MERGESORT DALAM TINGKAT *REGISTER TRANSFER LOGIC* BERBASIS *FIELD PROGRAMMABLE GATE ARRAY*

Ferry Wahyu Wibowo
Dosen STMIK AMIKOM Yogyakarta
ferrywahyu@gmail.com

Abstrak

Telah dibuat rangkaian yang berfungsi sebagai pengurut (sorter) menggunakan FPGA Xilinx Spartan-3E. Penelitian ini menggunakan FPGA sebagai piranti utama dan alat bantu yang digunakan adalah seperangkat komputer yang terinstal software Xilinx ISE 9.2i yang digunakan untuk meng-upload kode program ke FPGA. Paper ini mengulas sorter dengan 4 buah masukan yang mempunyai panjang 8 bit, yang dirancang di tingkat register transfer logic (RTL). Pengujian kinerja piranti ditampilkan dalam bentuk diagram bentuk gelombang pada perangkat lunak ISE 9.2i. Hasil dari penelitian diperoleh bahwa komponen FPGA Spartan-3E yang dikonsumsi adalah 26 slices flip-flop dari 9.312 (1%), 112 4-masukan LUT dari 9.312 (1%), 73 slices dari 4,656 (1%) dan waktu yang diperlukan untuk kerja rangkaian mergesort sebesar 16,029ns.

Kata kunci: FPGA, program, RTL, sorter, xilinx.

Pendahuluan

Pengurutan (*sorting*) merupakan salah satu algoritma dalam bidang ilmu komputer yang sering digunakan sebagai platform kinerja system, salah satu aplikasinya dalam jaringan (Sungun, et. al., 2006). *Field programmable gate array* (FPGA) mempunyai kekhasan desain rangkaian dalam pemenuhan optimasinya, yang berbeda ketika menggunakan mikrokontroler maupun *personal computer* (PC). FPGA diharapkan dapat digunakan untuk mengetahui tingkat desain rangkaian seoptimal mungkin, walaupun untuk tingkat desain rangkaian mempunyai banyak ragam (Wibowo, 2010).

Harkins et. al. (2005) menyatakan bahwa pengurutan menggunakan mikroprosesor lebih efisien dibandingkan

menggunakan FPGA, namun hal ini karena ada beberapa faktor yang mempengaruhinya, yaitu *memory bandwidth*, kecepatan *clock*, komputasional algoritma dan kemampuan algoritma untuk di-*pipeline*. Sedangkan, Mihhailov et. al. (2010) menggunakan rekursi dalam membuat algoritma pengurutnya dan mengaplikasikan teknik optimasi algoritmik dan arsitektural sehingga didapatkan hasil yang lebih baik daripada berbasis perangkat lunak.

Paper ini mengimplementasikan FPGA sebagai pengurut data masukan pada tingkat *register transfer logic* (RTL). Jumlah masukan data yang digunakan ada 4 buah yang masing-masing masukan data mempunyai panjang 8 bit (*word*).

Field Programmable Gate Array (FPGA)

FPGA merupakan piranti yang dapat diprogram berbentuk *integrated circuit* (IC) dan tersusun atas modul-modul logik bebas yang dapat dikonfigurasi. FPGA memiliki komponen logika yang dapat diprogram (*programmable logic*) yang terdiri dari *logic blocks* dan *programmable interconnects*. *Logic blocks* dapat diprogram untuk membuat fungsi-fungsi gerbang dasar seperti AND, OR, XOR ataupun yang lebih kompleks seperti *decoder* serta fungsi matematika sederhana, dan modul-modul logik tersebut terhubung melalui kanal-kanal penjalaran yang dapat diprogram. (Wibowo, 2010).

