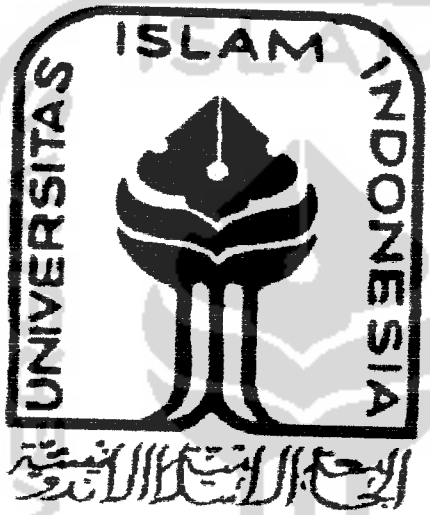
APLIKASI GERBANG LOGIKA PADA RANGKAIAN ELEKTRONIKA

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana
Jurusan Teknik Informatika



Oleh :

Nama : **Abdi**
No. Mahasiswa : **02 523 231**

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2007

LEMBAR PENGESAHAN PEMBIMBING

APLIKASI GERBANG LOGIKA PADA RANGKAIAN ELEKTRONIKA

TUGAS AKHIR

Oleh :

Nama : **Abdi**

NIM : **02 523 231**

Yogyakarta, Desember 2007

Pembimbing,

Drs. Supriyono M.Sc

LEMBAR PERNYATAAN KEASLIAN
HASIL TUGAS AKHIR

Saya yang bertandatangan di bawah ini,

Nama : Abdi

No. Mahasiswa : 02 523 231

Menyatakan bahwa seluruh komponen dan isi dalam Laporan Tugas Akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti bahwa ada beberapa bagian dari karya ini adalah bukan hasil karya saya sendiri, maka saya siap menanggung resiko dan konsekuensi apapun.

Demikian pernyataan ini saya buat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 11 Desember 2007

Abdi

LEMBAR PENGESAHAN PENGUJI

APLIKASI GERBANG LOGIKA PADA RANGKAIAN ELEKTRONIKA

TUGAS AKHIR

Oleh :

Nama : ABDI

NIM : 02 523 231

Telah Dipertahankan di Depan Sidang Penguji Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Jurusan Teknik Informatika
Fakulatas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, Desember 2007

Tim Penguji
Drs. Supriyono M.sc
Ketua

Yudi Prayudi, S.Si.,M.Kom
Anggota

Ami Fauziyah, ST.,MT
Anggota

Mengetahui,
Ketua Jurusan Teknik Informatika
Universitas Islam Indonesia

Yudi Prayudi, S.Si., M.Kom

HALAMAN PERSEMBAHAN

ripsi ini penulis persembahkan kepada :

- Ayahanda Abdullah Bursyah Almarhum, walaupun 18 th tidak menemani penulis tetapi doa restu ayah sangat berarti bagi penulis.
- Ibunda Sumiyati Abdullah yang selalu menyadarkan penulis untuk cepat merampungkan studi.
- Kakanda Azanah Abdullah, Azan Abdullah, Albi Abdullah yang sudah dengan susah payah membantu penulis baik spirit maupun materi.
- Papa Mertua Bapak Amrozi Ismunandar serta Mama Reni Yulianingsih yang dengan sabar menunggu menantunya ini lulus.
- Istri dan Ibu dari anakku Ayu Tyas Warastri..terimakasih atas supportnya dan menjaga si kecil saat penulis begadang mengerjakan skripsi ini.
- Jagoanku..Raihan Fatih Satrianara..tole..tole..bapak dah jadi sarjana le..
- Dosen Pembimbingku..Pak Supriyono..duh pak terimakasih kesabaran dan omelannya tanpa itu penulis takkan termotivasi..saya jadi kangen omelannya..
- Pak Muchlas..penulis buku "Rangkaian Digital" terimakasih pak atas waktu-waktu bapak menerima saya untuk konsultasi, sungguh sangat berarti buat saya..
- Sahabat Karibku Danang, Bendod, Mahendra, Terly, Maya, Meko..karena kalian penulis sanggup menjalani cobaan hidup..
- Teman –teman kampus Ambon, Ryan, Rooney, Daus, Firdi, Totong, Rachma, Tika, Favri..dll deh, tanpa kalian males kuliah bro..
- Teman-teman kos dulu..mas rio, mas dul, mas dian, danang, ikhsan, andi, brathaa, kohar(my brother), wawan, dan surip..hidup indah selama ngekos bareng kalian..
- Red Crispy Babarsari yang bersedia menerima penulis menjadi waiter paruh waktu..
- Abankirenk Advertising yang memercayakan penulis mendesign dan ngeFlash..
- Yamaha Vega AB 4221 SS yang dari SMU menemani penulis menjelajahi kota pelajar ini.
- Terakhir buat kopi, rokok, indomie rebus serta laptop sungguh kalian tercipta berguna.

HALAMAN MOTTO

...” Selesaikanlah Apa Yang Sudah Kamu Mulai ”...

KATA PENGANTAR

بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيْمِ

'salamu'alaikum Wr. Wb

Dengan mengucapkan alhamdulillah, puji dan syukur kehadirat Allah SWT yang ah melimpahkan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan tugas hir ini, yang berjudul "***Aplikasi Gerbang Logika Pada Rangkaian Elektronika***" ngan baik.

Laporan tugas ini disusun untuk melengkapi salah satu syarat guna memperoleh lar Sarjana Teknik Informatika pada Universitas Islam Indonesia dan atas apa yang lah diajarkan selama perkuliahan baik teori maupun praktek, disamping laporan itu ndiri yang merupakan rangkaian kegiatan yang harus dilakukan setelah tugas akhir ini elesai.

Penulisan dan penyelesaian tugas akhir ini tidak lepas dari saran, bimbingan, lukungan serta bantuan dari berbagai pihak. Untuk itu pada kesempatan kali ini penulis menyampaikan ucapan terima kasih kepada :

1. Bapak Fathul Wahid, ST., M.Sc., selaku Dekan Fakultas Teknologi Industri Universitas Islam
2. Bapak Yudi Prayudi, S.Si., M.Kom., selaku Ketua Jurusan Teknik Informatika.
3. Bapak Drs. Supriyono, M.Sc., selaku Dosen Pembimbing Tugas Akhir. Terima kasih atas segala bantuan, dukungan, bimbingan, pengetahuan serta mengajarkan kepada penulis apa arti dari sebuah tanggung jawab.
4. Bapak Muchlas , selaku tempat penulis berkonsultasi dan bukunya menjadi acuan penulis untuk menyelesaikan tugas akhir ini.
5. Kedua orang tuaku tercinta, Almarhum Bapak (Abdullah Bursyah) dan Ibu (Sumiyati Abdullah) yang tiada henti-hentinya melimpahkan kasih sayang yang

tulus, doa, dukungan, bimbingan, kesabaran, serta nasihat yang tidak ternilai harganya.

6. Kedua mertuaku tersayang, Papa (Amrozi Ismunandar) dan Mama (Reni Yulianingsih) yang selalu mendukung penulis dalam penyusunan skripsi ini.
7. Pendamping hidupku (Ayu Tyas Warastri) dan satria kecilku (Raihan Fatih Satrianara) yang selalu menjadikan penulis bersemangat tiap harinya.
8. Kakak-kakakku, Azanah Abdullah, Azan Abdullah, Albi Abdullah, dan ponakan-ponakan yang bandel, terimakasih atas doa, kesabaran, nasihat, pengertian, dukungan dan semangat yang selalu diberikan kepada penulis.
9. Sahabat – sahabat karibku Danang, Bendod, Mahendra dan Terly, yang selama ini membantu penulis dalam keadaan tertawa dan ber-airmata.
10. Teman-temanku Muh.Sigit Lestantyo, Ambon Fauzan, Pak Ryan, Pak Boss, Pak Hawi, Pak Hendra, Pak Suhadian, Bu Rahma, Bu Migma, Bu Anik, Bu Tata, Bu Demi, Bu Yanti dan Bu Nandha yang selalu menemani hari-hariku disaat kuliah .
11. Seluruh civitas akademika di lingkungan Teknik Informatika khususnya teman-teman Voip '02, teman-teman angkatan 2003, angkatan 2004, atas persahabatan, kebersamaan, dukungan, dan pengetahuan yang telah diberikan kepada penulis.
12. Semua pihak yang telah membantu dalam pembuatan hingga terselesaikannya tugas akhir ini, yang tidak penulis sebutkan satu persatu.

Akhir kata dengan ketulusan hati penulis panjatkan doa semoga apa yang telah mereka berikan dengan keikhlasan, mendapat pahala yang setimpal dari Allah SWT. Penulis menyadari dalam penulisan laporan tugas akhir ini masih jauh dari sempurna, karena keterbatasan kemampuan dan pengalaman. Penulis mengharapkan saran dan kritik yang bersifat membangun untuk memperbaiki tugas akhir ini semoga dapat bermanfaat bagi penulis khususnya dan pembaca pada umumnya.

Wassalamu'alaikum Wr. Wb

Yogyakarta, 9 Desember 2007

Abdi

ABSTRAKSI

Gerbang logika atau sering juga disebut gerbang logika Boolean merupakan sebuah sistem pemrosesan dasar yang dapat memproses *input-input* yang berupa bilangan biner menjadi sebuah *output* yang berkondisi yang akhirnya digunakan untuk proses selanjutnya. Gerbang logika dapat mengkondisikan *input-input* yang masuk kemudian menjadikannya sebuah *output* yang sesuai dengan apa yang ditentukan olehnya.dengan memanfaatkan teknologi yang ada, diperlukan suatu sistem yang dapat memberikan informasi yang berkaitan dengan proses gerbang logika pada rangkaian elektronika.

Sistem yang dibangun dengan tabel kebenaran (*truth table*) yang menunjukkan pengaruh pemberian level logika pada *input* suatu rangkaian logika terhadap keadaan level logika *output*nya. Dalam tabel tersebut ditampilkan semua kemungkinan pengaruh keadaan variable *input* terhadap keadaan *output* rangkaian logika. Melalui tabel kebenaran dapat diketahui watak atau karakteristik suatu rangkaian logika. Gerbang logika yang dibahas dalam penelitian ini adalah gerbang AND, gerbang OR, gerbang NOT, gerbang NAND, gerbang NOR dan gerbang XOR.

Hasil penelitian menunjukkan bahwa telah berhasil dibangun suatu aplikasi gerbang logika pada rangkaian elektronika, yang dapat dimanfaatkan oleh masyarakat (*user*) sebagai alat bantu dalam pembelajaran gerbang logika. Dengan memanfaatkan teknologi *action script* pada *macromedia flash* MX 2004, dapat mempercepat proses pengembangan sistem ini karena disertai operator-operator gerbang dasar sehingga diharapkan dapat mempercepat proses pembelajaran tentang gerbang logika bagi *user*.

Kata kunci :Gerbang Logika, Tabel Kebenaran, *Action Script*, Flash MX

TAKARIR

and	:	dan
application	:	aplikasi
asynchronous	:	tak serempak
carry	:	bawaan
counter	:	pencacah
code	:	kode
flowchart	:	diagram alir data
gate	:	gerbang
inventer	:	pembalik
input	:	masukan
interface	:	antarmuka
level	:	tingkatan
logic	:	logika
next	:	selanjutnya
not	:	tidak
output	:	keluaran
or	:	atau
previous	:	sebelumnya
proposition	:	proposisi
reset	:	kembali ke awal
script	:	kode bahasa pemrograman
software	:	perangkat lunak
system	:	sistem
tool	:	alat
truth	:	kebenaran
table	:	tabel
user friendly	:	mudah dipahami

DAFTAR ISI

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Logika adalah ilmu tentang penalaran (*reasoning*). Penalaran berarti mencari bukti validitas dari suatu argumen, mencari konsistensi dari pernyataan-pernyataan, dan membahas materi tentang kebenaran dan ketidakbenaran. Penekanan logika pada penarikan kesimpulan tentang validitas suatu argumen untuk mendapatkan kebenaran yang bersifat abstrak,yang dibangun dengan memakai kaidah-kaidah dasar logika tentang kebenaran dan ketidakbenaran yang menggunakan perangkai logika, yakni: "dan (*and*)", "atau (*or*)", "tidak (*not*)", "jika..maka..(*if..then implies*)", dan "..jika dan hanya jika..(*..if and only if..*)". Rangkaian elektronika pada dasarnya tersusun oleh rangkaian-rangkaian gerbang logika, misalnya gerbang *AND, OR, NAND*, dsb. [SOE06]

Gerbang-gerbang tersebut disusun berdasarkan logika matematika dengan bentuk tabel kebenaran. Untuk memudahkan proses tersusunnya rangkaian elektronika dengan basis tabel kebenaran, maka perlu dilakukan penelitian tentang hal tersebut diatas, dengan harapan dapat membantu proses perancangan rangkaian elektronika.

