Implementasi Metode Partisi Untuk Minimisasi Tabel Keadaan Pada Perancangan Rangkaian Sekuensial

Achmad Yani[1)], Antoni[2)]
[1)]Politeknik Negeri Medan
[2)]Universitas Islam Sumatera Utara
antmunthe@gmail.com

Abstrak

Penelitian ini dimaksudkan untuk membuat perangkat lunak (program) minimisasi tabel keadaan pada perancangan rangkaian logika untuk jenis rangkaian sekuensial dengan metode partisi. Perangkat lunak ini dibuat dengan menggunakan bahasa pemrograman Visual Basic. Program menerima masukan yang disimpan dalam file teks dengan format tertentu yang menyajikan tabel keadaan asli. Hasil minimisasi tabel keadaan disimpan dalam file teks. Tabel asli dan tabel hasil minimisasi ditampilkan untuk memudahkan pembacaan. Meskipun secara teoritis tidak ada batasan jumlah masukan dan jumlah keadaan pada metode ini, untuk minimisasi dengan komputer digital, jumlah masukan dan jumlah keadaan dibatasi oleh ruang memori komputer yang tersedia.

Kata Kunci: *Metode Partisi, Minimisasi Tabel Keadaan, Rangkaian Sekuensial*

I. PENDAHULUAN

Perancangan sistem digital untuk jenis rangkaian sekuensial yang terbentuk atas beberapa buah flip-flop akan menjadi lebih sederhana jika banyaknya flip-flop yang diperlukan dalam tahap realisasi rangkaian dapat diminimumkan. Untuk meminimumkan banyaknya flip-flop, maka pada tahap pembuatan tabel keadaan (*state table*) yang merupakan representasi transisi keadaan dari rangkaian sekuensial yang diinginkan, ada peluang untuk meminimumkan banyaknya keadaan (mereduksi keadaan). Banyaknya keadaan berkaitan langsung dengan banyaknya flip-flop yang diperlukan melalui hubungan berikut:

$$2^{N_{FF}-1}+1 \le N_S \le 2^{N_{FF}}$$

dengan N_{FF} adalah banyaknya flip-flop dan N_S adalah banyak keadaan. Ini berarti bahwa dengan mereduksi keadaan, maka banyaknya flip-flop yang diperlukan menjadi (berpeluang) berkurang yang pada akhirnya akan meminimumkan biaya pemakaian gerbang logika untuk komponen flip-flop.

Salah satu metode yang dapat digunakan untuk mereduksi keadaan (sehingga menghasilkan tabel keadaan yang minimum) dalam perancangan rangkaian sekuensial adalah metode partisi. Metode ini dapat dikerjakan secara manual. Namun demikian, jika tabel keadaan mengandung keadaan yang jumlahnya banyak sekali, maka proses pengerjaan menjadi lama dan beresiko terjadinya kesalahan sehingga hasilnya menjadi belum minimal.

Penelitian ini akan membuat minimisasi pada rangkaian logika, tetapi untuk jenis rangkaian sekuensial. Minimisasi dilakukan terhadap tabel keadaan pada perancangan rangkaian sekuensial dengan implementasi metode partisi menggunakan komputer. Perangkat lunak yang diperlukan untuk implementasi metode ini adalah bahasa pemrograman Visual Basic. Dengan otomatisasi pengerjaan metode partisi ini, diharapkan waktu dan biaya perancangan menjadi minimal, dan hasil minimisasi menjadi lebih baik.

Berdasarkan latar belakang pada bagian Pendahuluan di atas, permasalahan yang akan diteliti dapat dirumuskan sebagai berikut:

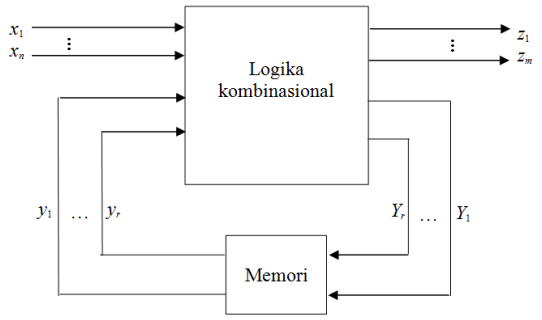
1) Bagaimana menyajikan tabel keadaan untuk suatu rangkaian sekuensial dalam bentuk yang tepat dan mudah untuk diolah dalam komputer digital?
2) Bagaimana prosedur baku minimisasi tabel keadaan dari suatu rangkaian sekuensial dengan metode partisi?
3) Bagaimana pembuatan algoritma untuk menerapkan prosedur baku di atas?
4) Bagaimana menerapkan algoritma di atas ke dalam bahasa pemrograman Visual Basic?