FPGA pada umumnya merupakan larik 2 dimensi *logic blocks* (*slices*) yang memiliki elemen memori, seperti *flip-flop* sederhana atau blok-blok memori yang lebih kompleks. Suatu hirarki dari *programmable interconnects* memungkinkan *logic blocks* untuk dapat saling berhubungan sesuai kebutuhan perancang sistem, seperti suatu *breadboard* yang dapat diprogram. *Logic blocks* dan *programmable interconnects* dapat diprogram oleh pemakai / perancang dalam mengimplementasikan berbagai macam fungsi logika, sehingga diberi istilah *field-programmable*. Pengertian terprogram (*programmable*) dalam FPGA, mirip dengan interkoneksi saklar dalam *breadboard* yang bisa diubah oleh perancang desain. Kelebihan dari FPGA adalah dapat dikonfigurasi oleh *end user*, tidak memerlukan proses fabrikasi

karena tersedia cara untuk mendukung *chip customized - very large scale integration* (VLSI) dalam mengimplementasikan *logic circuit*, *instant manufacturing*, *very-low cost prototype* dan pemrograman yang singkat untuk fungsi dan kemampuan yang setara dengan *applied specific integrated circuit* (ASIC).

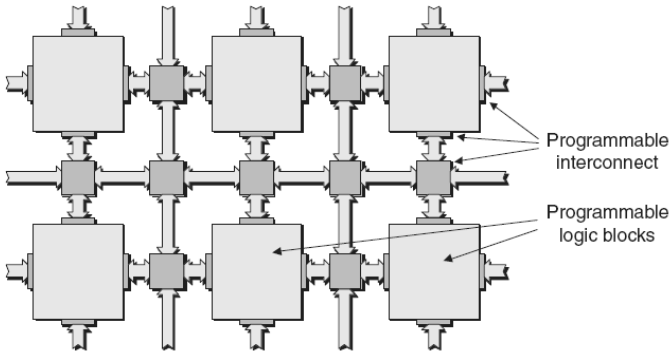
FPGA dapat diprogram menggunakan bahasa deskripsi Verilog dan *very high speed integrated circuit – hardware description language* (VHDL). Perkembangan FPGA sampai saat ini berlangsung dengan cepat dan banyak macam keluarga FPGA dengan kebutuhan perancangan dan perangkat perancangan (*design tools*) yang berbeda. Perusahaan yang memproduksi FPGA diantaranya adalah Xilinx, Altera, ACTEL, ATMEL, dan PLESSEY *Semiconductor*. Xilinx memproduksi beberapa jenis FPGA, yaitu VIRTEX, SPARTAN, XC3000, XC4000 dan XC5000. Pendekatan untuk menggabungkan *logic blocks* dan *interconnects* dari FPGA dengan *embedded microprocessor* dan periperal-periperal lainnya telah dilakukan. Tujuannya sebagai pelengkap dari sistem pada sebuah *chip* yang dapat diprogram.

Secara umum, FPGA akan lebih lambat jika dibandingkan dengan jenis *chip* yang lain, seperti *Application-Specific Integrated Circuit* (ASIC). Hal tersebut dikarenakan FPGA menggunakan daya yang besar dengan bentuk desain yang kompleks, walaupun demikian FPGA mempunyai harga yang murah, bisa diprogram mengikuti kebutuhan, dan kemampuan untuk diprogram ulang untuk mengkoreksi adanya *bugs*. Jenis FPGA dengan harga murah, biasanya tidak bisa diprogram dan dimodifikasi setelah proses desain dibuat (*fixed-version*). Chip FPGA yang lebih kompleks dapat diperoleh dari jenis FPGA yang dikenal dengan c*omplex-programmable logic device* (CPLD). FPGA pada awalnya dibuat sebagai pesaing dari CLPD dan bersaing pada bidang yang sama.

Arsitektur FPGA

Secara umum arsitektur bagian dalam dari IC FPGA terdiri atas tiga elemen utama yaitu *Input/Output Blocks* (IOBs),

Configurable Logic Block (CLB) dan Interkoneksi. *Configurable Logic Blocks* tersusun atas *look-up-table based complex structure* (struktur komplek berbasis tabel) dan *flip-flop* (FF) untuk implementasi rangkaian sekuensial. *Programmable Interconnect* berupa *wire segments* dan *programmable switches* yang menghubungkan antara *Configurable Logic Blocks* yang berbeda, sedangkan *input/output block* berperan sebagai *interface* antara *external package pin* dari *device* dan *internal user logic*. Gambar 1 mengilustrasikan arsitektur FPGA.



