Generator Set Uji untuk Diagnosis Kesalahan pada Rangkaian Digital Kombinasional *Demultiplexer 1 line to 4 line* IC 74ACT39 Menggunakan Metode Tabel Kesalahan

Denny Pratama[(1,a)], Ageng Sadnowo Repelianto[(1,b*)], Sri Purwiyanti[(1,c)]

[(1)]Magister Teknik Elektro, Universitas Lampung, Bandar Lampung, Indonesia, 35141
Email: [(a)]denpratama95@gmail.com, [(b)]ageng.sadnowo@eng.unila.ac.id (corresponding author), [(c)]sri.purwiyanti@eng.unila.ac.id*

Diterima (04 Maret 2025), Direvisi (20 April 2025)

Abstract. *A minor error that occurs in an IC manufacturing industry is a major problem if it goes undetected and the IC is used by the user. A method needs to be implemented to diagnose errors to ensure that the IC can function properly according to its purpose. The method used must be faster and more efficient in checking the IC. One of the methods that can be used for fault detection in an IC is the error table method. The error table method will map the possible faults that occur in an IC and then apply them to a test set generator with a diagnostic tree system. The design of a test set generator that operates based on the diagnostic tree principle using the error table method is one of the methods that can be used in fault detection. The implementation on a dual 1-line to 4-line demultiplexer 74ACT139 built in this research shows that this method is one of the effective approaches for fault diagnosis. As a result, all potential faults in a damaged IC under stuck logic conditions can be detected with an average efficiency of 93.01%, with the longest test level consisting of 16 test cycles achieving an efficiency of 87.5% at f_0 and f_{15}, while the shortest test consisted of 1 test cycle with an efficiency of 99.21% at f_1.*

Keywords: *Fault diagnosis, demultiplexer, efficiency, fault table, diagnostic tree.*

Abstrak. Kesalahan kecil yang terjadi dalam sebuah industri pembuatan IC merupakan masalah besar apabila lolos dan IC digunakan pengguna. Perlu adanya metode yang dilakukan untuk melakukan diagnosis kesalahan untuk memastikan bahwa IC dapat bekerja dengan baik sesuai dengan fungsinya. Metode yang digunakan harus lebih cepat dan lebih efisien dalam melakukan pengecekan IC. Salah satu metode yang dapat digunakan untuk melakukan pendeteksian kerusakan pada sebuah IC adalah metode tabel kesalahan. Metode tabel kesalahan akan memetakan kemungkinan kerusakan yang terjadi pada sebuah IC kemudian diaplikasikan pada sebuah generator set uji dengan sistem kerja pohon diagnosa. Perancangan generator set uji yang bekerja berdasarkan prinsip pohon diagnosa dengan metode tabel kesalahan merupakan salah satu metode yang dapat digunakan dalam mendeteksi kesalahan. Implementasi pada sistem *demultiplexer dual 1 line to 4 line* IC 74ACT139 yang dibangun pada penelitian ini menunjukkan bahwa metode ini salah satu yang dapat digunakan secara efektif untuk melakukan sistem diagnosis kesalahan. Hasilnya, seluruh potensi kerusakan pada IC yang rusak dalam kondisi *stuck* logika dapat dideteksi dengan rata-rata efisiensi sebesar 93,01% dengan pengujian level terpanjang sebanyak 16 kali pengujian dengan nilai efisiensi 87,5% pada f_0 dan f_{15} sedangkan pengujian terpendek sebanyak 1 kali pengujian dengan nilai efisiensi 99,21% pada f_1.

Kata kunci: maksimal 5 kata kunci, dipisahkan dengan tanda koma.

PENDAHULUAN

Industri *integrated circuit* (IC) melakukan pengujian berkala terhadap produknya menggunakan metode pengambilan sampel untuk memastikan kualitas produk. Pengujian dilakukan sesuai skala produksi dengan metode yang tepat untuk meningkatkan efisiensi dan menjaga kualitas tinggi [1]. Kondisi abnormal dapat

terjadi karena bencana alam, kecelakaan fisik, atau kegagalan peralatan [2]. Oleh karena itu, alat yang efektif sangat penting untuk memastikan kelancaran proses pengujian. Proses mendiagnosis gangguan pada IC melibatkan beberapa langkah, termasuk memeriksa *database* gangguan yang pernah terjadi, membandingkan sinyal *input* dan *output* pada rangkaian, dan menggunakan metode pendukung [3]. Salah satu metode yang dapat digunakan dalam proses pendeteksian kesalahan yaitu metode diagnosis kesalahan [4]. Diagnosis kesalahan bertujuan untuk mengidentifikasi ada atau tidaknya kesalahan, apabila ada kesalahan dapat diketahui di mana lokasi kesalahan berada [5]. Metode-metode ini dirancang dengan langkah-langkah sistematis untuk memudahkan penentuan lokasi kesalahan pada sebuah IC berada.

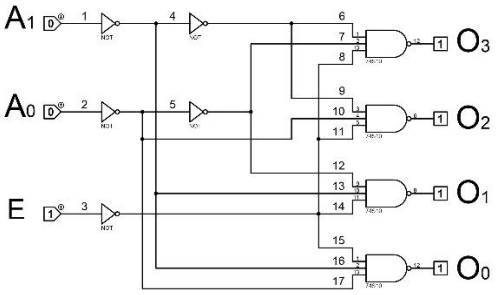
Diagnosis kesalahan adalah metode untuk mendeteksi kerusakan dengan efektif [6]. Pengujian dilakukan dengan cara minimal untuk menghemat biaya[7]. Penelitian ini menciptakan generator set uji untuk diagnosis kesalahan pada rangkaian digital kombinasional dengan *input* dan *output* jamak pada IC 74ACT139. Sebelumnya, penelitian telah berhasil dalam diagnosis rangkaian kombinasional dengan input jamak dan output tunggal[7][8]. Kebaharuan pada penelitian ini adalah proses pendeteksian kesalahan dilakukan pada *output* jamak. Oleh karena itu, diagnosis *input* dan *output* jamak yang diaplikasikan pada pohon diagnosa yang dibangun dapat mengurangi durasi dan biaya pengujian karena menggunakan *software* secara efisien dan mengurangi kebutuhan perangkat keras pengujian. Proses pendeteksian kesalahan pada *output* jamak memiliki sebuah prosedur baru yang digunakan sehingga menjadi kebaharuan pada penulisan karya ilmiah ini yang dilakukan dengan menggabungkan beberapa pohon diagnosa guna mempersingkat waktu pengujian.

