Implementasi *Greedy* Dalam Menemukan Rangkaian Logika Minimal Menggunakan *Karnaugh Map*

Aldy Wirawan 13511035
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
aldy.wirawan@students.itb.ac.id

Abstrak - Rangkaian logika adalah rangkaian elektrik yang terdiri dari beberapa gerbang logika. Rangkaian ini menghasilkan serangkaian hasil tertentu tergantung dari susunan gerbang logikanya. Seringkali suatu rangkaian logika masih dapat disederhanakan/diminimalisasi untuk menghemat *resource* yang ada. Salah satu teknik minimalisasi rangkaian logika adalah dengan menggunakan *Karnaugh Map*. Dalam makalah ini, penyelesaian *Karnaugh Map* akan dilakukan menggunakan algoritma *greedy*.

Istilah Indeks - *Greedy, Karnaugh Map*, Minimalisasi, Rangkaian Logika

I. PENDAHULUAN

Rangkaian logika sebagian besar diaplikasikan pada komputer digital. Selain itu, rangkaian logika juga merupakan pondasi dari sistem digital yang sedikit melibatkan komputasi numerik. Tombol menyalakan dan mematikan lampu merupakan salah satu contoh sistem digital yang tidak melibatkan banyak komputasi numerik.

Rangkaian logika melakukan operasi terhadap sinyal digital yang biasanya dibatasi dalam sejumlah nilai diskrit tertentu. Dalam rangkaian logika biner, nilai sinyal yang ada hanyalah dua, yaitu 0 dan 1. Dalam rangkaian logika desimal, nilai sinyal yang ada berada dalam jangkauan 0 sampai 9. Karena aplikasi rangkaian logika biner yang lebih sederhana daripada rangkaian logika lainnya, rangkaian logika biner lebih sering digunakan dan mempunyai peranan dominan dalam teknologi digital.

Tentunya rangkaian logika yang digunakan dalam suatu sistem digital harus didesain seminimal mungkin agar penggunaannya efisien. Salah satu cara untuk mencapai hal ini adalah dengan menggunakan *Karnaugh Map*. Biasanya penyelesaian *Karnaugh Map* dilakukan dengan pendekatan heuristik. Dalam makalah ini, penyelesaian *Karnaugh Map* akan dilakukan menggunakan algoritma *greedy*.

II. OPERASI LOGIKA

Operasi adalah prosedur untuk menghasilkan nilai yang baru dari satu atau beberapa input. Operasi terdiri dari konstanta, variabel, dan/atau operator. Operasi logika adalah operasi yang hanya melibatkan dua konstanta, yaitu 0 (melambangkan *false*) dan 1 (melambangkan *true*). Ada tiga operasi logika utama, yaitu AND (dilambangkan dengan perkalian), OR (dilambangkan dengan penjumlahan), dan NOT (dilambangkan dengan garis pendek horizontal).

III. EKSPRESI LOGIKA DAN TABEL KEBENARAN

Ekspresi logika terdiri dari satu atau beberapa operasi logika. Seperti halnya operasi logika, ekspresi logika menerima satu atau beberapa input dan menghasilkan output yang merupakan sebuah fungsi dari inputnya. Ekspresi logika disebut juga sebagai aljabar *boolean*. Seperti aljabar lainnya, terdapat beberapa aksioma yang mendasari ekspresi logika:

$$0 \cdot 0 = 0 \tag{1}$$

$$1 + 1 = 1 \tag{2}$$

$$1 \cdot 1 = 1 \tag{3}$$

$$0 + 0 = 0 \tag{4}$$

$$0 \cdot 1 = 1 \cdot 0 = 0 \tag{5}$$

$$1 + 0 = 0 + 1 = 1 \tag{6}$$

$$\text{Jika } x = 0, \text{maka } \bar{x} = 1 \tag{7}$$

$$\text{Jika } x = 1, \text{maka } \bar{x} = 0 \tag{8}$$

Selain itu, ekspresi logika juga memiliki sifat tertentu:

$$x \cdot y = y \cdot x \tag{9}$$

$$x + y = y + x \tag{10}$$

$$x \cdot (y \cdot z) = (x \cdot y) \cdot z \tag{11}$$

$$x + (y + z) = (x + y) + z \tag{12}$$

$$x \cdot (y + z) = x \cdot y + x \cdot z \tag{13}$$

$$x + y \cdot z = (x + y) \cdot (x + z) \tag{14}$$

$$x + x \cdot y = x \tag{15}$$

$$x \cdot (x + y) = x \tag{16}$$

$$x \cdot y + x \cdot y = x \tag{17}$$

$$(x + y) \cdot (x + y) = x \tag{18}$$

$$x \cdot y = x + y \tag{19}$$

$$x + y = x \cdot y \tag{20}$$

$$x + x \cdot y = x + y \tag{21}$$

$$x \cdot (x + y) = x \cdot y \tag{22}$$

$$x \cdot y + y \cdot z + x \cdot z = x \cdot y + x \cdot z \tag{23}$$

$$(x + y) \cdot (y + z) \cdot (x + z) = (x + y) \cdot (x + z) \tag{24}$$

Persamaan (9) dan (10) merepresentasikan sifat komutatif. Persamaan (11) dan (12) merepresentasikan sifat asosiatif. Persamaan (13) dan (14) merepresentasikan sifat distributif. Persamaan (15) dan (16) merepresentasikan sifat absorpsi. Persamaan (17) dan (18) merepresentasikan sifat kombinasi. Persamaan (19), (20), (21), dan (22) merepresentasikan teorema DeMorgan. Persamaan (23) dan (24) merepresentasikan konsensus.

Setiap sifat memiliki dua representasi karena dalam ekspresi logika terdapat prinsip dualitas, yaitu terdapat ekspresi logika lain yang valid dari sebuah ekspresi logika yang didapatkan dengan cara mengubah semua konstanta 0 menjadi 1 atau sebaliknya, dan mengubah semua operasi AND menjadi OR atau sebaliknya. Hal ini terlihat jelas dalam teorema DeMorgan.

Beberapa contoh dari ekspresi logika:

$$(\bar{x} + y)\overline{(y + x\bar{z})} + x\bar{y}\bar{z} \tag{25}$$

$$\bar{x}\bar{y}z + x\bar{y}\bar{z} + xyz \tag{26}$$

$$(x + 1)(y + 0z) \tag{27}$$

Ekspresi logika dapat mendeskripsikan semua output hasil kombinatorial dari variabel yang ada. Deskripsi dari semua hasil kombinatorial yang ada dituangkan dalam suatu tabel, yang dinamakan tabel kebenaran. Tabel kebenaran memiliki kolom sejumlah n+1 dan memiliki baris sejumlah 2^n dengan n merupakan jumlah variabel yang ada dalam suatu ekspresi kebenaran. Berikut beberapa contoh dari tabel kebenaran:

x	Y	f
0	0	0
0	1	0
1	0	0
1	1	1

$x \cdot y$

Tabel I. Tabel kebenaran operasi AND

x	y	f
0	0	0
0	1	1
1	0	1
1	1	1

$x + y$

Tabel II. Tabel kebenaran operasi OR

x	y	z	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

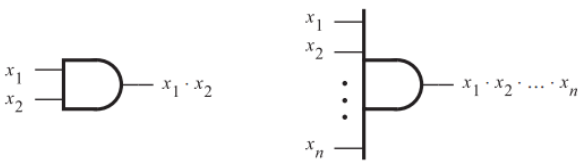
$\bar{x}\bar{y}z + x\bar{y}\bar{z} + xyz$

Tabel III. Tabel kebenaran ekspresi logika (26)

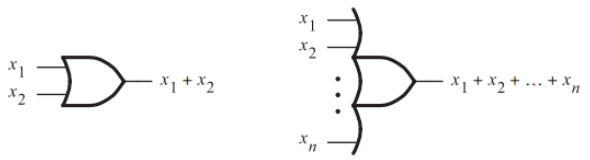
Kolom f dalam tabel kebenaran menampung output ekspresi logika dengan nilai masing-masing variabel berada pada baris yang sama.

