

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 mengompres 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

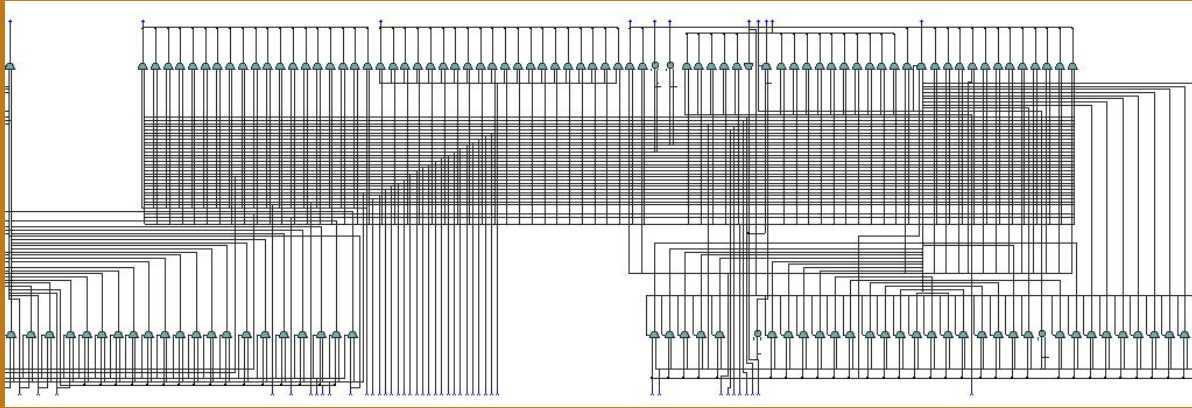
- Menyortir tiap node menurut frekuensi mereka
- Sort descending
- Algoritma bubble sort
-