Tujuan dari penelitian ini adalah membuat perangkat lunak aplikasi untuk otomatisasi pengerjaan proses minimisasi tabel keadaan pada perancangan rangkaian sekuensial dengan implementasi metode partisi menggunakan bahasa pemrograman Visual Basic.

Dari hasil penelitian yang akan dilakukan, manfaat yang dapat diambil adalah bahwa dengan tersedianya perangkat lunak aplikasi ini, proses minimisasi tabel keadaan dengan metode partisi akan menjadi jauh lebih cepat dan lebih mudah dibandingkan jika dikerjakan secara manual. Di samping itu, proses penyederhanaan akan menjadi jauh lebih teliti dan menghasilkan tabel keadaan yang benar-benar minimum yang pada akhirnya akan menyederhanakan pemakaian gerbang flip-flop pada tahap realisasi.

II. TINJAUAN PUSTAKA

Rangkaian sekuensial adalah salah satu jenis rangkaian logika digital di samping rangkaian kombinasi. Berbeda dengan rangkaian kombinasi, rangkaian sekuensial selain memiliki masukan dan keluaran, juga memiliki **keadaan sekarang** dan **keadaan berikut**. Rangkaian sekuensial dapat dimodelkan dengan diagram blok seperti Gambar 1.



Gambar 1 Model angkaian sekuensial

Pada Gambar 1, parameter (x_1, …, x_n) adalah masukan, parameter (z_1, …, z_m) adalah keluaran, dan paramater (y_1, …, y_r) dan (Y_1, …, Y_r) masing-masing adalah keadaan sekarang (*present state*) dan keadaan berikut (*next state*). Hubungan di antara parameter-parameter ini adalah sebagai berikut:

$$z_i = g_i\,(x_1, \ldots, x_n, y_1, \ldots, y_r),$$
$$i = 1, \ldots, m$$

$$Y_i = h_i\,(x_1, \ldots, x_n, y_1, \ldots, y_r),$$
$$i = 1, \ldots, r$$

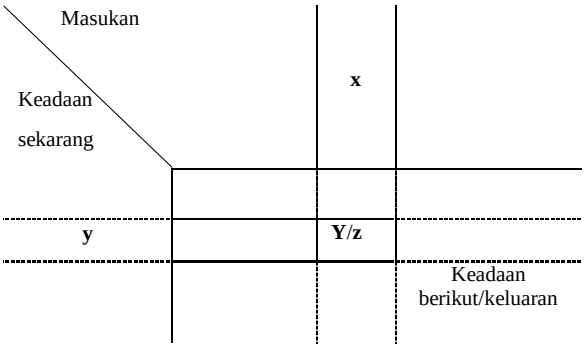
dengan g_i dan h_i adalah fungsi Boole (fungsi logika), dan z_i , x_i , y_i , dan Y_i semuanya adalah variabel biner (yang memiliki nilai logika 0 atau 1). Persamaan di atas dapat dituliskan dalam notasi vektor sebagai

$$\mathbf{z} = \mathbf{g}(\mathbf{x},\mathbf{y})$$

$$\mathbf{Y} = \mathbf{h}(\mathbf{x},\mathbf{y})$$

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_r \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_r \end{bmatrix}$$

Hubungan fungsional yang ada antara masukan, keluaran, keadaan sekarang, dan keadaan berikut dapat digambarkan dengan diagram keadaan atau tabel keadaan. Tabel keadaan untuk suatu rangkaian sekuensial dapat dibuat seperti Gambar 2



Gambar 2. Tabel keadaan

Pada Gambar 2, semua vektor masukan **x** dituliskan di bagian atas, sementara semua vektor keadaan sekarang **y** dituliskan di sebelah kiri. Isi (*entry*) tabel adalah berupa keadaan sekarang **Y** dan keluaran **z**. Tabel ini dibaca sebagai berikut: Untuk suatu masukan **x** dengan rangkaian sekuensial dalam keadaan **y**, rangkaian akan menghasilkan keadaan berikut **Y** dengan keluaran **z**.

