Jurnal TEKNO
(Civil Engineeering, Elektrical Engineeering and Industrial Engineeering)
Vol. 16, No. 2, Oktober 2019 , p-ISSN:1907-5243, e-ISSN: 2655-8416

Implementasi Reduksi Keadaan Rangkaian Digital Sekuensial Metode Bagan Implikasi

Theresia Prima Ari Setiyani[1], Yohanes Suyanto[2]

[1] Jurusan Teknik Elektro, Universitas Sanata Dharma, Yogyakarta, Indonesia
[2] Departemen Ilmu Komputer dan Elektronika, Universitas Gadjah Mada, Yogyakarta, Indonesia

E-mail: ariprima@usd.ac.id, yanto@ugm.ac.id

Abstract

The implementation of state reduction in sequential digital circuits is made for learning the topic of state reduction. The method used for state reduction is an implication chart. This method starts with reading the transition table state and transfered into the array structure. Based on this array structure a table or chart of initial implications is arranged. The next process is to change the contents of the table if there are cells that meet the requirements to be declared as identical or not identical. This process is repeated continuously until there is no change in cell contents. The state of reduction implementation is made using the Python programming language and PHP. The results of the implementation are successful for the state transition table with 1 input and 1 output.

Key words:*State reduction, implication chart, implication table, sequential digital circuit*

1. PENDAHULUAN

Rangkaian sekuensial terdiri atas rangkaian kombinasional dan elemen memori. Beberapa keluaran dari rangkaian kombinasional digunakan sebagai masukan bagian elemen memori. Keluaran dari elemen memori diumpanbalikkan ke bagian rangkaian kombinasional. Rangkaian sekuensial dinyatakan dalam deretan masukan, keluaran, dan keadaan internal. Dua tipe rangkaian sekuensial yaitu sinkron dan tak sinkron [1].

Prosedur klasik desain rangkaian sekuensial meliputi: (1) penentuan diagram transisi keadaan dari masalah yang ada; (2) penyusunan tabel transisi keadaan; (3) reduksi keadaan; (4) penentuan nilai keadaan; (5) penyusunan tabel kebenaran; (6) persamaan rangkaian; (7) gambar rangkaian. Minimisasi atau reduksi rangkaian dapat dilakukan pada langkah (3) dan (6). Langkah (6) merupakan minimisasi atau reduksi bentuk kombinasional sehingga dapat menggunakan metode persamaan aljabar boole, peta karnaugh atau metode tabulasi. Dalam makalah ini akan dibahas tentang reduksi keadaan [2].

Reduksi keadaan pada *finite state machine* (FSM) adalah masalah penting dalam sintesis rangkaian sekuensial, karena kompleksitas implementasi akhir terkait dengan jumlah keadaan (*state*) dalam diagram transisi keadaan atau *state transition diagram* (STG) . Untuk FSM komplit, masalah reduksi status dapat diselesaikan dalam polinomial [3] dan [4], algoritme yang paling efisien adalah [5]. Pendekatan standar untuk masalah ini didasarkan pada identifikasi keadaan yang identik atau kompatibel. Sejumlah sistem praktis yang didasarkan pada pendekatan ini telah diusulkan, berdasarkan pada pencacahan eksplisit [6] dan implisit [7].

1.1 Tinjauan Pustaka

Prosedur reduksi keadaan yang dikenal antara lain metode partisi keadaan yang ekuivalen dan tabel atau bagan implikasi [8] dan [9] . Prosedur tersebut mengacu pada definisi berikut:

1. Dua keadaan A dan B dikatakan ekuivalen jika dan hanya jika untuk setiap kemungkinan deretan masukan, rangkaian sekuensial menghasilkan deretan output yang sama, baik diawali dari keadaan A atau keadaan B.

2. Dua keadaan A dan B dikatakan ekuivalen sepanjang *k*, jika untuk setiap deretan masukan sepanjang *k*, rangkaian menghasilkan keluaran yang identik, baik diawali dari keadaan A atau B.

3. Dua keadaan A dan B dikatakan berbeda jika setidaknya ada satu deretan masukan yang menghasilkan keluaran yang berbeda tergantung pada awal keadaan dari A atau B.

1.1.1 Partisi ekuivalensi keadaan

Metode partisi ekuivalensi keadaan mempunyai tahapan sebagai berikut [8]:

1. Mulai dengan P_0, yang berisi semua keadaan rangkaian dalam satu blok.

2. Tentukan blok P_1 dengan mengamati keluaran dari tabel keadaan. Kelompokkan berdasarkan pola keluaran yang sama ke dalam satu blok.

