Huffman Encoding

For Lossless Data Compression

PA 23

OUTLINE

01 LATAR BELAKANG

02 IMPLEMENTASI

03 HASIL

04 KESIMPULAN

LATAR BELAKANG

Proyek ini berfokus pada implementasi algoritma Huffman Encoding untuk kompresi data lossless menggunakan pendekatan perangkat keras berbasis VHDL. Tujuan utama dari proyek ini adalah untuk menciptakan solusi yang efisien dalam menyimpan dan mengkompres data teks besar dengan memanfaatkan keunggulan FPGA untuk meningkatkan kecepatan dan efisiensi proses.

Untuk meningkatkan efisiensi proses kompresi, algoritma ini diimplementasikan menggunakan VHDL pada FPGA. FPGA dipilih karena kemampuannya untuk memproses data secara paralel dan secara real-time, yang sangat penting untuk menangani string data yang besar.

LATAR BELAKANG

Untuk meningkatkan efisiensi proses kompresi, algoritma ini diimplementasikan menggunakan VHDL pada FPGA. FPGA dipilih karena kemampuannya untuk memproses data secara paralel dan secara real-time, yang sangat penting untuk menangani string data yang besar. Proses implementasi mencakup:

1. Pemodelan Algoritma Huffman Encoding VHDL
2. Simulasi fungsionalitas dan akurasi algoritma sesuai dengan input data
3. Sintesis dan pengujian di platform FPGA guna evaluasi kerja perangkat keras dibandingkan perangkat lunak.

IMPLEMENTASI

HuffmanNode.vhd

- Menyatukan modul-modul utama:
- NodeGenerator
- NodeSorter
- NodeMerger
- HuffmanTranslator

Menghubungkan komponen lewat component instantiation dan penyambungan port (UUT)

Alur kerja dikendalikan clock

IMPLEMENTASI

ReadSort.vhd

- Membaca string dari sebuah file (Input.txt)
- Contoh : “abracadabra”
- Memecah string menjadi karakter-karakter uniknya (a, b, r, c, d)
- Menghitung kemunculan tiap karakter dalam string tersebut (a 5 / b 2 / r 2 / c 1 / d 1)
- Menuliskannya kembali ke dalam sebuah file (Output.txt)

IMPLEMENTASI

NodeGen.vhd

- Membaca Output.txt, yang berisi karakter hasil ReadSort
- Membuat node sebanyak karakter unik yang ada dalam string
- Abracadabra => 5 node (a, b, r, c, d)
- Node awal memiliki nilai karakter dan frekuensi

IMPLEMENTASI

NodeSort.vhd

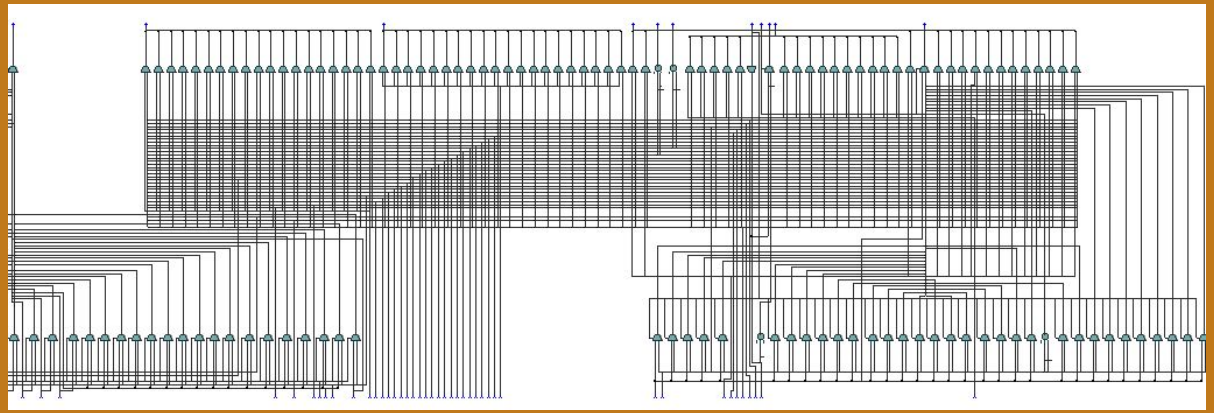
- Menyortir tiap node menurut frekuensi mereka
- Sort descending
- Algoritma bubble sort
-