METODE PENELITIAN

Mekanisme penelitian ini berbasis *Research and Development* (R&D) dan merupakan penelitian lanjutan dari grup kami sebelumnya pada rangkaian digital kombinasional [7][8]. Penelitian grup kami sebelumnya menggunakan rangkaian digital kombinasional dengan *output* tunggal, sehingga yang menjadi kebaharuan adalah *output* yang dipilih dalam mendeteksi kesalahan adalah *output* jamak dengan menggunakan IC 74ACT139. Perbedaan antara *output* tunggal dengan jamak adalah terdapat sebuah prosedur baru yaitu membuat irisan potongan *input* dan masing masing *output*. Pada IC 74ACT139 terdapat 4 *output* sehingga jumlah potongannya berjumlah 4. Masing-masing potongan akan menghasilkan sebuah pohon diagnosa yang akan digabung menjadi sebuah pohon diagnosa baru yaitu pohon diagnosa penggabungan potongan yang beririsan. Beririsan artinya jalur titik pengujian yang telah di lalui maka tidak digunakan kembali pada potongan selanjutnya. Artinya tidak ada titik pengujian yang masuk dalam 2 irisan.

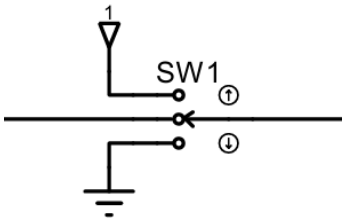
Sistem yang dibangun terdiri dari *hardware* dan *software,* adapun skema perancangan *hardware* pada penelitian ini terdiri atas PC *core i3* sebagai pengolahan data, unit mikrokontroller *Arduino MEGA* sebagai pusat perintah, *Project Board*, IC 74ACT139, IC 7473, IC 7404, IC 7410, kabel jumper, pemindah posisi dan multimeter digital. Berikut pada **Gambar 1** dan **Gambar 2** disajikan tampilan hardware dan potongan 1 *wiring diagram* pada IC 74ACT139.

Gambar 1. Hardware IC 74ACT139.



Gambar 2. Wiring diagram potongan 1 IC 74ACT139

Gambar 2 menunjukkan potongan 1 dari IC yang digunakan. Realisasinya, di dalam IC terdapat 2 rangkaian serupa sehingga dapat dilakukan pada salah satunya saja dan cukup mewakili keadaan sesungguhnya. Wiring diagram diatas juga menunjukkan rangkaian pengujian sistem dengan 17 titik pengujian pada *ic dual demultiplexer 1 to 4* dengan 3 *input* dan 4 *output* IC 74ACT139. Rangkaian dilakukan pengujian dengan mengkondisikan pada keadaan *stuck logika* berupa *stuck* @0 dan *stuck* @1 [7][8]. Kondisi *stuck* logika merupakan sebuah kondisi digital yang mungkin terjadi. Kondisi *stuck* @0 menyatakan bahwa kondisi *output* selalu dalam kondisi *low* atau 0, jika kondisi *stuck* @1 maka *output* akan selalu bernilai *high* atau 1. Masing-masing titik pengujian pada jalur pengujian pada **Gambar 2** di ilustrasikan keadaannya pada sebuah rangkaian *switch* pada **Gambar 3**. *Switch* merupakan pengkodisian kondisi *stuck* logika yang dimasukkan kedalam sistem untuk melakukan pengecekan kondisi.



Gambar 3. Rangkaian pengujian *stuck* logika

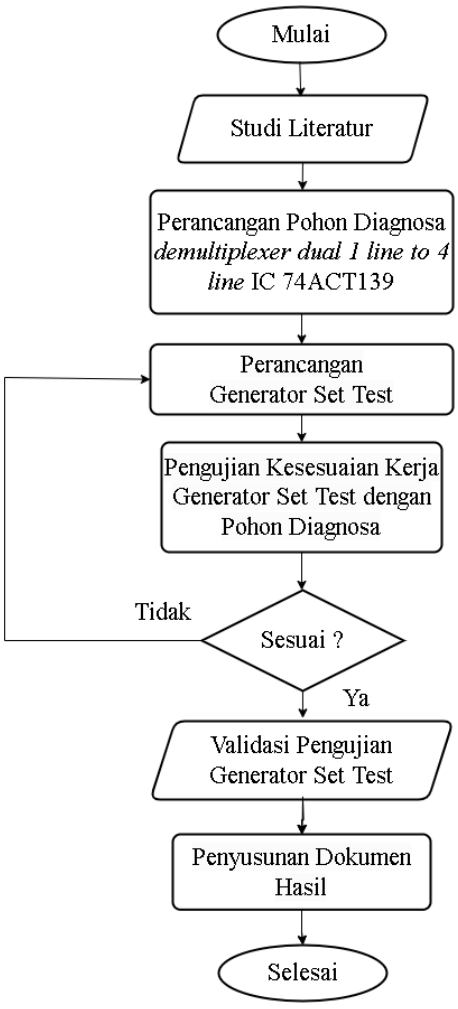
Kondisi *stuck* logika akan diatur untuk melakukan pemetaan dan pengujian sistem. Pada penelitian ini kondisi *stuck* logika diatur pada kesalahan tunggal. Artinya, setiap kondisi *stuck* hanya dilakukan pada salah satu switch saja yang dilakukan secara bergantian sampai seluruh *stuck* logika terwakili.

