Penggunaan Graf untuk Merepresentasikan Desain Digital Sekuensial dengan Metode Finite State Machine (FSM)

Zulfikar Hakim
Jurusan Teknik Informatika ITB, e-mail : zulfikar_165@students.itb.ac.id

Abstract *– Graf merupakan salah satu cabang Matematika Diskrit yang memiliki sangat banyak aplikasi dalam disiplin ilmu lainnya. Salah satunya adalah perancangan desain digital yang memanfaatkan rangkaian sekuensial. Salah satu metode perancangan rangkaian sekuensial adalah dengan memanfaatkan Finite State Machine (FSM). Untuk melakukan perancangan FSM inilah, penggunaan graf sangat mempermudah perancangan rangkaian sekuensial yang akan digunakan. FSM sendiri banyak digunakan dalam banyak peralata elektronik yang memanfaatkan otomatisasi elektris seperti mesin cuci elektrik, lift, dan lain sebagainya.*

Kata Kunci*: graf, Finite State Machine, desain digital, rangkaian sekuensial*

1. PENDAHULUAN

Graf merupakan salah satu cabang dalam matematika diskrit yang memiliki banyak aplikasi dalam disiplin ilmu lainnya. Graf banyak merepresentasikan objek-objek diskrit da hubungan di antara objek-objek tersebut.

Karena graf sangat representatif dalam merepresentasikan objek-objek diskrit – sesuai dengan definisinya di atas – graf sangat banyak digunakan dalam menyelesaikan banyak masalah. Berikut ini adalah beberapa masalah yang agak rumit namun dapat diselesaikan dengam mudah dengan graf :

1. *Traveling salesperson problem.* Graf digunakan untuk menyelesaikan masalah jarak terpendek seorang salesperson dalam turnya pada kota-kota yang jaraknya diketajui. Graf telah menyelesaikan masalah ini dengan metode Hamilton.
2. *Chinese Postman Problem.* Permasalahan yang dikemukakan adalah bagaimana seorang tukang pos dapat melewati jalan tepat sekali menuju setiap alamat tujuan surat.
3. *Graf coloring.*

Selain contoh di atas, graf juga digunakan dalam menyelesaikan masalah ketika seorang desainer digital ingin membuat otomasi mesin yang memanfaatkan *state* tertentu.

Untuk menjelaskan apakah *state* itu sebenarnya, marilah menarik suatu contoh yang menarik. Seekor kucing peliharaan angora (yang telah distrerilisasi – untuk mempermudah deskripsi masalah ini tentunya) biasanya memiliki hanya 4 kegiatan di dalam hidupnya : makan/minum, tidur, bermain dengan kucing peliharaan tetangga majikannya, serta dimandikan oleh sang majikan. 4 kegiatan itulah yang dapat dianalogikan sebagai state.

Hal ini sekaligus dapat digunakan untuk elektronika. Marilah mengambil satu contoh lagi yang lebih serius : mesin cuci otomatis berpintu depan. Mesin cuci jenis ini biasanya bersifat *full automatic* dan memiliki beberapa state tertentu untuk mewakili apa yang akan dikerjakannya pada suatu quantum waktu tertentu : mengisi air, berputar (membersihkan baju di dalamnya), membuang air, dan *spinning* (mengeringkan pakaian dengam memanfaatkan G-*force*).

Langkah-langkah di atas biasanya direpresentasikan oleh sebuah diagram FSM – yang pada hakikatnya adalah sebuah graf.

Setiap simpul mewakili *state*, dan setiap sisi mewakili dalam kondisi apakah suatu state akan pindah ke state lainnya atau tetap di state tersebut – yang biasanya memiliki gelang dalam simpulnya.

Berikut ini akan dijelaskan bagaimana suatu graf menjadi sangat penting dalam merepresentasikan suatu state dalam perancangan desain digital dengan rangkaian logika sekuensial, serta bagaimana metode perancangan FSM itu sendiri.