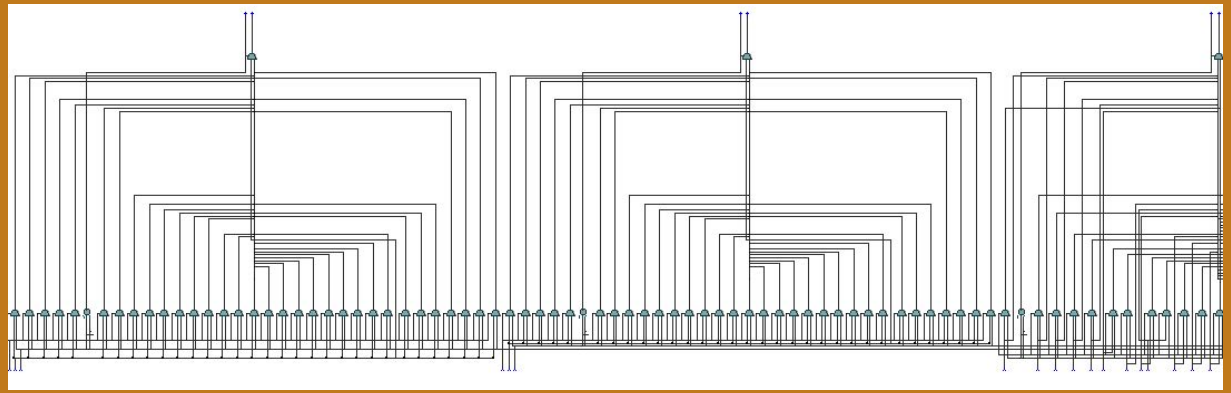
IMPLEMENTASI

HuffmanTranslation.vhd

- Mengambil data dari array
- Menghitung node dan menentukan leaf node
- melakukan traverse tree
- Menerjemahkan string ke biner