IMPLEMENTASI

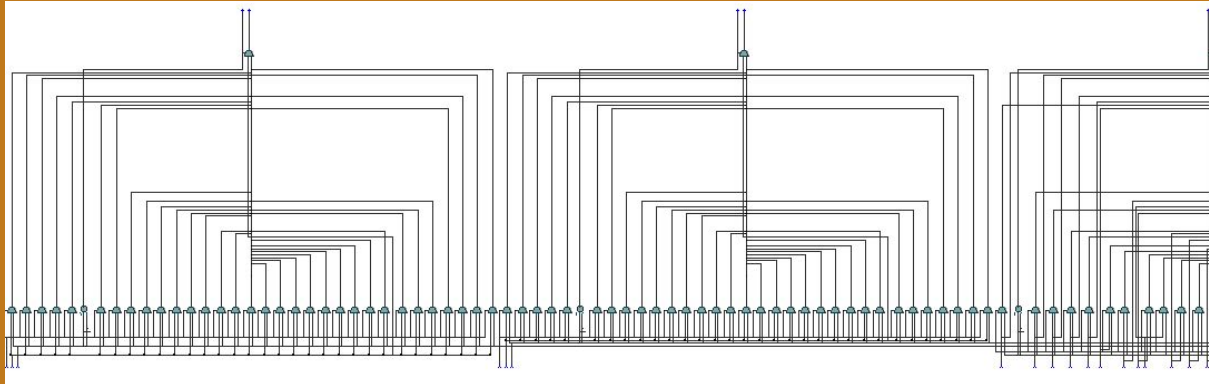
HuffmanTranslation.vhd

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

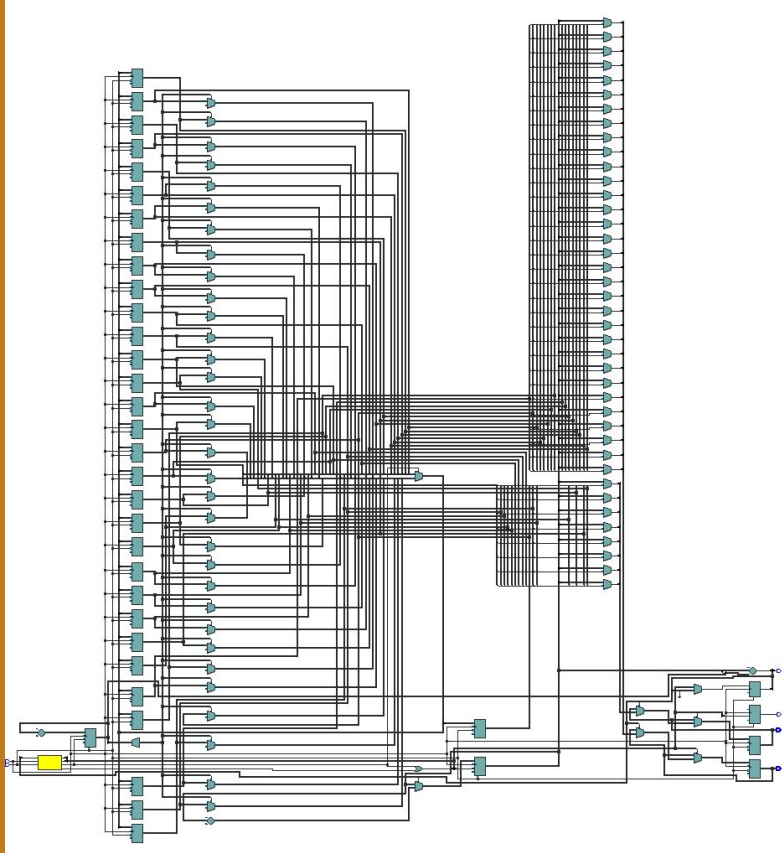
Schematics - ReadSort



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

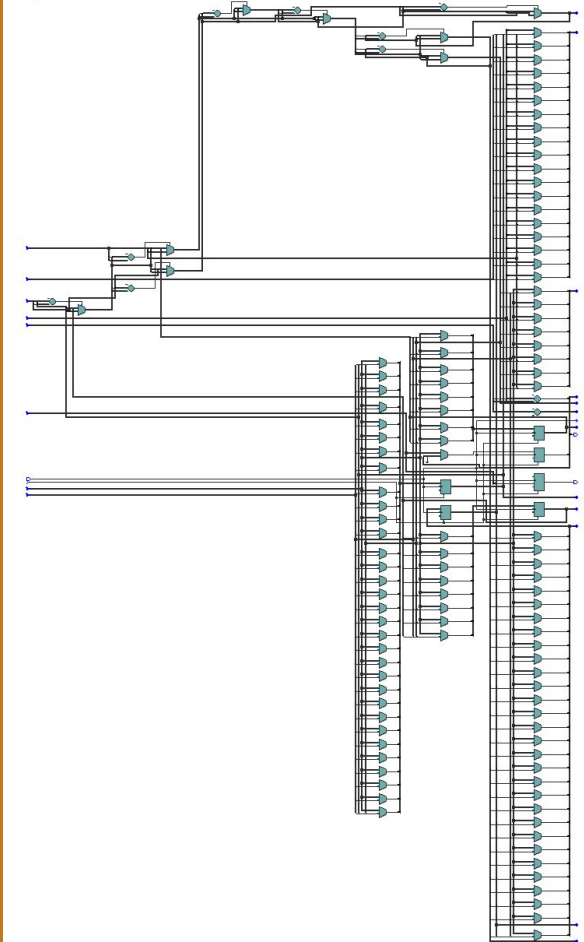


Schematics - NodeGenerator



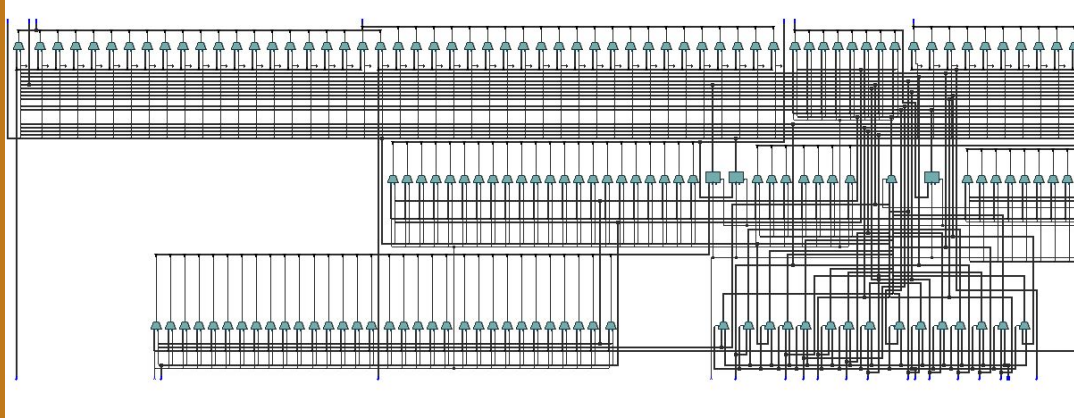
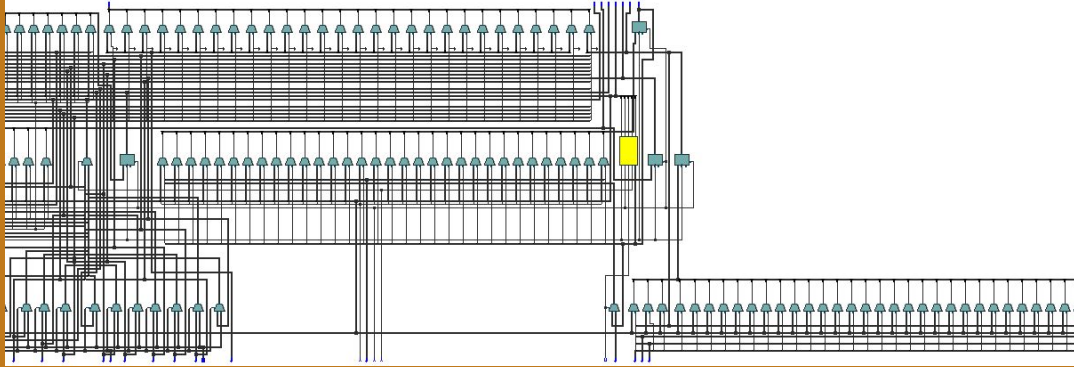
(Note : Line-line dynamic disubstitusi dengan nilai konstan ag dapat disintesis)