Secara umum dapat dikatakan bahwa dua keadaan adalah setara (*equivalent*) jika tidak dapat dibedakan antara keduanya. Dengan kata lain, tidak dapat ditentukan dalam keadaan yang mana dari kedua keadaan yang setara itu suatu rangkaian sekuensial dimulai dengan memberikan masukan dan mengamati keluaran. Jika kondisi ini ada untuk setiap urutan masukan, maka salah satu dari kedua keadaan itu adalah mubazir dan dapat dihilangkan dari tabel keadaan tanpa mengubah perilaku rangkaian.

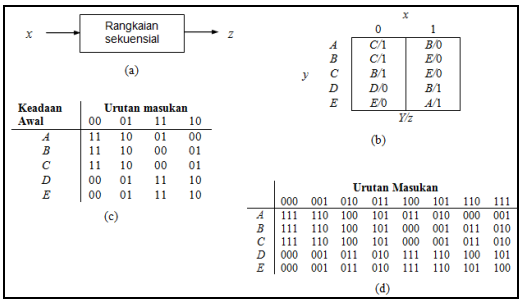
Keadaan S_1, S_2, …, S_j dari suatu rangkaian sekuensial dikatakan setara jika dan hanya jika, untuk setiap urutan masukan yang mungkin, urutan keluaran yang sama akan dihasilkan oleh rangkaian tanpa memandang apakah S_1, S_2, …, S_j keadaan awal atau bukan.

Definisi kesetaraan dapat juga dinyatakan dengan cara lain untuk pasangan keadaan. Misalkan S_k dan S_l adalah keadaan berikut dari rangkaian sekuensial jika masukan I_p diberikan sementara rangkaian dalam keadaan, masing-masing S_i dan S_j. Maka S_i dan S_j adalah setara jika dan hanya jika, untuk setiap masukan I_p yang mungkin,

1. keluaran yang dihasilkan oleh keadaan S_i sama dengan keluaran yang dihasilkan oleh keadaan S_j, dan
2. keadaan berikut S_k dan S_l setara.

Dua atau lebih keadaan dikatakan setara-*n* (*n-equivalent*) jika untuk urutan masukan yang sama dengan panjang *n* bit menghasilkan keluaran yang sama.

Kesetaraan keadaan dapat digambarkan melalui contoh sederhana seperti Gambar 3(a) dan (b).



Gambar 3 Keadaan mubazir
(a) Rangkaian sekuensial.
(b) Tabel keadaan.
(c) Urutan Keluaran untuk panjang 2 bit.
(d) Urutan Keluaran untuk panjang 3 bit.

Misalkan keadaan awal tidak diketahui. Jika masukan $x = 0$ diberikan ke rangkaian dan keluarannya $z = 1$, maka keadaan awalnya adalah *A* atau *B* atau *C*. Dengan cara yang sama, jika keluaran adalah z = 0 jika masukan $x = 0$ diberikan, keadaan awalnya adalah *D* atau *E*. Kesimpulannya, keadaan *A*, *B*, dan *C* adalah setara, dan keadaan *D* dan *E* juga setara untuk urutan masukan yang panjangnya 1 (disebut setara-1). Pada Gambar 3.c terlihat bahwa keadaan *B* dan *C*, dan keadaan *D* dan *E* adalah setara-2. Keadaan *B* dan *C* juga setara-3 seperti terlihat pada Gambar 3.d, dan selanjutnya dua keadaan ini juga setara-*K* untuk semua *K*.