3. Untuk menentukan blok-blok berikutnya yaitu P_i dari P_{i-1} dengan $i > 1$:

   a) Tentukan X_i *yang* merupakan urutan berikutnya dari setiap blok P_{i-1}; jika X_i dari suatu blok ada pada blok P_{i-1} yang sama, maka blok tidak dipecah, jika tidak maka blok dipecah.

   b) Ulang (a) untuk semua X_i.

4. Ulang langkah (3) hingga tidak ada lagi pemecahan blok.

1.1.2 Bagan implikasi

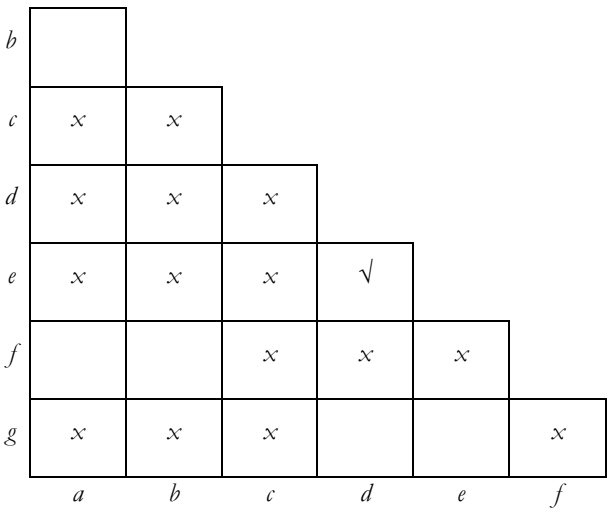
Metode bagan implikasi diawali dengan membuat bagan berupa kotak-kotak tersusun dalam bentuk segi tiga dengan panjang alas dan tinggi sebesar jumlah keadaan dikurangi satu. Sebagai contoh kasus adalah tabel transisi keadaan yang tertera pada Tabel 1.

Tabel 1. Transisi keadaan belum direduksi

Present state	*Next state*		*Output*	
	x=0	*x=1*	*x=0*	*x=1*
a	*d*	*b*	*0*	*0*
b	*e*	*a*	*0*	*0*
c	*g*	*f*	*0*	*1*
d	*a*	*d*	*1*	*0*
e	*a*	*d*	*1*	*0*
f	*c*	*b*	*0*	*0*
g	*a*	*e*	*1*	*0*

Berikut langkah-langkah selengkapnya:

1. Buat bagan implikasi seperti pada Gambar 1.
   Beri tanda centang pada perpotongan baris dan kolom yang menandakan bahwa pasangan baris dan kolom itu ekuivalen. Misalnya baris e dan kolom d adalah ekivalen karena mempunyai nilai output yang sama untuk semua kombinasi nilai input.
   Beri tanda silang untuk baris dan kolom yang jelas tidak ekuivalen.



Gambar 1. Bagan implikasi awal

2. Cek kotak yang kosong.
   Tulis syarat atau kondisi agar baris kolom tersebut ekuivalen. Contoh (a,b) akan ekuivalen jika (d,e) ekuivalen. Demikian juga (a,f) akan ekuivalen jika (c,d) ekuivalen. Hasil akhir dari langkah ini dapat dilihat pada Gambar 2.

b	*d,e*					
c	x	x				
d	x	x	x			
e	x	x	x	√		
f	*c,d*	*e,c* *a,b*	x	x	x	
g	x	x	x	*d,e*	*d,e*	x
	a	*b*	*c*	*d*	*e*	*f*

Gambar 2. Bagan implikasi setelah pengisian kondisi ekuivalen

3. Di langkah pertama, diketahui bahwa (d,e) adalah ekuivalen. Oleh karena itu (d,e) pada kolom b baris a dapat dibubuhi tanda centang dan didapatkan bagan Gambar 3.

b	*d,e* √					
c	x	x				
d	x	x	x			
e	x	x	x	√		
f	*c,d* x	*e,c* x *a,b*	x	x	x	
g	x	x	x	*d,e* √	*d,e* √	x
	a	*b*	*c*	*d*	*e*	*f*

Gambar 3. Bagan implikasi setelah pembubuhan tanda ekuivalen pada kotak ekuivalen bersyarat

Dengan demikian ada 4 kotak yang bertanda centang yaitu (a,b), (d,e), (d,g), dan (e,g). Dari tiga terakhir dapat disimpulkan bahwa keadaan d, e, dan g ekuivalen. Hasil reduksi keadaan tertera pada Tabel 2.

