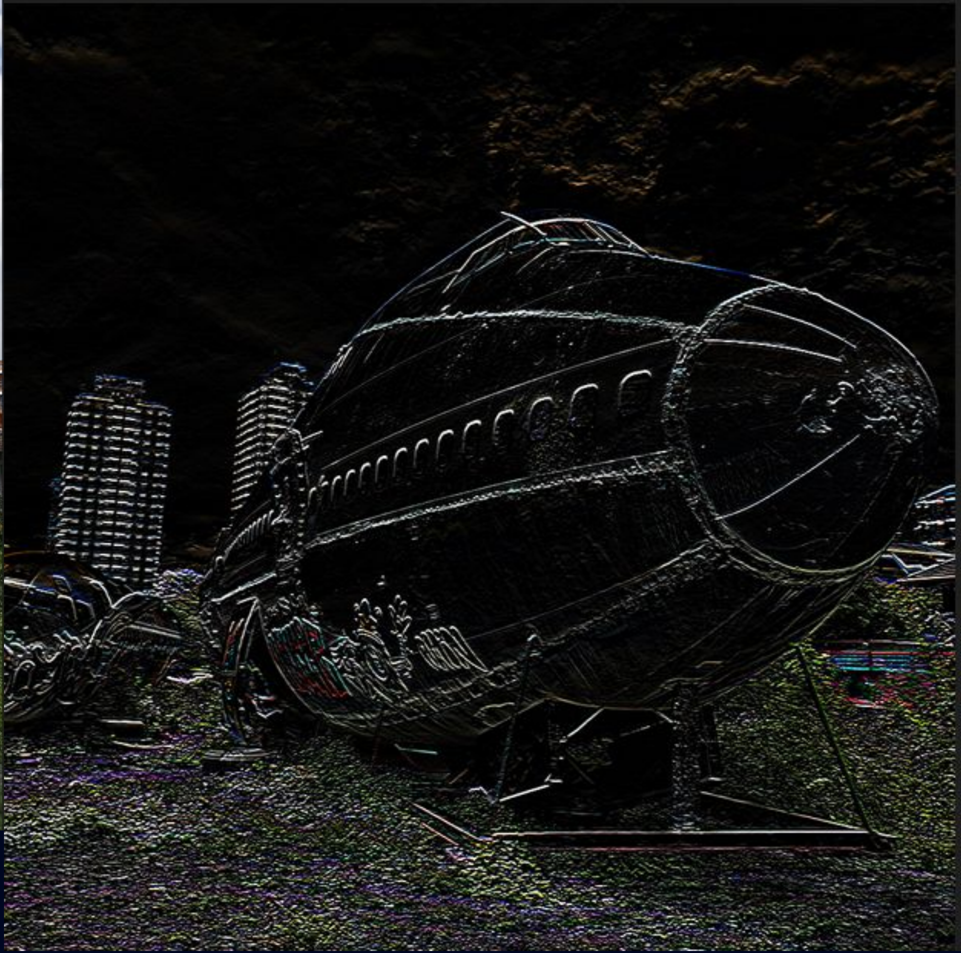


	Msgs
/cpu/CLK	1
/cpu/ENABLE	1
/cpu/RESET	0
/cpu/INSTRUCTION_RAW	0110
/cpu/state	IDLE
/cpu/PC	0
/cpu/ENABLE_ALU	0
/cpu/ENABLE_MMU	0
/cpu/ADDRESS_FLAG	0
/cpu/OPCODE	110
/cpu/INSTRUCTION	0110