Rangkaian digital penyucun IC terdiri dari 18 gerbang logika diantaranya NOT dan NAND yang digunakan yang membuat satu kesatuan fungsionalitas IC 74ACT139. Sebagai acuan dasar dari sistem logika adalah sebuah tabel kebenaran. Tabel kebenaran merupakan acuan dasar dalam melakukan perubahan kondisi. Tabel kebenaran berasal dari *output* natural dari *input* yang diberikan. Adapun *input* pada IC 74ACT139 adalah A_1, A_0 dan E sedangkan *output* terdiri dari O_3, O_2, O_1, O_0. Keterangan pada tabel akan di representasikan pada H adalah *high* atau bernilai 1, L adalah *low* atau bernilai digital 0 dan X adalah *immertial* atau diabaikan. Diabaikan merupakan kondisi high atau low tidak akan mempengaruhi output sistem. Berikut pada **Tabel 1** ditampilkan tabel kebenaran IC 74ACT139.

Tabel 1. Tabel Kebenaran IC 74ACT139

Input			*Output*			
E	A_0	A_1	O_0	O_1	O_2	O_3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	H	L	H	L	H	H
L	L	H	H	H	L	H
L	H	H	H	H	H	L

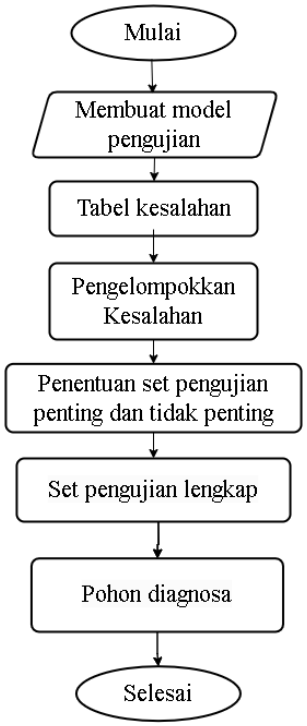
Tahapan penyusunan sebuah sistem pengujian yang ideal, diperlukan beberapa tahapan yang sistematis, diawali dengan studi literatur, kemudian melakukan perancangan pohon diagnosa pengujian, selanjutnya melakukan proses implementasi generator set tes. Prosedur pengujian generator set merupakan tahapan pengujian yang dilakukan untuk menemukan lokasi kesalahan berada dan memastikan sistem dapat bekerja optimal. Setelah sistem dapat melakukan diagnosis kesalahan maka hasil dapat divalidasi kemudian dilakukan penyusunan hasil pengujian penelitian. Berikut disajikan pada **Gambar 4** merupakan diagram alir penelitian yang dilakukan.



Gambar 4. Diagram alir penelitian

Tahapan impelementasi generator set uji yang dilakukan sistem meliputi perancangan pohon diagnosa. Dimana perancangan pohon diagnosa meliputi beberapa tahapan yang dilakukan. Pertama dilakukan proses pembuatan model pengujian. Tahapan ini melakukan proses pemisahan menjadi 4 bagian diagram yang beririsan. Pemisahan beririsan bertujuan supaya tidak ada *state* kesalahan ganda yang dilakukan sehingga kesalahan dapat dipetakan secara optimal.

Selanjutnya masing-masing potongan dilakukan pembuatan tabel kesalahan untuk memetakan kesalahan dengan melakukan *stuck* logika, melakukan pengelompokkan pada nilai *output* yang sama dari hasil pemetaan kesalahan, melakukan penentuan set pengujian penting dan tidak penting pada penelitian, menyusun set pengujian lengkap guna mengurutkan pengujian dan proses diakhiri dengan menyusun pohon diagnosa berdasarkan hasil dari set pengujian lengkap. Berikut pada **Gambar 5** disajikan prosedur pembuatan pohon diagnosa pengujian.



Gambar 5. Diagram alir pembuatan pohon diagnosa

Berdasarkan prosedur penelitian yang disajikan pada **Gambar 4** dan **Gambar 5** maka terdapat beberapa tahapan penyusunan yang harus dilakukan pada penelitian. Adapun prosedur yang disajikan merupakan sebuah alat bantu yang dapat digunakan sebagai acuan dalam proses penyusunan hasil dan pembahasan dari penelitian yang akan dilakukan.

HASIL DAN PEMBAHASAN

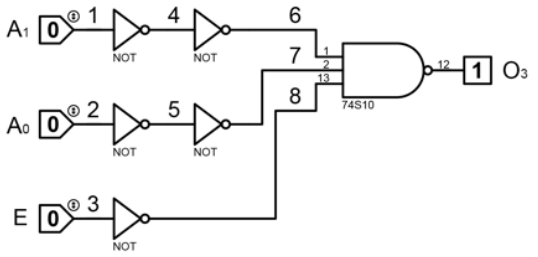
Tahapan Pelaksanaan Penelitian

Tahapan yang dilakukan dalam proses ini adlah implementasi dan pembahasan mengenai analisis melalui beberapa prosedur di antaranya melakukan pemecahan rangkaian pada **Gambar 2** menjadi empat bagian dengan *output* yang berbeda, melakukan proses pembuatan pohon diagnosa pada potongan 1 dengan hasil *output* O_3, potongan 2 dengan hasil *output* O_2, potongan 3 dengan hasil *output* O_1 dan potongan 4 dengan hasil *output* O_0. Masing-masing potongan dipastikan memiliki irisan sehingga tidak akan memiliki pengujian ganda pada setiap titiknya. Masing-masing potongan akan dilakukan proses perancangan pohon diagnosa seperti pembuatan tabel kesalahan, penentuan kelompok kesalahan, penentuan set uji dan pembuatan pohon diagnosa. Penentuan irisan pada pohon diagnosa ditujukan supaya tidak ada *state* kesalahan ganda dari masing-masing lokasi kesalahan. Pada saat melakukan penentuan lokasi kesalahan, dipastikan seluruh lokasi kesalahan masuk kedalam cakupan kesalahan. Teridentifikasi seluruh lokasi kesalahan akan memudahkan dalam proses diagnosis lokasi kesalahan. **Gambar 2** menunjukkan potongan lengkap dari *demultiplexer/ decoder dual 1 line to 4 line* IC 74ACT139. Pada tahap ini akan dipecah menjadi 4 bagian.