Gambar 1. Arsitektur FPGA (Maxfield, 2004)

Fungsi logik dan interkoneksi FPGA ditentukan oleh data yang tersimpan pada sel memori statik internal. Ada beberapa cara untuk membuat data konfigurasi ke dalam sel memori, yaitu dilakukan secara otomatis pada waktu catu daya diberikan maupun dengan membaca konfigurasi data dari eksternal serial atau byte paralel PROM (mode *master*) atau data dapat dituliskan pada FPGA dari *external device* (mode *slave* dan peripheral).

Proses Implementasi FPGA

Rancangan rangkaian yang akan diimplementasikan ke dalam FPGA dideskripsikan menggunakan VHDL (*Very High Speed Integrated*

Circuit Hardware Description Language), kemudian melakukan langkah optimasi, pemetaan teknologi, penempatan dan penjalaran.

Optimasi logik
Langkah ini memodifikasi ekspresi Boole untuk mengoptimalkan penggunaan sumber daya dengan tujuan meminimalisasi area dan kecepatan eksekusi atau kombinasi keduanya.

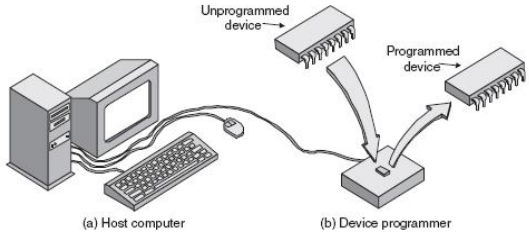
Pemetaan Teknologi
Pemetaan teknologi mentransformasikan ungkapan Boole ke dalam FPGA yaitu *logic blocks*, tujuan optimasinya untuk mengoptimisasi area dengan meminimalkan penggunaan *block* dan mengoptimisasi tunda dengan meminimalkan jumlah *stage* pada *critical path*.

Penempatan
Salah satu penempatan berbasis *simulated annealing* digunakan untuk menempatkan masing-masing *block* dalam larik FPGA, sehingga meminimalkan panjang seluruh interkoneksi yang diperlukan untuk penempatan.

Piranti Lunak Penjalaran
Piranti lunak penjalaran bertugas mengatur *FPGA's wire segments* dan menentukan *programmable switches* untuk menghubungkan FPGA *blocks*, memastikan seluruh koneksi telah terbentuk dan meminimalkan *propagation delay* pada *time-critical connections* (bagian dari rangkaian yang mempunyai *delay* terpanjang yang sangat menentukan waktu tunda keseluruhan).

Bagian Pemrograman
Bagian pemrograman mengkonfigurasi *chip* setelah tahap penempatan dan penjalaran. Seluruh proses pemrograman memakan waktu beberapa menit sampai beberapa jam, tergantung besar program yang digunakan. Pemrograman FPGA berbeda dengan pemrograman

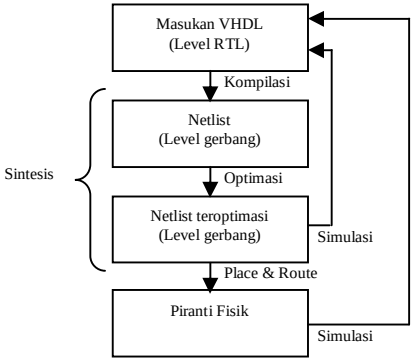
komputer yang sekuensial. Gambar 2 mengilustrasikan cara rekonfigurasi FPGA via port serial via PC.