2. FINITE STATE MACHINE : DEFINISI, REPRESENTASI, DAN PROSES PENDESAINAN

2.1. Definisi formal *Finite State Machine (FSM)*

FSM adalah sebuah formalisme matematis yang sesungguhnya terdiri dari :

- Himpunan *state*. Dalam graf, *state* biasanya dinyatakan dalam sebuah *node* yang berisi keterangan mengenai nama state serta representasi *state* dalam bilangan biner
- Himpunan input dan himpunan output. Input direpresentasikan dengan sisi berarah yang dapat jadi merupakan suatu kondisi bagaimana state dapat berpindah ke *state* lainnya atau tetap pada *state* yang sama. Himpunan output menyatakan output yang diberikan dalam sebuah state dan direpresentasikan dalam node bersama dengan state yang telah diberikan.
- Suatu *initial state* – *state* awal. State awal merupakan suatu *state* ketika sistem dimulai. Direpresentasikan secara grafis dengan sebuah sisi berarah tanpa *state* sumber yang mengarah kepada *state* awal.
- Suatu kondisi bagaimana *state* dapat berpindah. Sekalipun input bisa jadi menjadi kondisi suatu *state* berpindah, namun input tidak selalu menjadi satu-satunya kondisi. Contohnya adalah ketika sistem menggunakan register lokal untuk menyimpan suatu nilai dalam sebuah *datapath.*

2.2 Beberapa contoh format representasi Finite State Machine

Beberapa representasi dapat digunakan untuk merepresentasikan suatu *state*. Untuk membahas bagaimana *state* dapat direpresentasikan, kita akan mengambil suatu contoh masalah yang akan digunakan dalam berbagai kesempatan dalam membahas FSM dalam makalah ini.

Misalkan sebuah sistem dengan lampu tiga warna dikontrol oleh tombol power. Ketika tombol ditekan, lampu akan bergantian menyala setiap satu detik sekali dengan urutan merah – kuning – hijau. Namun, ketika tombol tidak ditekan lampu akan mati dan kembali menyala jika ditekan dengan urutan yang tetap : merah menyala pertama

Untuk menyelesaikan masalah di atas, kita akan menggunakan tiga representasi FSM yang keseluruhannya memang dapat digunakan untuk mendeskripsikan FSM – namun keefektifannya sama sekali tidak identik.

2.2.1. Representasi diagram waktu (*timing diagram*)

Tiap bit dalam komponen desain digital pada hakikatnya hanya memiliki dua buah kemungkinan nilai : 1 atau 0. Timing diagramlah yang merepresentasikan nilai ini.

Permasalahannya adalah diagram tersebut tidak menggambarkan secara eksplisit bagaimana suatu state dapat dikonversikan menjadi suatu kode biner tertentu untuk dapat dikenali oleh sistem desain digital yang akan dirancang. Tipe ini juga tidak merepresentasikan bagaimana suatu state dapat berubah ke state lainnya. Hal ini

sedikit mempersulit desainer ketika harus membaca FSM yang rumit dengan banyak input, output, dan juga kondisi perubahan state. Representasi FSM dengan diagram waktu dapat dilihat di gambar 1 lampiran makalah ini.

Perubahan nilai output dan perubahan *state* dipengaruhi oleh *clock*, suatu denyut yang selalu berubah nilainya menjadi 1, 0, dan menjadi satu lagi secara periodik. Pengaruh denyut clock pada register dan perubahan state berada di luar cakupan makalah ini.

2.2.2 Tabel kebenaran

Tabel kebenaran sangatlah penting untuk merancang suatu sistem digital, pembuatan tabel kebenaran memang merupakan bagian dari perancangan desain digital dengan state di dalamnya. Namun tanpa menggunakan diagram FSM yang direpresntasikan dengan graf, pembacaannya akan sedikit rumit. Contoh representasi FSM dengan tabel kebenaran dapat disaksikan pada tabel 1 lampiran makalah ini.

Tabel kebenaran ini memang menampilkan setiap informasi yang dibutukan oleh setiap state dan kondisi perpindahnnya, namun masih agak susah dibaca karena tidak secara grafis merepresentasikan kondisi yang telah dibicarakan sebelumnya.

