Penerapan Graf dan Logika dalam Perancangan Rangkaian Digital dengan Studi Kasus Jam Digital

James Jaya 13511089[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]james.jaya@s.itb.ac.id

Matematika diskrit merupakan cabang dari ilmu matematika yang merupakan salah satu dasar yang cukup penting pada bidang rekayasa. Teknik informatika dan teknik elektro merupakan cabang ilmu yang cukup banyak menggunakan aplikasi dari matematika diskrit. Makalah ini membahas beberapa topik dari matematika diskrit yang digunakan pada salah satu topic teknik elektro, sistem digital. Topik matematika diskrit yang digunakan ini adalah logika, aljabar Boolean, dan teori graf.

Subtopik dari sistem digital yang dibahas adalah perancangan rangkaian sekuensial. Makalah ini menggunakan rangkaian jam digital sebagai salah satu studi kasus dalam pemaparan aplikasi logika, aljabar Boolean, dan teori graf dalam sistem digital.

Kata kunci: jam digital, logika, Boolean, finite-state machine, graf, sekuensial.

I. PENDAHULUAN

Sistem digital adalah salah satu topik inti dari program studi teknik elektro. Topik ini merupakan dasar yang harus dimiliki sebelum mendapatkan topik yang lebih lanjut seperti sistem mikroprosesor dan arsitektur komputer. Oleh karena itu, dapat dikatakan bahwa sistem digital merupakan dasar perancangan perangkat keras komputer. Posisinya yang merupakan topik inti membuat sistem digital dimasukkan sebagai salah satu topik pada program studi lain yang mempunyai hubungan cukup dekat dengan komputer, misalnya ilmu komputer dan teknik komputer. Salah satu kompetensi yang akan dimiliki setelah topik sistem digital ini adalah kemampuan merancang rangkaian digital yang kompleksitasnya tidak terlalu tinggi.

Pada topik sistem digital, dikenal dua jenis rangkaian yaitu rangkaian kombinasional dan rangkaian sekuensial. Rangkaian kombinasional adalah rangkaian yang memiliki keluaran yang hanya bergantung pada kombinasi input saat itu, sedangkan rangkaian sekuensial adalah rangkaian yang keluarannya bergantung pada keluaran sebelumnya dan bergantung pada masukkannya [1]. Rangkaian kombinasional dapat dikatakan relatif lebih sederhana daripada rangkaian sekuensial karena permalahan didalamnya cukup diselesaikan dengan aljabar boolean saja. Di lain pihak, rangkaian sekuensial adalah rangkaian yang sebenarnya terdiri dari beberapa rangkaian kombinasional sehingga permasalahan di dalamnya lebih kompleks. Pemecahan masalah di dalamnya memerlukan penerapan dari graf dan pohon.

II. DASAR TEORI

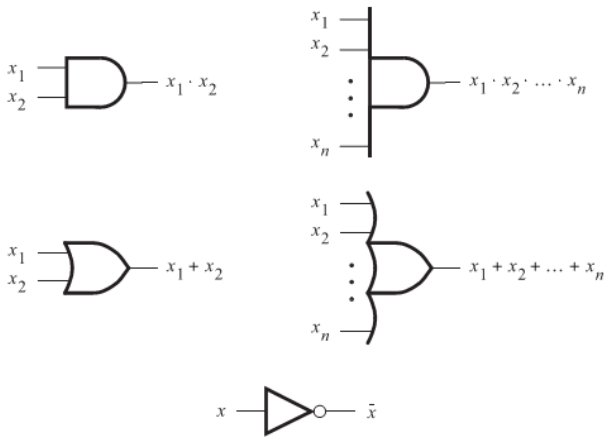
A. Logika

Proposisi adalah sebuah pernyataan yang dapat bernilai benar atau salah [1]. Beberapa proposisi dapat digunakan untuk membuat pernyataan logika yang lebih kompleks lagi. Proposisi yang dibentuk dengan beberapa proposisi yang dihubungkan dengan operator logika disebut proposisi majemuk [1].

Terdapat empat operator logika, yaitu konjungsi, disjungsi, implikasi, dan biimplikasi. Konjungsi (∧) adalah operator logika yang membentuk pernyataan bernilai benar apabila kedua operannya bernilai benar, selain itu salah [5]. Konjungsi biasa dituliskan AND atau DAN atau dalam notasi '∧'. Disjungsi adalah operator logika yang membentuk pernyataan bernilai benar apabila salah satu operannya bernilai benar dan bernilai salah apabila kedua operannya bernilai salah [5]. Disjungsi biasa ditulis OR atau ATAU atau '∨'. Implikasi (→) adalah operator logika yang operan pertamanya disebut hipotesis dan operan keduanya disebut konklusi. Implikasi membentuk pernyataan bernilai salah apabila hipotesisnya benar namun konklusinya salah dan bernilai benar untuk kondisi selain itu [5]. Misal p adalah hipotesis dan q adalah konklusi, maka jika keduanya dihubungkan dengan operator implikasi dapat ditulis "jika p, maka q" atau dalam notasi p → q. Biimplikasi (↔) adalah operator logika yang membentuk pernyataan bernilai benar apabila kedua operannya memiliki nilai kebenaran yang sama [5].

B. Gerbang Logika

Gerbang logika merupakan implementasi elektronis dari operator-operator logika. Gerbang logika ini dapat dibentuk dengan menggunakan transistor. Contoh dari gerbang logika sederhana adalah gerbang AND, OR, dan NOT [4]. Terdapat pula variasi dari gerbang-gerbang logika ini, seperti NAND, NOR, XOR, dan XNOR.