Gambar 2. Pemrograman FPGA melalui PC (Maxfield, 2004)

VHDL bahasa pendukung FPGA

VHDL adalah bahasa deskripsi perangkat keras yang mirip dengan bahasa tingkat tinggi, seperti C, Pascal, Fortran dan sebagainya. VHDL melukiskan perilaku rangkaian elektronik atau sistem pada tata algoritma, bukan implementasi. Kompiler perangkat keras modern dapat membaca deskripsi tingkat tinggi sistem elektronik dan menterjemahkannya menjadi konfigurasi *bit string* yang digunakan untuk mengkonfigurasi sebuah *chip* FPGA. Gambar 3 mengilustrasikan aliran VHDL.



Gambar 3. Ringkasan aliran VHDL

Bagian kode VHDL terdiri dari 3 bagian pokok, yaitu deklarasi *library*, terdiri dari daftar semua *library* yang digunakan dalam desain; *entity*, menjelaskan pin-pin masukan dan keluaran dari rangkaian; dan *architecture*, terdiri dari kode VHDL yang sesuai untuk menjabarkan rangkaian.

Sebuah library merupakan kumpulan potongan kode yang digunakan dalam pemrograman. Struktur library, biasanya ditulis dalam bentuk *function*, *procedure* atau *component* yang ditempatkan di dalam *package* yang dikompile pada library tujuan. Untuk mendeklarasikan *library* pada desain, diperlukan dua baris kode, yaitu nama *library* dan klausa.

```
LIBRARY nama_library;
USE nama_library.nama_paket.bagian_paket;
```

Ada tiga paket *library* berbeda yang biasanya digunakan dalam sebuah desain, yaitu ieee.std_logic_1164 (dari *library* ieee), standar (dari *library* std) dan work (*work library*).

Mergesort

Sorter merupakan piranti yang tertanam algoritma untuk menyusun data masukan yang acak, kemudian mengurutkannya dengan membagi dua bagian masukan secara terpisah. Jika kedua masukan telah disusun, maka akan membentuk urutan data dari hasil penggabungan dua buah data masukan. Salah satu pengurutan cara yang digunakan adalah *mergesort*, cara kerja algoritma tersebut ditunjukkan pada gambar 4.

```
Masukan data : D0 6C 48 19
Masukan data tersebut dibagi menjadi dua :
D0 6C       |     48 91
Kedua masukan data terbagi tersebut disusun sendiri menjadi,
6C D0       |     48 91
Setelah itu dibuat menjadi,
48
6C D0       |     91
-------------------------
48 6C
D0          |     91
-------------------------
48 6C 91
D0
-------------------------
48 6C 91 D0
Sehingga nilai yang terurut adalah
48 6C 91 D0
```

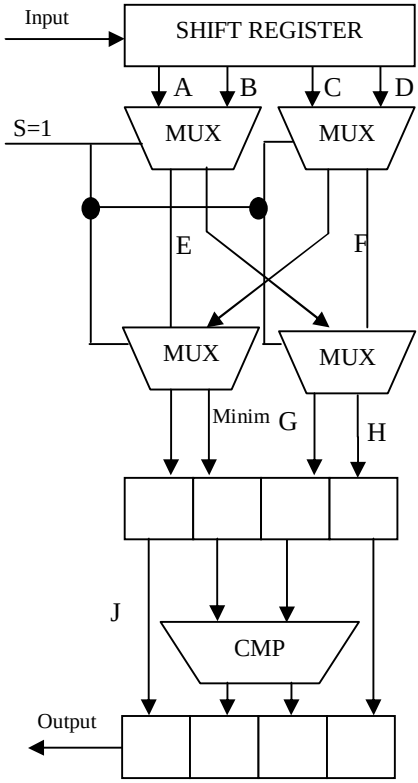
Gambar 4. Ilustrasi *mergesort*

Metodologi Penelitian

Rancangan Global

Rangkaian global dari pengurut (*sorter*) dalam paper ini menggunakan FPGA Xilinx Spartan-3E dengan 8-bit masukan dan keluaran. Sistem rangkaian pokok yang dirancang menggunakan VHDL terdiri dari *shift register*, multiplekser, rangkaian pembanding (*compare*) dan RAM. Komponen-komponen tersebut dibuat secara terpisah, kemudian dipanggil untuk digabungkan menjadi satu.