2.2.3. Representasi dengan diagram FSM berupa graf

Seperti yang dijelaskan pada bab sebelumnya, representasi FSM dengan graf adalah dengan menyertakan informasi nama *state*, kode *state*, output pada *node*, serta informasi syarat perubahan *state* pada *edge*. Representasi tersebut secara detail telah dilampirkan, tepatnya pada gambar 2. Pada gambar, output 001 adalah urutan nyala lampu dengan format merah – kuning – hijau. Sehingga 001 berarti merah mati, kuning mati, serta hijau hidup. Hal ini hanyalah bertujuan untuk mempermudah penulisan.

Di dalam representasi ini, kedudukan *edge* sangatlah penting untuk melihat sifat perpindahan *state*. *Edge* yang berbentuk gelang berarti *state* tidak berubah pada suatu kondisi tertentu, seperti gelang yang terdapat pada *state* menunggu. Gelang tersebut memiliki makna *state* tidak akan berubah jika x (input tombol) adalah 0 (mati)

Representasi ini sangatlah cocok dengan kebutuhkan kita, dengan memperlihatkan setiap informasi *state*, output, input, dan kondisi yang dibutuhkan *state* untuk berpindah ke *state* lainnya, atau tetap pada *state* tersebut.

2.3. Pembuatan *cotroller* berbasiskan FSM

Untuk melihat lebih jelas bagaimana suatu graf yang sederhana dapat menghasilkan rangkaian rumit yang bersifat low-level (berupa gerbang AND,OR,NOT, dan sebagainya), maka suatu metode untuk membuat *controller* tidak dapat dilewat pembahasannya. Berikut ini adalah beberapa langkah yang dapat digunakan untuk membuat sebuah *controller* yang mengadopsi FSM yang telah diberikan.

Berikut ini adalah bagaimana membuat sebuah controller dari FSM yang telah diketahui diagram graf FSM-nya :

- Langkah pertama
  - Dapatkan graf FSM-nya. Graf dapat merepresentasikan bagaimana kebiasaan dari sebuah FSM dan hubungannya dengan input, output, kode *state*, dan bagaimana suatu *state* berpindah ke state lainnya atau tetap di dalam *state*nya.
- Langkah kedua
  - Membuat arsitektur umum, yaitu suatu arsitektur yang menghubungkan antara suatu

rangkaian kombinasional – suatu rangkaian yang outputnya bergantung penuh kepada perubahan input dan gerbang logika di dalamnya – dan *register state* beserta input dan output yang telah dideskripsikan di dalam FSM

- Langkah ketiga
  - *State encoding* ; memberikan kode *state* unik untuk setiap *node* di dalam graf yang merepresentasikan kode *state*. Banyaknya bit dalam kode state akan merepresentasikan berapa banyak *state* yang akan digunakan. Misalnya, kode 2 bit akan dapat merepresentasikan 4 state, sedangkan kode 1 bit hanya merepresentasikan 2 *state* saja.
- Langkah keempat
  - membuat tabel *state* berisi tabel kebenaran seperti representasi FSM yang telah dijelaskan di atas
- Langkah kelima
  - Mengimplementasikan logika kombinasional yang telah didapatkan dari tabel sebelumnya

3. FSM DAN IMPLEMENTASI RANGKAIAN

Untuk melihat bagaimana graf menjadi peranan yang sangat penting dalam pembuatan implementasi FSM ke dalam rangkaian logika kombinasional, saya akan mencoba mengimplementasikan masalah yang telah dibahas pada bab sebelumnya pada sebuah rangkaian kombinasional melalui langkah-langkah yang telah disebutkan pada bab 2.3.

3.1. Mendapatkan graf yang merepresentasikan FSM

Graf yang merepresentasikan FSM telah dijabarkan dalam bab sebelumnya dan terdapat pada lampiran, gambar 2.