Gambar 1 Gerbang Logika Sederhana [4]

C. Aljabar Boolean

Aljabar Boolean adalah aturan dasar logika dari logika Boolean yang membentuk struktur matematika [2]. Aljabar Boolean pertama kali dikemukakan oleh George Boole pada tahun 1854 yang melihat bahwa himpunan dan logika proposisi mempunyai sifat yang serupa [2].

Pada aljabar Boolean, AND dituliskan '.', OR dituliskan '+', dan NOT dituliskan dengan memberikan garis di atas operan yang akan dikenakan operator atau dengan tanda "'", misal a'. Operator-operator ini berlaku seperti operator pada aljabar biasa, misalnya 0.0 = 0, 0.1 = 0, 0 + 1 = 1.

Fungsi Boolean adalah pemetaan dari Bn ke B melalui ekspresi Boolean, kita menuliskannya sebagai f : $B^n \rightarrow B$, Bn adalah himpunan yang beranggotakan pasangan terurut ganda-n di dalam daerah asal B. Contoh fungsi Boolean: f(x, y, z) = xyz'.

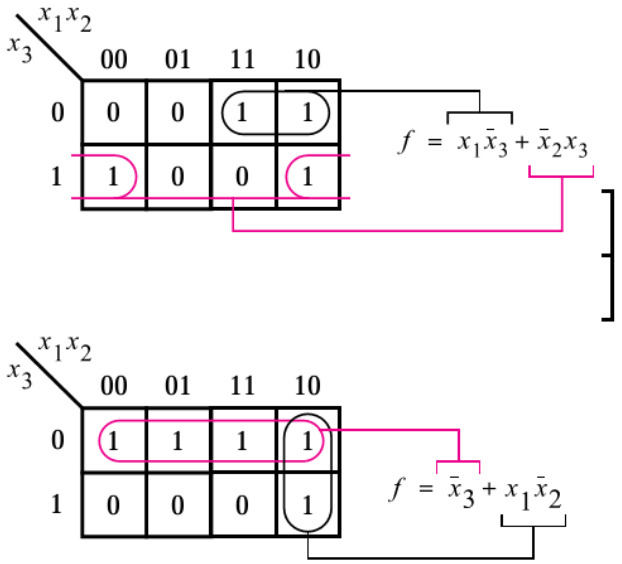
Fungsi Boolean biasanya mengandung ekspresi yang tidak perlu dan menyebabkannya menjadi panjang. Ada fungsi yang ekivalen (memiliki nilai kebenaran yang sama) namun bentuknya lebih sederhana. Proses penyederhanaan fungsi disebut minimisasi fungsi. Minimisasi fungsi salah satunya dapat dilakukan dengan Karnaugh Map.

Karnaugh Map atau K-Map adalah diagram terbentuk dari kotak-kotak yang bersisian. Tiap kotak merepresentasikan minterm dan minterm dikatakan bertetangga jika hanya berbeda 1 literal [2].

x_1	x_2	x_3	
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7

x_3 \ x_1x_2	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

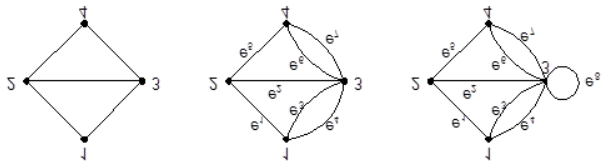
Gambar 2 Tabel kebenaran tiga literal dan K-Map-nya [4]



Gambar 3 Penggunaan K-Map untuk Minimisasi [4]

D. Graf

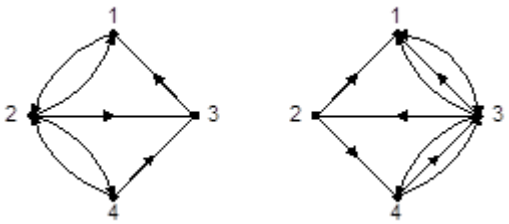
Graf G didefinisikan sebagai pasangan himpunan (V, E), ditulis dengan notasi G = (V, E), yang dalam hal ini V adalah himpunan tidak-kosong dari simpul-simpul (vertices atau node) dan E adalah himpunan sisi (edges atau arcs) yang menhubungkan sepasang simpul [3]. Graf digunakan untuk merepresentasikan objek diskrit dan hubungan antara objek-objek tersebut.



Gambar 4 Graf Sederhana, Graf Ganda, Graf Semu [3]

Terdapat beberapa dua jenis graf jika digolongkan menurut ada tidaknya sisi ganda, yaitu graf sederhana dan graf tidak sederhana [3]. Graf sederhana adalah graf yang tidak memiliki sisi ganda [3]. Graf tidak sederhana adalah graf yang memiliki sisi ganda atau gelang [3]. Graf yang memiliki sisi ganda juga merupakan graf ganda dan graf yang memiliki gelang juga merupakan graf semu. Pada Gambar 4, di graf semu terdapat sisi gelang yaitu sisi (3,3).

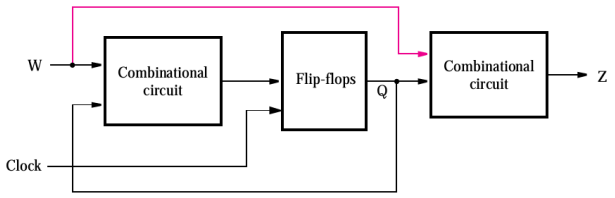
Berdasarkan orientasi arah pada sisi, graf dapat dibedakan menjadi dua, yaitu graf berarah dan graf tidak-berarah. Graf tidak-berarah adalah graf yang sisinya tidak memiliki orientasi arah sedangkan graf berarah memiliki orientasi arah.