Tabel 2. Transisi keadaan setelah direduksi

Present state	*Next state*		*Output*	
	x=0	*x=1*	*x=0*	*x=1*
a	*d*	*a*	*0*	*0*
c	*d*	*f*	*0*	*1*
d	*a*	*d*	*1*	*0*
f	*c*	*a*	*0*	*0*

1.2 Metodologi Penelitian

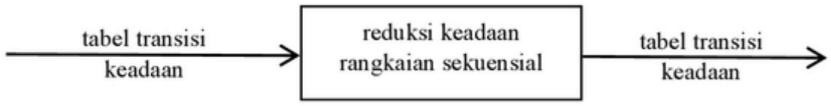
Penelitian ini dilakukan dengan langkah-langkah sebagai berikut

a. studi literatur: hasil yang didapat berupa dokumen, artikel, buku yang berisi pemahaman tentang teori reduksi keadaan pada rangkaian sekuensial metode bagan implikasi;
b. perancangan algoritme: didapat algoritme reduksi keadaan pada rangkaian sekuensial;
c. implementasi algortime: didapat program reduksi keadaan pada rangkaian sekuensial;
d. uji coba program: didapat hasil uji coba;
e. pembahasan: didapat hasil pembahasan tentang implementasi metode reduksi keadaan pada rangkaian sekuensial.

2. PEMBAHASAN

2.1 Konsep

Aplikasi yang dibuat mempunyai masukan berupa tabel transisi keadaan dan keluaran berupa tabel transisi keadaan yang telah direduksi. Tabel memuat kolom *present state, next state,* dan *output*. Gambar 4 menunjukkan diagram blok aplikasi reduksi keadaan.



Gambar 4. Diagram blok aplikasi reduksi keadaan

2.2 Perancangan dan implementasi

Masukan dan keluaran aplikasi berupa tabel. Oleh karena itu keduanya cocok menggunakan tipe data larik. Indeks larik dapat menggunakan nama keadaan. Misalnya ‘a’, ‘b’, ‘c’, dan seterusnya sampai dengan ‘z’. Elemen larik berisi *next state* dan *output*. Contoh bentuk masukan adalah (dalam PHP)

```
$input=[
        'a'=>[['d','b'],[0,0]],
        'b'=>[['e','a'],[0,0]],
        'c'=>[['g','f'],[0,1]],
        'd'=>[['a','d'],[1,0]],
        'e'=>[['a','d'],[1,0]],
        'f'=>[['c','b'],[0,0]],
        'g'=>[['a','e'],[1,0]]
];
```

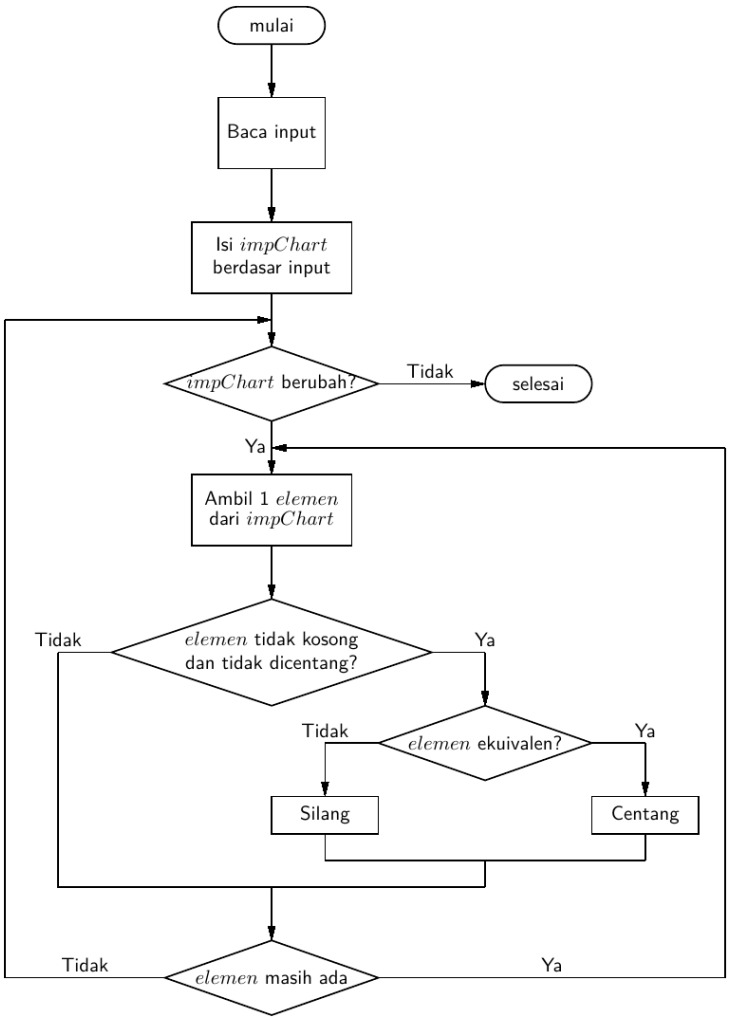
Masukan tersebut maksudnya untuk *present state* 'a' mempunyai *next state* 'd' untuk masukan x=0 dan 'b' untuk masukan x=1 serta keluaran '0' untuk masukan x=0 dan '0' juga untuk masukan x=1.