1.2. Rumusan Masalah

Rumusan masalah adalah bagaimana merancang dan membangun tabel kebenaran dengan logika matematika untuk aplikasi gerbang logika pada rangkaian elektronika.

1.3. Batasan Masalah

Pembatasan masalah disini bukan saja untuk menyederhanakan persoalan yang dihadapi, tetapi juga untuk menyederhanakan persoalan tersebut agar tidak menyimpang dari yang diinginkan. Yang menjadi batasan-batasan penelitian tugas akhir ini adalah :

Aplikasi ini hanya khusus mencakup bidang logika proposisional.

Rangkaian elektronika yang disusun yaitu rangkaian gerbang dasar, *decoder & encoder, half & full adder, flip-flop, multiplexer & demultiplexer, counter,* dan *register.* Perangkat lunak yang akan digunakan untuk membangun aplikasi ini adalah Macromedia Flash MX 2004 dan dijalankan di *platform* Windows XP.

1.4. Tujuan Tugas Akhir

Adapun tujuan yang hendak dicapai dalam penyusunan tugas akhir ini adalah membangun sistem tabel kebenaran dengan logika matematika untuk aplikasi gerbang logika pada rangkaian elektronika.

1.5. Manfaat Tugas Akhir

Dengan adanya penelitian diharapkan dapat memberikan manfaat antara lain :

a. Memberikan kemudahan bagi pengguna mengenal gerbang-gerbang logika pada rangkaian elektronika sebagai alat bantu ajar.

b. Mengimplementasikan gerbang-gerbang logika pada rangkaian elektronika sehingga membantu proses perancangan rangkaian elektronika.

1.6. Metodologi Penelitian

Metodologi penelitian adalah suatu cara berurutan yang dilakukan dalam penelitian. Metode yang digunakan untuk membantu dalam membangun tabel kebenaran dengan logika matematika untuk aplikasi gerbang logika pada rangkaian elektronika ini adalah :

a. Survei

Dilakukan untuk mengidentifikasi masalah dan kebutuhan, serta cara kerja dan ruang lingkup sistem yang akan dibuat. Survei ini dilakukan dengan dua cara:

1. Studi Pustaka, mempelajari buku-buku, artikel, situs dan skripsi yang berhubungan dengan permasalahan pada tugas akhir ini.
2. Wawancara, dilakukan pada beberapa dosen serta mahasiswa teknik elektronika yang mengerti tentang rangkaian digital.

b. Analisa

Dilakukan untuk mendapatkan pemahaman dari sistem yang akan diimplementasikan dengan jalan mendokumentasikan hasil proses pemahaman tersebut, juga untuk mengetahui kekurangan dan kelebihan sistem serta mengidentifikasi spesifikasi sistem yang akan dirancang.

c. Perancangan

Memodelkan sistem berdasar hasil analisa sehingga diperoleh gambaran penyelesaian dari permasalahan yang terdeteksi dari tahapan analisa. Gambaran ini akan digunakan sebagai acuan pada tahap implementasi dengan melibatkan teknologi yang mendukung berupa perangkat lunak.

d. Pemrograman

Merupakan tahapan implementasi dari hasil analisa dan perancangan dengan melibatkan teknologi yang mendukung berupa perangkat lunak.

e. Pengujian

Diperlukan untuk mengetahui apakah sistem dapat berjalan dengan baik, dan apakah sistem dapat menghasilkan keputusan yang baik, benar serta menguntungkan.

f. Analisa Hasil

Lanjutan dan merupakan langkah akhir penyusunan, menganalisis hasil dari sistem yang dibuat.

1.7. Sistematika Penulisan

Untuk memberikan gambaran secara menyeluruh mengenai masalah yang akan dibahas, sistematika penulisan tugas akhir ini dibagi dalam beberapa bab.

BAB I PENDAHULUAN

Bab ini berisi deskripsi umum isi tugas akhir yang meliputi latar belakang masalah, identifikasi masalah, batasan masalah, tujuan penyusunan tugas akhir, manfaat penyusunan tugas akhir, metode penelitian tugas akhir dan sistematika penulisan.

BAB II LANDASAN TEORI

Bab ini berisi landasan teori yang digunakan untuk merancang dan membangun aplikasi untuk Membangun Tabel Kebenaran Dengan Logika Matematika Untuk Aplikasi Gerbang Logika Pada Rangkaian Elektronika dan gambaran umum tentang Macromedia Flash MX 2004 sebagai bahasa pemrograman yang digunakan untuk membangun aplikasi ini.

BAB III METODOLOGI

Bab ini memuat uraian tentang metode analisis kebutuhan perangkat lunak yang dipakai serta dibahas juga kebutuhan masukan, kebutuhan keluaran dan antar muka yang digunakan, dan juga memuat tentang perancangan sistem meliputi perancangan diagram alir data (*flow chart*).

BAB IV HASIL DAN PEMBAHASAN

Bab ini berisi tentang perancangan perangkat lunak yang meliputi perancangan diagram alir data.

BAB V SIMPULAN DAN SARAN

Bab ini memuat kesimpulan yang merupakan rangkuman dari hasil analisis kinerja perangkat lunak dan saran-saran berdasarkan keterbatasan-keterbatasan yang ditemukan serta asumsi-asumsi yang dilihat selama pembuatan perangkat lunak.

BAB II

LANDASAN TEORI

2.1 Tabel Kebenaran (*Truth Table*).

Tabel kebenaran atau *truth table* merupakan tabel yang menunjukkan pengaruh pemberian level logika pada *input* suatu rangkaian logika terhadap keadaan level logika *output*nya. Dalam tabel tersebut ditampilkan semua kemungkinan pengaruh keadaan variabel *input* terhadap keadaan *output* rangkaian logika.

Oleh karena itu, tabel kebenaran mencerminkan watak atau karakteristik suatu rangkaian logika. Tabel kebenaran sebagai alat untuk menunjukkan watak rangkaian logika diperkenalkan pertama kali oleh George Boole pada 1854, kemudian dikembangkan oleh Claude Shannon dari *Bell Labs*.

Definisi lain dari tabel kebenaran yaitu suatu tabel yang menunjukkan secara sistematis datu demi satu nilai-nilai kebenaran sebagai hasil kombinasi dari proposisi-proposisi yang sederhana[SOE06].

2.2 Logika Matematika

Logika matematika (*mathematical logic*) adalah cabang ilmu di bidang Matematika yang memperdalam masalah logika, atau lebih tepatnya memperjelas logika dengan kaidah-kaidah matematika.

Logika matematika sendiri terus berkembang, mulai dari logika proposisional, logika predikat, pemograman logika, dan sebagainya. Perkembangan terakhir ilmu logika adalah logika *fuzzy*, atau di Indonesia disebut logika kabur atau logika samar.

Logika dalam ilmu komputer digunakan sebagai dasar untuk belajar bahasa pemograman, struktur data, kecerdasan buatan, teknik/sistem digital, basis data, teori komputasi, rekayasa perangkat lunak, sistem pakar, jaringan saraf tiruan, dan lain-lainnya yang mempergunakan logika secara intensif. Salah satu contoh yang populer adalah sistem digital, yaitu bidang ilmu yang didasari oleh logika untuk membuat gerbang logika (*logic gates)* dan arsitektur komputer sebagai inti *mikroprocessor,* otak komputer atau *central processing unit.*

2. 2.1 Logika Proposisional

Logika proposisional atau *sentennial logic* dikembangkan oleh GeorgeBoole dan Augustus De Morgan yang memberinya nama Logika Simbolik, karena logika tersebut bekerja dengan memanipulasi simbol-simbol. Dasar pemberian nilainya kemudian diformulasikan pada Tabel Kebenaran (*Truth Table)* yang diperkenalkan oleh Emil L. Post (1897-1954) dan Ludwig J.J Wittgeinstein (1889-1951).

Proposisi atau *proposition* adalah setiap pernyataan yang hanya memiliki satu nilai benar dan salah, sehingga logika proposisional yang menangani atau memproses atau memanipulasi penarikan kesimpulan secara logis. Proposisi –proposisi dapat digabung dan dimanipulasi sedemikian rupa dengan berbagai cara sehingga membentuk proposisi yang rumit. Penggabungan tersebut dilakukan dengan perangkai-perangkai (*connectives)* sehingga disebut proposisi majemuk (*compund propositions).*

2.3 Gerbang Logika

Gerbang logika atau sering juga disebut gerbang logika boolean merupakan sebuah sistem pemrosesan dasar yang dapat memproses *input-input* yang berupa bilangan biner menjadi sebuah *output* yang berkondisi yang akhirnya digunakan untuk proses selanjutnya. Gerbang logika dapat mengkondisikan *input-input* yang masuk kemudian menjadikannya sebuah *output* yang sesuai dengan apa yang ditentukan olehnya.

Jadi sebenarnya, gerbang logika inilah yang melakukan pemrosesan terhadap segala sesuatu yang masuk dan keluar ke dan dari komputer. Maka dari itu, sebenarnya sebuah perangkat komputer merupakan sebentuk kumpulan gerbang-gerbang digital yang bekerja memproses sesuatu *input*, menjadi *output* yang di inginkan.

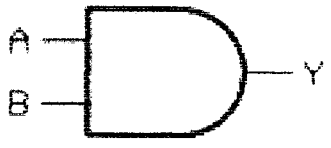
2.3.1 Gerbang – Gerbang Dasar

Gerbang logika boolean itu sendiri terdiri dari beberapa jenis. Masing-masing dapat melakukan proses yang berbeda. Maka dari itu, gerbang-gerbang ini nantinya akan disatukan untuk membentuk sebuah sistem pemrosesan yang lebih besar lagi. Berikut ini adalah jenis-jenisnya:

2.3.1.1 Gerbang AND

Gerbang AND memiliki karakteristik logika di mana jika kedua *input* yang masuk adalah bernilai 0, maka hasil *output*nya pasti akan bernilai 0. Jika kedua *input* diberi nilai 1, maka hasil *output* akan bernilai 1 pula. Logika gerbang AND bisa diumpamakan sebagai sebuah rangkaian dengan dua buah saklar yang disusun secara paralel. Jika salah satunya memutuskan hubungan rangkaian, maka hasil yang dikeluarkan dari rangkaian tersebut adalah 0. Tidak peduli saklar manapun yang diputuskan maka hasil akhirnya adalah 0. Ketika

kedua buah saklar terhubung dengan rangkaian bersamaan, maka hasil akhirnya barulah bernilai 1.



Gambar 2.1 Simbol Gerbang AND 2 *input*.

2.3.1.2 Gerbang NOT

Gerbang NOT sering disebut juga dengan istilah *inverter* atau pembalik. Logika dari gerbang ini adalah membalik apa yang di-input ke dalamnya. Biasanya *input*-nya hanya terdiri dari satu kaki saja. Ketika input yang masuk adalah 1, maka hasil *output*-nya adalah 0. Jika *input* yang masuk adalah 0, maka hasil *output*-nya adalah 1. Banyak sekali penerapan gerbang NOT ini pada rangkaian digital, meskipun fungsinya sangat sederhana.

Gambar 2.2 Simbol Gerbang NOT.

2.3.1.3 Gerbang OR

Gerbang OR dapat dikatakan memiliki karakteristik "memihak 1", di mana karakteristik logikanya akan selalu mengeluarkan hasil *output* bernilai 1 apabila ada satu saja *input* yang bernilai 1. Jadi gerbang logika ini tidak peduli berapa nilai *input* pada kedua sisinya, asalkan salah satunya atau kedua-duanya bernilai 1, maka *output*nya pasti juga akan bernilai 1. Logika gerbang OR ini

dapat diumpamakan sebagai sebuah rangkaian dengan dua buah saklar yang terpasang secara seri.

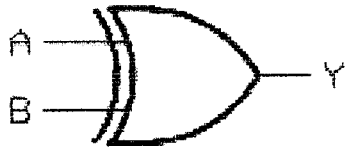
Apabila salah satu saklar memutuskan hubungan (bernilai 0), maka *output*-nya tetaplah bernilai 1 karena *input* yang lain tidak akan terputus hubungannya dengan *output*. Apabila kedua *input* bernilai 0, maka *output* barulah benar-benar terputus atau bernilai 0. Jika keduanya bernilai 1, maka *output* juga akan bernilai 1.

Gambar 2.3 Simbol Gerbang OR 2 *input*.