Gambar 5 Graf Berarah [3]

E. Rangkaian Sekuensial

Rangkaian sekuensial adalah rangkaian yang keluarannya bergantung dari keluaran sebelumnya dan mungkin juga bergantung pada input saat itu [4]. Rangkaian sekuensial biasa terdiri dari beberapa rangkaian kombinasional dan beberapa flip-flop [4] Terdapat beberapa jenis flip-flop, misalnya D flip-flop, T flip-flop, dan JK flip-flop. Flip-flop berfungsi sebagai memori yang menyimpan output dari proses sebelumnya.



Gambar 6 Diagram Blok Rangkaian Sekuensial Sederhana [4]

Rangkaian sekuensial biasa dirancang dengan model finite-state machine. Finite-state machine (FSM) adalah model mesin di mana state atau keadaan dari mesin tersebut berhingga jumlahnya. Berdasarkan pengaruh input, terdapat dua jenis FSM. Pertama, FSM Moore adalah FSM yang outputnya tidak bergantung pada inputnya. Kedua, FSM Mealy adalah FSM yang outputnya bergantung pada inputnya dan statenya.

III. Aplikasi Logika, Aljabar Boolean, dan Graf pada Perancangan Rangkaian Sekuensial

A. Logika dan Aljabar Boolean

Rangkaian digital sekuensial merupakan rangkaian logika. Input dan outputnya selalu terhubung oleh sebuah fungsi. Pada rangkaian logika, nilai logika yang dikenal adalah 0 dan 1. Ini sangat erat kaitannya dengan prinsip yang ada pada logika dan aljabar boolean.

Seringkali terdapat banyak cara untuk mendapatkan output-output tertentu untuk input-input yang tertentu. Persamaan Boolean yang dapat dibuat tak terhingga jumlahnya. Namun, permasalahannya adalah menentukan fungsi logika yang paling sederhana untuk permasalahan yang sama.

Kompleksitas fungsi logika sangat berpengaruh pada performansi dari rangkaian logika. Fungsi yang kompleks akan membuat gerbang logika yang dibutuhkan semakin banyak, artinya transistor yang digunakan akan semakin banyak. Jumlah transistor yang semakin banyak ini berdampak pada tiga aspek. Pertama, biaya yang diperlukan semakin besar. Semakin banyak transistor yang digunakan tentu membuat modal biaya yang semakin besar juga. Kedua, tempat yang digunakan menjadi lebih luas. Pada masa sekarang, ukuran integrated circuit atau transistor sangat penting. Transistor yang banyak akan menempati ruang yang lebih banyak dan tentu membuat kinerja rangkaian kurang efektif. Ketiga, banyak transistor berarti membuat delay rangkaian menjadi semakin besar. Delay yang besar dan berbeda-beda membuat rangkaian tidak berfungsi dengan baik, serta dapat menimbulkan glitch.

Cara penyederhanaan fungsi logika dengan aljabar Boolean ada beberapa. Pertama, fungsi logika dapat disederhanakan dengan hukum-hukum yang ada di aljabar Boolean. Misal ada persamaan

$$f(x,y,z) = x'yz' + xy'z' + xx'z + xyz + x'y'z + \text{ y'yz}$$

Penggunaan hukum aljabar Boolean dapat menyederhanakan persamaan tersebut, dalam kasus ini digunakan hukum komplemen untuk menghilangan suku ke-3 dan ke-6.

$$f(x,y,z) = x'yz' + xy'z' + xyz + x'y'z$$

Kemudian dengan hukun distributif, bentuk persamaan itu menjadi

$$f(x,y,z) = z'(x'y + xy') + z(xy + x'y')$$

Penggunaan dua hukum aljabar Boolean telah membuat persamaan logika menjadi jauh lebih sederhana. Pada persamaan awal, dibutuhkan 6 buah gerbang AND tiga input, 1 buah gerbang OR enam input, dan 3 buah gerbang NOT. Sebuah gerbang AND tiga input memerlukan 8 transistor, artinya enam buah gerbang memerlukan 48 transistor. Sebuah gerbang OR enam input memerlukan 14 transistor. Gerbang NOT memerlukan 2 transistor, tiga gerbang NOT berarti 6 transistor. Transistor yang diperlukan untuk mewujudkan rangkaian dengan fungsi tersebut adalah 68 transistor.

Fungsi logika yang telah disederhanakan memerlukan gerbang AND dua input sebanyak 6 buah, 1 buah gerbang OR 2 input dan 3 buah gerbang NOT. Kebutuhan transistornya adalah 36 + 6 + 6 = 48 transistor. Sehingga terbukti bahwa aljabar Boolean dapat menghemat kebutuhan transistor.