Algoritme untuk aplikasi ini dapat dilihat pada Gambar 5.

```
Inisialisasi impChart
Baca input
Isi impChart sesuai dengan masukan input
iter <- 0
While ada perubahan isi impChart
        Foreach elemen dalam impChart
                if elemen tidak kosong dan tidak centang
                        if elemen ekuivalen
                                Tandai centang
                        Else If elemen tidak ekuivalen
                                Tandai tidak ekuivalen
                        Endif
                Endif
        Endforeach
Endwhile
```

Gambar 5. Algoritme reduksi keadaan

Bentuk lain dari algoritme Gambar 5 adalah diagram alir seperti pada Gambar 6. Implementasi dari algoritme Gambar 5 dalam Python ada pada Gambar 7. Variabel inputState adalah variabel masukan berisi tabel transisi keadaan. Variabel change digunakan sebagai *flag* tanda adanya perubahan dalam impChart jika bernilai 1, atau tidak ada perubahan jika bernilai 0. Fungsi checkEquiv digunakan untuk mendeteksi apakah elemen dalam impChart ekuivalen antara baris dan kolomnya. Selain diimplementasikan dalam Python algoritme tersebut juga diimplementasikan dalam PHP. Implementasi algoritme tersebut dalam PHP dapat dilihat pada Gambar 8.



Gambar 6. Diagram alir reduksi keadaan

```
NUMSTATE=len(inputState)

change=1
while change==1:
        change=0
        col=0
        while col<NUMSTATE:
                row=col+1
                while row<NUMSTATE:
                        cell=impchart[row][col]
                        if cell != '--' and cell != "V":
                                c = cell.split(" ")
                                if len(c)>1:
                                        cek2=checkEquiv(c[1],impchart)
                                else:
                                        cek2=""
                                cek1=checkEquiv(c[0],impchart)
                                if cek1=="--" or cek2=="--":
                                        cek='--'
                                else:
                                        if cek1=="V" and (cek2=="V" or cek2==""):
                                                cek='V'
                                        else:
                                                cek=cek1+" "+cek2

                                impchart[row][col]=cek.strip()
                        if cell != impchart[row][col]:
                                change=1
                        row +=1

                col +=1

        displayChart(impchart)
```

Gambar 7. Implementasi algoritme reduksi keadaan dalam Python

```
echo "<h3>Bagan awal</h3>";

displayChart($impchart);

$change=1; $iterasi=0;
while ($change==1){
        $change=0;
        $col=0;
        while ($col<$NUMSTATE){
                $row=$col+1;
                while ($row<$NUMSTATE){
                        $cell=$impchart[$row][$col];
                        if ($cell != '--' and $cell != "V"){
                                $c = explode(" ",$cell);
                                if (count($c)>1){
                                        $cek2=checkEquiv($c[1],$impchart);
                                }else{
                                        $cek2="";
                                }
                                $cek1=checkEquiv($c[0],$impchart);
                                if ($cek1=="--" or $cek2=="--"){
                                        $cek='--';
                                }else{
                                        if ($cek1=="V" and ($cek2=="V" or $cek2="")){
                                                $cek='V';
                                        }else{
                                                $cek=$cek1." ".$cek2;
                                        }
                                }
                                $impchart[$row][$col]=trim($cek);
                        }
                        if ($cell != $impchart[$row][$col]){
                                $change=1;
                        }
                        $row +=1;
                }

                $col +=1;
        }

        $iterasi++;
        echo "<h3>Iterasi {$iterasi}</h3>";
        displayChart($impchart);
}
```