2.3.1.4 Gerbang X-OR (*Exlusive OR)*

Gerbang XOR merupakan singkatan dari kata *Exclusive*-OR. Sesuai dengan namanya, gerbang logika ini merupakan versi modifikasi dari gerbang OR. Jika pada gerbang OR Anda akan mendapatkan hasil *output* yang serba 1 jika salah satu *input* atau keduanya bernilai 1, tidak demikian dengan XOR. Gerbang logika ini hanya akan mengeluarkan hasil *output* bernilai 1 jika hanya salah satu *input* saja yang bernilai 1. Maksudnya jika kedua *input* bernilai 1, maka hasil *output*-nya tetaplah 0.

Jadi dengan demikian, logika XOR tidak akan membiarkan kedua *input* bernilai sama. Jika sama, maka hasil *output*-nya adalah 0.



Gambar 2.4 Simbol Gerbang XOR 2 *input*.

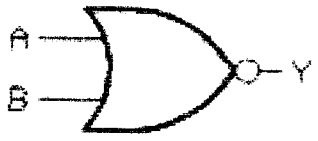
2.3.1.5 Gerbang NAND

Gerbang logika NAND merupakan modifikasi yang dilakukan pada gerbang AND dengan menambahkan gerbang NOT didalam prosesnya. Maka itu, mengapa gerbang ini dinamai NAND atau NOTAND. Logika NAND benar-benar merupakan kebalikan dari apa yang dihasilkan oleh gerbang AND. Di dalam gerbang logika NAND, jika salah satu *input* atau keduanya bernilai 0 maka hasil *output*-nya adalah 1. Jika kedua *input* bernilai 1 maka hasil *output*-nya adalah 0.

Gambar 2.5 Simbol Gerbang NAND 2 *input*.

2.3.1.6 Gerbang NOR

Gerbang NOR atau NOT-OR juga merupakan kebalikan dari gerbang logika OR. Semua *input* atau salah satu *input* bernilai 1, maka *output*-nya akan bernilai 0. Jika kedua input bernilai 0, maka *output*-nya akan bernilai 1.

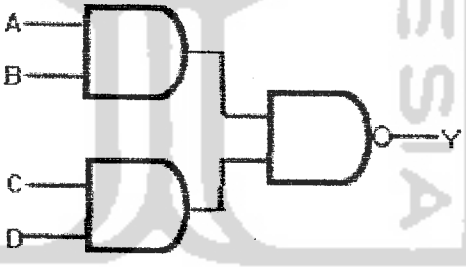


Gambar 2.6 Simbol Gerbang NOR 2 *input*.

2.3.2 Kombinasi Gerbang Dasar

2.3.2.1 Gerbang AND–NAND

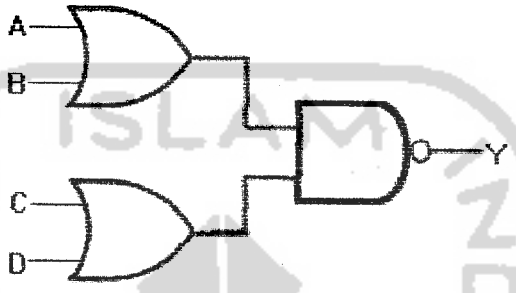
Gerbang AND–NAND merupakan rangkaian gabungan gerbang AND dengan gerbang NAND. Pada gerbang AND–NAND ini, kedua input yang masuk diproses melalui gerbang AND terlebih dahulu kemudian *output* dari gerbang AND tersebut diproses lagi sebagai input pada gerbang NAND, sehingga dari proses terakhir pada gerbang NAND didapat *output* gerbang AND-NAND.



Gambar 2.7 Rangkaian gerbang AND-NAND.

2.3.2.2 Gerbang OR-NAND

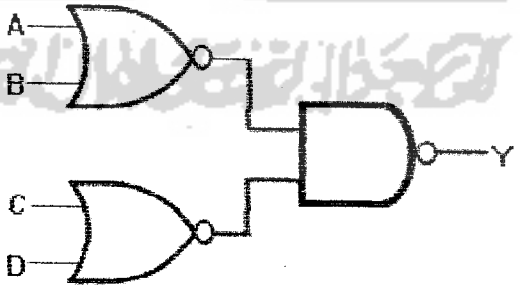
Gerbang OR-NAND prosesnya sama seperti gerbang AND-NAND diatas, *output* pada gerbang OR dijadikan *input* pada gerbang NAND schingga didapat *output* dari gerbang OR-NAND.



Gambar 2.8 Rangkaian gerbang OR-NAND.

2.3.2.3 Gerbang NOR-NAND

Sama seperti gerbang- gerbang sebelumnya, proses pada gerbang NOR-NAND didapat dari *output* pada gerbang NOR dan diteruskan menjadi *input* pada gerbang NAND.

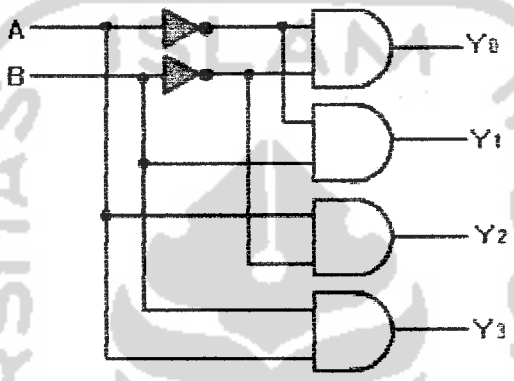


Gambar 2.9 Rangkaian Gerbang NOR-NAND.

2.4 *Decoder* dan *Encoder*

2.4.1 *Decoder*

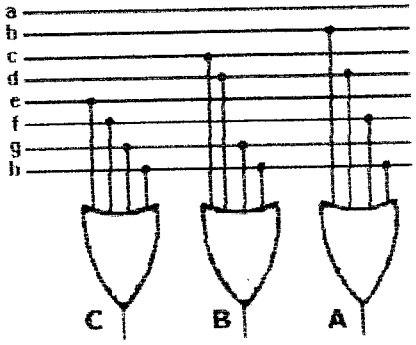
Decoder merupakan rangkaian logika yang berfungsi mengkode ulang atau menafsirkan kode-kode biner yang ada pada *input*nya menjadi data asli pada *output*nya. Contoh : *decoder* 2 ke 4 berfungsi menafsirkan kode-kode biner 2 bit menjadi data asli bilangan desimal 0 sampai dengan 3.



Gambar 2.10 Rangkaian *Decoder* 2 ke 4.

2.4.2 *Encoder*

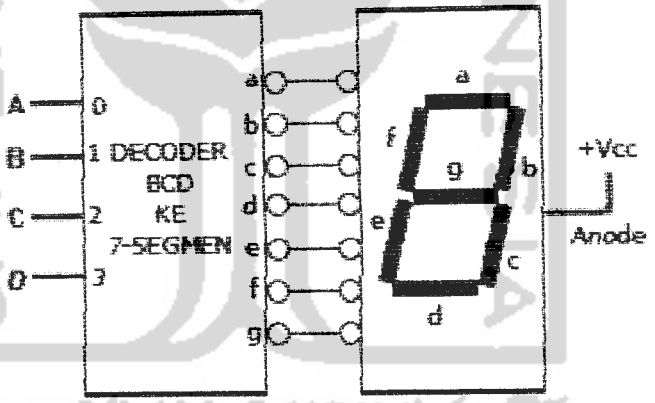
Encoder merupakan rangkaian logika yang berfungsi mengubah data yang ada pada *input*nya menjadi kode-kode biner pada *output*nya. Contoh *encoder* oktal ke biner atau disebut juga *encoder* 8 ke 3, berfungsi mengubah data bilangan oktal pada inputnya menjadi kode biner 3-bit pada *output*nya.



Gambar 2.11 Rangkaian *Encoder* 8 ke 3.

2.4.3 Decoder BCD ke Peraga 7-Segmen

Agar data dalam bentuk kode BCD dapat langsung ditampilkan pada peraga 7-segmen, maka diperlukan rangkaian *decoder* yang menghasilkan sinyal-sinyal penggerak peraga 7-segmen.

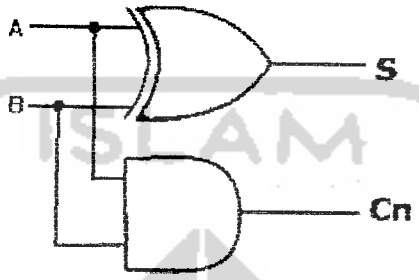


Gambar 2.12 Simbol *Decoder* BCD ke Peraga 7-Segmen.

2.5 *Half* dan *Full Adder*

2.5.1 *Half Adder*

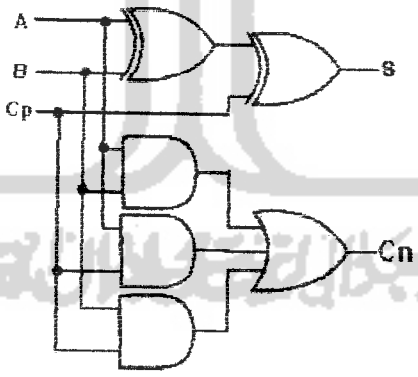
Half Adder merupakan rangkaian penjumlah 2-bit biner yang tidak menyertakan bawaan sebelumnya(*previous carry*) pada inputnya.



Gambar 2.13 Rangkaian *Half Adder*

2.5.2 *Full Adder*

Full Adder adalah rangkaian penjumlah yang menyertakan bawaan sebelumnya (*previous carry*) pada inputnya.



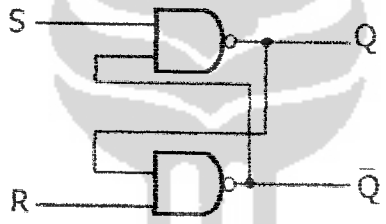
Gambar 2.14 Rangkaian *Full Adder*

2.6 *Flip – flop*

Elemen penyimpanan rangkaian logika sekuensi adalah *flip-flop*. *Flip-flop* merupakan sel biner yang mampu menyimpan data 1-bit , sehingga sel ini dinamakan pula memori 1-bit. Ciri-ciri *flip-flop* yang paling menonjol adalah memiliki dua buah *output*, yakni satu buah *output* dari data yang disimpan dan lainnya merupakan komplemennya.[MUC05]

2.6.1 *Flip Flop Set Reset*

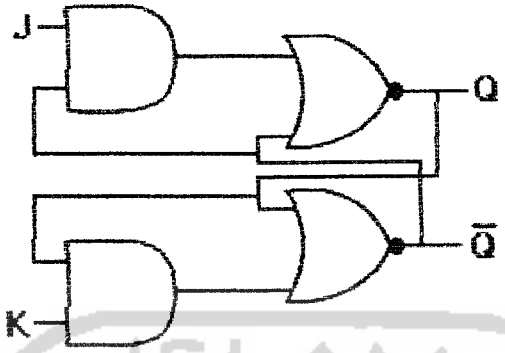
Flip flop set-reset atau disingkat flip SR merupakan memori yang melakukan penyimpanan data dengan cara memberi sinyal pada *input Set* (S) dan *Reset* (R) yang dimilikinya.



Gambar 2.15 Rangkaian *Flip-Flop* SR

2.6.2 *Flip Flop JK*

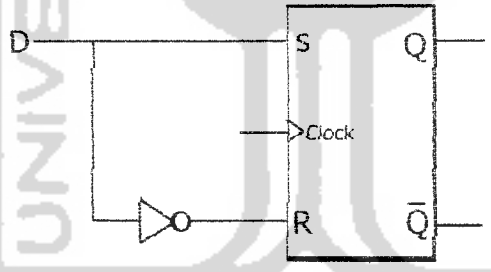
Kelemahan *flip-flop* SR adalah munculnya *output* yang tidak dapat didefinisikan ketika *input* S dan R tinggi untuk jenis NOR dan rendah untuk jenis NAND. Untuk menanggulangi keadaan tersebut, maka dikembangkan *flip-flop* JK. Jadi, *flip-flop* JK dibangun untuk mengantisipasi keadaan terlarang pada *flip-flop* SR.[MUC05]



Gambar 2.16 Rangkaian *Flip Flop* JK

2.6.3 Flip Flop D

Selain *flip-flop* SR dan JK terdapat pula *flip-flop* D. sesuai dengan namanya, input *flip-flop* ini adalah D. *Flip-flop* D dibangun dengan menggunakan *flip-flop* SR.