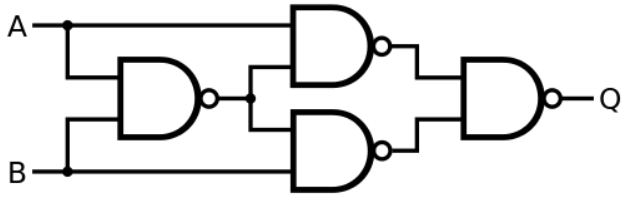
Gerbang OR dan gerbang AND sebenarnya dibuat dari gerbang NOR dan gerbang NAND yang dinegasikan. Artinya adalah OR adalah NOR yang dihubungkan dengan NOT dan AND adalah NAND yang dihubungkan dengan NOT. NOT adalah gerbang yang dibentuk dari dua transistor. Penggunaan gerbang NAND dan NOR untuk membentuk OR dan AND dapat menghemat dua transitor. Performansi dari rangkaian juga akan lebih baik karena transistor yang digunakan lebih sedikit yang artinya delay yang terjadi juga lebih sedikit. Berkaitan dengan hal ini, ada hukum aljabar Boolean yang sangat erat kaitannya. Hukum tersebut adalah hukum De Morgan.

$$(AB)' = A' + B'$$
$$(A + B)' = A'B'$$

Penyederhanaan juga dapat dilakukan dengan definisi. Misal definisi x XOR y (x $\oplus$ y) adalah x'y + xy'. Melalui definisi XOR itu, persamaan (x), fungsi tersebut juga dapat dituliskan menjadi

$$f(x, y, z) = x \oplus y \oplus z$$

Diketahui bahwa gerbang XOR terbentuk dari 4 buah gerbang NAND, yang artinya diperlukan 16 buah transistor. Persamaan ini memerlukan dua buah gerbang XOR, berarti bentuk persamaan ini hanya memerlukan 32 buah transistor.



Gambar 7 Struktur Gerbang XOR [4]

Penyederhanaan dengan hukum-hukum aljabar Boolean seringkali memakan waktu dan seringkali didapat hasil yang kurang sederhana. Oleh karena itu, terdapat metode lain yang menggunakan prinsip ini tetapi lebih efektif. Pertama adalah Karnaugh Map atau yang lebih dikenal dengan K-Map dan yang kedua adalah penggunaan bentuk cube dan hypercube, serta yang ketiga ada metode Tabular.

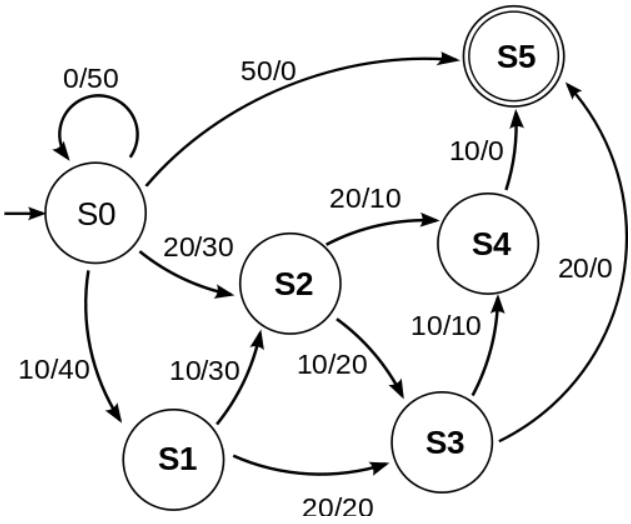
B. Graf

Pada perancangan rangkaian sekuensial, graf sangat berperan penting. Salah satu model dalam perancangan rangkaian sekuensial adalah finite-state machine (FSM). FSM adalah pemodelan mesin di mana mesin memiliki sejumlah state kerja yang dapat berubah tergantung pada keadaan saat itu. State mempengaruhi keluaran dari sistem.

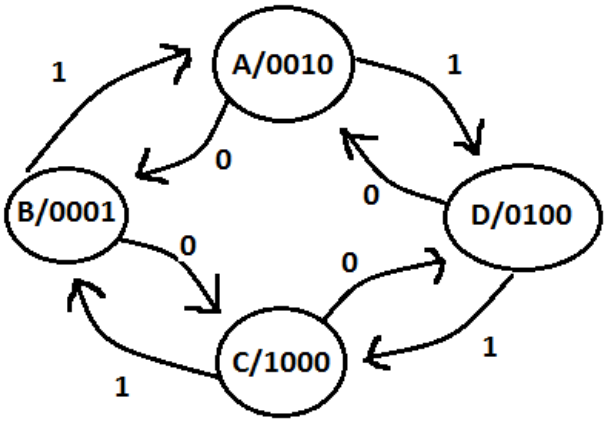
Perancangan FSM dapat dilakukan dengan beberapa cara, namun salah satu yang termudah adalah penggunaan graf. Graf yang digunakan pada perancangan FSM adalah graf berarah. Simpul pada graf berarah ini mewakili state mesin, sedangkan orientasi arah mewakili state berikutnya dari state yang sekarang.

Perancangan FSM dengan graf berarah membantu manusia secara visual, sehingga dengan mudah dapat diteliti state mana yang masih kurang lengkap. Graf berarah juga dengan mudah memperlihatkan berapa state yang terdapat pada FSM.

Penggambaran FSM dengan graf berarah yang disebut state diagram ini juga sangat membantu dalam penyederhanaan rangkaian. Rangkaian yang paling sederhana adalah rangkaian yang statenya paling sedikit. Metode penyederhanaan yang digunakan adalah implication chart dan row matching.