3.2. Membuat arsitektur standar yang sesuai dengan keperluan sistem yang akan dibuat

Arsitektur standar untuk keperluan dapat dilihat di lampiran gambar 3. Dalam arsitektur tersebut terdapat suatu rangkaian kombinasional yang digambarkan dengan diagram blok (*block diagram*). Di dalamnya akan terdapat rangkaian yang mengadopsi keperluan yang telah dijabarkan di dalam graf diagram FSM yang telah kita buat sebelumnya.

State register berfungsi untuk menyimpan nilai state yang telah di-*encode* sebelumnya sehingga sistem mengenali pada state manakah sistem sedang berada. Cara kerja register berada di luar jangkauan makalah ini.

3.3. *States Encoding*

Untuk dapat dikenali oleh sistem dan *state register*, maka setiap state harus memiliki kode uniknya masing – masing. Oleh karena itu setiap *state* akan diberikan kodenya sendiri-sendiri dengan panjang 2-bit. Hal ini dikarenakan *state* yang dibutuhkan hanyalah ada empat dan kemungkinan nilai pada angka biner 2 bit adalah tepat 4.

Berikut ini adalah kode untuk masing – masing state :

Nama State	Kode State
MENUNGGU	00
MERAH	01
KUNING	10
HIJAU	11

Kode state juga telah dimasukkan ke dalam graf FSM yang ada pada gambar 1 lampiran makal ini.

3.4.Membuat tabel *state* berupa tabel kebenaran

Tabel kebenaran dapat dilihat di lampiran,

tabel 1

3.5. Implementasi rangkaian.

Berdasarkan *state* tabel yang telah dibuat sebelumnya, maka setiap output akan memiliki persamaannya sendiri sehingga menghasilkan output yang diinginkan, apakah bernilai satu atau nol. Berikut ini adalah rangkaian yang telah diimplementasikan, digambarkan pada gambar 3 pada lampiran makalah ini.

4. KESIMPULAN

Graf sangat merepresentasikan FSM, baik apa saja kondisi yang dibutuhkan FSM untuk berpindah state atau tetap pada state-nya, output yang dihasilkan, dan yang paling utama adalah sifat dari FSM itu sendiri.

DAFTAR REFERENSI

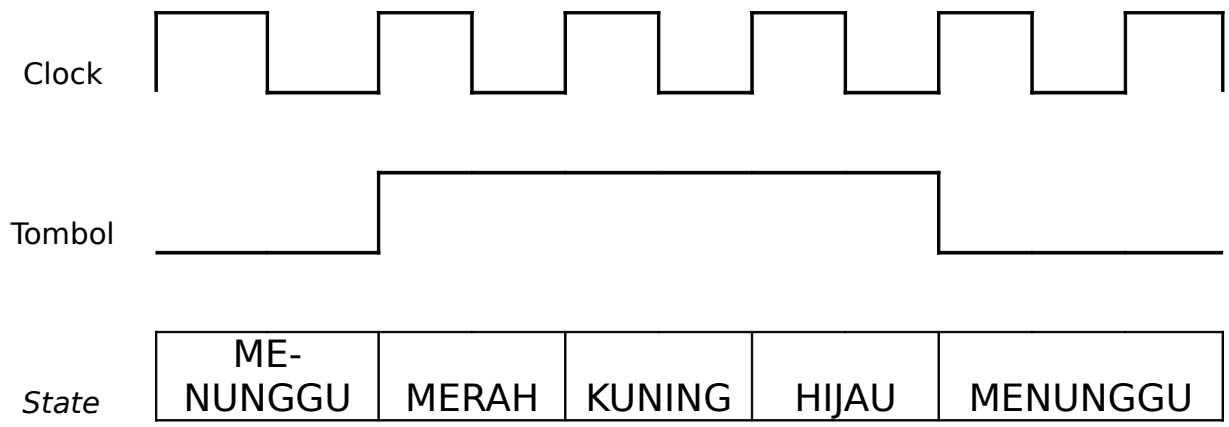
[1] Munir, Rinaldi. “Diktat Struktur Diskrit”

[2] Vahid, Frank.2007. *Digital Design*. New Jersey: John Wiley & Sons, Inc. Page 113 - 127