1. Pohon Diagnosa Potongan Pertama

Pohon diagnosa disusun berdasarkan rangkaian gerbang logika digital NOT dan NAND. Gerbang logika potongan pertama disusun hingga menghasilkan sebuah lokasi kesalahan yang berjumlah delapan titik. Lokasi kesalahan yang sudah masuk cakupan potongan pertama tidak perlu lagi masuk dalam potongan selanjutnya. Gambar 4 merupakan potongan satu pengujian dengan jumlah titik pengujian 8 titik *stuck@0* dan *stuck@1*

Tahapan awal pembuatan pohon diagnosa pengujian dilakukan dengan membuat tabel kesalahan. Tabel kesalahan merupakan tabel yang merepresentasikan seluruh kendala *stuck* logika yang terjadi pada setiap jalur yang disajikan pada **Tabel 2**.



Gambar 6. Diagram pengujian potongan 1

Tabel 2. Tabel kesalahan IC 74ACT139 potongan 1

A_1	A_0	E	Y_0	Y_{10}	Y_{11}	Y_{20}	Y_{21}	Y_{30}	Y_{31}	Y_{40}	Y_{41}	Y_{50}	Y_{51}	Y_{60}	Y_{61}	Y_{70}	Y_{71}	Y_{80}	Y_{81}
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	0	1	0	0	1	0	1	0	1	1	0	1	0	1	0
1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	0

Berdasarkan hasil diatas, dikelompokkan apabila terdapat *output* pengujian dengan hasil yang *similar*. Berdasarkan hasil pengelompokkan, maka hasilnya tabel disajikan seperti pada **Tabel 3**. Dilanjutkan dengan melakukan proses penentuan set uji. Penentuan set uji dilakukan dengan menguji masing masing kelompok kesalahan dengan cara melakukan operasi XOR terhadap seluruh element pada masing-masing baris. Terdapat pengujian 2, 4, 6, 7. Baris tersebut dipilih karena selain baris tersebut bernilai 0 untuk keseluruhan elemennya sehingga tidak perlu dilakukan pengujian untuk penentuan set uji. Hasil penentuan set uji akan disajikan pada **Tabel 4** secara berturut berikut

Tabel 3. Kelompok kesalahan IC 74ACT139 potongan 1

A_1	A_0	E	f_0	f_1	f_2	f_3	f_4	
			Y_0	Y_{61}	Y_{81}	Y_{21}	Y_{41}	Y_{51}
							Y_{31}	Y_{60}
				Y_{40}		Y_{71}	Y_{20}	Y_{70}
				Y_{11}	Y_{30}	Y_{50}	Y_{10}	Y_{80}
0	0	0	1	1	1	1	1	
0	0	1	1	1	1	1	1	
0	1	0	1	0	1	1	1	
0	1	1	1	1	1	1	1	
1	0	0	1	1	1	0	1	
1	0	1	1	1	1	1	1	
1	1	0	0	0	0	0	1	
1	1	1	1	1	0	1	1	

Tabel 4. Penentuan set uji potongan 1

Uji	0	0	0	0	1	1	1	2	2	3
	1	2	3	4	2	3	4	3	4	4
2	1	0	0	0	1	1	1	0	0	0
4	0	0	1	0	0	1	0	1	0	1
6	0	0	0	1	0	0	1	0	1	1
7	0	1	0	0	1	0	0	1	1	0

Eliminasi baris dan kolom dominan dilakukan hingga mendapatkan pengujian keseluruhan dengan identifikasi warna hijau. Tahapan dilakukan hingga setiap perbandingan masuk dalam cakupan set uji.

Uji	0	0	0	0	1	1	1	2	2	3
	1	2	3	4	2	3	4	3	4	4
2	1				1	1	1			
4			1			1		1		1
6				1			1		1	1
7		1			1			1	1	

Uji	0	0	0	2	2	3
	2	3	4	3	4	4
4		1		1		1
6			1		1	1
7	1			1	1	

Gambar 7. Menentukan set uji pohon diagnosa pengujian 2 dan 6

Uji	0	0	2
	2	3	3
4		1	1
7	1		1

Uji	0
	3
4	1

Gambar 8. Menentukan set uji pohon diagnosa pengujian 7 dan 4

Gambar 7 bagian atas menunjukkan proses eliminasi baris pengujian 6 yang berasal dari **Tabel 4**. Proses dilakukan dengan melakukan eliminasi terhadap baris yang memiliki elemen 1. Kemudian dilanjutkan pada **Gambar 7** bawah merupakan gambar hasil eliminasi yang dilanjutkan kembali proses eliminasinya pada level 2 yaitu pengujian 6. Proses dilanjut melakukan eliminasi pengujian 7 dan pengujian 4 seperti pada **Gambar 8**.

Setelah pengujian lengkap, langkah terakhir dari setiap potongan adalah membuat pohon diagnosa pengujian. Pohon

diagnosa pengujian merupakan percabangan pengujian yang akan merujuk pada sebuah kelompok kesalahan. Kelompok kesalahan merupakan respresentasi dari diagnosis kesalahan. Ri merupakan mutlak selisih antara 0 dan 1 dalam setiap baris. $f_{(1)}$ adalah jumlah 1 dalam suatu baris, $f_{(0)}$ adalah jumlah 0 dalam suatu baris. Seperti dijelaskan pada persamaan 1.

$$R_i = |f_{(1)} - f_{(0)}| \quad (1)$$

Nilai R_i yang terkecil dipilih sebagai pengujian yang pertama terlebih dahulu dari setiap hasil eliminasi. Karena R_i sama, maka kita dapat menentukan pengujian mana terlebih dahulu yang dilakukan, dipilih kemudian pengujian 2 dengan objek eliminasi yaitu kelompok kesalahan. **Gambar 9** merupakan tabel pengujian level 1 dalam setiap pengujian akan menghasilkan 1 atau 0 yang akan merujuk pada sebuah lokasi kesalahan. Kelompok nilai 0 merupakan kesalahan f_1 dan kelompok nilai 1 merupakan kelompok kesalahan f_0, f_2, f_3 dan f_4 yang merujuk pada pengujian selanjutnya di level 2. Pada level 2 disajikan pada **Gambar 10** dimana *output* nilai 0 merujuk pada sebuah lokasi kesalahan f_3 dan *output* nilai 1 merujuk pada f_0, f_2 dan f_4. **Gambar 11** merupakan pengujian level 3 dimana *output* nilai 1 merupakan lokasi kesalahan yaitu f_4 dan *output* bernilai 0 akan merujuk pengujian selanjutnya dengan lokasi kesalahan f_0 atau f_2. Gambar 12 merupkan pengujian akhir dimana pada pengujian tersebut diberikan nilai 7 dimana ketika *output* bernilai 1 maka merujuk pada f_0 dan apabila bernilai 0 maka merujuk pada lokasi kesalahan f_2. f_0 akan selalu berada pada level akhir pengujian dikarekanakan f_0 adalah kondisi tidak ada kesalahan

f_0	f_1	f_2	f_3	f_4	Uji	R_i
1	0	1	1	1	2	3
1	1	1	0	1	4	3
0	0	0	0	1	6	3
1	1	0	1	1	7	3