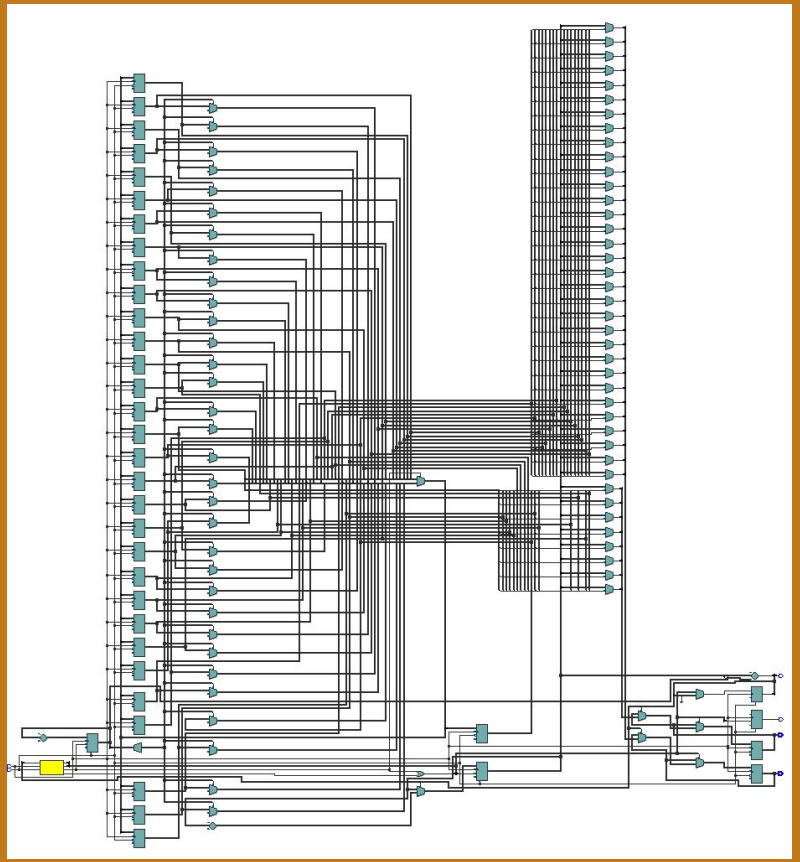
Schematics - ReadSort





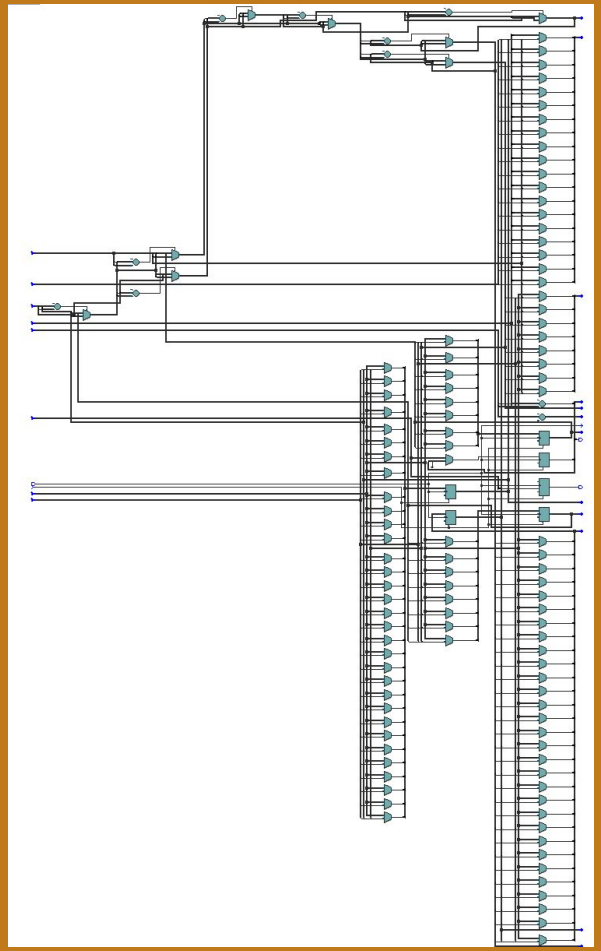
(Note : Line-line dynamic disubstitusi dengan nilai konstan agar dapat disintesis)

Schematics - NodeGenerator



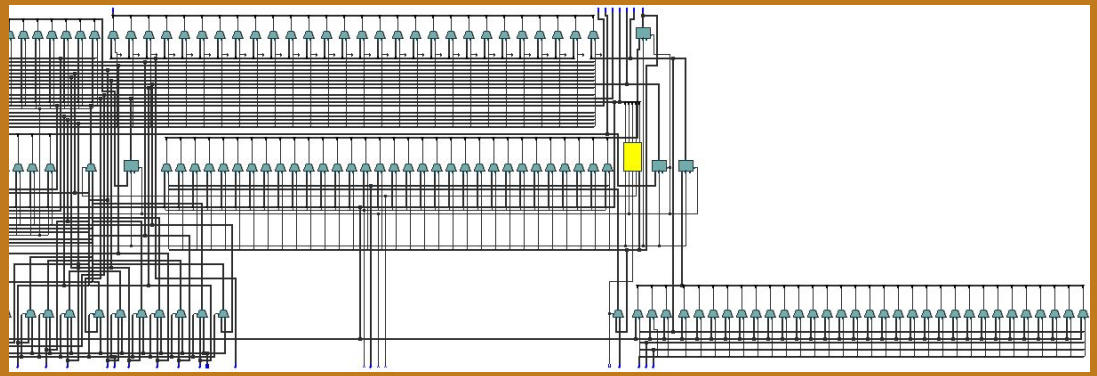
(Note : Line-line dynamic disubstitusi dengan nilai konstan ag
dapat disintesis)