Schematics - NodeSorter



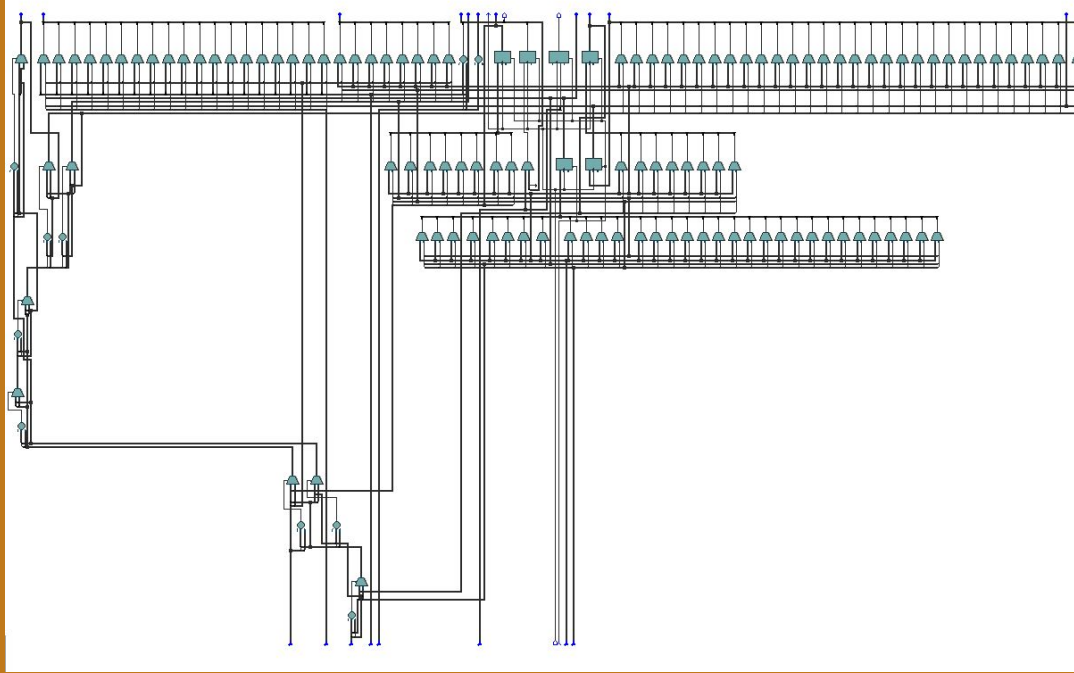
(Note : Line-line dynamic disubstitusi dengan nilai konstan agar dapat disintesis)

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

	Hops	
# headsortick	1	
# headsortreset	0	
# headsort/data_ready	0	
# headsort/sorted_char	01110110	01101001 01101111 01110000 01110010 01110011 01111011 01111010
# headsort/sorted_freq	2	1 3 2
# headsort/done	1	
# headsort/sorted_c...	espruv	espruv
# headsort/sorted_c...	11111322200...	11111322200... 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 64

Hasil file Output

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

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
v$Sim 281s: run
# ** Note: Node generation completed. Total nodes: 8

```

[illegible][illegible]