Gambar 9. Menentukan urutan pengujian level 1

f_0	f_2	f_3	f_4	Uji	Ri
1	1	0	1	4	2
0	0	0	1	6	2
1	0	1	1	7	2

Gambar 10. Menentukan urutan pengujian level 2

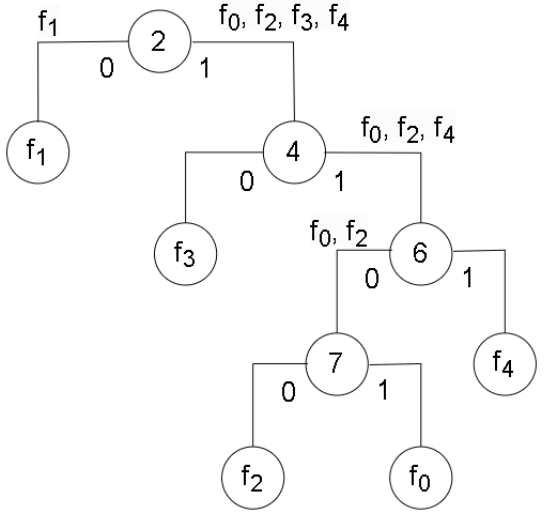
f_0	f_2	f_4	Uji	R_i
0	0	1	6	1
1	0	1	7	1

Gambar 11. Menentukan urutan pengujian level 3

f_0	f_2
1	0

Uji
7

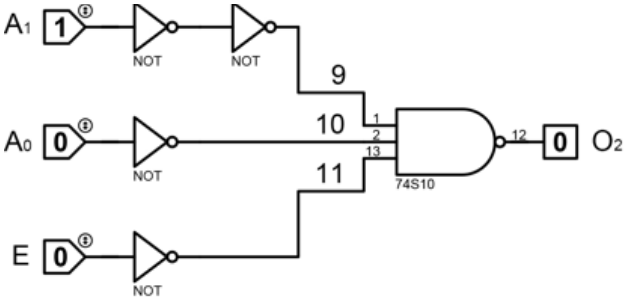
Gambar 12. Menentukan urutan pengujian level 4



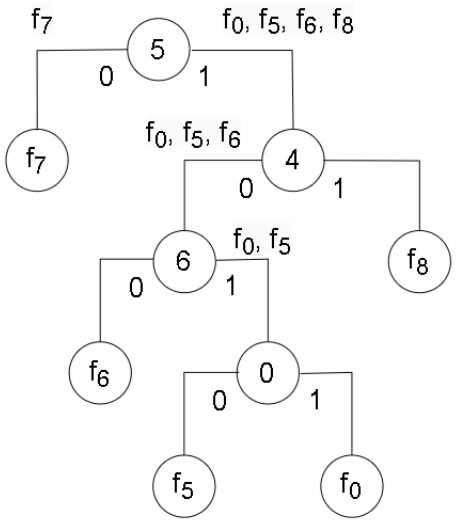
Gambar 13. Pohon diagnosa potongan 1

Gambar 13 disajikan pohon diagnosa hasil pengujian potongan 1 dengan 4 level pengujian.

2. Pohon Diagnosa Potongan Kedua



Gambar 14. Diagram pengujian potongan 2

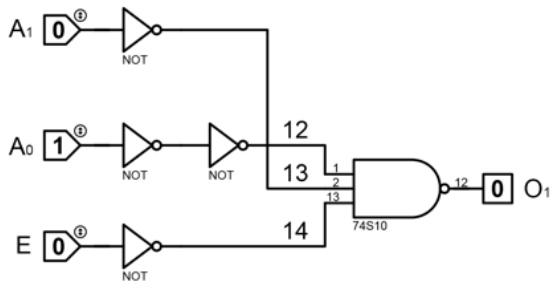


Gambar 15. Pohon diagnosa potongan 2

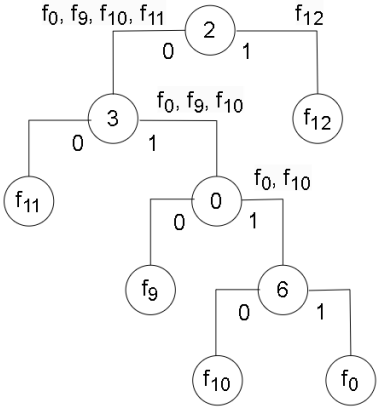
Pohon diagnosa potongan 2 diambil pada irisan rangkaian uji yang belum terindentifikasi pada potongan 1 namun dalam cakupan *output* 2. Pada tahapan ini potongan mencakup pengujian jalur 9, 10 dan 11. **Gambar 14** menunjukkan cakupan pada *output* 2 atau disebut O_2. Hasil pohon diagnosa pada rangkaian seperti disajikan pada **Gambar 15.**

3. Pohon Diagnosa Potongan Ketiga

Pohon diagnosa potongan 3 merupakan irisan rangkaian uji yang belum terindentifikasi pada potongan 1 dan potongan 2 pada cakupan *output* 1. Pada tahapan ini potongan mencakup titik pengujian jalur 12, 13 dan 14. **Gambar 16** menunjukkan cakupan pada *output* 3 atau disebut O_1. Hasil pohon diagnosa pada rangkaian seperti pada **Gambar 17**.