Gambar 8 Contoh State Diagram Graf Berarah FSM Mealy (http://commons.wikimedia.org/wiki/File:CPT-FSM-Mealy-02.svg)



Gambar 9 Contoh State Diagram Graf Berarah FSM Moore (http://i116.photobucket.com/albums/o33/Timmymna/elec/FSM2.png)

IV. STUDI KASUS: JAM DIGITAL

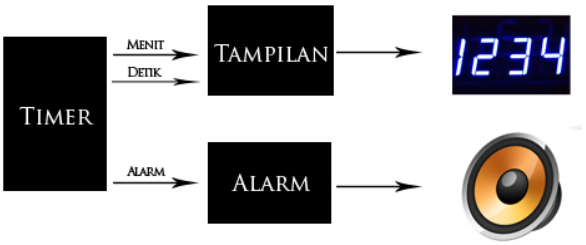
A. Spesifikasi

Jam digital yang dirancang ini memiliki batasan spesifikasi. Jam digital ini menggunakan sistem waktu 24-jam, bukan AM-PM. Tampilan jam digital menggunakan display 7-segment. Jam ini juga dilengkapi alarm yang dapat dikonfigurasi waktunya. Perancangan ini dilakukan dengan field-programmable gate arrays (FPGA) dengan bahasa deskripsi perangkat keras VHDL. Perancangan dilakukan dengan pemodelan finite-state machine (FSM).

B. Diagram Blok

Diagram blok adalah gambaran aliran sinyal dari input sampai system mengeluarkan output. Pada perancangan ini, diagram blok digunakan untuk merepresentasikan aliran sinyal antar komponen. Diagram blok merupakan sebuah graf berarah, di mana simpul merepresentasikan komponen dan orientasi arah menunjukkan arah aliran sinyal yang masuk-keluar dari setiap komponen.

Pada kasus ini, sistem rangkaian jam digital yang dibuat memiliki beberapa blok yaitu blok timer, blok alarm, dan blok tampilan. Blok timer adalah komponen yang menjadi "otak" dari sistem ini. Di blok ini dapat dilakukan penggantian mode, pengaturan waktu, pengaturan alarm, dan blok ini merupakan blok yang menghitung jalannya waktu. Blok tampilan adalah blok yang berfungsi untuk menampilkan waktu saat ini. Blok alarm adalah blok yang berfungsi untuk menyalakan alarm.



Gambar 10 Graf Berarah Blok Diagram Rangkaian Jam Digital

Blok timer mengirimkan sinyal integer menit dan detik, kemudian blok tampilan akan mengubahnya menjadi binary-coded-decimal (BCD) code sehingga dapat ditampilkan pada 7-segment. Blok timer juga mengirimkan sinyal alarm yang membuat alarm berbunyi apabila waktunya sudah seperti pengaturan. Blok timer sendiri memiliki input sinyal ganti mode dan empat sinyal untuk mengatur waktu, yaitu tambah menit, kurang menit, tambah detik, dan kurang detik.

C. Blok Tampilan

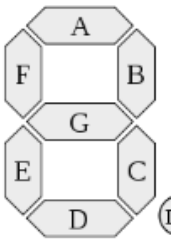
Blok tampilan terdiri dari dua komponen, yaitu konverter integer ke BCD dan konverter BCD ke 7-segment. Konverter integer ke BCD memiliki batasan maksimum dua digit integer. Blok ini menggunakan prinsip logika dan aljabar Boolean.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity decTobcd is
  port (
    dec: in integer;
    bcd: out std_logic_vector(7 downto 0)
  );
end decTobcd;

architecture behavioral of decTobcd is

signal first : integer;
signal second: integer;
signal bcdfirst : std_logic_vector(7 downto
0);
signal bcdsecond: std_logic_vector(3 downto
0);