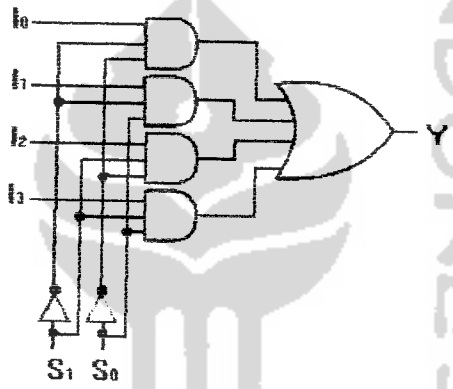


Gambar 2.17 Rangkaian *Flip Flop* D

2.7 *Multiplexer dan Demultiplexer*

2.7.1 *Multiplexer*

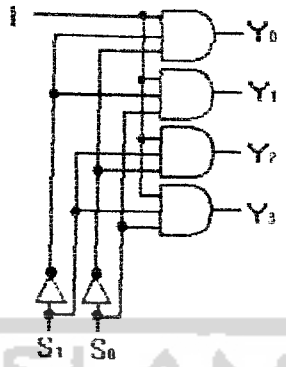
Multiplexer merupakan rangkaian logika yang berfungsi memilih data yang ada pada *input*nya untuk disalurkan ke *output*nya dengan bantuan sinyal pemilih atau sinyal control. Kata *multiplexer* sering dikemukakan dalam bentuk singkatan yakni MUX. *Multiplexer* disebut juga sebagai pemilih data (*data selector*).



Gambar 2.18 Rangkaian *Multiplexer* 4 ke 1

2.7.2 *Demultiplexer*

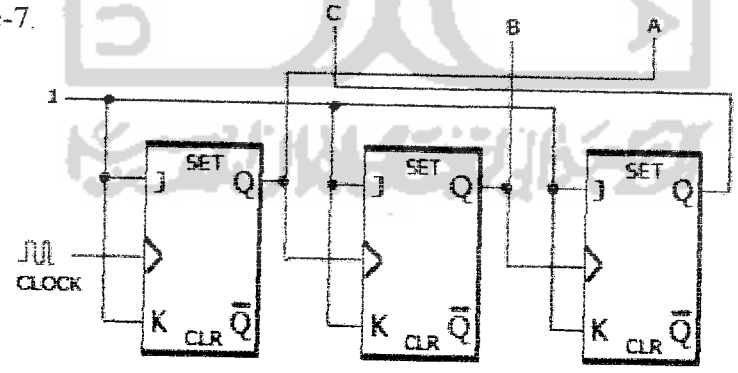
Demultiplexer merupakan rangkaian logika yang berfungsi menyalurkan data yang ada pada *input*nya ke salah satu dari beberapa *output*nya dengan bantuan sinyal pemilih atau sinyal control. Dalam penyebutannya, *demultiplexer* sering dikemukakan dalam bentuk singkatan saja yakni DEMUX. *Demultiplexer* disebut juga penyalur data (*data distributor*), dan fungsinya merupakan kebalikan dari *multiplexer*.



Gambar 2.19 Rangkaian *Demultiplexer* 1 ke 4

2.8 Pencacah (*counter*)

Pencacah atau *counter* merupakan rangkaian logika sekuensi yang berfungsi mencacah atau menghitung jumlah pulsa *clock* yang masuk. Menurut jumlah pulsa yang dapat dicacah, terdapat jenis pencacah modulo 2^n (n=1,2,3,4,...), contoh pencacah modulo-4, pencacah modulo-8, pencacah modulo-16. Jika *clock* ke-0 dinyatakan sebagai keadaan awal pencacah, jumlah pulsa yang dapat dicacah oleh pencacah modulo-8 adalah 8 buah. Pada pencacah modulo-8, *output* akan *reset* pada *clock* ke-8 sehingga pencacah ini hanya mampu mencacah pulsa *clock* ke-0 sampai pulsa *clock* ke-7.



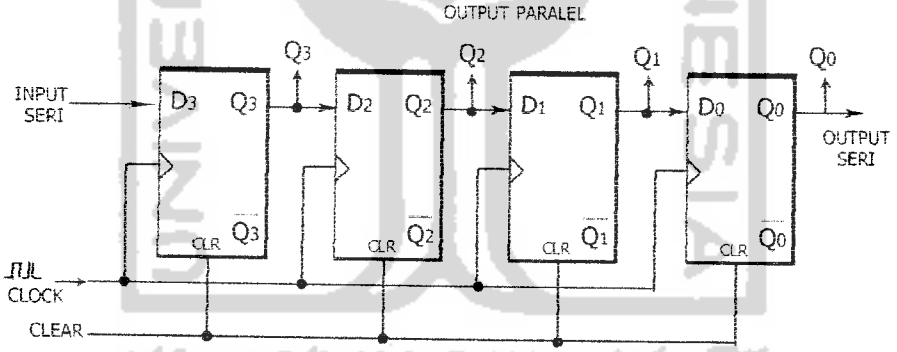
Gambar 2.20 Simbol *Counter Modulo-8* dengan *flip flop* JK

2.9 *Register*

Flip flop merupakan elemen logika yang berfungsi menyimpan data. Data-data yang disimpan pada elemen tersebut berbentuk keadaan biner yang dapat berupa angka maupun huruf yang disusun dalam format kode seperti BCD dan ASCII. Oleh karena data-data itu berbentuk suatu keadaan biner yang panjangnya lebih dari satu bit maka untuk menyimpannya diperlukan elemen yang terdiri atas beberapa *flip-flop*, dan elemen seperti itu dinamakan *register*.

2.9.1 *Register* Geser (*Shift Register*)

Register geser merupakan rangkaian logika yang melakukan penyimpanan data secara seri dengan memasukkan data bit demi bit. Disebut *register* geser karena dalam memindahkan data dari *input* ke *output*nya, *register* ini melakukan penggeseran bit yang ada di dalam elemen-elemennya.

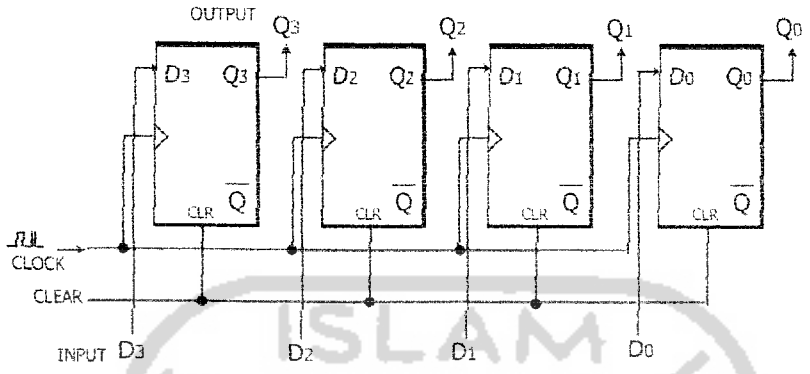


Gambar 2.21 Rangkaian *Register* geser 4-bit

2.9.2 *Register* Paralel

Register paralel merupakan rangkaian logika yang memiliki input dan *output* berupa saluran data paralel dengan panjang n-bit. Penyimpanan data pada *register* paralel dilakukan dengan cara menempatkan data yang akan

disimpan pada *input* paralel, dan untuk memindahkan data tersebut ke *output*nya dilakukan dengan memberikan pulsa *clock*.



Gambar 2.22 Rangkaian *register* paralel 4-bit

2.10 Rangkaian Elektronika

Rangkaian elektronika didefinisikan sebagai kesatuan dari komponen-komponen elektronika baik pasif maupun aktif yang membentuk suatu fungsi pengolahan sinyal (*signal processing*).

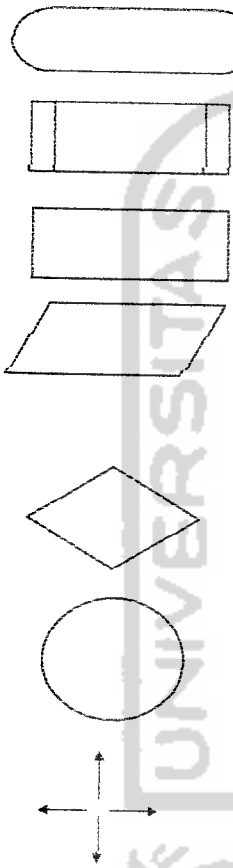
Berdasarkan sifat sinyal yang diolah, terdapat dua jenis rangkaian elektronika yakni rangkaian analog dan rangkaian digital. Rangkaian analog adalah rangkaian elektronika yang mengolah sinyal listrik kontinyu, sedangkan rangkaian digital adalah rangkaian elektronika yang mengolah sinyal listrik diskrit [MUC05].

2.11 Diagram Alir

Untuk membangun program komputer, maka langkah awal yang perlu dipersiapkan adalah membangun diagram alir program berdasarkan persoalan / masalah yang akan diselesaikan. Diagram alir (*flow chart*) ini terdiri dari simbol-simbol yang mewakili fungsi-fungsi langkah program yang menunjukkan urutan yang akan dikerjakan.

Bagan alir adalah suatu bagan yang berisi simbol-simbol grafis yang menunjukkan arah aliran kegiatan dan data yang terjadi dalam sebuah program. Secara umum, bagan alir bisa dikelompokkan menjadi bagan alir sistem (*system flowchart*) dan bagan alir program (*program flowchart*).

Adapun simbol-simbol diagram alir tersebut adalah sebagai berikut :



: menunjukkan awal dan akhir program

: definisi atau keterangan

: proses perhitungan

: *input* dan *output*

: pilihan

: menunjukkan hubungan arus yang terputus dengan sambungannya

: garis penghubung antar simbol

2.12 Gambaran Umum Macromedia

Macromedia adalah sebuah perusahaan perangkat lunak yang bergerak di bidang grafis dan pengembangan web. Perusahaan ini didirikan pada tahun 1992 dan telah berkembang pesat pada tahun 1990-an dan 2000-an. Pada Desember 2005

Macromedia diakuisisi salah satu perusahaan saingannya, Adobe Systems, tetapi Adobe sementara ini masih tetap menggunakan nama Macromedia pada sejumlah programnya.

Macromedia didirikan pada tahun 1992 melalui *merger* antara Authorware Inc. (perusahaan pembuat Authorware) dan MacroMind-Paracomp (perusahaan pembuat Macromind Director). Hingga pertengahan 1990-an, Macromedia Director yang digunakan untuk memproduksi CD-ROM dan kios-kios informasi masih merupakan produk unggulan Macromedia, namun seiring meningkatnya popularitas World Wide Web Macromedia menciptakan Shockwave, sebuah *plugin* Director bagi penjelajah web serta pada tahun 1996 mengakuisisi dua perusahaan berorientasi web, FutureWave Software (yang membuat FutureSplash Animator - yang kemudian berkembang menjadi Flash) dan iBand Software (pembuat perangkat lunak *authoring* HTML - yang digunakan sebagai dasar untuk mengembangkan Dreamweaver).

Tahun 2001 Macromedia mengakuisisi Allaire, yang mengembangkan ColdFusion sebelum pada akhirnya pada tahun 2005 Macromedia sendiri dibeli oleh Adobe.

2.12.1 Macromedia Flash

Macromedia Flash adalah program animasi interaktif berbasis *vector* yang sering digunakan pada web desain, sebagai sebuah program animasi, flash memiliki kelebihan dari program animasi lainnya karena adanya fasilitas *action script* sehingga animasi bisa menjadi lebih interaktif.

Sebagai program yang diproduksi oleh Macromedia, flash bisa diintegrasikan dengan program-program Macromedia yang lain seperti : Freehand, Fireworks, Dreamweaver, maupun Macromedia Director.

Output dari Flash adalah dalam format *Flash Movie File* (*swf). Dalam file ini sound dapat dikompresikan menjadi ukuran yang lebih kecil dengan format MP3 sehingga file Flash tetap dalam ukuran kecil tanpa merubah kualitas suara.

Selain untuk animasi pada web ataupun *full web pages* , *flash movie* dapat dijalankan secara *stand alone player* dengan fasilitas projektor. Dari projektor ini dengan beberapa jenis program *screensaver maker* yang ada seperti *screenweaver, screentime*, dan lain-lain dapat diaplikasikan lebih lanjut menjadi sebuah screensaver yang menawan. Jenis aplikasi flash yang lain adalah membuat interaktif *games*.

Hingga kini flash telah sampai pada versi 9.0 yang makin lengkap dengan fasilitas script yang makin mudah untuk digunakan.