Original

Sobel Vertical

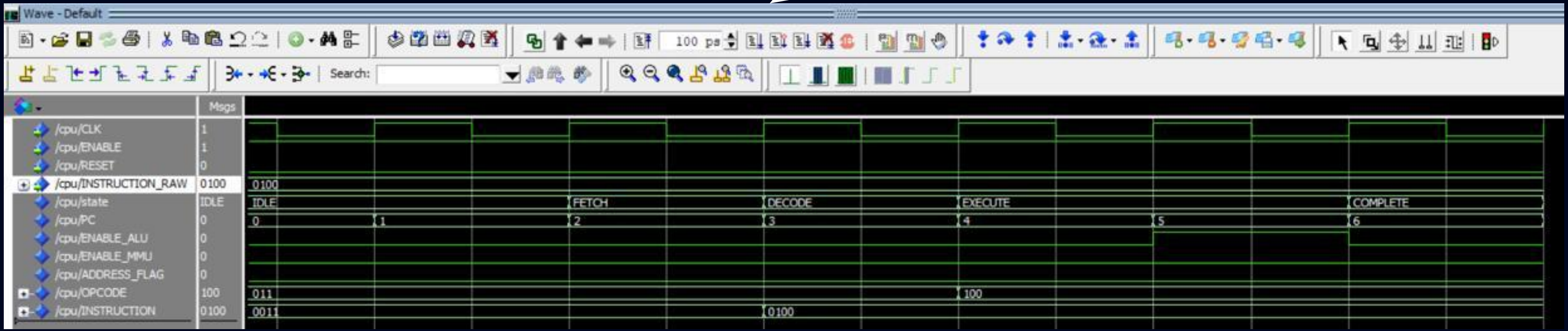


Sharpening

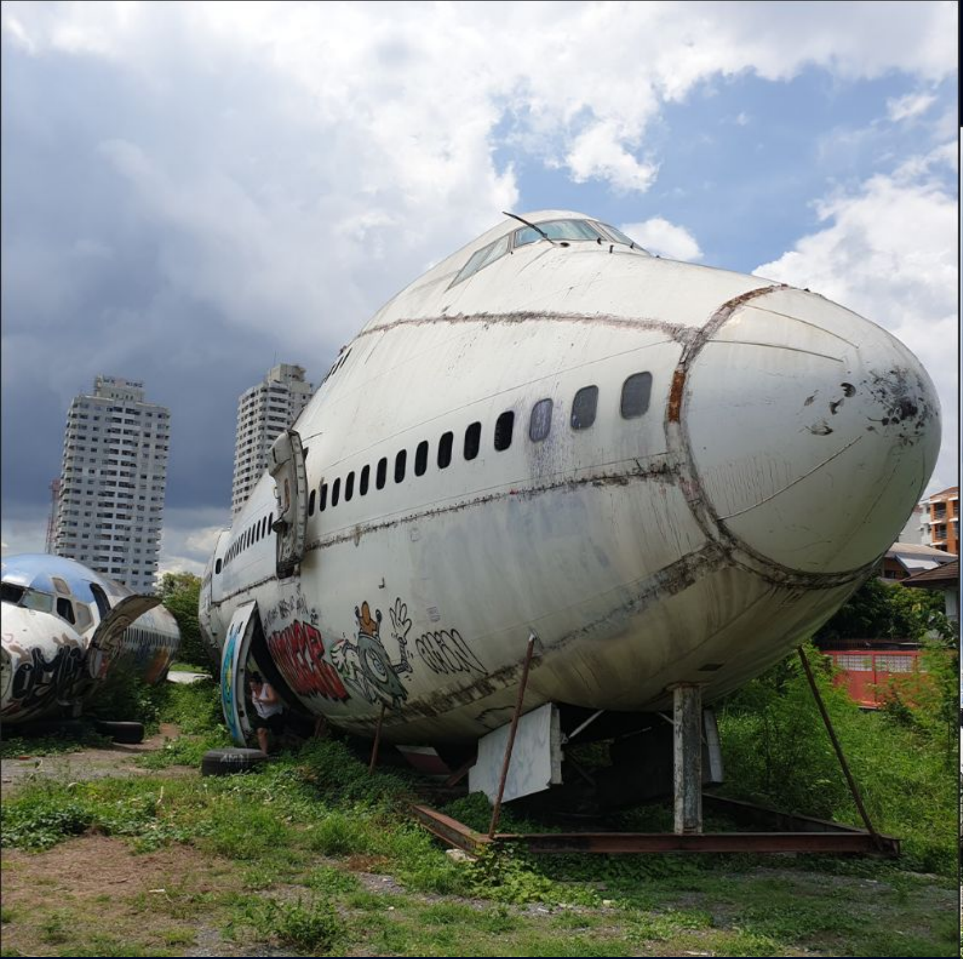
$$K = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Kernel yang dipakai