begin
  first <= dec/10;
  second <= dec - first*10;
  bcdfirst <= conv_std_logic_vector(first*16,
8);
  bcdsecond <=
conv_std_logic_vector(second,4);
  bcd <= (bcdfirst) + bcdsecond;
end behavioral;
```

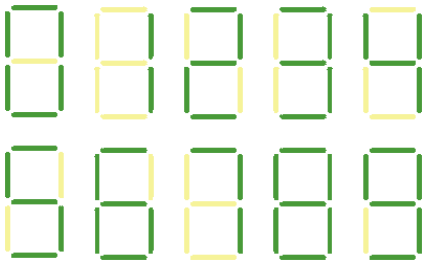
Gambar 11 Script VHDL untuk Konverter Integer ke BCD

Proses yang dilakukan skrip pada Gambar 11 adalah sebagai berikut. Misal dec = 15, maka first = 15/10 = 1 dan second = 15-1*10 = 5. Tujuan dari first dan second hanyalah untuk memisahkan digit dari bilangan dec. Kemudian bcdfirst diisi dengan first yang di-konvert menjadi bilangan biner 8-bit, namun first dikalikan 16 terlebih dahulu, sehingga bcdfirst = 00010000. Tujuan pengalian dengan 16 adalah supaya bilangan biner yang terbentuk bergeser empat bit ke kiri. Sinyal bcdsecond diisi dengan second yang di-convert menjadi bilangan biner 4-bit, berarti dalam contoh kasus ini bcdsecond = 0101. Akhirnya, keluaran dari komponen ini adalah bcd, bcd = bcdfirst + bcdsecond = 00010000 + 0101 = 00010101. Kode binary 00010101 adalah BCD dari decimal 15.

Kode BCD yang diperoleh pada proses itu kemudian menjadi input dari komponen berikutnya yaitu konverter BCD ke 7-segment. 7-segment bekerja dengan cara mengaktifkan segmen per segmen, artinya ada tujuh input untuk tujuh segmen. Lampu sebuah segmen akan menyala apabila diberikan logika 1 dan mati ketika diberikan logika 0 (active high). Bilangan yang dapat ditampilkan pada sebuah tampilan 7-segment adalah bilangan 1-digit, sehingga yang dapat ditampilkan hanya 0-9. Untuk menampilkan bilangan 2-digit diperlukan dua tampilan 7-segment.



Gambar 12 Tampilan 7-segment



Gambar 13 Angka 0-9 untuk Tampilan 7-segment (http://1.bp.blogspot.com/_-dCZKOAZ9SE/TRSuuHd-wBI/AAAAAAAAABc/pzkQ8DOoZUI/s1600/seven_segment_displays.gif)

BCD untuk angka 0-9 memiliki panjang 4-bit dan 7-segment memerlukan 7 input, sehingga misal BCD 4-bit diwakili oleh D3, D2, D1, dan D0 dan 7 input untuk 7-segment adalah A, B, C, D, E, F, G yang pembagiannya terlihat seperti pada Gambar 12. Melalui ini, dapat dibentuk tabel kebenaran dari komponen konverter BCD ke 7-segment yang terlihat pada Tabel 1.

Melalui Tabel 1, dapat dicari fungsi Boolean dari A, B, C, D, E, F, G dari D3, D2, D1, dan D0. Akan digunakan K-Map untuk mencari fungsi Boolean tersebut. K-Map terlihat seperti Gambar 14.

A

D1D0/ D3D2	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11				
10	1	1		

B

D1D0/ D3D2	00	01	11	10
00	1	1	1	1
01	1	0	1	0
11				
10	1	1		

C

D1D0/ D3D2	00	01	11	10
00	1	1	1	0
01	1	1	1	1
11				
10	1	1		

D

D1D0/ D3D2	00	01	11	10
00	1	0	1	1
01	0	1	0	1
11				
10	1	1		

E

D1D0/ D3D2	00	01	11	10
00	1	0	0	1
01	0	0	0	1
11				
10	1	0		

F

D1D0/ D3D2	00	01	11	10
00	1	0	0	0
01	1	1	0	1
11				
10	1	1		

G

D1D0/ D3D2	00	01	11	10
00	0	0	1	1
01	1	1	0	1
11				
10	1	1		

Gambar 14 K-Map untuk 7-Segment

Melalui K-Map untuk 7-segment ini, dapat diperoleh fungsi Boolean-nya:

- A = (D1+D0+D2')(D3+D2+D1+D0')
- B = (D1+D0'+D2')(D2'+D1'+D0)
- C = (D2+D1'+D0)
- D = (D1+D0+D2') (D3+D2+D1+D0') (D2'+D1'+D0')
- E = D2'D0'+D1D0'
- F = (D3+D2+D0')(D3+D2+D1') (D3+D1'+D0')
- G = (D3+D2+D1)(D2'+D1'+D0')

Fungsi Boolean tersebut kemudian diimplementasikan dengan skrip VHDL. Karena BCD yang digunakan adalah 8-bit yang diperoleh dari komponen sebelumnya, maka terdapat 14 segmen yang akan diprogram pada skrip ini. A-G adalah segmen untuk angka satuan dan H-N adalah segmen untuk angka puluhan.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity bcd7 is
  port (
    x: in STD_LOGIC_VECTOR (7 downto 0);
    A, B, C, D, E, F, G: out STD_LOGIC;
    H, I, J, K, L, M, N: out STD_LOGIC
  );
end bcd7;

architecture behavioral of bcd7 is
begin
  A <= (x(1) OR x(0) OR not x(2)) AND (x(3) OR
x(2) OR x(1) OR not x(0));
  B <= (not x(2) OR x(1) OR not x(0)) AND (not
x(2) OR not x(1) OR x(0));
  C <= x(2) OR not x(1) OR x(0);
  D <= (not x(2) OR x(1) OR x(0)) AND (x(3) OR
x(2) OR x(1) OR not x(0)) AND (not x(2) OR
not x(1) OR not x(0));
  E <= (not x(2) AND not x(0)) OR (x(1) AND
not x(0));
  F <= (x(3) OR x(2) OR not x(0)) AND (x(3) OR
x(2) OR not x(1)) AND (x(3) OR not x(1) OR
not x(0));
  G <= (x(3) OR x(2) OR x(1)) AND (not x(2) OR
not x(1) OR not x(0));

  H <= (x(5) OR x(4) OR not x(6)) AND (x(7) OR
x(6) OR x(5) OR not x(4));
  I <= (not x(6) OR x(5) OR not x(4)) AND (not
x(6) OR not x(5) OR x(4));
  J <= x(6) OR not x(5) OR x(4);
  K <= (not x(6) OR x(5) OR x(4)) AND (x(7) OR
x(6) OR x(5) OR not x(4)) AND (not x(6) OR
not x(5) OR not x(4));
  L <= (not x(6) AND not x(4)) OR (x(5) AND
not x(4));
  M <= (x(7) OR x(6) OR not x(4)) AND (x(7) OR
x(6) OR not x(5)) AND (x(7) OR not x(5) OR
not x(4));
  N <= (x(7) OR x(6) OR x(5)) AND (not x(6) OR
not x(5) OR not x(4));
end behavioral;
```

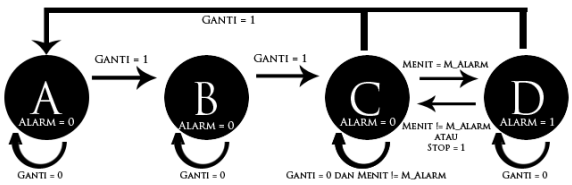
Gambar 15 Skrip VHDL untuk Konverter BCD 8-bit ke 7-segment

D. Blok Timer

Blok timer merupakan inti dari rancangan ini. Blok timer berfungsi untuk menggerakkan waktu dan melakukan pengaturan. Oleh karena fungsinya yang cukup banyak dan kompleks, maka kerjanya perlu dirancang dengan sistematis. Perancangan blok ini dilakukan dengan model finite-state machine (FSM). Model ini merupakan model mesin di mana mesin memiliki beberapa keadaan (state) yang jumlahnya hingga.