Metode partisi melibatkan penentuan secara berlanjut partisi P_K , $K = 1, 2, 3, \ldots, l$, di mana setiap P_K tersusun dari sejumlah blok, yang masing-masing blok terdiri atas sekelompok satu atau lebih keadaan. Keadaan yang terdapat dalam suatu blok dari P_K adalah setara-*K*. Dengan kata lain, suatu rangkaian sekuensial dengan keadaan S_1, S_2, S_3, S_4, S_5, jika $P_K = (S_1S_3)\ (S_2S_4)(S_5)$, maka P_K mengandung tiga blok, dan S_1 dan S_3 adalah setara-*K*, seperti halnya S_2 dan S_4. S_5 tidak setara-*K* dengan satu pun keadaan yang lain dalam rangkaian sekuensial. Untuk lebih jelasnya, rangkaian sekuensial pada Gambar 3.b akan digunakan sebagai contoh dalam menggambarkan prosedur pengerjaan metode partisi.

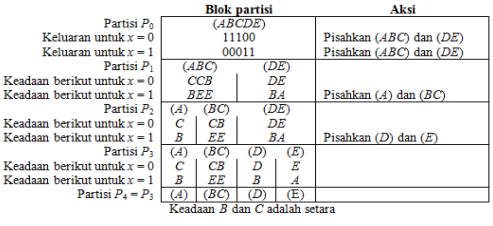
Prosedur Partisi

- **Langkah 1**. Partisi pertama P_1 dibentuk dengan dua atau lebih keadaan dalam blok yang sama dari P_1 jika dan hanya jika **keluarannya** identik untuk setiap masukan. Untuk contoh pada Gambar 3.b, $P_1 = (ABC)(DE)$, sehingga semua keadaan di dalam masing-masing blok adalah setara-1.
- **Langkah 2**. Partisi berikutnya P_K , $K = 2, 3, 4, \ldots, l$, diturunkan dengan menempatkan dua atau lebih keadaan dalam blok yang sama dari P_K jika dan hanya jika untuk setiap nilai masukan, **keadaan berikutnya** semuanya terletak di dalam satu **blok tunggal** P_{K-1}.
- **Langkah 3**. Jika $P_{K+1} = P_K$, yaitu, jika partisi berulang, keadaan dalam masing-masing blok P_K yang setara-*K* adalah setara-(*K*+1), setara-(*K*+2), dan seterusnya, dan P_K dikatakan **partisi kesetaraan**. Dalam contoh ini, jelas terlihat bahwa $P_4 = P_3$ dan dengan demikian keadaan *B* dan *C* adalah setara-*K* untuk setiap *K*; yaitu, kedua keadaan ini (*B* dan *C*) adalah setara.

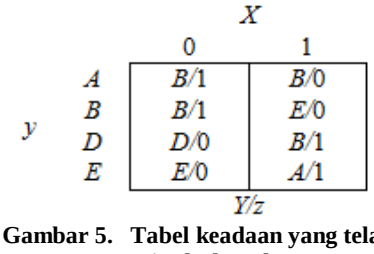
Dalam mengerjakan prosedur partisi untuk contoh ini, perlu diperiksa kelompok keadaan dalam setiap blok P_1. Dalam blok pertama, keadaan berikut untuk *A*, *B*, dan *C* dengan $x = 0$ semuanya terletak dalam blok yang sama P_1. Namun demikian, untuk $x = 1$, keadaan berikut *A* terletak dalam blok dari P_1 yang berbeda dengan keadaan berikut dari *B* dan *C*. Dengan demikian, blok (*ABC*) yang terdapat dalam P_1 dibagi menjadi blok-blok (*A*)(*BC*) dalam P_2 seperti pada Gambar 4. Keadaan berikut dari *D* dan *E* terletak dalam P_1 yang sama baik untuk $x = 0$ maupun $x = 1$, sehingga *D* dan *E* akan tetap dalam blok P_2 yang sama. Maka $P_2 = (A)(BC)(DE)$ dan keadaan dalam masing-masing blok adalah setara-2. Dengan demikian, P_2 tepat bersesuaian dengan Gambar 3.c.

Partisi P_3 diperoleh dengan menguji setiap blok dari P_3. Keadaan berikut dari *B* dan *C* terletak dalam blok P_2 yang sama untuk setiap masukan, sehingga blok (*BC*) tetap dalam P_3. Namun demikian, keadaan berikut untuk *D* dan *E* dengan $x = 1$ terletak dalam blok-blok yang berbeda dari P_2, sehingga kedua keadaan ini harus muncul dalam blok yang berbeda dari P_3. Dengan demikian, $P_3 = (A)(BC)(D)(E)$.