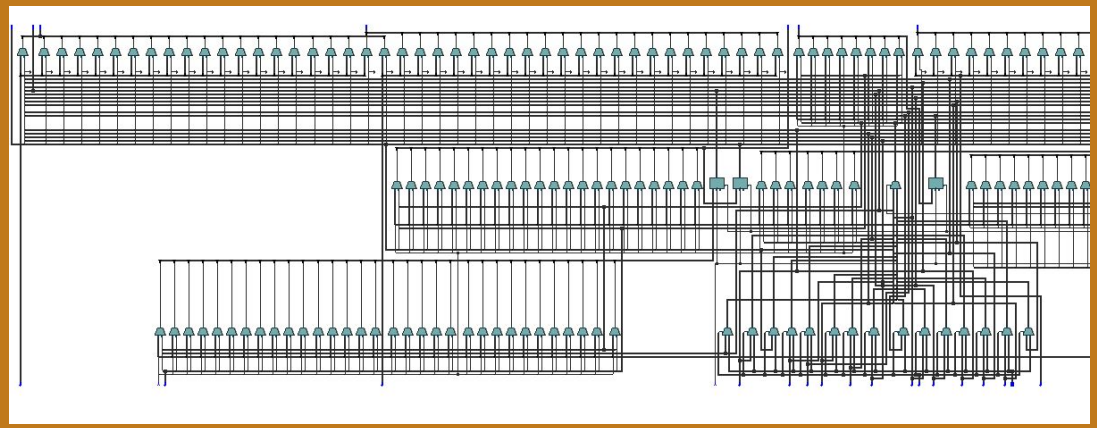
Schematics - NodeSorter



(Note : Line-line dynamic disubstitusi dengan nilai konstan agar dapat disintesis)

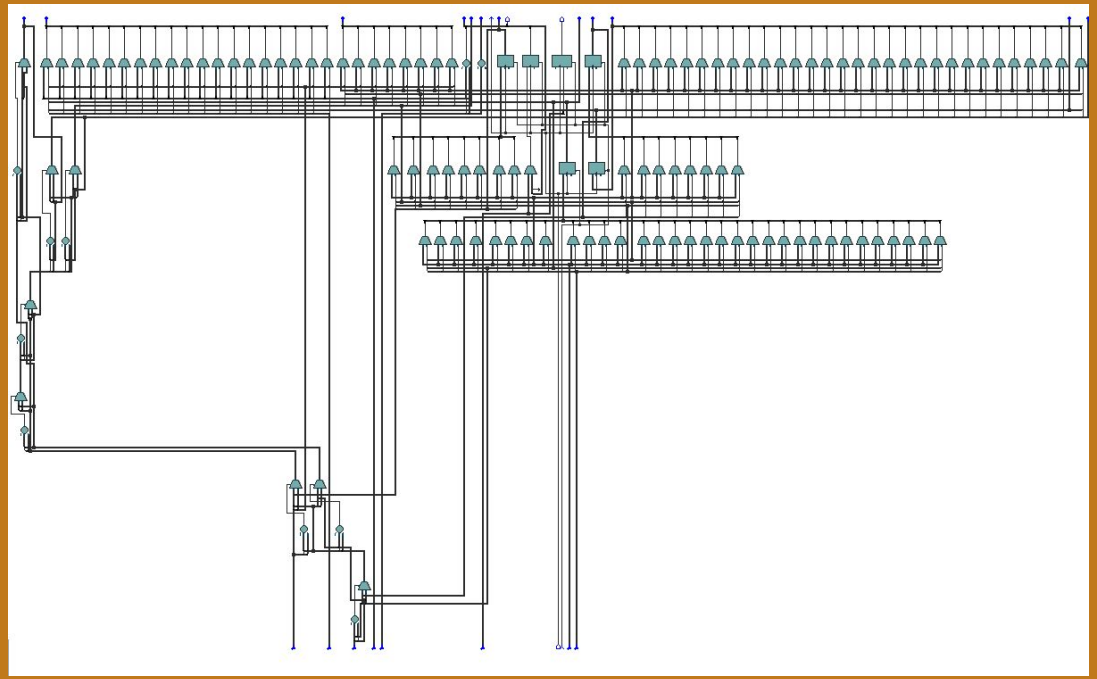
Schematics - NodeMerger





(Note : Line-line dynamic disubstitusi dengan nilai konstan agar dapat disintesis)

