 ISSN: 1411-3201

FINITE STATE MACHINE UNTUK PENGENDALI ELEVATOR BERBASIS FIELD PROGRAMMABLE GATE ARRAY

Ferry Wahyu Wibowo
STMIK AMIKOM Yogyakarta
ferrywahyuwibowo@scientist.com

ABSTRAKSI

Paper ini membahas mengenai penggunaan diagram finite state machine (FSM) pada software ISE 9.2i yang digunakan dalam pembuatan pengendali elevator pada field programmable gate array (FPGA). State diagram merupakan salah satu fitur desain masukan yang terdapat pada software ISE. Proses desain masukan state diagram merupakan finite state machine yang terdiri dari masukan, keluaran dan state keadaan. State keadaan dapat berupa konkuren dan sekuensial yang diberlakukan dalam suatu state untuk eksekusi proses tertentu.

Kata Kunci : *Elevator, FPGA, FSM*

PENDAHULUAN

Field programmable gate array (FPGA) mempunyai tiga masukan pemrograman yaitu skematik, bahasa deskripsi perangkat keras dan state diagram (Wibowo, 2011). Desain skematik digunakan untuk membangun piranti didasarkan pada blok rangkaian yang sifatnya *drag-place* dan menghubungkan antar blok dengan pengkabelan (*wiring*) pada *working windows*. Prinsip ini pada dasarnya sangat mudah sekali, namun mempunyai kelemahan dalam pengembangan rangkaiannya karena sifatnya yang rumit dan tidak bebas jika ingin merancang rangkaian yang berbeda dari pilihan blok rangkaian yang sudah disediakan, ditambah dengan waktu yang diperlukan cukup lama jika ingin mendesain menggunakan gerbang logika dasar. Kelemahan dari masukan untuk mengkonfigurasi FPGA yang menggunakan desain skematik dapat diatasi menggunakan desain masukan bahasa deskripsi perangkat keras (*hardware description language* atau disingkat dengan HDL). Prinsip bahasa deskripsi ini cukup efektif dalam mendesain suatu blok rangkaian secara bebas dengan menentukan watak / sifat rangkaian yang diperlukan (Wibowo, 2010). Penggunaan *state diagram* walaupun tidak begitu familiar sebagaimana desain skematik dan HDL, namun pada *software* ISE mempunyai fitur *state diagram* sebagai salah satu masukan konfigurasi untuk FPGA (Wibowo, 2011). Desain masukan *state diagram* akan dikonversi menjadi HDL dengan pemrograman yang kompleks.

Penelitian Munoz, et. al. (2008) mengimplementasikan FPGA Spartan 3 untuk memvalidasi lima algoritma untuk sistem elevator dengan menerapkan prinsip sistem *elevator* yang terdiri dari *elevator group control system* (EGCS) dan *micro-processed sub-system* yang menggunakan *local control system* (LCS) pada setiap *elevator*. Algoritma tersebut diaplikasikan pada LCS, sedangkan EGCS berdasarkan pada logika fuzzy (FEGCS). FEGCS berjalan pada PC yang menentukan algoritma terbaik untuk mengurangi waktu tunggu dan konsumsi daya. Metode sederhana dalam mengimplementasikan *reconfigurable elevator controller* menggunakan FPGA Xilinx Spartan 3AN dikemukakan oleh Jayawardana, et. al. (2011) dengan implementasi jumlah lantai dengan masukan dan keluaran tertentu. Implementasi yang sederhana dimulai dari 3 lantai, sedangkan kendali sistem diimplementasikan dalam *ladder logic* pada *programmable logic control* (PLC). Kode *very high speed integrated circuit* (VHSIC) *hardware description language* (VHDL) difungsikan untuk kendali elevator yang dapat direkonfigurasi untuk jumlah lantai dalam FPGA.

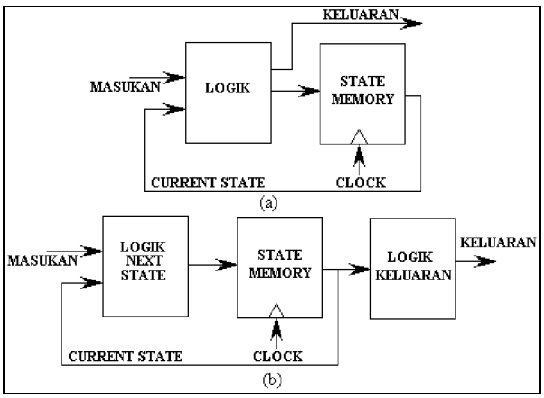
Paper ini membahas mengenai sistem elevator menggunakan kode VHDL yang diaplikasikan pada FPGA. Metode pendekatan yang digunakan untuk menjalankan sistem *elevator* adalah *Finite State Machine* (FSM). Hasil yang didapatkan dari sintesis kode VHDL adalah konsumsi komponen-komponen FPGA.