Komponen yang digunakan hanya memuat satu desain komponen saja, jika terdapat dua komponen yang sama dalam perancangan, maka cukup satu komponen saja yang dipanggil. *Clock* yang digunakan dalam perancangan *mergesort* dalam tingkat register transfer logic ini menggunakan 50 MHz atau sekitar 20 ns, sehingga waktu yang diperlukan untuk mengisi *shift register* sebesar 1280 ns.



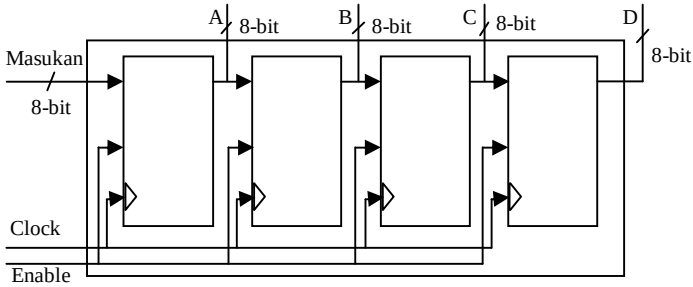
Gambar 5. Rancangan modul *mergesort*

Gambar 5 menunjukkan rancangan modul seluruh rangkaian *sorter*. Prinsip kerjanya dengan memasukkan 4 masukan 8-bit sinyal acak yang dipilih melalui aktif *enable* pada *shift register* dan perpindahan sinyal dalam register dikendalikan melalui *clock* aktif tinggi. Setelah terpilih 4 sinyal masukan, masing-masing multiplekser akan memilih 2 sinyal masukan sebagai nilai kecil dan besar (jika dipilih S = 0 maka akan dipilih nilai yang kecil diantara keduanya, sedangkan jika dipilih S = 1 maka nilai yang besar yang akan dipilih). Misalkan jika S = 0 maka nilai yang kecil akan dipilih, kemudian nilai dari kedua multiplekser ini akan dibandingkan lagi melalui muliplekser berikutnya, untuk mencari nilai yang lebih kecil lagi. Jika nilai terkecilnya sudah terpilih maka nilainya akan dimasukkan dan disimpan pada *buffer* terendah yakni bit 0 sampai 7 dan akan dibandingkan melalui rangkaian pembanding dengan masukan multiplekser, jika keluaran dan masukan multipleksernya sama, maka nilai masukan lain akan dipilih dan dimasukkan pada *buffer* bit 8 sampai 15. Sedangkan, jika S = 1 maka nilai yang besar akan dipilih, kemudian nilai kedua multiplekser akan dibandingkan lagi sebagaimana pemilihan S = 0. Namun, untuk mencari nilai yang lebih besar perlu dimasukkan pada *buffer* bit 24 sampai 31, kemudian nilainya akan dibandingkan melalui rangkaian *comparator*, sehingga didapatkan nilai masukan terbesar kedua yang ditempatkan pada bit 16 sampai 23. Sinyal akan dikeluarkan dari *buffer* dengan cara *clocking*.

Rangkaian *Shift-Register*

Rangkaian *shift-register* yang dirancang mempunyai 8-bit data masukan dan 32-bit data keluaran (A,B,C dan D). Nilai awal pada sinyal tersimpan dalam rangkaian ini adalah “00000000” yang berupa heksadesimal. Port rangkaian ini selain terdapat port masukan dan keluaran juga terdapat port masukan *clock* dan *enable*, masing-masing merupakan 1-bit. Sinyal *clock* bertugas merambatkan setiap masukan 8-bit sehingga sinyal masukan merupakan masukan serial, sedangkan

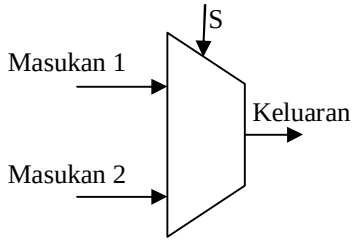
enable digunakan untuk memilih 4 masukan 8-bit yang akan diurutkan (lihat gambar 6).