Schematics - HuffmanTranslator



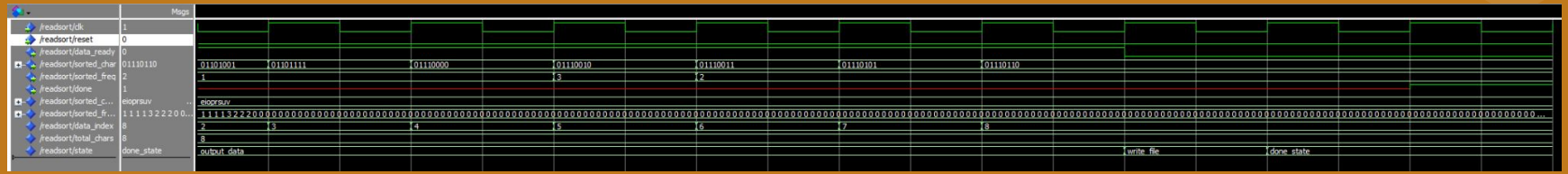
(Note : Line-line dynamic disubstitusi dengan nilai konstan agar dapat disintesis)

Pengujian Program

supersurvivor

Isi file Input = “supersurvivor”

Eksekusi ReadSort



Hasil file Output

e 1
i 1
o 1
p 1
r 3
s 2
u 2
v 2

Pengujian Program

Output dipindahkan ke nodeGen
Eksekusi nodeGen

```
run
# ** Note: Character read: u, Frequency read: 2
#    Time: 2 ns  Iteration: 0  Instance: /huffmanprocessor/node_gen
# ** Note: Receiving new node - Char: s Freq: 2
#    Time: 2 ns  Iteration: 0  Instance: /huffmanprocessor/node_sort
# ** Note: Character already exists at index 5
#    Time: 2 ns  Iteration: 0  Instance: /huffmanprocessor/node_sort
run
# ** Note: Outputting node: Char = u, Freq = 2
#    Time: 2100 ps  Iteration: 0  Instance: /huffmanprocessor/node_gen
# ** Note: Receiving new node - Char: s Freq: 2
#    Time: 2100 ps  Iteration: 0  Instance: /huffmanprocessor/node_sort
# ** Note: Character already exists at index 5
#    Time: 2100 ps  Iteration: 0  Instance: /huffmanprocessor/node_sort
run
# ** Note: Receiving new node - Char: u Freq: 2
#    Time: 2200 ps  Iteration: 0  Instance: /huffmanprocessor/node_sort
# ** Note: Adding new node at index 6
#    Time: 2200 ps  Iteration: 0  Instance: /huffmanprocessor/node_sort
run
# ** Note: Character read: v, Frequency read: 2
#    Time: 2300 ps  Iteration: 0  Instance: /huffmanprocessor/node_gen
# ** Note: Receiving new node - Char: u Freq: 2
#    Time: 2300 ps  Iteration: 0  Instance: /huffmanprocessor/node_sort
# ** Note: Character already exists at index 6
#    Time: 2300 ps  Iteration: 0  Instance: /huffmanprocessor/node_sort
run
# ** Note: Outputting node: Char = v, Freq = 2
#    Time: 2400 ps  Iteration: 0  Instance: /huffmanprocessor/node_gen
# ** Note: Receiving new node - Char: u Freq: 2
#    Time: 2400 ps  Iteration: 0  Instance: /huffmanprocessor/node_sort
# ** Note: Character already exists at index 6
#    Time: 2400 ps  Iteration: 0  Instance: /huffmanprocessor/node_sort
run
# ** Note: Receiving new node - Char: v Freq: 2
#    Time: 2500 ps  Iteration: 0  Instance: /huffmanprocessor/node_sort
# ** Note: Adding new node at index 7
#    Time: 2500 ps  Iteration: 0  Instance: /huffmanprocessor/node_sort
VSIM 281> run
# ** Note: Node generation completed. Total nodes: 8
```