Gambar 8. Implementasi algoritme reduksi keadaan dalam PHP

2.3 Uji coba

Uji coba pertama dilakukan dengan menggunakan data masukan seperti Tabel 1 dan diubah dalam format Python dan PHP menjadi seperti pada Gambar 9.

```
nextState={\
'a':{0:['d','b'],1:[0,0]},\
'b':{0:['e','a'],1:[0,0]},\
'c':{0:['g','f'],1:[0,1]},\
'd':{0:['a','d'],1:[1,0]},\
'e':{0:['a','d'],1:[1,0]},\
'f':{0:['c','b'],1:[0,0]},\
'g':{0:['a','e'],1:[1,0]}}
```

```
$nextState=[
'a'=>[['d','b'],[0,0]],
'b'=>[['e','a'],[0,0]],
'c'=>[['g','f'],[0,1]],
'd'=>[['a','d'],[1,0]],
'e'=>[['a','d'],[1,0]],
'f'=>[['c','b'],[0,0]],
'g'=>[['a','e'],[1,0]]
];
```

Gambar 9. Data masukan dalam format Python dan PHP

Hasilnya adalah

Bagan awal

b	d,e					
c	--	--				
d	--	--	--			
e	--	--	--	V		
f	c,d	c,e a,b	--	--	--	
g	--	--	--	d,e	d,e	--
	a	b	c	d	e	f