Hasil Modelsim

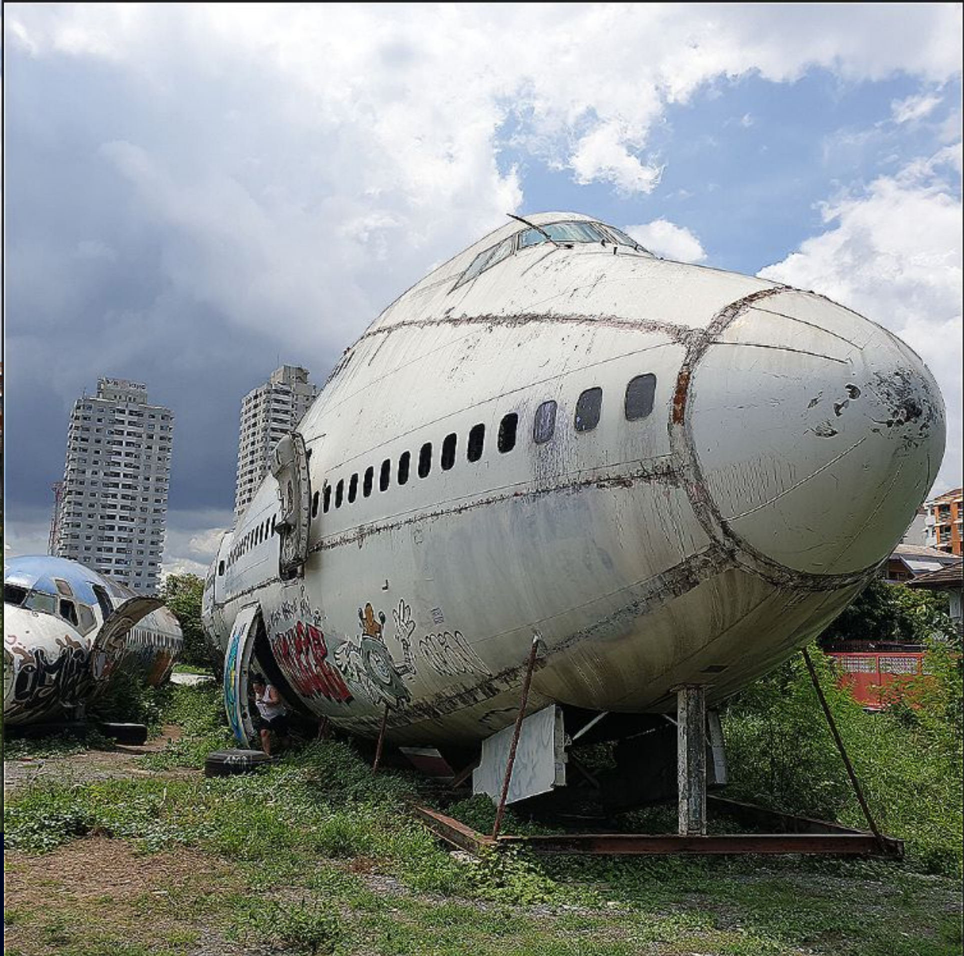


	Msgs
/cpu/CLK	1
/cpu/ENABLE	1
/cpu/RESET	0
/cpu/INSTRUCTION_RAW	0100
/cpu/state	IDLE
/cpu/PC	0
/cpu/ENABLE_ALU	0
/cpu/ENABLE_MMU	0
/cpu/ADDRESS_FLAG	0
/cpu/OPCODE	100
/cpu/INSTRUCTION	0100



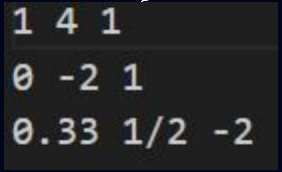
Original

Sharpening

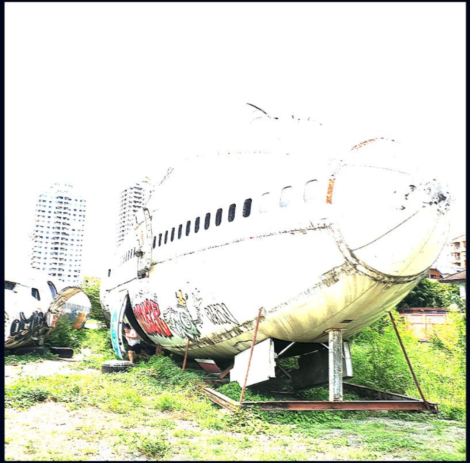


Custom

Kernel yang dipakai

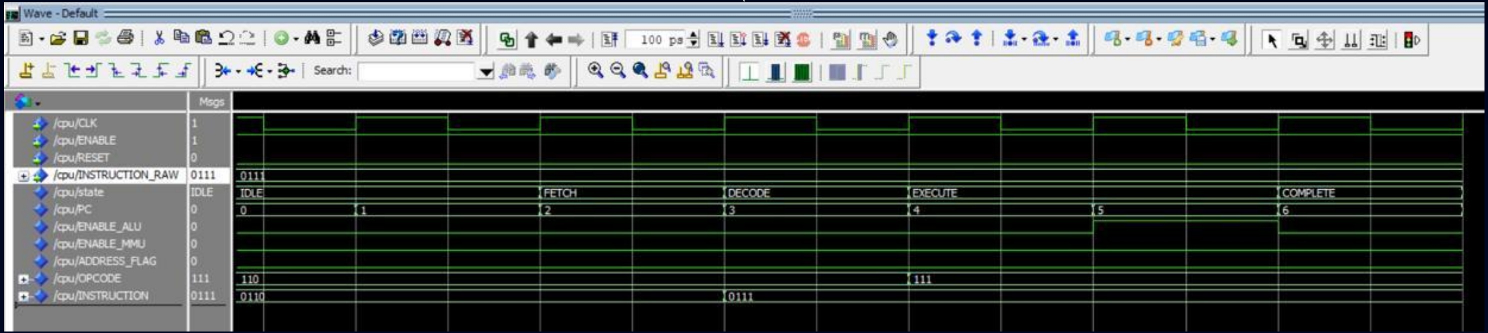


```
1 4 1
0 -2 1
0.33 1/2 -2
```



Output

Hasil Modelsim



```
/cpu/CLK              1
/cpu/ENABLE           1
/cpu/RESET            0
/cpu/INSTRUCTION_RAW  0111
/cpu/state            IDLE
/cpu/PC               0
/cpu/ENABLE_ALU       0
/cpu/ENABLE_MMU       0
/cpu/ADDRESS_FLAG     0
/cpu/OPCODE           111
/cpu/INSTRUCTION      0111
```

Terima Kasih!