Prosedur memperoleh partisi yang berlanjut diulang terus sampai kondisi yang dinyatakan dalam Langkah 3 diperoleh. Hasil akhir menunjukkan bahwa $P_4 = P_3$. Blok (*BC*) dalam partisi P_3 berarti bahwa keadaan *B* setara dengan keadaan *C*. Maka tabel keadaan pada Gambar 3.b dapat disederhanakan dengan cara menghapus baris *C*, dan mengganti setiap kemunculan keadaan *C* dalam tabel dengan keadaan *B*, yang hasilnya dapat dilihat pada Gambar 5.



Gambar 4. Kesetaraan keadaan dengan metode partisi



Gambar 5. Tabel keadaan yang telah Disederhanakan

III. METODOLOGI PENELITIAN

Penelitian yang akan dilakukan menggunakan metodologi sebagai berikut:

(1) Menyajikan tabel keadaan untuk suatu rangkaian sekuensial dalam bentuk yang tepat dan mudah untuk diolah dalam komputer digital.
(2) Menuliskan algoritma proses minimisasi tabel keadaan dari suatu rangkaian sekuensial dengan metode partisi.
(3) Mengimplementasikan algoritma di atas ke dalam bahasa pemrograman Visual Basic.
(4) Menguji beberapa contoh tabel keadaan sebagai masukan untuk disederhanakan dengan program yang telah dibuat.
(5) Membandingkan hasil minimisasi melalui penggunaan program aplikasi yang dibuat dengan hasil penyederhanaan yang dilakukan secara manual.

IV. HASIL DAN PEMBAHASAN

4.1 Algoritma Proses Minimisasi

Dengan menggunakan simbol-simbol parameter yang sama seperti pada bagian Tinjauan Pustaka, maka algoritma proses minimisasi tabel keadaan dengan metode partisi dapat dibuat dengan menggunakan *flowchart* seperti pada Gambar 6.

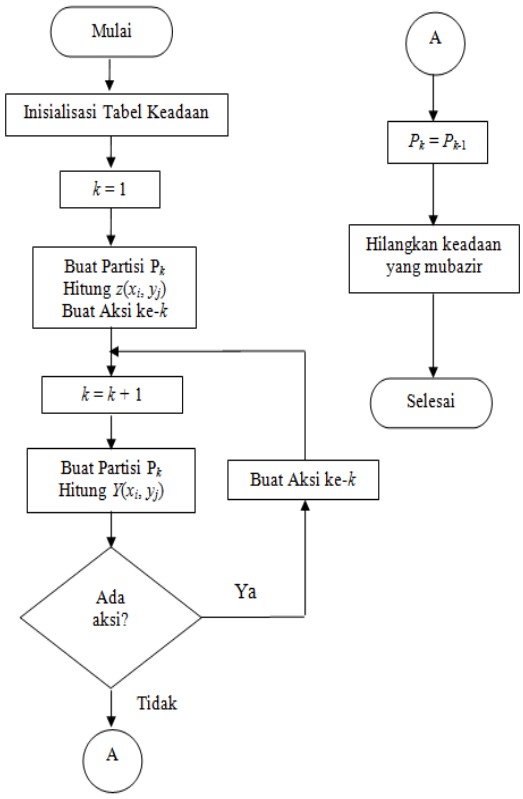
Sebagai inisialisasi, untuk $k = 0$, partisi awal (partisi nol, P_0) berisi himpunan dari semua keadaan sekarang. Untuk $k = 1$, partisi P_1 diperoleh setelah dievaluasi nilai keluaran z untuk setiap keadaan sekarang dan setiap masukan yang menghasilkan aksi pemisahan blok jika keluarannya berbeda. Untuk $k = 2, 3, 4, \ldots$, partisi P_k diperoleh setelah dievaluasi nilai keadaan berikut Y untuk setiap keadaan sekarang dan setiap masukan yang (kemungkinan) menghasilkan aksi pemisahan blok jika keadaan berikutnya tidak terletak dalam satu blok yang sama pada partisi sebelumnya (P_{k-1}). Proses partisi berulang terus jika ada aksi pemisahan blok dari hasil evaluasi nilai keadaan berikut. Jika tidak ada lagi aksi pemisahan blok, maka tercapai kondisi $P_k = P_{k-1}$. Tabel akhir hasil minimisasi diperoleh dengan cara menghilangkan keadaan-keadaan yang mubazir.