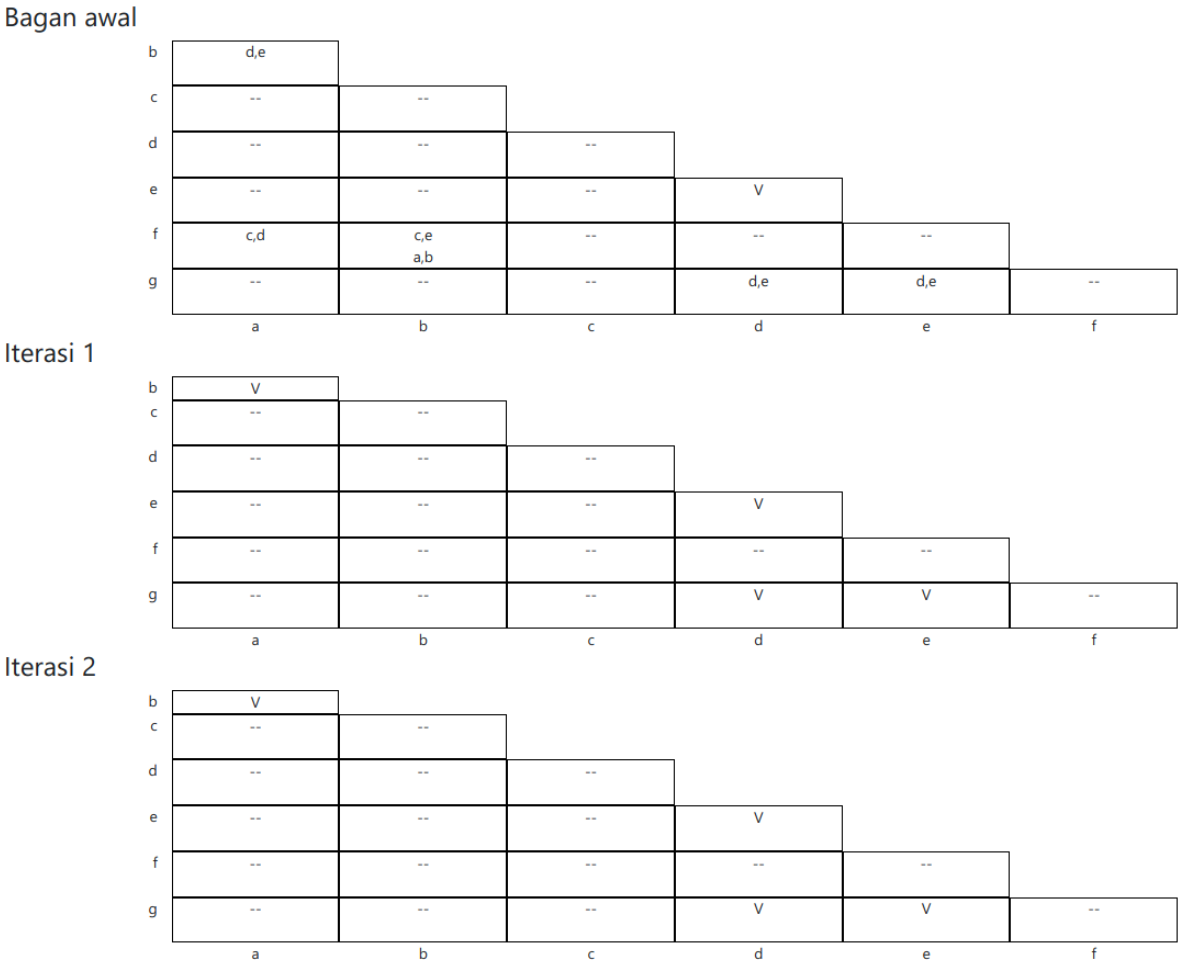
Iterasi ke- 1

b	V					
c	--	--				
d	--	--	--			
e	--	--	--	V		
f	--	--	--	--	--	
g	--	--	--	V	V	--
	a	b	c	d	e	f

Iterasi ke- 2

b	V					
c	--	--				
d	--	--	--			
e	--	--	--	V		
f	--	--	--	--	--	
g	--	--	--	V	V	--
	a	b	c	d	e	f

Dengan demikian (a,b), (d,e), (d,g), dan (e,g) ekuivalen. Hasil ini sesuai dengan Gambar 3. Hasil keluaran dari implementasi menggunakan PHP ada pada Gambar 10.



Gambar 10. Keluaran dari implementasi dalam PHP

Dengan mereduksi b menjadi a, g menjadi e, e menjadi d, maka tabel transaksi keadaan menjadi seperti pada Tabel 2.

2.4 Hasil

Tabel 3 adalah tabel transisi keadaan yang pada awalnya mempunyai 6 keadaan dengan satu masukan dan satu keluaran. Setelah direduksi menjadi 5 keadaan karena (a,c) ekuivalen.

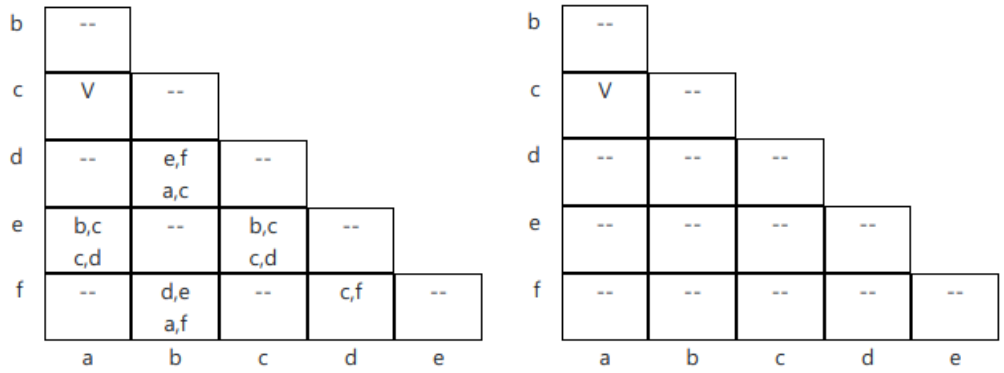
Tabel 3. Transisi keadaan dengan 6 keadaan sebelum dan sesudah direduksi

Present state	*Next state*		*Output*	
	x=0	*x=1*	*x=0*	*x=1*
a	*b*	*c*	*0*	*0*
b	*e*	*a*	*1*	*0*
c	*b*	*c*	*0*	*0*
d	*f*	*c*	*1*	*0*
e	*c*	*d*	*0*	*0*
f	*d*	*f*	*1*	*0*

Present state	*Next state*		*Output*	
	x=0	*x=1*	*x=0*	*x=1*
a	*b*	*a*	*0*	*0*
b	*e*	*a*	*1*	*0*
d	*f*	*a*	*1*	*0*
e	*a*	*d*	*0*	*0*
f	*d*	*f*	*1*	*0*

Bentuk bagan awal dan akhir untuk proses Tabel 3 ada pada Gambar 11. Pada awalnya keadaan (a,c) jelas ekuivalen. Keadaan (a,e) ekuivalen jika (b,c) dan (c,d) ekuivalen. Namun (b,c) tidak ekuivalen sehingga (a,c) menjadi tidak

ekuivalen. Keadaan (b,d) menjadi tidak ekuivalen karena (e,f) tidak ekuivalen. Demikian juga (b,f) yang bersyarat (d,e), (c,e) yang bersyarat (b,c), dan (d,f) yang bersyarat (c,f) menjadi tidak ekuivalen. Dengan demikian hanya (a,c) yang ekuivalen.