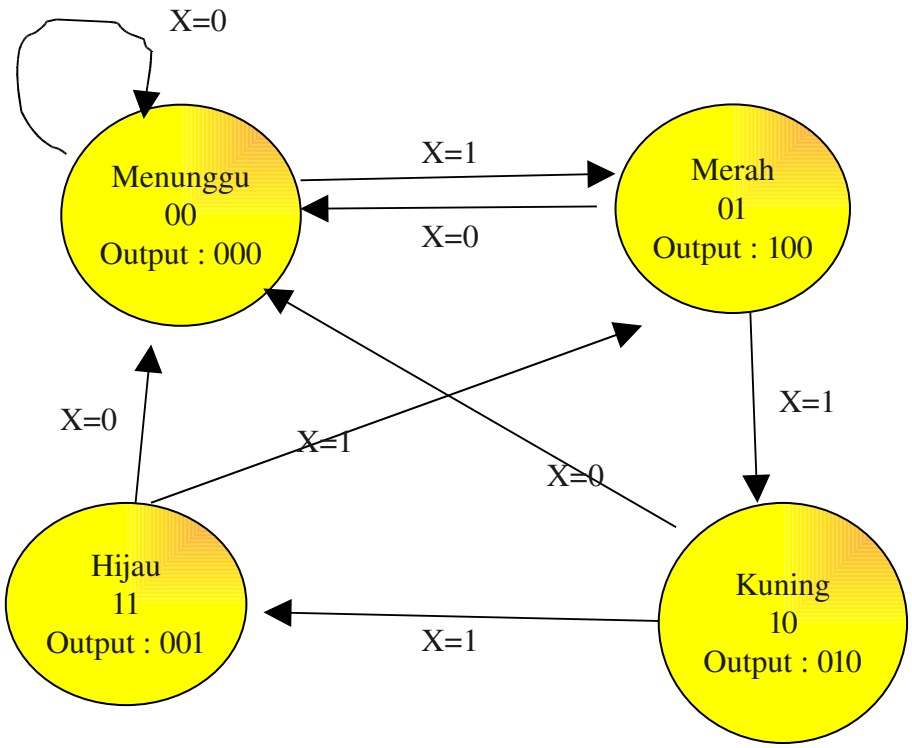
LAMPIRAN
GAMBAR DAN TABEL BERSESUAIAN

Nama State	Kode State		Input	State Berikutnya		Output		
	s1	s0		n1	n0	Merah	Kuning	Hijau
Menunggu	0	0	0	0	0	0	0	0
			1	0	1	0	0	0
Merah	0	1	0	0	0	1	0	0
			1	1	0	1	0	0
Kuning	1	0	0	0	0	0	1	0
			1	1	1	0	1	0
Hijau	1	1	0	0	0	0	0	1
			1	0	1	0	0	1

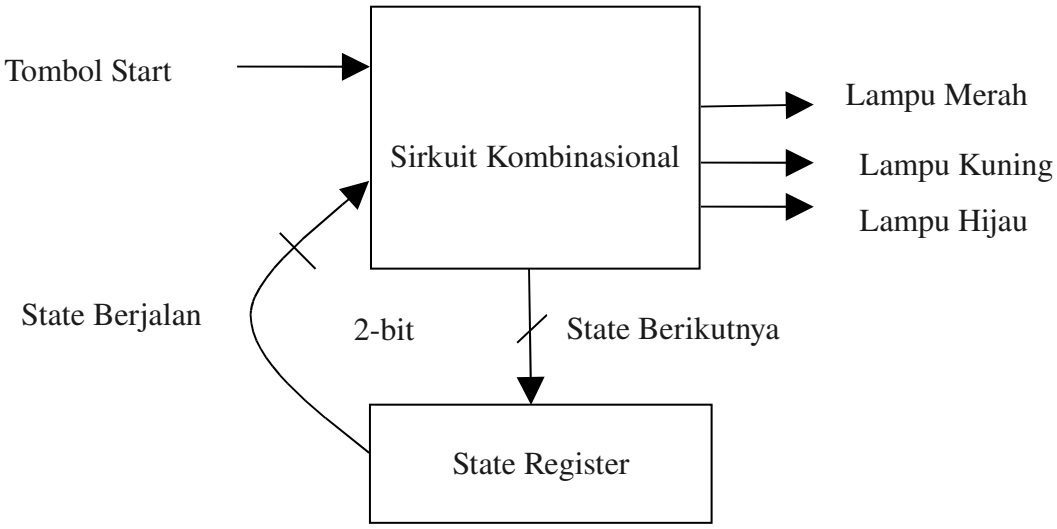
Tabel 1
Tabel Kebenaran yang Ekivalen dengan FSM