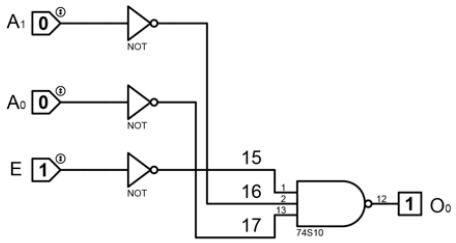


Gambar 16. Diagram pengujian potongan 3

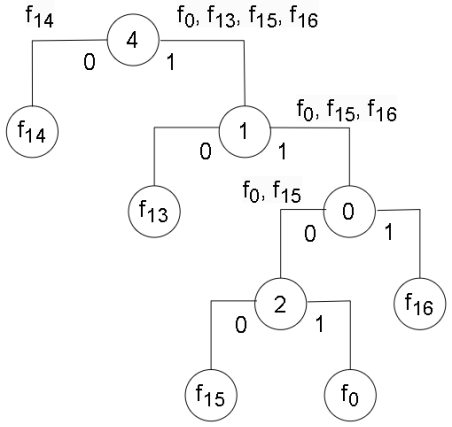


Gambar 17. Pohon diagnosa potongan 3

4. Pohon Diagnosa Potongan Keempat



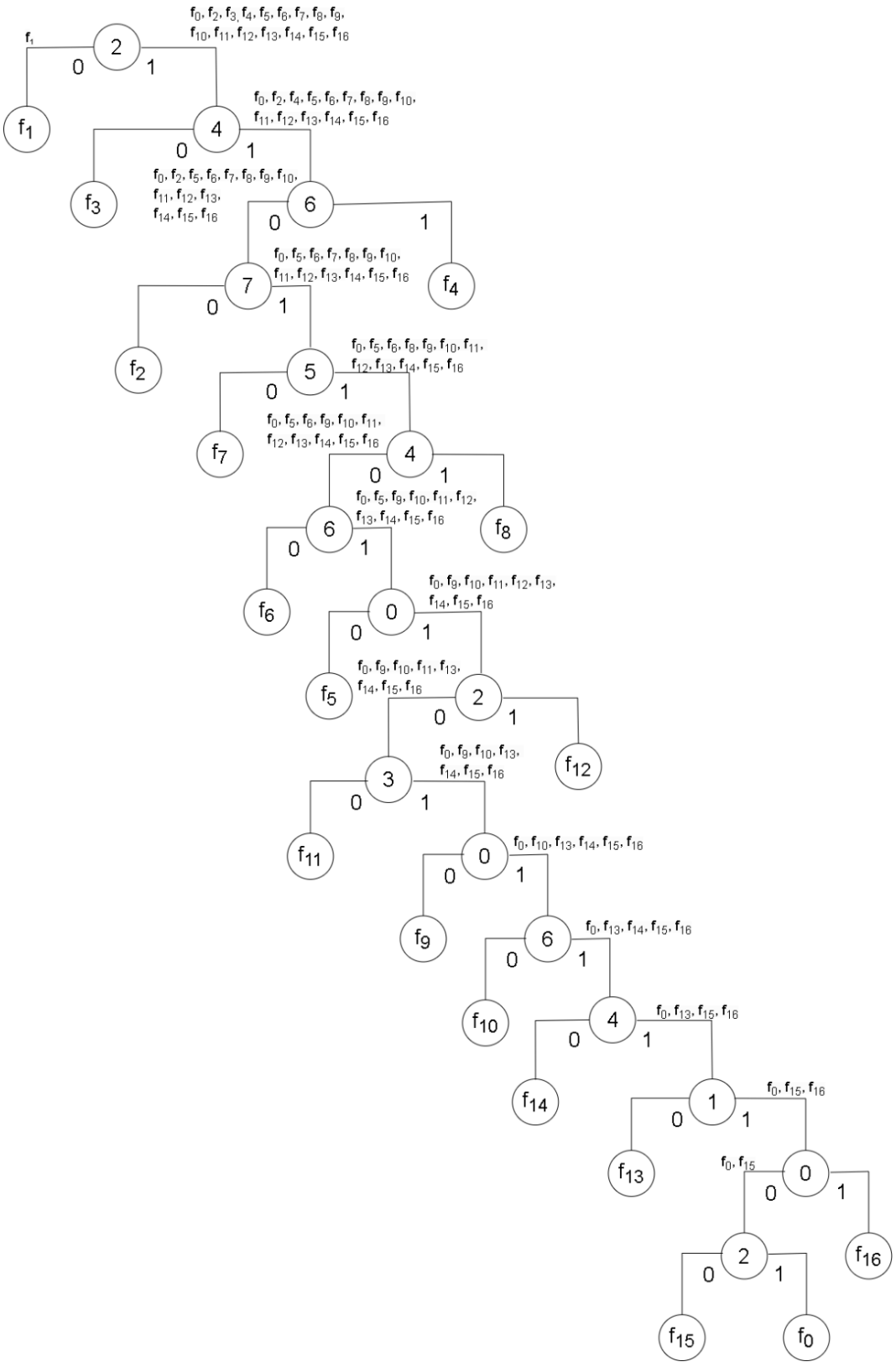
Gambar 18. Diagram pengujian potongan 4



Gambar 19. Pohon diagnosa potongan 4

Pohon diagnosa potongan 4 merupakan irisan rangkaian uji yang belum terindentifikasi pada potongan 1, potongan 2 dan potongan 4 pada cakupan *output* 4. Pada tahapan ini potongan mencakup pengujian jalur 15, 16 dan 17. **Gambar 18** menunjukkan cakupan pada *output* 4 atau disebut O_0. Hasil pohon diagnosa pada rangkaian seperti pada **Gambar 19.**

5. Penggabungan Pohon Diagnosa

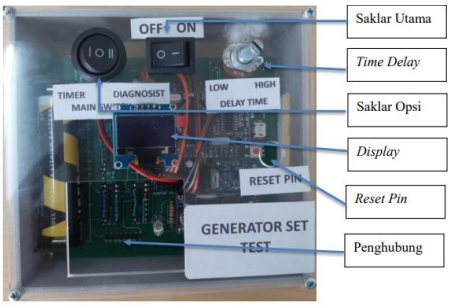


Gambar 20. Gabungan pohon diagnosa

Berdasarkan pengujian potongan, selanjutnya dilakukan penggabungan pohon diagnosa dari keseluruhan potongan. Dimulai dari potongan pertama, keadaan f_0 pada potongan pertama akan lanjut pada pohon diagnosa pada potongan kedua. Keadaan f_0 pada potongan kedua akan dilanjutkan pada pohon diagnosa potongan ketiga hingga pada f_0 pada potongan ketiga akan dilanjut pada pohon diagnosa potongan keempat. Berdasarkan skenario berikut, maka seluruh kesalahan dapat masuk dalam cakupan diagnosis kesalahan. Sehingga seluruh kemungkinan kerusakan pada IC 74ACT139 dapat terpetakan dan dapat di definisikan dengan baik. Terpetakan dengan baik diartikan bahwa proses diagnosis kesalahan dapat berjalan benar dengan mengetahui seluruh kemungkinan apabila terjadi kesalahan.