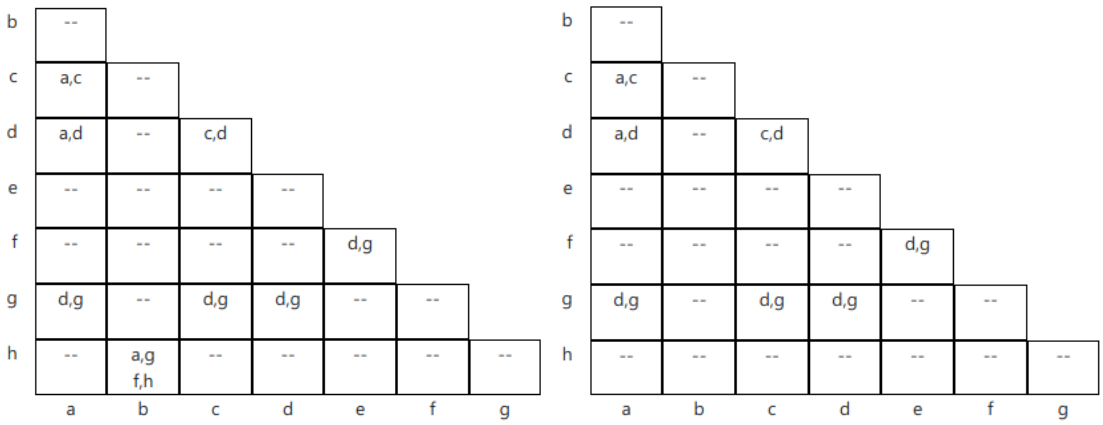
Gambar 11. Keluaran dari untuk data dari Tabel 3 bagan awal dan akhir iterasi

Hasil untuk data lain ada pada Tabel 4 yang awalnya mempunyai 8 keadaan dengan 1 masukan dan 1 keluaran menjadi 4 keadaan. Hasil reduksi tabel ini didapat dari keluaran aplikasi yang terlihat pada Gambar 12. Keadaan (a,c) ekuivalen jika (a,c) ekuivalen. Dengan demikian syarat (a,c) ekuivalen adalah dirinya sendiri. Oleh karena itu dapat dinyatakan bahwa (a,c) ekuivalen. Hal ini berlaku pula untuk keadaan (a,d), (c,d), dan (d,g). Keadaan (b,h) ekuivalen jika (a,g) dan (f,h) ekuivalen. Keadaan (a,g) ekuivalen karena (d,g) ekuivalen. Namun karena (f,h) tidak ekuivalen maka (b,h) tidak ekuivalen. Keadaan (e,f) ekuivalen karena (d,g) ekuivalen.

Tabel 4. Transisi keadaan dengan 8 keadaan sebelum dan sesudah direduksi

Present state	*Next state*		*Output*	
	x=0	*x=1*	*x=0*	*x=1*
a	*d*	*a*	*0*	*1*
b	*a*	*f*	*1*	*0*
c	*d*	*c*	*0*	*1*
d	*d*	*d*	*0*	*1*
e	*c*	*d*	*1*	*1*
f	*c*	*g*	*1*	*1*
g	*g*	*g*	*0*	*1*
h	*g*	*h*	*1*	*0*

Present state	*Next state*		*Output*	
a	*a*	*a*	*0*	*1*
b	*a*	*e*	*1*	*0*
e	*a*	*a*	*1*	*1*
h	*a*	*h*	*1*	*0*



Gambar 12. Keluaran dari untuk data dari Tabel 4 bagan awal dan akhir iterasi

Keadaan (a,c), (a,d), (a,g), (c,d), (c,g), dan (d,g) ekuivalen sehingga (a,c,d,g) ekuivalen. Keadaan (e,f) juga ekuivalen. Dengan demikian tinggal keadaan a, b, e, dan h.