METODE PENELITIAN

Finite State Machine (FSM) sering digunakan oleh perancang logik digital untuk mengkonfigurasi sistem sinkronus dengan proses sekuensial. FSM mempunyai dua jenis struktur, yaitu Mealy dan Moore. FSM mempunyai *state memory* yang digunakan untuk menyimpan *current state* dari mesin, dimana beberapa *flip-flop* di-*clock* oleh sinyal *clock* tunggal. *State vector* (ada yang menyebutnya sebagai *current state* atau hanya *state* saja) merupakan data yang tersimpan pada *state*

 ISSN: 1411-3201

memory. *Next state* dari mesin merupakan fungsi dari *state vector* dan masukan. Keluaran FSM jenis Mealy merupakan fungsi *state vector* dan masukan (lihat gambar 1(a)), sedangkan keluaran FSM jenis Moore merupakan fungsi dari *state vector* saja (lihat gambar 1(b)).



Gambar 1. Struktur *Finite State Machine*

Logik pada *state machine* didefinisikan dalam sebuah proses. Mesin Mealy dan Moore dapat terjadi *spike* pada keluaran logiknya, namun pada keadaan tertentu *spike* dapat diabaikan. Sinyal data dapat terjadi *spike* sesaat setelah terjadi *active clock edge* pada desain sinkronus, padahal sinyal data harus stabil. Kestabilan sinyal data dapat dipenuhi dengan cara menyamakan keluaran dengan *state machine*, dan menambahkan keluaran ter-*clock* pada mesin Moore / Mealy.

Sistem pengendali *elevator* pada penelitian ini didasarkan pada proses pemilihan lantai. Prinsipnya hanya memilih lantai yang akan dituju, berhenti pada suatu lantai tertentu, menggerakkan motor agar kotak *elevator* naik atau turun menggunakan prinsip *finite state machine* Moore. Prinsip tersebut digambarkan dalam bentuk *state diagram* yang kemudian dituliskan dalam bentuk kode bahasa deskripsi VHDL untuk mengkonfigurasi FPGA Xilinx Spartan 3E.

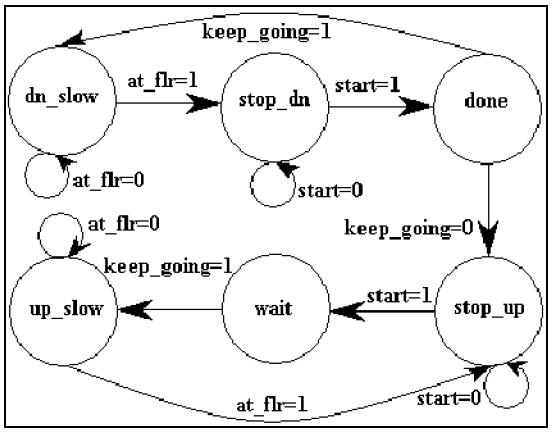
HASIL PENELITIAN DAN PEMBAHASAN

Prinsip kerja sistem pengendali elevator menggunakan model *finite state machine* Moore untuk rangkaian logika sekuensial. Model ini sangat membantu perancangan sistem yang menggunakan urutan proses. Sinyal pergerakan elevator yang didasarkan pada prinsip *Finite State Machine* mempunyai tiga masukan sinyal, yaitu sinyal masukan, proses *state* keadaan dan sinyal keluaran. Sinyal masukan *clock* (clk) terhubung dengan *clock* eksternal 50 MHz pada FPGA Xilinx Spartan-3E *starter kit*; sinyal masukan *reset* (rst) merupakan sinyal yang digunakan untuk me-*reset* proses keadaan perubahan sinyal, sinyal masukan ini menggunakan penekanan tombol untuk membangkitkan kondisi logika; sinyal *keep_going*, *start*, *at_flr* merupakan sinyal masukan untuk melakukan kegiatan yang digunakan oleh *elevator*; sinyal flr merupakan sinyal keluaran untuk menunjukkan keberadaan kotak *elevator* pada suatu lantai tertentu; sinyal *brake*, *dn*, *up* merupakan sinyal keluaran untuk menunjukkan kondisi *elevator*; sinyal *sreg* merupakan tipe data yang digunakan untuk mendefinisikan suatu kondisi *state*. *Encoding* dilakukan secara sekuensial dan *default* secara *software*; sinyal *pr_sreg*, *next_sreg* menggunakan jenis data *sreg* yang mempunyai *state* : *dn_slow*, *stop_dn*, *done*, *stop_up*, *wait*, dan *up_slow*; *state done* dan *wait* merupakan *state* yang mempunyai keadaan transisi *elevator* untuk menunggu adanya sinyal penekanan tombol. Keluaran yang digunakan untuk mengetahui apakah alurnya sudah berjalan sesuai dengan yang diharapkan, maka diperlukan alat yang digunakan untuk mendeteksi perjalanan *state* dan keluarannya. Piranti yang digunakan adalah *light emitting diode* (LED) yang terdapat pada *board* FPGA Xilinx Spartan-3E *starter kit*. LED 6 dan 7 difungsikan untuk menunjukkan lantai, sedangkan LED 0, 1 dan 2 difungsikan untuk menunjukkan operasi *state* yang sedang berlangsung.