Gambar 6. *Shift register*

Multiplekser

Modul multiplekser bertugas untuk memilih nilai terkecil jika S = 0 dan untuk memilih nilai terbesar jika S = 1. Bagan dari rangkaian multiplekser ditunjukkan pada gambar 7.

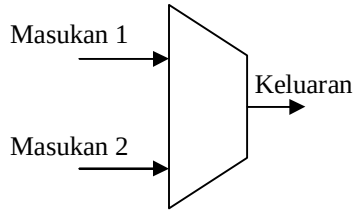


Gambar 7. Bagan Multiplekser

Rangkaian pembanding (*comparator*)

Rangkaian pembanding menggunakan prinsip seperti multiplekser. Prinsip kerja dari rangkaian ini adalah, jika nilai S = 0 maka nilai terkecil keluaran dari multiplekser akan dibandingkan dengan masukan multiplekser. Fungsi rangkaian ini untuk mencari nilai yang tidak sama, sehingga sinyal tersebut akan dimasukkan pada

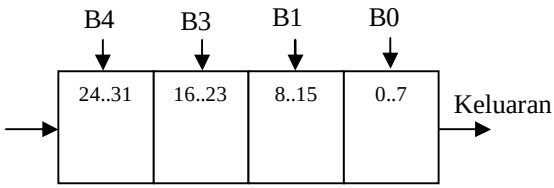
urutan tempat sebelumnya, begitu juga jika nilai S = 1, namun untuk pemilihan nilainya menggunakan nilai yang besar (lihat gambar 8).



Gambar 8. Bagan rangkaian pembanding

Rangkaian Keluaran

Rangkaian keluaran sistem *sorter* ini menggunakan keluaran 8-bit, sehingga nilai yang terurut tadi akan dikeluarkan satu-persatu dari sinyal byte 0, byte 1, byte 2 dan byte 3. Rangkaian penyimpan data yang digunakan berdasarkan *finite state machine* (lihat gambar 9).

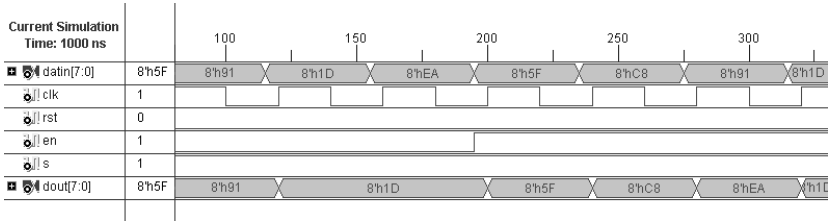


Gambar 9. Bagan rangkaian keluaran

Hasil dan Pembahasan

Penelitian ini menggunakan 4 masukan 8-bit yang bernilai acak yang dipilih datanya ketika masukan sinyal di-*enable*. Masukan 8-bit satu-persatu ditempatkan dan digeser seiring dengan *clock rising edge* sebagai data masukan pada sinyal *shift register* sinkron. Untuk memicu pengolahan data diperlukan *selectable* s, nilai s=’0’ digunakan untuk memilih masukan 8-bit yang paling rendah dan nilai s=’1’ digunakan untuk memilih masukan 8-bit yang paling tinggi. Sinyal masukan terpilih dan terkondisi tersebut dimasukkan dalam