IV. GERBANG LOGIKA

Suatu operasi logika dapat direpresentasikan secara elektronik menggunakan transistor, menghasilkan sebuah elemen rangkaian yang disebut sebagai gerbang logika. Sesuai karakteristik operasi logika, suatu gerbang logika memiliki satu atau beberapa input dan output yang merupakan fungsi dari input gerbang logika tersebut. Untuk memudahkan representasi suatu gerbang logika, dibuatlah konvensi simbol grafik terhadap gerbang logika yang ada.



(a) Gerbang Logika AND



(b) Gerbang Logika OR

(c) Gerbang Logika NOT

Gambar 1. Simbol grafik dari gerbang logika dasar

V. RANGKAIAN LOGIKA

Rangkaian logika adalah gabungan gerbang logika yang membentuk suatu fungsi logika. Misalkan akan didesain sebuah fungsi dengan dua input, yaitu x_1 dan x_2, maka kedua input tersebut merepresentasikan sebuah saklar yang dapat terbuka (bernilai 0) atau tertutup (bernilai 1). Fungsi rangkaian akan terus memonitor status saklar dan memproduksi output yang sesuai.

Biasanya rangkaian logika didesain berdasarkan keluaran tertentu yang diharapkan. Keluaran yang diharapkan tersebut berbentuk tabel kebenaran. Misalkan tabel kebenaran yang ingin diproduksi adalah sebagai berikut:

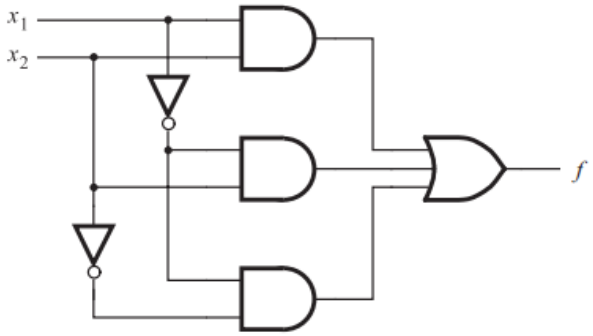
x_1	x_2	$f(x_1,x_2)$
0	0	1
0	1	1
1	0	0
1	1	1

Tabel IV. Contoh tabel kebenaran yang ingin diproduksi

Tabel kebenaran tersebut dapat direpresentasikan menggunakan sebuah ekspresi logika. Untuk setiap x_1 dan x_2 yang menghasilkan 1, x_1 dan x_2 nya ditulis dalam operasi AND. Kemudian semua operasi AND dari x_1 dan x_2 digabungkan menggunakan operasi OR. Hasil ekspresi logika dari tabel IV adalah sebagai berikut:

$$f(x_1,x_2) = \overline{x_1}\,\overline{x_2} + \overline{x_1}x_2 + x_1x_2 \tag{28}$$

Rangkaian logika dari ekspresi tersebut adalah seperti ini:



Gambar 2. Rangkaian logika dari ekspresi logika (28)

Rangkaian logika memiliki dua bentuk ekspresi logika, yaitu *sum-of-products* dan *product-of-sums*. Ekspresi logika (28) berbentuk *sum-of-products*, yang pembentukannya berdasarkan *minterms*, atau nilai 1 pada kolom f di tabel kebenaran. Bentuk *product-of-sums* merupakan kebalikan dari *sum-of-products*, sehingga pembentukannya berdasarkan *maxterms*, atau nilai 0 pada kolom f di tabel kebenaran. Jika format *product-of-sums* adalah:

$$x_1x_2x_3\ldots x_n + x_1x_2x_3\ldots x_n + \ldots \tag{29}$$

Maka format *sum-of-products* adalah kebalikannya yaitu sebagai berikut:

$$(x_1 + x_2 + x_3 + \ldots + x_n)\cdot(x_1 + x_2 + x_3 + \ldots + x_n)\cdot(\ldots \tag{30}$$