2.12.2 Action Script

ActionScript adalah bahasa yang ditulis untuk *software* Macromedia Flash. Flash asalnya dibuat sebagai *software* animasi web dan *ActionScript* diintegrasikan ke representasi *timeline*. *ActionScript* berbasiskan pada *JavaScript* dan *Processing* dibuat menggunakan Java, jadi terdapat beberapa kesamaan antara kedua sistem tersebut.:

a. Menggambar

Kode *ActionScript* untuk menggambar sangat dioptimasi dan bentuk digambar ke layar lebih cepat dibanding pada *Processing*. Hal ini memungkinkan aplikasi Flash bisa berukuran besar dalam dimensi piksel dan sangat memanfaatkan tranparansi dan grafis antialias.

b. Menghitung/Kalkulasi

ActionScript sangat lambat dalam melakukan kalkulasi dan Processing jauh lebih baik untuk aplikasi yang memuat sejumlah besar elemen.

c. Operasi Piksel

ActionScript dibuat untuk *vector drawing* dan dengan demikian mengabaikan kemungkinan untuk secara langsung memanipulasi piksel. *Processing* menyediakan fungsi dan struktur data untuk sederhananya mengakses piksel dari suatu gambar dan membuat potensi untuk menulis banyak program menarik.

d. 3D

ActionScript tidak secara internal merepresentasikan struktur spasial/ruang dalam tiga dimensi, sementara itu pustaka gambar *Processing* secara inheren 3-D.

e. Objek

ActionScript secara inheren berorientasi objek, sementara *Processing* dapat ditulis dalam *style* prosedural maupun berorientasi objek. Hal ini membuat pelajar yang belajar dengan *Processing* untuk memahamai program pertama yang dibuatnya tanpa harus terlebih dahulu memahami metafor *object*-oriented

BAB III

METODOLOGI

3.1 Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan perangkat lunak merupakan bagian awal yang sangat penting dan paling mendasar dalam pembuatan sebuah sistem aplikasi. Apabila terdapat kesalahan pada analisis ini, maka akan berdampak pula terhadap tahapan-tahapan selanjutnya. Kesalahan yang ada dapat mengakibatkan ketidaksempurnaan pada perangkat lunak yang akan dibuat. Ketidaksempurnaan tersebut bisa saja menyebabkan alur jalannya program yang tidak tepat, sehingga proses yang berjalan pun menjadi tidak efisien. Walaupun dalam prakteknya perangkat lunak atau program yang dibuat dapat berjalan seperti yang diinginkan, namun dalam tahap evaluasi dan pengembangannya justru akan terjadi malfungsi.

Dengan adanya analisis kebutuhan sistem ini, maka diharapkan perangkat lunak yang akan dibuat dapat dinilai kinerjanya. Dengan demikian kelebihan ataupun kelemahan dari sistem dapat diketahui, agar nantinya dapat dilakukan perbaikan dalam pengembangannya.

3.1.1. Metode Analisis

Metode yang digunakan dalam pembuatan sistem "Membangun Tabel Kebenaran Dengan Logika Matematika Untuk Aplikasi Gerbang Logika Pada Rangkaian Elektronika" ini adalah dengan metode analisis yang berarah alir data.

Pada metode transformasi *input*, proses dan *output* dinyatakan dalam diagram alir atau *flow chart.*

3.1.2. Hasil Analisis

Berdasarkan analisis yang telah dilakukan maka dapat diketahui apa saja yang akan menjadi masukan sistem, proses-proses sistem, keluaran sistem, fungsi atau metode yang digunakan oleh sistem, kebutuhan perangkat keras, kebutuhan perangkat lunak serta antarmuka sistem yang akan dibuat, sehingga sistem yang dibangun sesuai dengan apa yang diharapkan.

3.1.2.1 Analisis Kebutuhan Masukan

Kebutuhan masukan atau *input* yang diperlukan untuk memenuhi kebutuhan dalam implementasi "Membangun Tabel Kebenaran Dengan Logika Matematika Untuk Aplikasi Gerbang Logika Pada Rangkaian Elektronika" ini adalah data berupa masukan nilai biner (0 atau 1) yang diperlukan untuk mendapatkan hasil akhir dari keluaran sistem.

3.1.2.2 Analisis Kebutuhan Proses

Berdasarkan analisis yang telah dilakukan maka dapat diketahui apa saja yang menjadi masukan sistem, keluaran sistem, spesifikasi fungsi atau metode yang digunakan oleh sistem, kebutuhan perangkat keras, kebutuhan perangkat lunak serta antar muka sistem yang akan dibuat, sehingga sistem yang nantinya sesuai dengan

apa yang diharapkan. Analisis kebutuhan juga bermanfaat sebagai dasar evaluasi setelah program selesai dibangun.

Masukan data untuk perangkat lunak pada aplikasi gerbang logika pada rangkaian elektronika terdiri dari beberapa proses, antara lain:

a. Proses pemilihan rangkaian elektronika.

   Pada proses ini, *user* akan memilih macam-macam rangkaian elektronika yang tersedia. Secara garis besar rangkaian elektronika yang dapat dipilih *user* antara lain: rangkaian gerbang dasar, *decoder & encoder, half & full adder, flip-flop, multiplexer & demultiplexer, counter,* dan *register.*

b. Proses pemilihan sub rangkaian elektronika.

   Pada proses ini, *user* akan memilih sub rangkaian elektronika yang akan diproses oleh sistem. Secara garis besar sub rangkaian elektronika yang dapat dipilih *user* antara lain, misalnya pada rangkaian gerbang-gerbang dasar yaitu: gerbang AND, gerbang OR, gerbang NOT, gerbang NAND, gerbang NOR, gerbang EXOR, gerbang AND-NAND, gerbang OR-NAND, dan gerbang NOR-NAND.

3.1.2.3 Analisis Kebutuhan Keluaran

Keluaran yang diinginkan dalam sistem ini adalah tabel kebenaran yang memiliki nilai prioritas menyeluruh tertinggi dari setiap rangkaian elektronika yang ada, sehingga diharapkan *user* dapat mengerti dan paham jika nilai biner yang *user* masukkan akan mendapat keluaran yang dapat dijelaskan dengan tabel kebenaran.

3.1.2.4 Kebutuhan Perangkat Lunak

Perangkat lunak yang dibutuhkan untuk pengembangan dan implementasi dari pembuatan sistem aplikasi antara lain :

1. Sistem operasi berbasiskan Windows *XP*.
2. *Web browser internet explorer* versi 5 keatas, sebagai alternatif untuk menampilkan aplikasi ini.
3. Macromedia Flash MX 2004, sebagai *tool* untuk membuat aplikasi ini.
4. Adobe Photoshop CS, sebagai *tool* untuk membuat *image* yang dibutuhkan aplikasi.

3.1.2.5 Kebutuhan Perangkat Keras

Perangkat keras adalah bagian dari sistem komputer yang harus ada sebagai media berjalannya perangkat lunak. Perangkat-perangkat keras tersebut meliputi :

1. Processor Intel Pentium M 1.73GHz.
2. Memory dengan kapasitas minimal 256MB.
3. *Harddisk* minimal 500MB.
4. *Mouse.*
5. *Keyboard.*

3.1.3 Antarmuka Sistem

Antarmuka atau *interface* merupakan sarana komunikasi yang menjadi perantara antara *user* dengan sistem aplikasi. Oleh karena itu antarmuka dari sistem yang akan dibuat harus *user friendly*, artinya pengguna dapat menggunakan perangkat lunak dengan mudah tanpa harus mempelajarinya terlebih dahulu. Sehingga dapat meminimalkan kesalahan, baik kesalahan masukan, proses maupun

keluaran yang dihasilkan dari sistem. Dalam tahap perancangan arsitektur perangkat lunak ini akan dijelaskan rincian format masukan, proses dan keluaran.

Antarmuka yang dibutuhkan dalam pembuatan sistem ini antara lain :

1. Halaman utama.
2. Antarmuka untuk memilih menu rangkaian elektronika yang tersedia..
3. Antarmuka untuk gerbang dasar.
4. Antarmuka untuk gerbang AND.
5. Antarmuka untuk gerbang OR.
6. Antarmuka untuk gerbang NOT.
7. Antarmuka untuk gerbang NAND.
8. Antarmuka untuk gerbang NOR.
9. Antarmuka untuk gerbang X-OR.
10. Antarmuka untuk gerbang AND-NAND.
11. Antarmuka untuk gerbang OR-NAND.
12. Antarmuka untuk gerbang NOR-NAND.
13. Antarmuka untuk *decoder & encoder*.
14. Antarmuka untuk *decoder*.
15. Antarmuka untuk *encoder*.
16. Antarmuka untuk peraga 7-segmen.
17. Antarmuka untuk *decoder* BCD ke peraga 7-segmen.
18. Antarmuka untuk *half & full adder*.
19. Antarmuka untuk *half adder*.
20. Antarmuka untuk *full adder*.
21. Antarmuka untuk *flip-flop*.
22. Antarmuka untuk *flip-flop set -reset*.
23. Antarmuka untuk *flip-flop* JK.
24. Antarmuka untuk *flip-flop* D.
25. Antarmuka untuk *multiplexer & demultiplexer*.

26. Antarmuka untuk *multiplexer*.
27. Antarmuka untuk *demultiplexer*.
28. Antarmuka untuk *multiplexer & demultiplexer*.
29. Antarmuka untuk pencacah (*counter*)..
30. Antarmuka untuk *asynchronous counter mod-8*.
31. Antarmuka untuk *register*.
32. Antarmuka untuk *register* geser 4-bit.
33. Antarmuka untuk *register parallel*

3.2 Perancangan Perangkat Lunak

Perancangan perangkat lunak adalah tahapan lanjutan dari pembuatan sistem setelah hasil dari proses analisis kebutuhan perangkat lunak diketahui. Perancangan ini meliputi perancangan dari alur jalannya proses sistem dengan diagram alir, dan perancangan antarmuka sistem (*interface*).

3.2.1 Metode Perancangan Perangkat Lunak

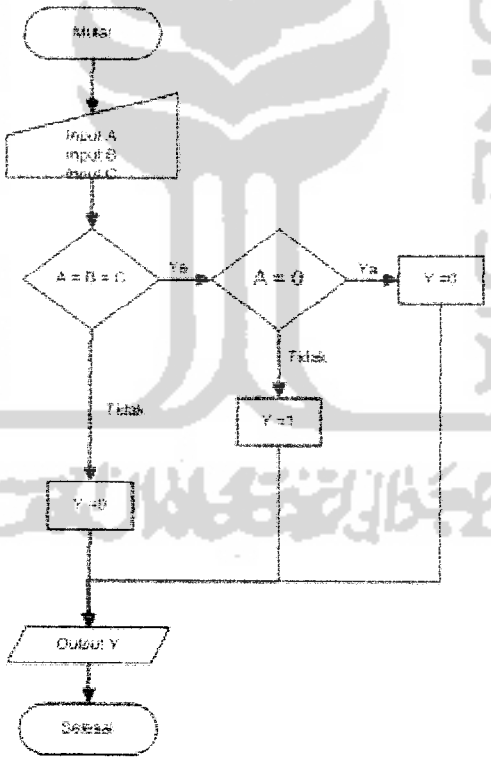
Metode perancangan yang digunakan dalam pembuatan "Membangun Tabel Kebenaran Dengan Logika Matematika Untuk Aplikasi Gerbang Logika Pada Rangkaian Elektronika" ini adalah dengan menggunakan *Flow Chart* atau Diagram Alir. *Flow Chart* merupakan metode yang digunakan pada metodologi pengembangan sistem yang terstruktur. Dengan menggunakan notasi-notasi, *Flow Chart* menggambarkan alir data dari sistem secara logika tanpa mempertimbangkan lingkungan fisik dimana data tersebut mengalir atau lingkungan fisik dimana data tersebut disimpan.

3.2.2 Hasil Perancangan Perangkat Lunak

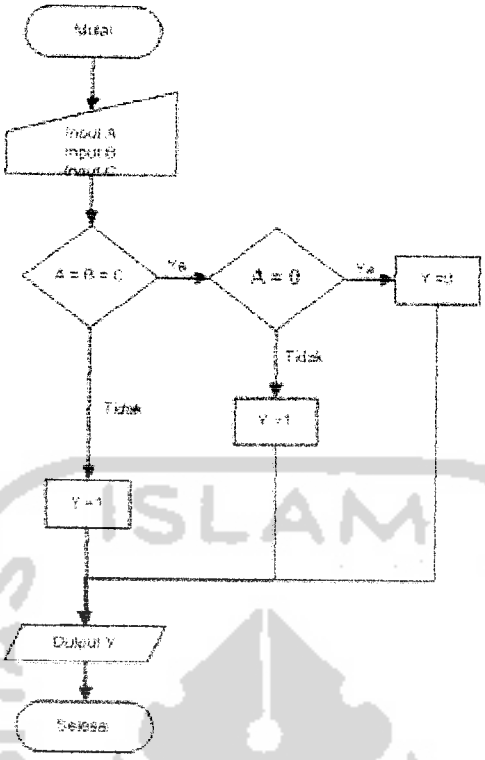
Hasil dari perancangan "Membangun Tabel Kebenaran Dengan Logika Matematika Untuk Aplikasi Gerbang Logika Pada Rangkaian Elektronika" disesuaikan dengan penerapan metode-metode dalam perancangan, yaitu metode perancangan terstruktur (*Structured Design Method*).