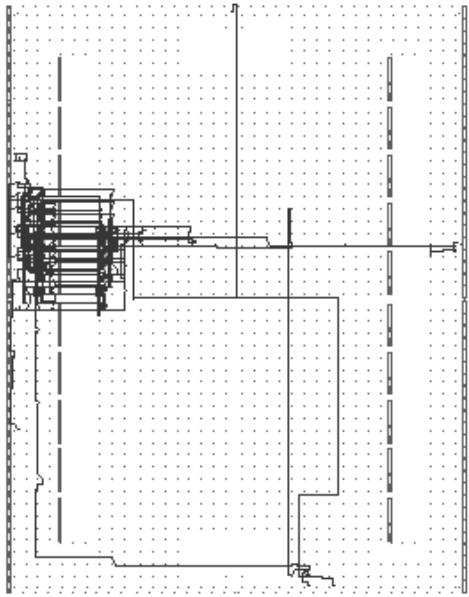
suatu *buffer*, kemudian untuk sinyal masukan terendah kedua dan sinyal masukan terbesar kedua dibandingkan untuk mendapatkan sinyal terendahnya yang akan ditempatkan pada sinyal masukan 8-bit port 8-15 dari rangkaian berupa *finite state machine.* Isi data akan dikeluarkan satu-persatu dari *register* seiring dengan *clock.* Diagram bentuk gelombang dari *mergesort* ditunjukkan pada gambar 10.



Gambar 10. Diagram bentuk gelombang dari *mergesort* 4x8-bit

Gambar 10 menggunakan *clock* sebesar 20ns dan waktu yang diperlukan sebelum eksekusi sinyal data masukan sebesar 5ns. Sinyal *reset* rst direndahkan untuk mendapatkan rangkaian *finite state machine* (FSM). Sinyal *select* s dibuat bernilai tinggi agar multiplekser yang digabungkan dengan komparator mendapatkan data yang bernilai besar, sedangkan sinyal *enable* en digunakan untuk mengaktifkan rangkaian *finite state machine* agar keluaran data dapat diurutkan dari nilai yang terkecil ke nilai yang terbesar.

Komponen yang diperlukan untuk membuat *mergesort* ini terdiri dari 1 FSM, 34 register *flip-flop* dan 6 komparator, yang terdiri dari 1 8-bit komparator sama dan 5 8-bit komparator lebih besar. Konsumsi FPGA yang digunakan 26 *slices flip-flop* dari 9.312 (1%), 112 4-masukan LUT dari 9.312 (1%), 73 *slices* dari 4,656 (1%). Konsumsi komponen pada FPGA ditunjukkan pada gambar 11.



Gambar 11. Floorplan dari rangkaian *mergesort*

Waktu yang diperlukan untuk kerja rangkaian *mergesort* ini sebesar 16,029ns, yang terdiri dari waktu logik sebesar 10,885ns (67,9%) dan waktu jalar 5,144ns (32,1%).

Kesimpulan

Penelitian ini dapat disimpulkan bahwa untuk membuat *mergesort* 4x8-bit pada tingkat register transfer logic yang mengimplementasikan FPGA dihasilkan sekitar 1% komponen yang dimiliki oleh FPGA Spartan-3E dan mempunyai waktu eksekusi logik sebesar 10,885ns dan waktu eksekusi jalar 5,144ns.

Daftar Pustaka

Harkins, J., El-Ghazawi, T., El-Araby, E., Huang, M. (2005). Performance and Analysis of Sorting Algorithms on the SRC 6 Reconfigurable Computer. *MAPLD International Conference*.

Maxfield, C. M. (2004). *The Design Warrior's Guide to FPGAs*. USA: Newnes.

Mihhailov, D., Sklyarov, V., Skliarova, I., Sudnitson, A. (2010). Optimization of FPGA-based Circuits for Recursive Data Sorting. *12th Biennial Baltic Electronics Conference (BEC2010)*, 129-132.

Sungun, S., Şenol, Y, Gündüzal, P. (2006). Minimal Sorting Network Realization on FPGA for Genetic Programming. *Erciyes Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 22*(1-2), 26-36.

Wibowo, F. W. (2010). Implementasi FPGA untuk Pengukuran Daya Listrik. *Seminar Nasional Fisika (SNF 2010)*, FI106-1 – FI106-6.

Wibowo, F. W. (2010). The Conservative Structure of Synthesizing Read Only Memory Design Using VHDL on FPGA. *International Seminar on Industrial Engineering and Management, 4*, 231-235.