Dari tabel IV, bentuk *product-of-sums* yang dihasilkan adalah sebagai berikut:

$$f(x_1,x_2) = \overline{x_1} + x_2 \tag{31}$$

Dapat dilihat bahwa pada bentuk *product-of-sums* nilai yang diberikan operasi NOT adalah x_1, dikarenakan sudut pandang *product-of-sums* yang berkebalikan dengan *sum-of-products*.

Dari contoh yang telah dipaparkan diatas dapat dilihat bahwa ternyata rangkaian logika pada gambar 2 masih dapat disederhanakan dengan mengimplementasikan ekspresi logika (31). Teknik penyederhanaan dari sebuah ekspresi logika secara umum ada dua, yaitu penyederhanaan ekspresi menggunakan aksioma dan sifat dari ekspresi logika itu sendiri, dan penyederhanaan menggunakan *Karnaugh Map*. Dalam makalah ini, hanya akan dibahas penyederhanaan menggunakan *Karnaugh Map*.

VI. KARNAUGH MAP

Cara mencari suatu ekspresi minimal adalah dengan mereduksi jumlah operasi sebuah ekspresi yang ada. *Karnaugh Map* merupakan metode sistematik dalam

melakukan hal itu. *Karnaugh Map* merupakan representasi tabel kebenaran dalam bentuk matriks. Berikut merupakan contoh dari tabel kebenaran pada tabel III yang dikonversi ke *Karnaugh Map*:

x \ yz	**00**	**01**	**11**	**10**
0	0	0	0	1
1	1	0	1	0

Tabel V. Karnaugh Map dari tabel kebenaran pada tabel III

Salah satu kelebihan dari *Karnaugh Map* adalah urutannya disusun sedemikian rupa sehingga tiap sel yang bersebelahan hanya berbeda dalam satu nilai variabel saja. Melalui *Karnaugh Map* tersebut, sebuah ekspresi dapat diminimalisasi dengan menggunakan *minterm* maupun *maxterm*. Minimalisasi *minterm* memperhatikan entri variabel yang menghasilkan nilai 1, sedangkan minimalisasi *maxterm* memperhatikan entri variabel yang menghasilkan nilai 0. Sebagai contoh, ambil tabel kebenaran pada tabel IV dan konversi tabel tersebut ke *Karnaugh Map*:

x_2 \ x_1	**0**	**1**
0	1	1
1	0	1

Tabel VI. Karnaugh Map dari tabel kebenaran pada tabel IV

Akan dilakukan minimalisasi menggunakan *minterm*, sehingga yang diperhatikan adalah nilai 1. Suatu nilai 1 yang bersebelahan bisa direpresentasikan hanya dengan variabel yang tidak berubah saja. Sebagai contoh, lihat *Karnaugh Map* pada tabel VI yang telah ditandai berikut:

x_2 \ x_1	**0**	**1**
0	1	1
1	0	1

Tabel VII. Karnaugh Map pada tabel VI yang telah ditandai secara horizontal

x_2 \ x_1	**0**	**1**
0	1	1
1	0	1

Tabel VIII. Karnaugh Map pada tabel VI yang telah ditandai secara vertikal

Penandaan pada tabel VII dapat direpresentasikan dengan $\overline{x_2}$ karena pada kedua nilai 1 tersebut nilai x_2 tetap 0, sedangkan penandaan pada tabel VIII dapat direpresentasikan dengan x_1 karena nilai x_1 tetap yaitu 1. Maka didapat persamaan minimalnya yaitu sama dengan (31).

Hubungan nilai 1 yang bersebelahan dapat lebih dari 2 nilai 1, dengan syarat bentuk hubungan nilai 1 yang terbentuk adalah persegi atau garis, dengan jumlah nilai 1 totalnya genap. Nilai yang berada pada ujung-ujung *Karnaugh Map* juga dapat dihubungkan satu sama lain.