3.2.2.1 Diagram Alir Sistem

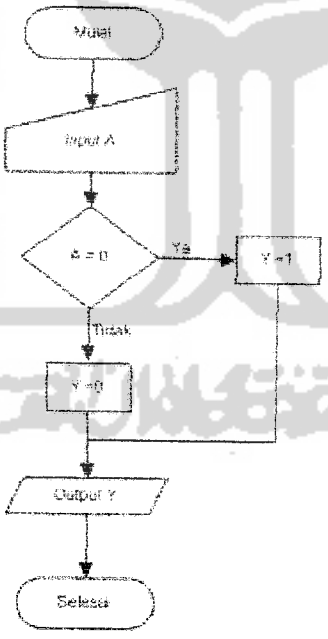
Bagan alir sistem digunakan untuk menggambarkan keseluruhan langkah kerja dan sistem yang akan dibuat juga akan digunakan untuk menentukan langkah-langkah kerja, mulai dari perancangan antarmuka sampai pembuatan laporan-laporan yang dibutuhkan pemakai.



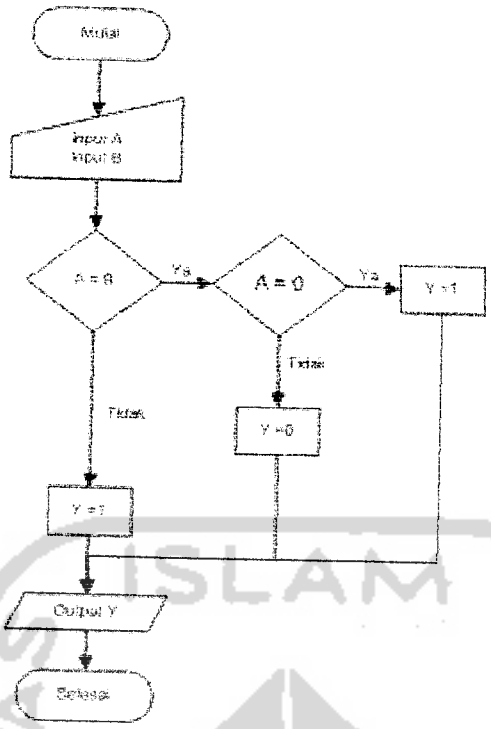
Gambar 3.1 Diagram Alir Gerbang AND



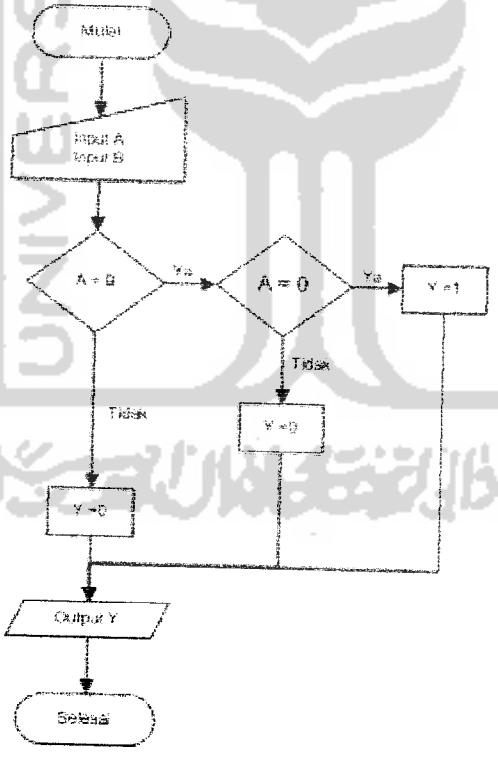
Gambar 3.2 Diagram Alir Gerbang OR



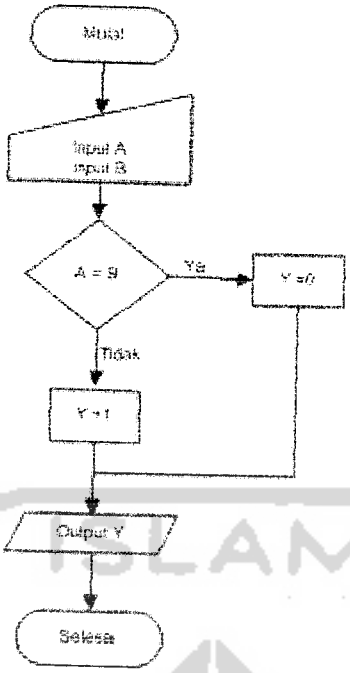
Gambar 3.3 Diagram Alir Gerbang NOT



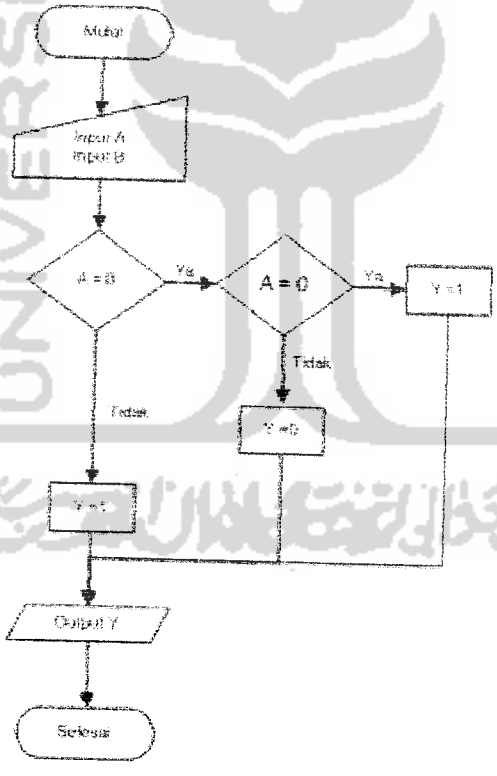
Gambar 3.4 Diagram Alir Gerbang NAND



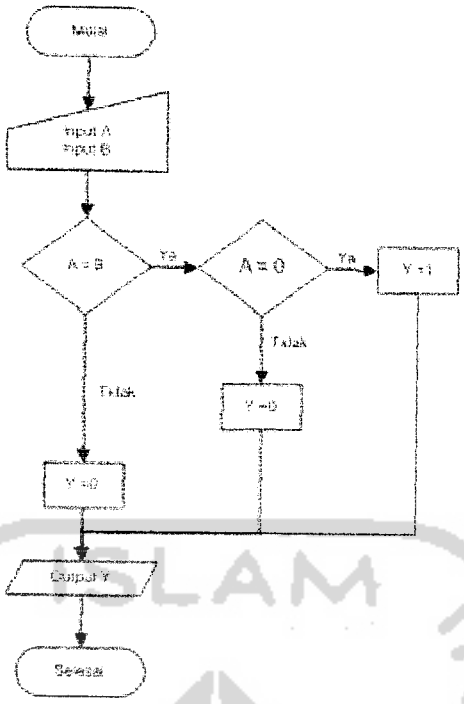
Gambar 3.5 Diagram Alir Gerbang NOR



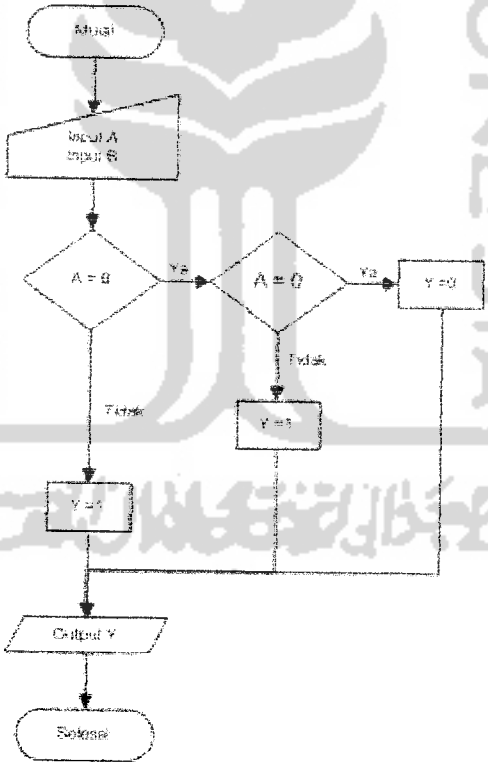
Gambar 3.6 Diagram Alir Gerbang XOR



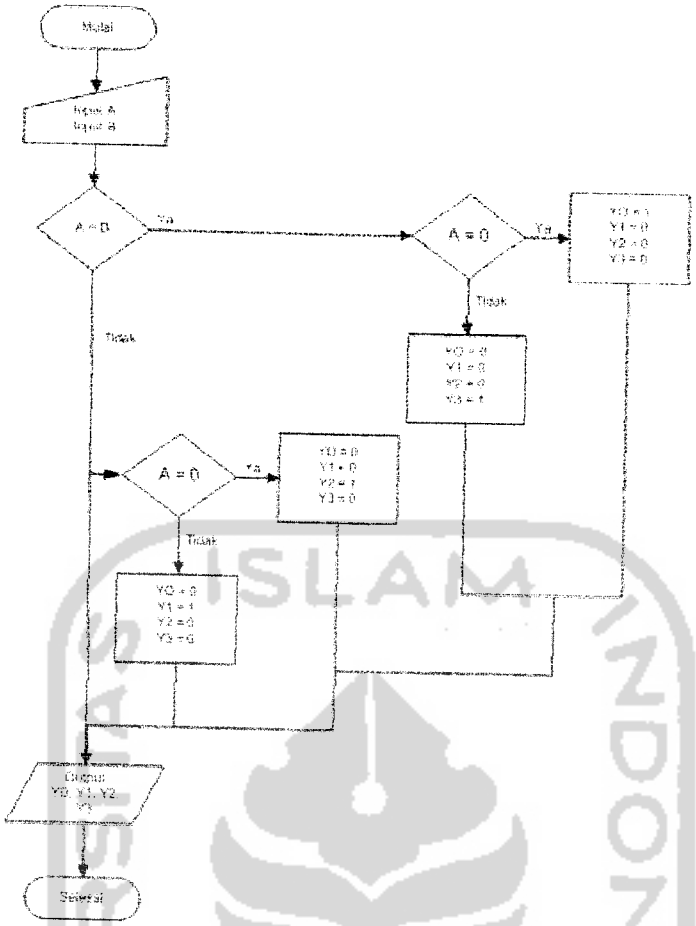
Gambar 3.7 Diagram Alir Rangkaian AND-NAND



Gambar 3.8 Diagram Alir RangkaianOR-NAND

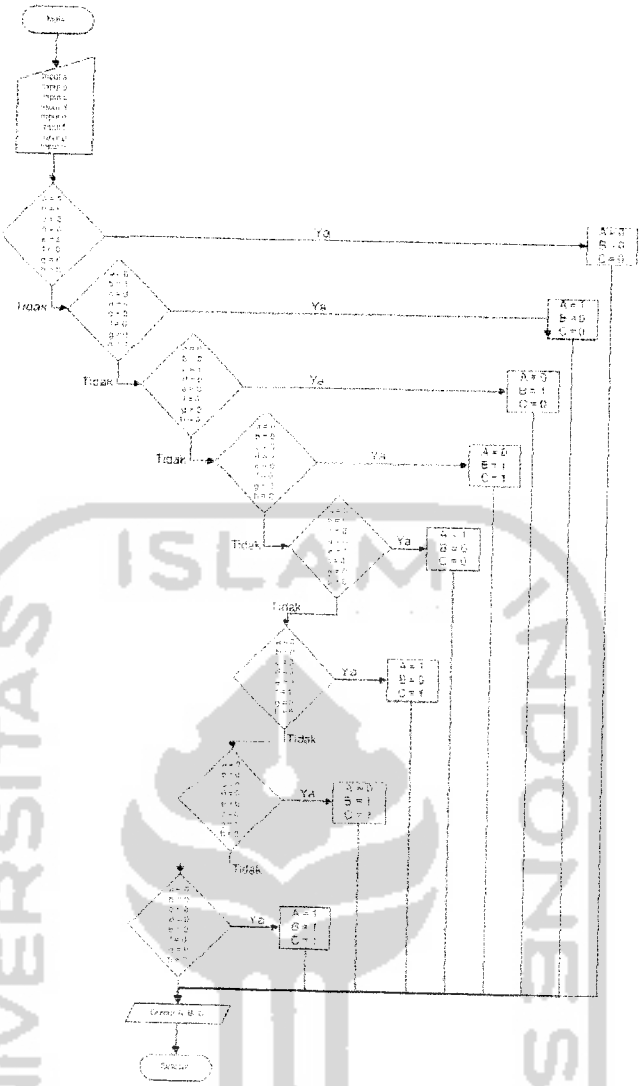


Gambar 3.9 Diagram Alir Rangkaian NOR-NAND



Gambar 3.10 Diagram Alir *Decoder*

Adapun *output* Y0 agar dapat bernilai 1 didapat dari komplemen dari A *and* B atau persamaannya Y0= (*not*A) *and* (*not*B). Selanjutnya untuk Y1, agar dapat menghasilkan nilai 1 maka persamaannya Y1= (*not*B) *and* A. Untuk Y2= B *and* (*not*A) dan untuk Y3= A *and* B.