Blok timer memiliki beberapa input dan output. Input dari blok ini yang pertama adalah clock yang berfungsi untuk menangkap perubahan yang ada di sistem. Input kedua, input ganti state, tombol ini yang mengatur perubahan state pada sistem. Input ketiga adalah input untuk pengaturan waktu, terdapat empat input, yaitu tambah menit, kurang menit, tambah detik, dan kurang detik. Output blok ini adalah integer menit dan detik untuk tampilan serta sinyal alarm yang mengatur aktif tidaknya alarm.

Perancangan FSM dapat dilakukan dengan berbagai cara, namun salah satu yang termudah adalah membuat graf berarah yang disebut dengan state diagram. Dengan membuat graf berarah ini, perancangan bersifat lebih intuitif. Pada perancangan jam digital ini, pertama ditentukan dahulu state paling awal dari jam. Anggap state awal jam adalah mode pengaturan, sehingga pengguna jam yang pertama kali menyalakan langsung masuk ke state ini. Sebut state pengaturan ini state A. Pada state A, input pengaturan waktu yang ada empat dapat digunakan untuk mengatur waktu sekarang. Waktu yang diatur tertampil pada tampilan 7-segment. Pemberian input 1 pada sinyal ganti membuat state berpindah ke B.



Gambar 16 State Diagram Graf Berarah Blok Timer

State B adalah pengaturan alarm. Pada state ini, pengguna juga dapat menggunakan input pengaturan waktu sama seperti state A, perbedaannya adalah waktu yang diatur bukan waktu sekarang tetapi waktu alarm. Tampilan waktu alarm juga terlihat pada tampilan 7-segment. Pembrian input 1 pad sinyal ganti membuat state pindah ke C.

State C adalah state pengoperasian jam normal. Jam berhitung maju detik per detik jika tidak diberikan input apapun. State berpindah ke state A ketika input ganti = 1. Ketika menit menunjukkan menit yang sama ketika alarm diatur, maka state berpindah ke state D.

State D adalah state alarm. Pada state ini, waktu berjalan seperti biasa namun ditambah suara alarm berbunyi. Alarm akan berbunyi selama menit saat itu sama dengan menit alarm dan selama tombol stop tidak ditekan. State berpindah ke A jika tombol ganti state digunakan. Setelah tombol stop ditekan atau menit sudah berganti, maka jam kembali ke state pengoperasian normal, C. Berikut ini adalah skrip VHDL yang merupakan implementasi dari FSM yang telah dirancang.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity timer is
 port (
   clock: in STD_LOGIC;
   ganti: in STD_LOGIC;
   plus_jam: in STD_LOGIC;
   min_jam: in STD_LOGIC;
   plus_menit: in STD_LOGIC;
   min_menit: in STD_LOGIC;
   menit: buffer integer range 0 to 23;
   detik: buffer integer range 0 to 59;
   jam_alarm: buffer integer range 0 to 23;
   menit_alarm: buffer integer range 0 to 59;
   alarm: buffer STD_LOGIC
   stop: in STD_LOGIC;
```

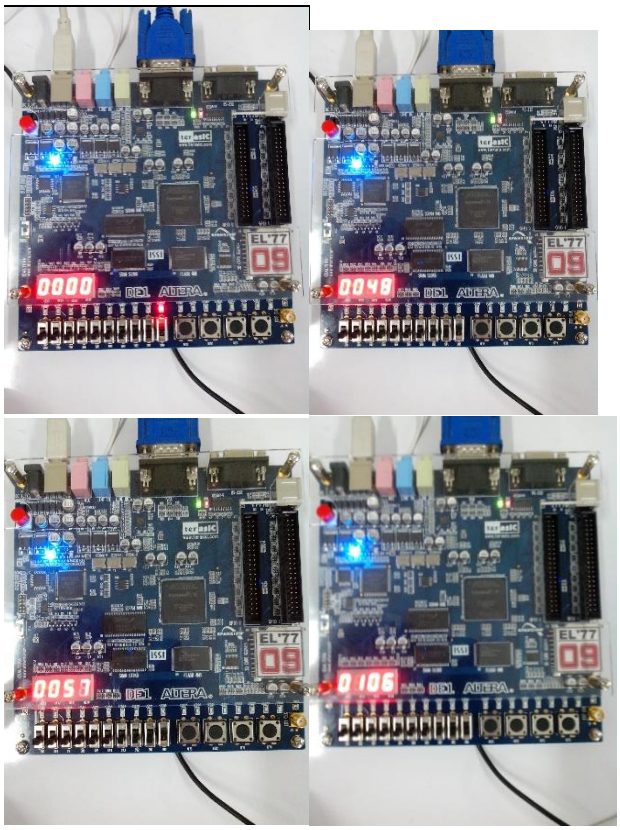
```
 );
end timer;