Analisis Hasil

Proses perancangan implementasi dilakukan dengan menggunakan bantuan mikrokontroller arduino MEGA sebagai device utama. Digunakan bantuan rangkaian yang serupa dengan IC yang disusun menggunakan gerbang logika dan beberapa saklar untuk membentuk potongan de*multiplexer dual 1 line to 4 line* 74ACT139. Berdasarkan titik uji terdapat 15 lokasi kemungkinan kesalahan dengan kondisi *stuck* logika. Diharapkan sistem yang dibuat dapat menjangkau seluruh lokasi kesalahan sama seperti skenario yang dibuat pada pohon diagnosa.

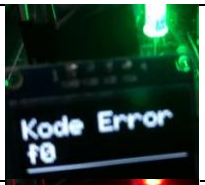
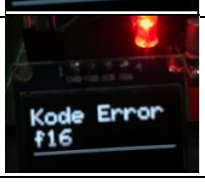


Gambar 21. Generator set uji

Gambar 21 merupakan geneator set uji yang digunakan untuk melakukan pengujian pada IC 74ACT139. Penggunaan metode pohon diagnosa diharapkan dapat mempersingkat waktu pengujian sehingga waktu pengujian dapat diminimalisir. Jika menggunakan manual pengujian dijalankan sebanyak 128 kali dikarenakan memiliki *input* 3 dan *output* 4. 128 merupakan perkalian antara jumlah kemungkinan *input* dan *output*. Berikut merupakan hasil dari pengujian lokasi kesalahan dari rangkaian digital kombinasional IC demultiplexer dengan minimal level penngujian 1 dan maksimal level pengujian 16. Berikut merupakan pengujian yang dilakukan pada circuit under test yang susunannya disesuaikan dengan IC 74ACT139.

Tabel 5 merupakan pengkondisian atas lokasi kesalahan yang muncul pada tampilan *output* pada LCD generator set uji. Hasil menunjukkan kesesuaian antara pohon diagnosa yang dibangun dan kondisi yang muncul setelah diimplementasikan pada generator set uji berbasis arduino MEGA. **Tabel 6** akan menampilkan hasil pengujian keseluruhan dalam setiap level pengujian.

Tabel 5. Pengujian pada titik lokasi kesalahan

No	Lokasi Kesalahan	Level Pengujian	Tampilan
1	Tidak ada kesalahan	16	
2	Jalur 13 S@1	12	
3	Jalur 17 S@0 , atau Jalur 16 S@0 , atau Jalur 15 S@0	15	

Tabel 6. Prosedur Pengujian Generator Set Uji

Lokasi Kesalahan	Pengujian	Level Pengujian																Keterangan
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
f_1	PD	2																kode eror f_1
Jalur 1 S@1 Jalur 6 S@1 Jalur 4 S@0	U	0																
f_3	PD	2	4															kode eror f_3
Jalur 2 S@1 Jalur 7 S@1 Jalur 5 S@0	U	1	0															
f_4	PD	2	4	6														kode eror f_4
Jalur 4 S@1 Jalur 3 S@1 Jalur 2 S@0 Jalur 1 S@0 Jalur 5 S@1 Jalur 6 S@0 Jalur 7 S@0 Jalur 8 S@0	U	1	1	1														
f_2	PD	2	4	6	7													kode eror f_2
Jalur 8 S@1 Jalur 3 S@0	U	1	1	0	0													
f_7	PD	2	4	6	7	5												kode eror f_7
Jalur 11 S@1	U	1	1	0	1	0												
f_8	PD	2	4	6	7	5	4											kode eror f_8
Jalur 10 S@0 Jalur 11 S@0	U	1	1	0	1	1	1											
f_6	PD	2	4	6	7	5	4	6										kode eror f_6
Jalur 10 S@1	U	1	1	0	1	1	0	0										
f_5	PD	2	4	6	7	5	4	6	0									kode eror f_5
Jalur 9 S@1	U	1	1	0	1	1	0	1	0									
f_{12}	PD	2	4	6	7	5	4	6	0	2								kode eror f_{12}
Jalur 14 S@0 Jalur 13 S@0 Jalur 12 S@0	U	1	1	0	1	1	0	1	1	1								
f_{11}	PD	2	4	6	7	5	4	6	0	2	3							kode eror f_{11}
Jalur 14 S@1	U	1	1	0	1	1	0	1	1	0	0							
f_9	PD	2	4	6	7	5	4	6	0	2	3	0						kode eror f_9
Jalur 12 S@1	U	1	1	0	1	1	0	1	1	0	1	0						
f_{10}	PD	2	4	6	7	5	4	6	0	2	3	0	6					kode eror f_{10}
Jalur 13 S@1	U	1	1	0	1	1	0	1	1	0	1	1	0					
f_{14}	PD	2	4	6	7	5	4	6	0	2	3	0	6	4				kode eror f_{14}
Jalur 16 S@1	U	1	1	0	1	1	0	1	1	0	1	1	1	0				
f_{13}	PD	2	4	6	7	5	4	6	0	2	3	0	6	4	1			kode eror f_{13}
Jalur 15 S@1	U	1	1	0	1	1	0	1	1	0	1	1	1	1	0			
f_{16}	PD	2	4	6	7	5	4	6	0	2	3	0	6	4	1	0		kode eror f_{16}
Jalur 17 S@0 Jalur 16 S@0 Jalur 15 S@0	U	1	1	0	1	1	0	1	1	0	1	1	1	1	1	1		
f_{15}	PD	2	4	6	7	5	4	6	0	2	3	0	6	4	1	0	2	kode eror f_{15}
Jalur 17 S@1	U	1	1	0	1	1	0	1	1	0	1	1	1	1	1	0	0	
f_0	PD	2	4	6	7	5	4	6	0	2	3	0	6	4	1	0	2	kode eror f_0
Tidak ada kesalahan	U	1	1	0	1	1	0	1	1	0	1	1	1	1	1	0	1	