State awal adalah keadaan *dn_slow* yang mempunyai syarat untuk melanjutkan ke keadaan berikutnya, yaitu *state stop_dn*, maka harus memasukkan kondisi *at_flr* berlogika *high*. Jika tidak didapatkan logika *high* pada kondisi *at_flr* maka *state* tersebut tidak akan melanjutkan ke *state* berikutnya. Logika *high* tersebut digunakan untuk menunjukkan bahwa kondisi telah terpenuhi dengan berada pada lantai yang dituju, sehingga *state* akan melanjutkan ke *state* berikutnya, yaitu *state stop_dn*. *State stop_dn* merupakan *state* yang mempunyai tugas untuk menunggu masukan *start* untuk berlogika *high*. Jika sinyal logika *high* pada *state stop_dn* telah terpenuhi maka akan menunggu sinyal masukan logika *high* dari *keep_going*. Jika telah terpenuhi logika *high* pada *keep_going*, maka *state* akan berpindah ke keadaan *dn_slow*, namun jika mendapatkan sinyal masukan *low* pada *keep_going* maka *state* akan berpindah ke *state stop_up*. *State stop_up* merupakan *state* yang mempunyai tugas untuk menunggu sinyal masukan yang berlogika *high*

 ISSN: 1411-3201

pada *start*. Jika sinyal masukan *high* telah terpenuhi pada *start*, maka akan menunggu sinyal masukan berlogika *high* pada *keep_going*. Jika sinyal masukan berlogika *high* telah terpenuhi pada *keep_going*, maka sinyal akan menuju ke *state* berikutnya, yaitu *state up_slow*. *State up_slow* merupakan *state* yang berfungsi untuk menunggu sinyal masukan berlogika *high* pada *at_flr*. Jika telah terpenuhi sinyal masukan berlogika *high* pada *at_flr*, maka *state* akan berpindah ke *state* berikutnya, yaitu *state stop_up*. *State diagram* dari masukan, *state*, dan keluaran sistem pengendali *elevator* ini ditunjukkan sebagaimana pada gambar 2.



Gambar 2. *State diagram* pengendali *elevator*

Hasil sintesis kode VHDL yang diperoleh dari sistem pengendali *elevator* berbasis FPGA Spartan-3E menghasilkan konsumsi komponen yang terdiri dari *slice*, *slice flip-flop*, 4-masukan *Look-Up-Table* (LUT), *input-output* (IO), *bonded* IOB (IOB) dan *global clock* (GCLK) sebagaimana ditunjukkan pada tabel 1.

Tabel 1 Konsumsi Komponen FPGA Spartan-3E untuk Sistem Pengendali *Elevator*

Komponen	Jumlah terpakai	Jumlah Keseluruhan	Prosentase (%)
Slice	10	4656	0%
Slice Flip-Flop	16	9312	0%
4 masukan LUT	13	9312	0%
IO	10	-	-
IOB	10	232	4
GCLK	1	24	4

Analisis periode *clock* yang digunakan sebesar 3,551ns atau 281,611 MHz. Periode *clock* tersebut didapatkan dari 2,307ns untuk logik (65%) dan 1,244ns untuk *route* (35%)).

KESIMPULAN

Pengendali *elevator* berbasis FPGA lebih mudah jika menerapkan prinsip *finite state machine* untuk proses sekuensial. Proses ini didapatkan hasil yang lebih spesifik dengan penggambaran *state diagram*. Pengendali *elevator* pada paper ini belum menggunakan sistem yang lebih kompleks untuk menerapkan penekanan tombol permintaan *elevator* di masing-masing lantai, buka tutup pintu *elevator*, deteksi kelebihan muatan, tombol dan komunikasi darurat. Paper ini hanya menjelaskan pengendali *elevator* untuk kondisi naik, turun dan sampai lantai yang dituju saja.

DAFTAR PUSTAKA

Jayawardana, H.P.A.P, Amarasekara, H.W.K.M., Peelikumbura, P.T.S., Jayathilaka, W.A.K.C., Abeyaratne, S.G., Dewasurendra, S.D., 2011, Design and Implementation of a Statechart Based Reconfigurable Elevator Controller, *6th IEEE International Conference on Industrial and Information Systems* (ICIIS 2011), pp. 352-357.

Munoz, D. M., Llanos, C. H., Ayala-Rincon, M., van Els, R. H., 2008, FPGA Implementation of Dispatching Algorithms for Local Control of Elevator Systems, *IEEE International Symposium on Industrial Electronics* (ISIE 2008).

Wibowo, F. W., 2010, Mergesort dalam Tingkat Register Transfer Logic Berbasis Field Programmable Gate Array, *Jurnal Ilmiah DASI*, Vol. 11 No. 4, pp. 33-47.

Wibowo, F. W., 2011, System on Chip untuk Mesin Pengepkan Barang Berbasis FPGA, *Prosiding Seminar Teknik Informatika* (STI 2011), pp. B-52-B-59.

Wibowo, F. W., 2011, Interoperability of Reconfiguring System on FPGA Using a Design Entry of Hardware Description Language, *Proceedings of International Conference on Advances in Computing, Control, and Telecommunication Technologies 2011*, pp. 79-83.