4.2 Penyajian Tabel Keadaan sebagai Masukan bagi Program

Sebagai masukan yang akan dibaca program, maka tabel keadaan asli yang akan dibaca program disimpan dalam file teks ASCII dengan format sebagai berikut:

x {x1 x2 … xn} [; Komentar]
y {y1 y2 … yr} [; Komentar]
y1 {daftar entry tabel keadaan untuk baris keadaan y1}
y2 {daftar entry tabel keadaan untuk baris keadaan y2}
yr {daftar entry tabel keadaan untuk baris keadaan yr

dengan **n** adalah banyaknya masukan dan **r** adalah banyaknya keadaan. Komentar yang terletak di dalam tanda kurung siku adalah *optional* (dapat dihilangkan) karena hanya berfungsi untuk keterangan. Setiap *entry* dalam daftar entry dari tabel keadaan asli dituliskan dengan dipisahkan spasi.



Gambar 6. Flowchart proses minimisasi dengan metode partisi

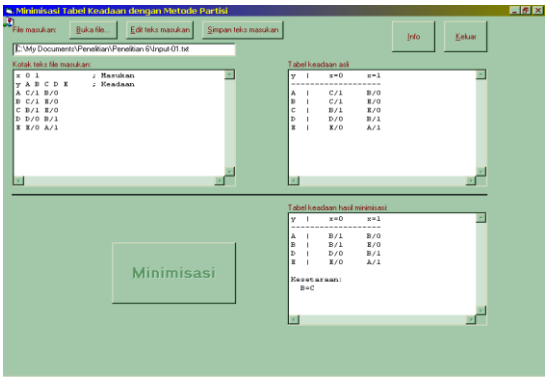
4.3 Implementasi Program dengan Visual Basic

Perangkat lunak untuk mengerjakan minimisasi tabel keadaan berbasis metode partisi dibuat dengan menggunakan Visual Basic 6. Berdasarkan algoritma yang harus dikerjakan pada metode ini, maka dibuat beberapa subrutin sebagai berikut:

1. Baca_File_Masukan()
2. Inisialisasi()
3. Evaluasi_Nilai_Keluaran()
4. Buat_Partisi()
5. Buat_Aksi()
6. Evaluasi_Nilai_Keadaan_Berikut()
7. Ada_Aksi()
8. Kesetaraan_Keadaan()
9. Tampilkan_Tabel_Keadaan()

4.4 Pengujian Hasil Program

Gambar 7 adalah tampilan hasil program untuk contoh seperti pada bagian Tinjauan Pustaka



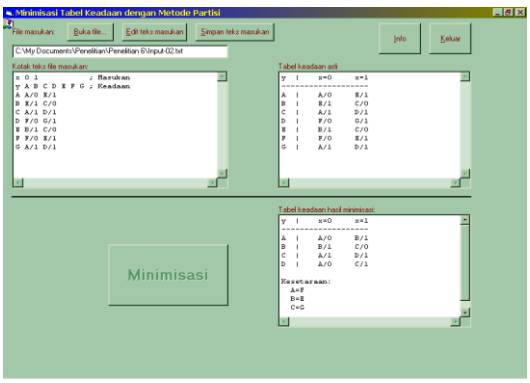
Gambar 7. Tampilan Hasil Program: Contoh 1

File masukan yang isinya berupa penyajian tabel keadaan asli dituliskan seperti berikut:

```
x 0 1          ; Masukan
y A B C D E ; Keadaan
A C/1 B/0
B C/1 E/0
C B/1 E/0
D D/0 B/1
E E/0 A/1
```

Baris **x 0 1** berarti bahwa masukan x ada dua buah, yaitu 0 dan 1. Baris **y A B C D E** berarti bahwa keadaan ada lima buah, yaitu A, B, C, D, dan E. Kelima baris berikutnya menyatakan *entry* (isi) tabel keadaan asli. Kotak teks file masukan yang berisi representasi tabel keadaan diperjelas bentuknya dengan kotak teks di sebelah kanannya yang berlabel 'Tabel keadaan asli'. Dengan menekan *button* **Minimisasi**, maka proses minimisasi tabel keadaan dilakukan sehingga menghasilkan tabel keadaan yang terdapat pada kotak teks yang berlabel 'Tabel keadaan hasil minimisasi'. Di sini tampak bahwa banyaknya keadaan telah berkurang dari 5 (yaitu A, B, C, D, dan E) menjadi 4 (yaitu A, B, D, dan E) karena adanya kesetaraan B = C, sehingga semua kemunculan keadaan C dalam tabel keadaan asli digantikan dengan keadaan B, dan baris C dihilangkan.