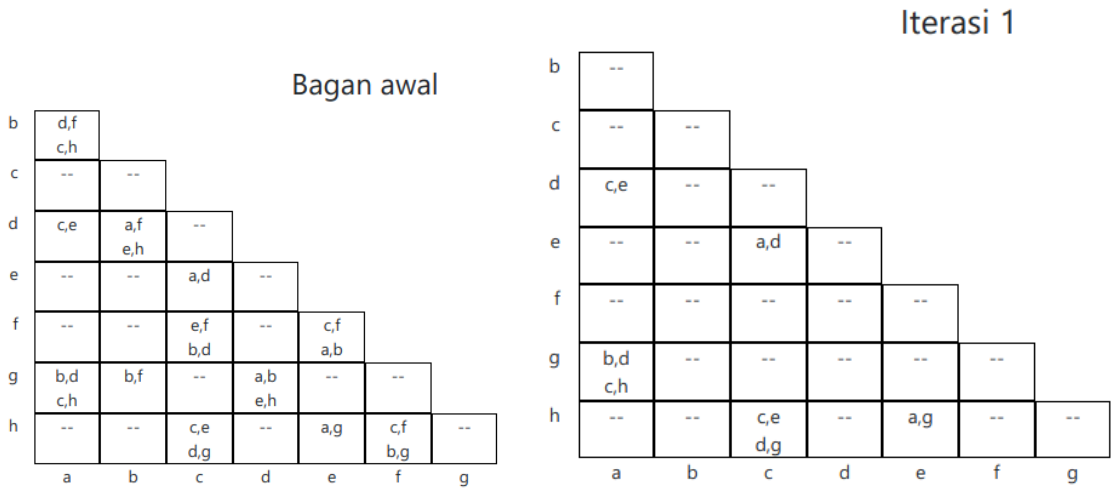
3. KESIMPULAN

Implementasi reduksi keadaan pada desain rangkaian digital sekuensial berhasil dilakukan dengan bahasa Python dan PHP. Uji coba dengan beberapa keadaan dengan 1 masukan dan 1 keluaran berhasil dengan baik. Selanjutnya dapat diuji dengan lebih dari 1 masukan dan lebih dari 1 keluaran.

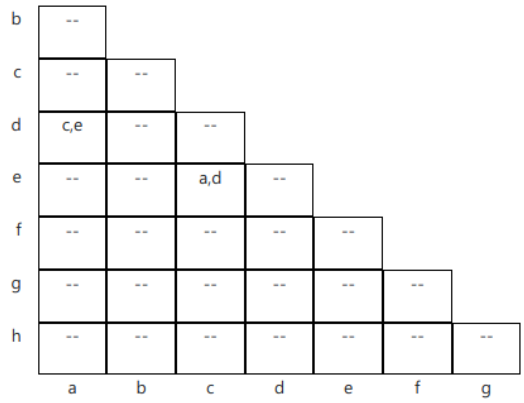
DAFTAR PUSTAKA

[1] M.M. Mano, 2002, Digital Design, 3rd Edition, Prentice Hall Inc.
[2] M. Murdoca, dan V. Heuring, 2002, *Principles of Computer Architecture*, Prentice Hall.
[3] Huffman, D. A., 1954, The synthesis of sequential switching circuits, *Journal of Franklin Inst.*, part I, vol. 257, no. 3, pp. 161–190.
[4] E. F. Moore, 1956, Gedanken experiments on sequential machines, in *Automata Studies*, C. E. Shannon and J. McCarthy, Eds. Princeton, NJ: Princeton Univ. Press.
[5] J.E. Hopcroft, 1971, *n log n algorithm for minimizing states in finite automata*, Stanford Univ., Satnford, CA, Tech. Rep. CS 71/190.
[6] J.K. Rho, G. Hachtel, F. Somenzi, and R. Jacoby, 1994, Exact and heuristic algorithms for the minimization of incom pletely specified state machines, *IEEE Trans. Computer-Aided Design*, vol. 13, pp. 167–177.
[7] T. Kam, T. Villa, R. Brayton, and A. Sangiovanni-Vincentelli, , A fully implicit algorithm for exact state minimization, in *Proc. ACM/IEEE Design Automation Conf.*, 1994, pp. 684–690.
[8] S.G. Shiva, 1998, *Introduction to Logic Design,* CRC Press
[9] A. B. Marcovitz, 2010, *Introduction to logic design 3rd ed.*, McGraw-Hill Higher Education, Boston, USA.

LAMPIRAN



Iterasi 2



Iterasi 3