Tabel 7. Persentase efisiensi pengujian

Lokasi Kesalahan	Level Pengujian	Persentase Efisiensi (%)
f_1	1	99,22
f_3	2	98,44
f_4	3	97,66
f_2	4	96,88
f_7	5	96,09
f_8	6	95,31
f_6	7	94,53
f_5	8	93,75
f_{12}	9	92,97
f_{11}	10	92,19
f_9	11	91,41
f_{10}	12	90,63
f_{14}	13	89,84
f_{13}	14	89,06
f_{16}	15	88,28
f_{15}	16	87,50
f_0	16	87,50

Tabel 6 telah menunjukkan dimana lokasi kesalahan berada secara utuh dan mencakup seluruh lokasi kesalahan. Terlihat, untuk mencapai maksimal pengujian dibutuhkan 16 level pengujian untuk mengecek apakah ada atau tidaknya kesalahan pada sistem IC yang dibangun. Minimal pengujian adalah 1 level pada kerusakan yang terjadi pada f_1 yaitu jalur 1 mengalami stuck logika 1 atau jalur 6 mengalami stuck logika 1 atau jalur 4 mengalami stuck logika 1 dan pengujian terpanjang yaitu level 15 pada f_{15} dan f_0. Lokasi kesalahan f_{15} berada pada jalur 17 mengalami stuck logika 1 dan f_0 merupakan tanpa kesalahan. Keberhasilan pohon diagnosa salah satunya dapat dicapai apabila f_0 menjadi pengujian akhir dikarenakan tanpa kesalahan harus terdeteksi setelah semua lokasi kesalahan terpetakan dengan baik. Hasil dapat ditunjukkan total penghematan yang terjadi rata-rata adalah 93,01 % dengan maksimal sebesar 99,21 % kerusakan terjadi pada f_1 dan minimal 87,5% pada f_0 dan f_{15}. Berikut adalah persentase penghematan dari masing masing pengujian yang disajikan pada **Tabel 7**. Penggunaan metode pohon diagnosa dalam melakukan diagnosis kesalahan yang diimplementasi menggunakan arduino mega sebagai mikrokontroller terbukti menghemat waktu dalam proses pendeteksian kesalahan secara optimal.

KESIMPULAN

Generator set uji yang dibangun untuk mendeteksi kesalahan kondisi stuck logika pada *demultiplexer dual 1 line to 4 line* 74ACT139 berdasarkan pohon diagnosa yang mengacu pada tabel kesalahan telah berhasil dibuat. Penggunaan metode ini menghasilkan efisiensi waktu pengujian dengan rata-rata adalah 93,01 %. Pendeteksian kesalahan ter pendek yaitu pada f_1 yaitu sebanyak 1 level pengujian yang artinya hanya satu pengujian dapat terdeteksi kerusakan pada f_1 dengan efisiensi 99,21%. Kemudian pengujian terpanjang pada f_0 dan f_{15} sebanyak 16 level pengujian dengan efisiensi sebesar 85,5%.

DAFTAR PUSTAKA

[1] S. Heo and J. H. Lee, "*Fault Detection and Classification using Artificial Neural Networks*" *Int. Symp. Adv. Control Chem. Process.*, vol. 1, pp. 464–469, 2018.

[2] A. Yadav and Y. Dash, "*An Overview of Transmission Line Protection by Artificial Neural Network: Fault Detection, Fault Classification, Fault Location, and Fault Direction Discrimination*" *Advances in Artificial Neural Systems*, vol. 2014, pp. 1–20, 2014.

[3] R. Isermann, "*Model-Based Fault-Detection and Diagnosis - Status and Applications*" *Annu Rev Control*, vol. 29, no. 1, pp. 71–85, 2005.

[4] D. Miljković, "*Fault Detection Methods: A Literature Survey*" *MIPRO 2011/CTS*, pp. 110–115, 2011.

[5] G. R. Yang and X. J. Wang, "*Artificial Neural Networks for Neuroscientists: A Primer*" *Neuron*, vol. 107, no. 6, pp. 1048–1070, 2020.

[6] R. Liu, B. Yang, E. Zio, and X. Chen, "*Artificial Intelligence for Fault Diagnosis of Rotating Machinery: A Review*" *Mechanical Systems and Signal Processing*, vol. 108. Academic Press, pp. 33–47, 2018.

[7] R. A. Setiawan, "Rancang Bangun Generator Set Uji untuk Diagnosis Kesalahan Rangkaian *Multiplexer Quad 2 Line To 1 Line IC 74157* Menggunakan Metode Tabel Kesalahan Berbasis Arduino Nano Atmega328", Magister Teknik thesis, Fakultas Teknik, Universitas Lampung , Bandar Lampung 2023.

[8] A.S. Repelianto, "Penggunaan Metoda Tabel Kesalahan untuk Diagnosis Kesalahan Rangkaian Digital Kombinasional", Teknik Elektro, Universitas Sriwijaya, Palembang, 1996.

[9] Sm Thamarai, K. Kuppusamy, and T. Meyyappan, "*Fault Detection and Test Minimization Methods for Combinational Circuits-A Survey*," *Int. J. Comput. Trends Technol.*, vol. 2, no. 2, pp. 140–146, 2011.

[10] A. Smith, A. Veneris, M. Fahim Ali, and A. Viglas, "*Fault diagnosis and Logic Debugging using Boolean Satisfiability,*" *IEEE Trans. Comput. Des. Integr. Circuits Syst.*, vol. 24, no. 10, pp. 1606–1620, Oct. 2005.