architecture behavioral of timer is

constant jamMax : integer := 23;
constant jamMin : integer := 0;
constant menitMax : integer := 59;
constant menitMin : integer := 0;

type State is (A, B, C, D);
signal currentstate: State := A;

begin
 process(currentstate, clock)
 begin
 if (clock'event and clock = '1') then
   case currentstate is
     when A =>
       if (ganti = '0') then
         if (plus_jam = '0') then
           if (jam < jamMax) then
             jam <= jam + 1;
           else
             jam <= jamMin;
           end if;
         elsif (min_jam = '0') then
           if (jam > jamMin) then
             jam <= jam - 1;
           else
             jam <= jamMax;
           end if;
         end if;
         if (plus_menit = '0') then
           if (menit < menitMax) then
             menit <= menit + 1;
           else
             menit <= menitMin;
           end if;
         elsif (min_menit = '0') then
           if (menit > menitMin) then
             menit <= menit - 1;
           else
             menit <= menitMax;
           end if;
         end if;
         currentstate <= A;
       else
         currentstate <= B;
       end if;
     when B =>
       if (ganti = '0') then
         if (plus_jam = '0') then
           if (jam_alarm < jamMax) then
             jam_alarm <= jam + 1;
           else
             jam_alarm <= jamMin;
           end if;
         elsif (min_jam = '0') then
           if (jam_alarm > jamMin) then
             jam_alarm <= jam_alarm - 1;
           else
             jam_alarm <= jamMax;
           end if;
         end if;
         if (plus_menit = '0') then
           if (menit_alarm < menitMax) then
             menit_alarm <= menit_alarm + 1;
           else
             menit_alarm <= menitMin;
           end if;
         elsif (min_menit = '0') then
           if (menit_alarm > menitMin) then
             menit_alarm <= menit_alarm - 1;
           else
             menit_alarm <= menitMax;
           end if;
         end if;
         currentstate <= B;
       else
         currentstate <= C;
       end if;
     when C =>  --HITUNG TIMER
         if (ganti = '1') then
           currentstate <= A;
           else
             if (menit < menitMax) then
               menit <= menit + 1;
             else
               if (jam < jamMax) then
                 jam <= jam + 1;
                 menit <= menitMin;
               else
                 jam <= jamMin;
                 menit <= menitMin;
               end if;
             end if;
             if (menit = menit_alarm and jam = jam_alarm)
then
               currentstate <= D;
             else
               currentstate <= C;
             end if;
           end if;
       when D =>
         if (ganti = '1') then
           currentstate <= A;
           alarm <= '0';
         else
           alarm <= '1';
           if (menit < menitMax) then
             menit <= menit + 1;
           else
             if (jam < jamMax) then
               jam <= jam + 1;
               menit <= menitMin;
             else
               jam <= jamMin;
               menit <= menitMin;
             end if;
           end if;
           if (menit = menit_alarm and jam = jam_alarm
and stop = '0') then
             currentstate <= D;
           else
             currentstate <= C;
             alarm <= '0';
           end if;
         end if;
     end case;
   end if;
   end process;
end behavioral;
```

E. Blok Alarm

Blok alarm adalah komponen yang berfungsi untuk menerima input sinyal alarm. Jika alarm bernilai 0, maka alarm mati. Jika alarm bernilai 1, maka alarm berbunyi. Pembahasan mengenai cara kerja blok alarm adalah di luar konteks topik makalah ini.

F. Pengujian

Pengujian dilakukan dengan board FPGA Altera DE-1. Pengujian berhasil dilakukan dan rangkaian bekerja sesuai dengan yang diharapkan. Hasil pengujian dapat dilihat pada Gambar 17.



Gambar 17 Pengujian Jam Digital pada Board FPGA

V. KESIMPULAN

Perancangan rangkaian digital melibatkan beberapa aspek dalam matematika diskrit, yaitu logika, aljabar Boolean, dan graf. Logika dan aljabar Boolean digunakan untuk menyederhanakan fungsi logika sehingga sistem bekerja lebih efektif dan lebih hemat. Graf digunakan untuk perancangan sistem kerja rangkaian yang mengikuti model finite-state machine (FSM).

VII. UCAPAN TERIMA KASIH

Terima kasih kepada Tuhan atas segala kemampuan yang diberikan-Nya sehingga makalah ini selesai tepat waktu. Terima kasih kepada dosen matakuliah struktur diskrit IF2091, Dra. Harlili dan Dr. Rinaldi Munir untuk ilmu-ilmu yang sudah dibagikan. Terima kasih kepada orang tua saya dan teman-teman saya yang selalu mendukung apapun yang saya lakukan.

REFERENSI

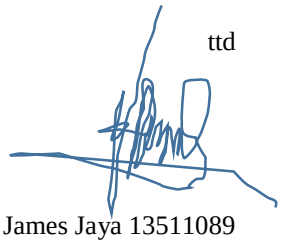
[1] Rinaldi Munir, Matematika Diskrit. Bandung: Informatika, 2009, ch. 1.
[2] Rinaldi Munir, Matematika Diskrit. Bamdung: Informatika, 2009, ch. 7.
[3] Rinaldi Munir, Matematika Diskrit. Bamdung: Informatika, 2009, ch. 8.
[4] Stephen Brown dan Zvonko Vranesic, *Fundamentals of Digital Logic with VHDL Design.* New York: McGraw-Hill, 2009.
[5] K. H. Rosen, Discrete Mathematics and Its Application. New York: McGraw-Hill, 2009.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 16 Desember 2012

ttd



James Jaya 13511089