Bila suatu nilai 1 tidak memiliki nilai 1 yang bersebelahan, maka mau tidak mau nilai tersebut tetap harus direpresentasikan dalam ekspresi tanpa menyederhanakan variabel untuk nilai tersebut.

VII. Algoritma Greedy

Algoritma *greedy* membentuk solusi dari suatu permasalahan secara langkah per langkah. Prinsip algoritma ini adalah "*take what you can get now!*" atau mengambil solusi terbaik dalam suatu langkah tanpa melihat konsekuensi ke depannya. Langkah yang telah diambil tidak dapat dibatalkan lagi.

Algoritma *greedy* memiliki skema dalam menyelesaikan suatu masalah. Elemen-elemen yang menyusun algoritma *greedy* adalah himpunan kandidat (C), himpunan solusi (S), fungsi seleksi, fungsi kelayakan, dan fungsi obyektif. Himpunan kandidat adalah elemen-elemen pembentuk solusi. Himpunan solusi adalah kumpulan kandidat yang terpilih sebagai solusi persoalan. Fungsi seleksi merupakan fungsi pemilihan kandidat untuk menjadi bagian dari himpunan solusi. Fungsi kelayakan memeriksa apakah kandidat yang terpilih memberikan solusi yang layak. Terakhir, fungsi obyektif memberikan tujuan/solusi yang ingin dicapai.

Ambil suatu contoh persoalan penukaran uang. Misalkan ada beberapa satuan uang, yaitu 1,3,4, dan 5. Akan ditukar uang senilai 20 terhadap satuan uang tersebut, berapa jumlah satuan uang paling sedikit untuk menukar uang tersebut?

Dengan algoritma *greedy*, diambil nilai terbesar, yaitu 5. Proses ini terus diulang hingga nilai dari uang yang telah diambil sama dengan nilai yang ingin dicapai, yaitu 20. Solusi yang dihasilkan algoritma *greedy* adalah empat buah satuan 5, yang merupakan jumlah satuan uang paling sedikit (paling optimal). Himpunan kandidatnya adalah satuan 1,3,4, dan 5. Himpunan solusinya adalah empat buah satuan 5. Fungsi seleksinya adalah memilih nilai tertinggi dari satuan yang ada. Fungsi kelayakannya adalah memeriksa apakah nilai total dari himpunan solusi sudah melebihi jumlah uang yang ingin dicapai. Terakhir, fungsi obyektifnya adalah jumlah koin yang digunakan minimum.

Tinjau lagi persoalan yang sama, dengan nilai uang yang mau ditukar berbeda, yaitu 7, Dengan *greedy*, akan didapatkan hasil sebuah satuan 5 dan dua buah satuan 1, padahal solusi optimalnya adalah sebuah satuan 4 dan sebuah satuan 3. Hal ini menunjukan bahwa solusi yang dihasilkan dari *greedy* belum tentu optimal

VIII. IMPLEMENTASI GREEDY DALAM KARNAUGH MAP

Asumsikan seluruh kemungkinan hubungan yang ada dalam suatu *Karnaugh Map* dapat dibangkitkan. Pembangkitan seluruh kemungkinan hubungan dapat dilakukan dengan *bruteforce*, yaitu dengan melihat semua kemungkinan yang ada dan membangkitkan hubungan yang memenuhi salah satu kemungkinan tersebut. Maka, penyederhanaan menggunakan *Karnaugh Map* dapat dilakukan dengan algoritma *greedy* berskema berikut:

- Himpunan kandidat : seluruh hubungan yang mungkin dalam suatu *Karnaugh Map*
- Himpunan solusi : hubungan pada *Karnaugh Map* yang melingkupi seluruh nilai *minterm/maxterm*, tergantung dari penyederhanaan yang mau dilakukan
- Fungsi seleksi : memilih hubungan dengan jumlah nilai *minterm/maxterm* terbanyak
- Fungsi kelayakan : hubungan yang dipilih sudah merupakan hubungan yang memuat nilai *minterm/maxterm* terbanyak (dengan implikasi hubungan yang dipilih sudah menghasilkan kombinasi variabel yang paling sederhana) dan ada setidaknya satu nilai *minterm/maxterm* yang belum dilingkup pada himpunan solusi sebelumnya
- Fungsi obyektif : menghasilkan ekspresi logika yang paling sederhana dari *Karnaugh Map*

Berikut adalah *pseudocode* dari algoritma *greedy* yang diambil:

Deklarasi

```
kemungkinan : array of string {menampung semua kemungkinan hubungan yang ada}

solusi : array of string {menampung hasil greedy}

procedure solve()
{prosedur untuk menyelesaikan pemilihan hubungan pada Karnaugh Map}
Deklarasi
-
Algoritma
        while (nilai maxterm/minterm belum terlingkupi semua) do
                pilih kemungkinan hubungan Karnaugh Map dengan jumlah maxterm/minterm paling banyak terlingkupi dengan syarat ada maxterm/minterm yang belum terlingkupi di pilihan sebelumnya. jika ada dua kemungkinan hubungan dengan jumlah yang sama dan memenuhi syarat, pilih secara acak
```

Algoritma

```
Bangkitkan semua kemungkinan dalam Karnaugh Map dan masukkan ke dalam larik kemungkinan
solve()
```

Sebagai contoh, akan diambil *Karnaugh Map* pada tabel VI. Setelah dibangkitkan semua kemungkinan yang ada, maka, semua kemungkinannya adalah tabel VII, tabel VIII, dan tiga tabel lagi, yaitu jika masing-masing nilai 1 pada tabel VI direpresentasikan sendiri-sendiri:

x_2 \ x_1	**0**	**1**
0	1	1
1	0	1

Tabel IX. Kemungkinan hubungan pada Karnaugh Map jika nilai 1 pada saat x_1 dan x_2 0 diambil sendiri

x_2 \ x_1	**0**	**1**
0	1	1
1	0	1

Tabel X. Kemungkinan hubungan pada Karnaugh Map jika nilai 1 pada saat x_1 1 dan x_2 0 diambil sendiri

x_2 \ x_1	**0**	**1**
0	1	1
1	0	1

Tabel XI. Kemungkinan hubungan pada Karnaugh Map jika nilai 1 pada saat x_1 dan x_2 1 diambil sendiri

Algoritma *greedy* dengan spesifikasi yang telah disebutkan sebelumnya akan memilih secara acak dari tabel VII dan VIII, karena pada tabel VII dan VIII nilai 1 yang terdapat dalam hubungannya adalah 2, sedangkan pada tabel IX, X, dan XI nilai 1 yang terdapat dalam hubungannya hanyalah 1. Misalkan algoritma *greedy* memilih tabel VII, maka karena semua nilai 1 belum terlingkupi semua (nilai 1 pada saat x_1 dan x_2 1) , algoritma *greedy* harus memilih lagi, dan yang dipilih adalah tabel VIII, karena tabel VIII memiliki nilai 1 terbanyak dalam hubungannya (yaitu 2) dan memenuhi syarat setidaknya ada satu nilai yang belum dilingkup pada himpunan solusi sebelumnya (nilai 1 pada saat x_1 dan x_2 1).