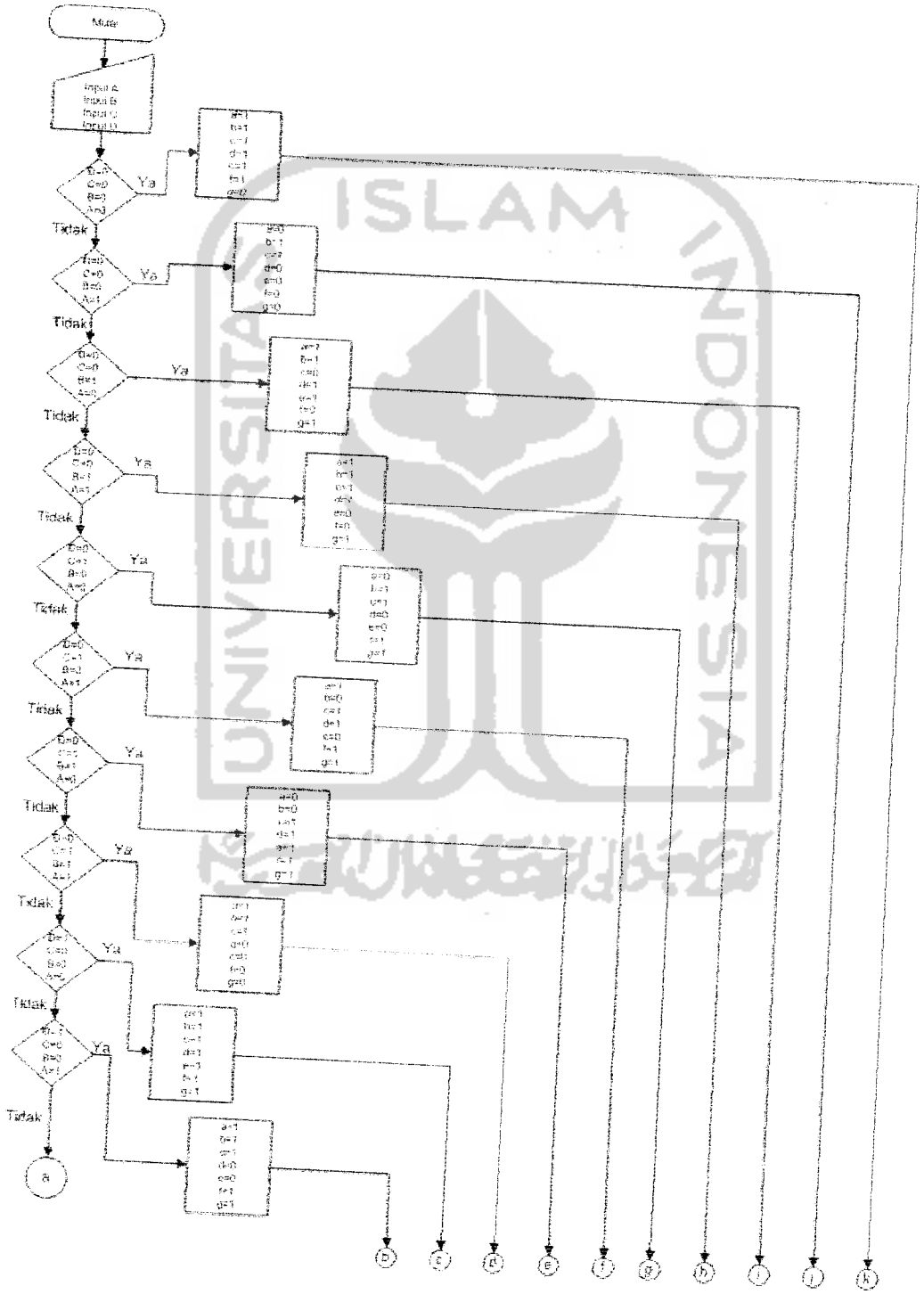


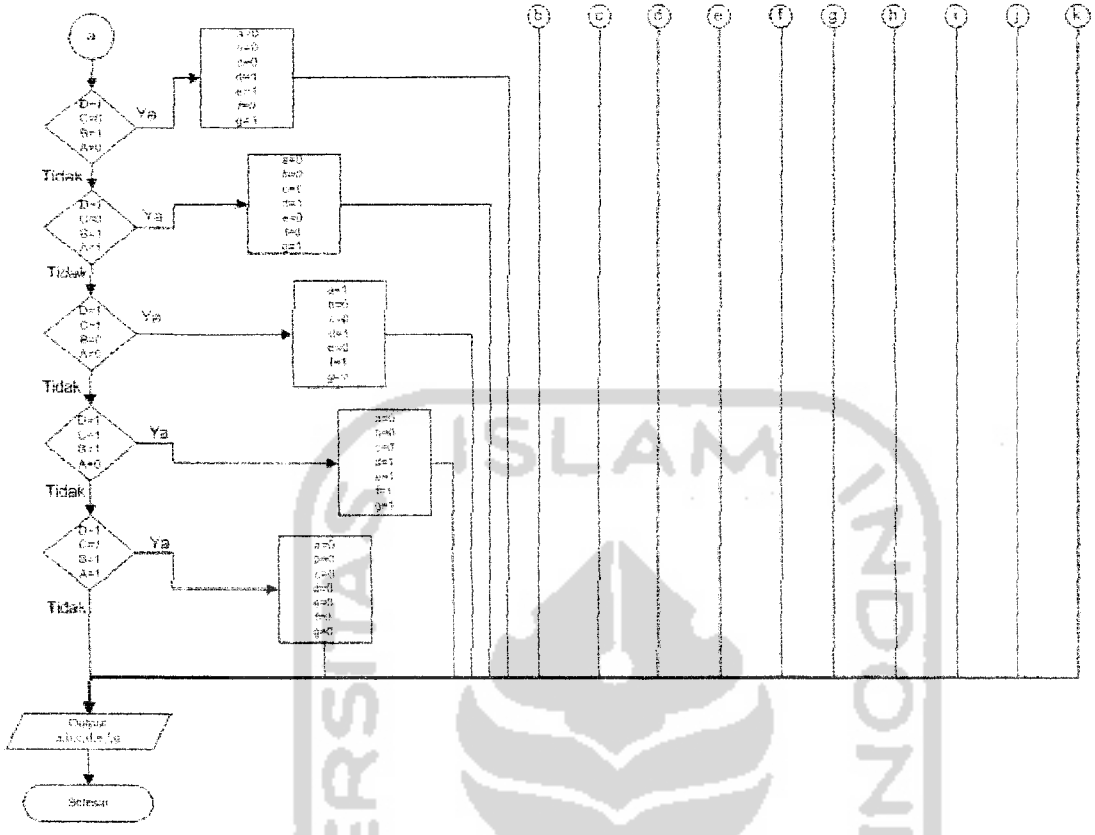
Gambar 3.11 Diagram Alir *Encoder* 8 ke 3

Encoder pada gambar 3.11 diatas merupakan *encoder* yang jenisnya bukan prioritas, artinya untuk menghasilkan kode biner pada *output*nya, hanya boleh ada 1 saklar saja pada *output*nya yang tertutup (*ON*). Maksudnya disini hanya boleh salah satu *input* saja yang bernilai 1 dan yang lain harus 0.

Pada bagian *output* A terdapat 4 nilai logika tinggi atau 1 yang bersesuaian dengan *input* a, *input* c, *input* e, *input* g sehingga *output* A merupakan operasi OR

dari *input* a, *input* c, *input* e, dan *input* g atau A= a + c + e + g. Pada *output* B terdapat 4 nilai logika tinggi yang bersesuaian dengan *input* b, *input* c, *input* f dan *input* g, sehingga B = b + c + f + g. Sedangkan pada *output* C, nilai logika tingginya bersesuaian dengan *input* d, *input* e, *input* f dan *input* g, sehingga C= d + e + f + g.





Gambar 3.12 Diagram Alir *Decoder BCD* ke Peraga 7-Segmen

Adapun persamaan dari *decoder BCD* ke peraga 7-segmen adalah :

a = ((notD) and (notC) and B) or ((notD) and C and A) or (D and (notB) and A) or ((notC) and (notB) and (notA))

b = ((notD)and B and A) or (D and (notC) and (notB)) or ((notD) and (notC)) or ((notB) and (notA))

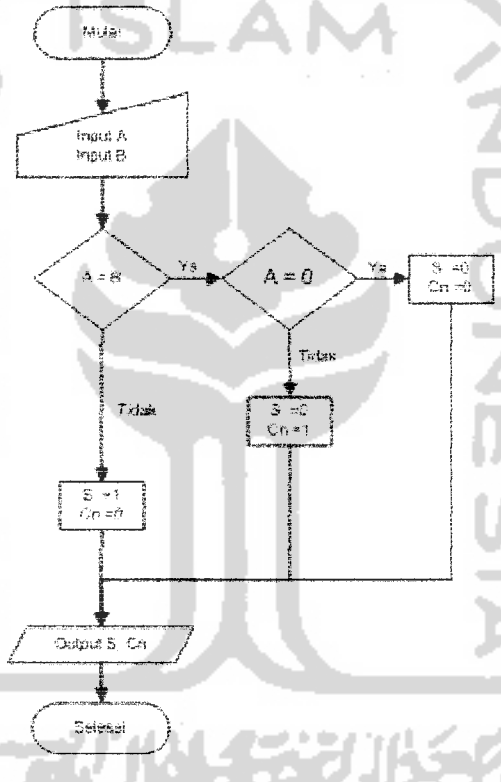
c= ((notC) and B and A) or ((notD) and C) or ((notC) and (notB))

d= (C and (notB) and A) or ((notC) and B and A) or ((notC) and (notA)) or (B and(notA))

e= ((notC) and (notA)) or (B and(notA))

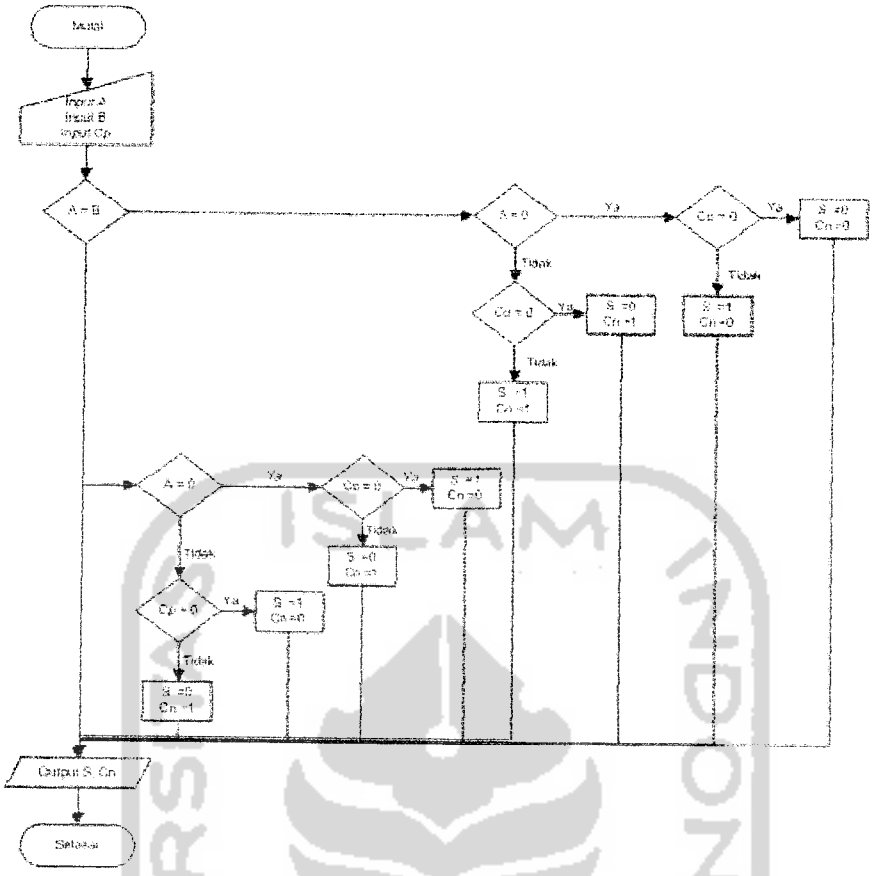
f= ((notB) and (notA)) or (C and(notA)) or (C and (notB)) or (D and(notB))

g= (B and (notA)) or ((notC) and B) or (C and (notB)) or (D and(notC))