Untuk contoh lain, dapat dilihat pada Gambar 8



Gambar 8. Tampilan Hasil Program: Contoh 2

V. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari penelitian yang dilakukan, dapat diambil beberapa kesimpulan sebagai berikut:

1. Metode partisi, salah satu metode untuk menyederhanakan tabel keadaan pada perancangan rangkaian sekuensial, secara teoritis dapat dikerjakan secara manual. Metode ini dikerjakan dengan mengikuti prosedur tertentu yang sistematis. Dengan sifat pengerjaan yang sistematis ini, maka metode ini cocok dan mudah diterapkan dengan menggunakan komputer digital melalui pemrograman.
2. Perangkat lunak minimisasi tabel keadaan berbasis metode partisi yang dibuat dengan Visual Basic 6 pada penelitian ini sangat menguntungkan para perancang rangkaian digital karena mengefektifkan waktu penyederhanaan dan juga memperoleh ketelitian yang sangat tinggi yang pada gilirannya akan dapat meminimumkan biaya realisasi rangkaian digital yang dihasilkan karena dapat meminimumkan pemakaian elemen memori berupa flip-flop.
3. Secara teoritis, tidak ada batasan jumlah keadaan untuk metode partisi. Namun demikian, untuk otomatisasi dengan komputer digital, maka jumlah keadaan untuk tabel keadaan dibatasi oleh ruang memori komputer yang tersedia.

5.2 Saran

1. Perangkat lunak hasil penelitian ini dapat dikembangkan, misalnya dengan menambahkan simulasi proses pengerjaan minimisasi dalam metode partisi secara visual. Hal ini dapat membantu, misalnya, untuk melihat cara kerja metode partisi bagi mahasiswa yang sedang mempelajari metode ini dalam mata kuliah Rangkaian Logika atau Perancangan Sistem Digital.
2. Perangkat lunak ini dapat juga dikembangkan dengan menampilkan (menggambarkan) realisasi rangkaian sekuensial hasil minimisasi secara visual.

DAFTAR PUSTAKA

[1] Nelson, Victor P., H.T. Nagle, B.D. Carroll, and J. D. Irwin, 1995, *Digital Logic Circuit Analysis and Design*, Prentice-Hall International, Inc., New Jersey.

[2] Malvino, A. P., and D. P. Leach, 1986, *Digital Principles and Applications*, McGraw-Hill, New York.

[3] Tarigan, Pernantin, 2001, *Rangkaian Logika Digital*, USU Press, Medan.

[4] Dwihono, 1996, *Rangkaian Logika*, Penerbit INDAH, Surabaya.

[5] Texas Instruments, Inc., 1971, *Designing with TTL Integrated Circuits*, McGraw-Hill.
[6] Yani, Achmad, 2003, *Perancangan dan Pembuatan Perangkat Lunak Untuk Menyederhanakan Fungsi Logika Dengan Metode Quine-McCluskey Menggunakan Visual Basic*, Penelitian Dana Rutin Politeknik Negeri Medan.
[7] Yani, Achmad, 2004, *Implementasi Metode Tabulasi Quine-McCluskey Untuk Minimisasi Fungsi Logika Berkeluaran Ganda Menggunakan Visual Basic*, Penelitian Dana Rutin Politeknik Negeri Medan.
[8] Marcovitz, A.B., 2010, *Introduction to logic design 3rd ed.*, McGraw-Hill Higher Education, Boston, USA.
[9] Setiyani dan Suyanto, 2019, *Implementasi Reduksi Keadaan Rangkaian Digital Sekuensial Metode Bagan Implikasi***,** Jurnal Tekno, vol. 16, no. 2, pp. 23–34, Oct.