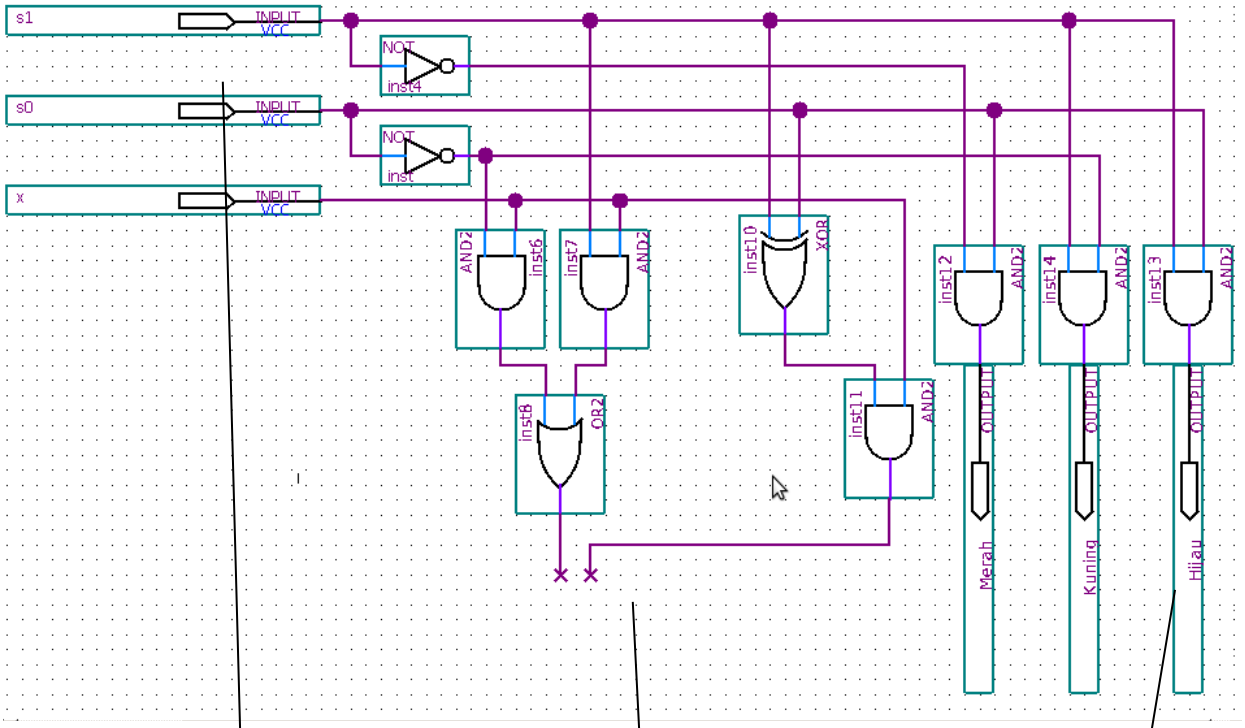
Gambar 1
Timing Diagram dapat digunakan untuk merepresentasikan sifat FSM
(Asumsi : tidak ada delay)



Gambar 2
FSM yang direpresentasikan dengan graf berarah



Gambar 3
Arsitektur Umum Rangkaian



Gambar 4
Isi dari diagram blok “Sirkuit Kombinasional”