Untuk memperjelas penggunaan *greedy* dalam *Karnaugh Map*, akan ambil permasalahan yang lebih rumit. Misalkan ada tabel kebenaran sebagai berikut:

a	b	c	d	f
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Tabel XII. Tabel kebenaran dengan empat variabel

Bila tabel kebenaran tersebut dikonversi menjadi *Karnaugh Map*, maka *Karnaugh Map* yang terbentuk:

ab / cd	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	0	1	0
10	1	0	0	1

Tabel XIII. Karnaugh Map dari Tabel XII

Dari *Karnaugh Map* tersebut, semua kemungkinan hubungannya adalah:

ab / cd	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	0	1	0
10	1	0	0	1

Tabel XIV. Tujuh kemungkinan jika nilai 1 diambil sendiri-sendiri

ab / cd	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	0	1	0
10	1	0	0	1

Tabel XV. Dua kemungkinan jika hubungan angka 1 berbentuk vertikal pada ujung-ujung Karnaugh Map diambil

ab / cd	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	0	1	0
10	1	0	0	1

Tabel XVI. Dua kemungkinan jika hubungan angka 1 berbentuk horizontal pada ujung-ujung Karnaugh Map diambil

ab / cd	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	0	1	0
10	1	0	0	1

Tabel XVII. Satu kemungkinan jika hubungan angka 1 berbentuk horizontal pada tengah Karnaugh Map diambil

ab / cd	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	0	1	0
10	1	0	0	1

Tabel XVIII. Satu kemungkinan jika hubungan angka 1 berbentuk vertikal pada tengah Karnaugh Map diambil

ab / cd	00	01	11	10
00	1	0	0	1
01	0	1	1	0
11	0	0	1	0
10	1	0	0	1

Tabel XIX. Satu kemungkinan jika hubungan angka 1 berbentuk persegi pada ujung-ujung Karnaugh Map diambil

Dari empat belas kemungkinan tersebut, yang algoritma *greedy* akan memilih tabel XIX, karena melingkupi paling banyak nilai 1, yaitu empat. Kemudian, dari algoritma *greedy* akan memilih satu secara acak dari hubungan yang melingkupi dua nilai 1, yaitu diantara tabel XV, XVI, XVII, dan XVIII. Tetapi karena pada tabel XV dan XVI semua nilai 1-nya telah terlingkupi oleh tabel XIX, bila terpilih sekalipun, tabel XV dan XVI tidak lolos fungsi kelayakan, sehingga tabel yang masuk himpunan solusi adalah XVII atau XVIII. Kemudian, karena masih ada nilai 1 yang belum terlingkupi, maka algoritma *greedy* akan memilih tabel diantara XVII dan XVIII yang belum dipilih di tahap sebelumnya. Setelah

pemilihan tersebut, semua nilai 1 telah terlingkupi, sehingga didapatlah ekspresi yang paling sederhana, yaitu:

$$\bar{b}\bar{d} + \bar{a}bd + abd \tag{32}$$

Hasil dari *greedy* yang didesain pasti menghasilkan solusi yang optimal untuk permasalahan ini. Hal ini dikarenakan penyelesaian *Karnaugh Map* harus memilih hubungan antar nilai dengan jumlah nilai terbanyak agar dapat ditemukan ekspresi yang minimal.

VIII. KESIMPULAN

Algoritma *greedy* mampu mencari ekspresi logika minimal menggunakan *Karnaugh Map* dengan asumsi semua kemungkinan hubungan *Karnaugh Map* dapat dibangkitkan.

REFERENCES

[1] Munir, Rinaldi. 2009. Diktat Kuliah IF2211 Strategi Algoritma. Program Studi Teknik Informatika STEI ITB.

[2] Brown, Stephen & Zvonko Vranesic. 2009. *Fundamental of Digital Logic with VHDL Design Third Edition*. Mc Graw-Hill

[3] *Karnaugh Map*. 17 Desember 2013 (20.00) <http://www.facstaff.bucknell.edu/mastascu/elessonshtml/Logic/Logic3.html>

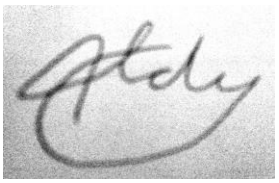
[4] *Logic Expression*. 17 Desember 2013 (20:00) <http://dept-info.labri.u-bordeaux.fr/~strandh/Teaching/AMP/Common/Strandh-Tutorial/expressions.html>

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, 20 Desember 2013

ttd



Aldy Wirawan
13511035