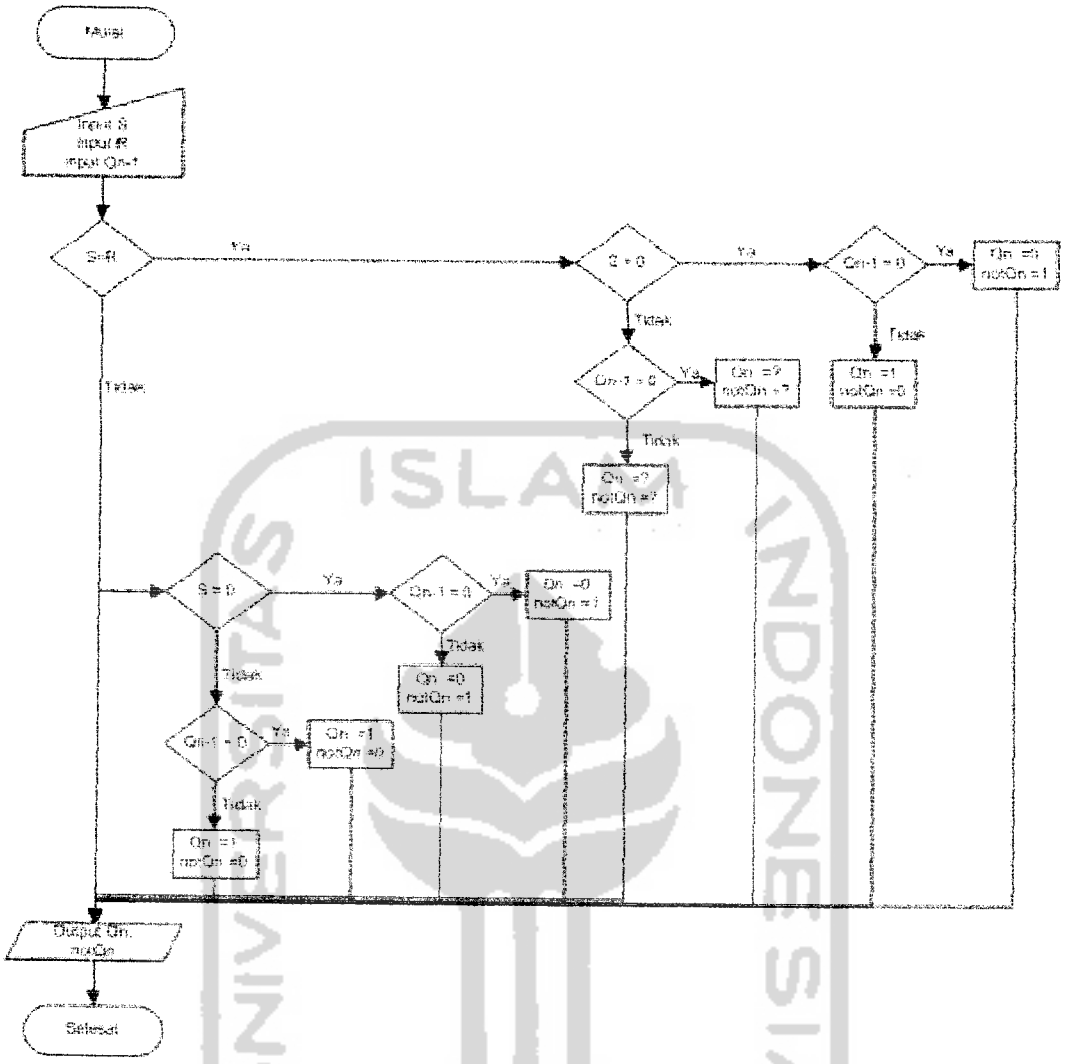
Gambar 3.13 Diagram Alir *Half Adder*

Adapun *output* S membentuk operasi XOR pada rangkaiannya dan *output* Cn membentuk operasi AND. Sehingga bisa diketahui bahwa S = A *xor* B dan untuk bawaan berikutnya atau *next carry* (Cn), Cn = A *and* B.



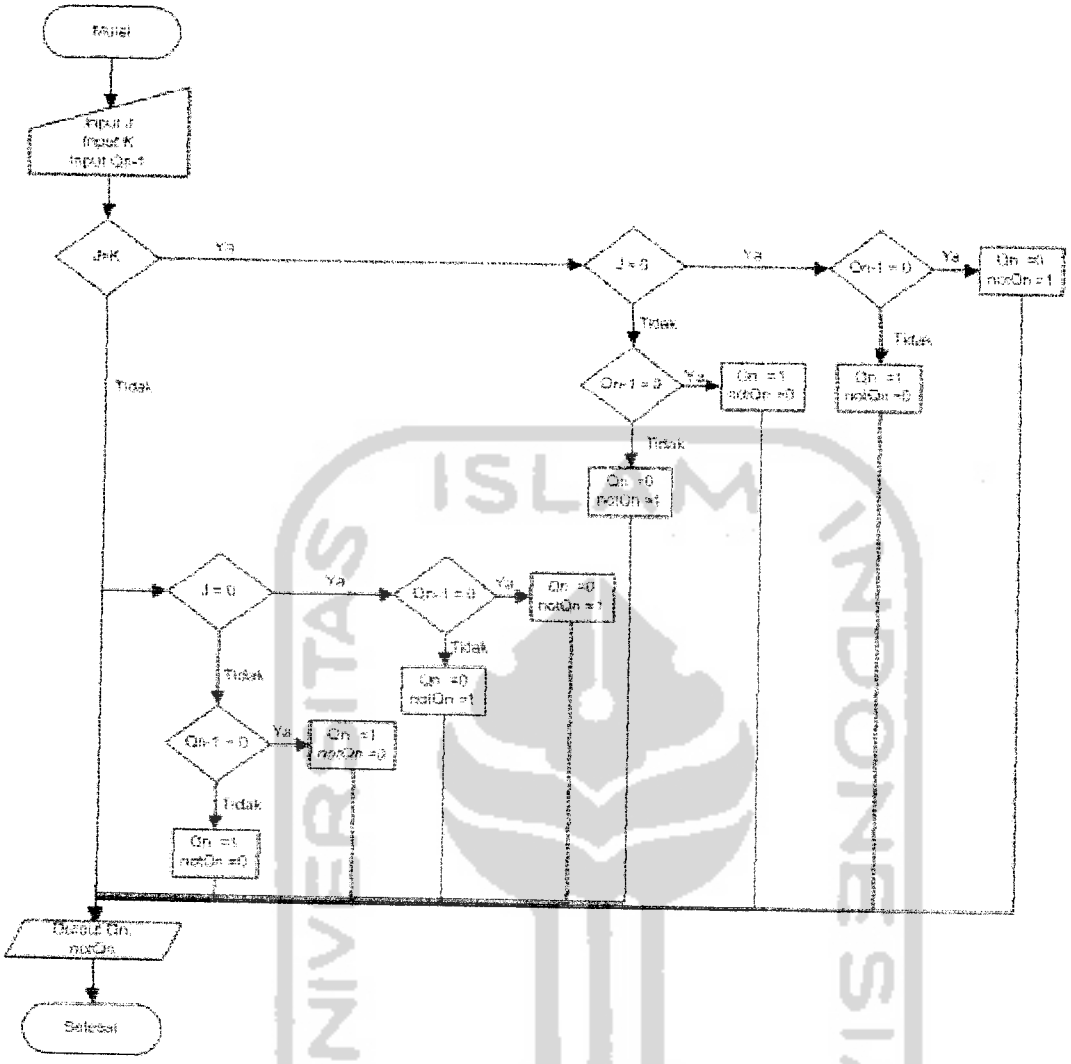
Gambar 3.14 Diagram Alir Full Adder

Adapun *output* S didapat dari operasi XOR 3-*input*, jadi : S= A *xor* B *xor* C, sedangkan untuk *output* Cn didapat dari operasi AND dan OR sehingga Cn= (A *and* B) *or* (A *and* Cp) *or* (B *and* Cp).



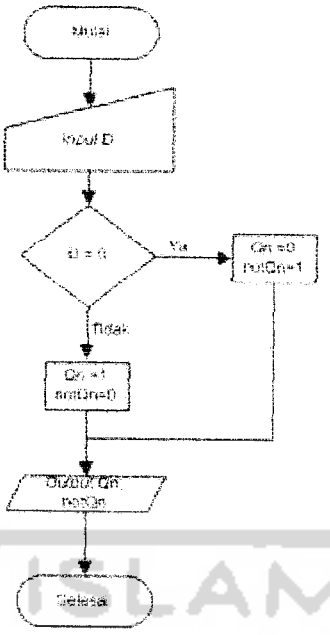
Gambar 3.15 Diagram Alir *Flip-flop Set Reset*

Qn merupakan keadaan *output* sekarang dan Qn-1 keadaaan *output* sebelumnya, maka persamaan *output flip-flop* SR yang dibangun dengan menggunakan gerbang NOR dapat dinyatakan dalam Qn= *not* (R *or*(*not*Qn-1)) dan untuk *not*Qn= *not* (S or Qn-1).



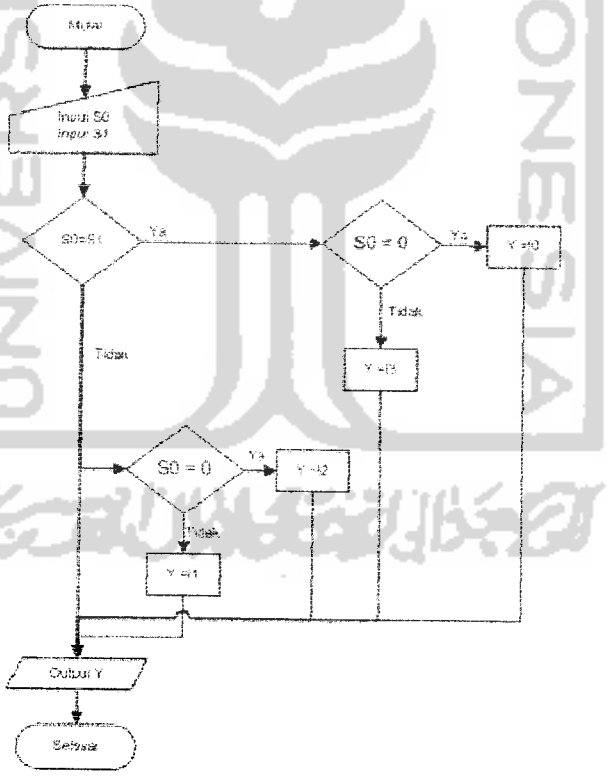
Gambar 3.16 Diagram Alir *Flip-flop* JK

Adapun persamaan Qn dan *not*Qn untuk *flip-flop* JK, adalah : Qn= (J *and* (*not*Qn-1)) *or* ((*not*K) *and* Qn-1). Dan untuk *not*Qn = ((*not*J) *and* (*not*Qn-1)) *or* (K *and* Qn-1).



Gambar 3.17 Diagram Alir *Flip-flop* D

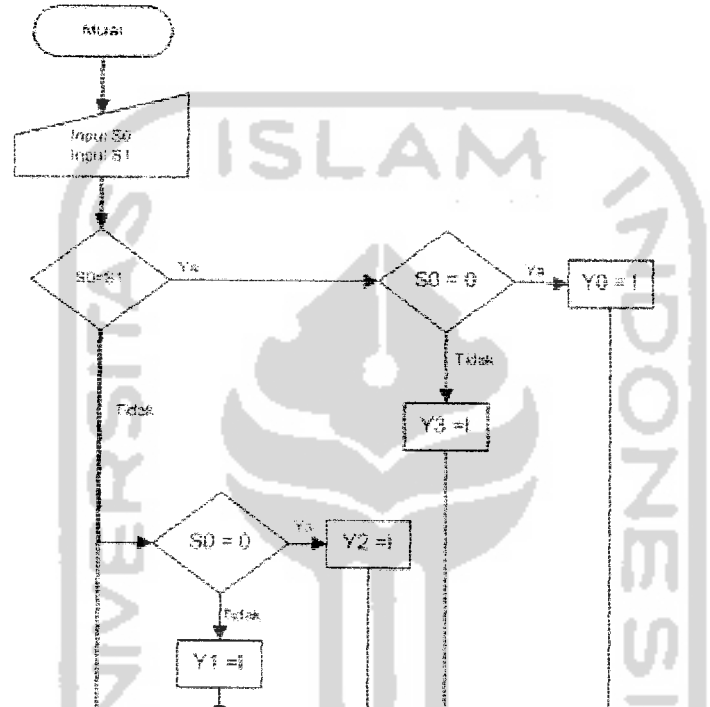
Untuk *flip-flop* D persamaannya adalah : Qn = D dan *not*Qn = *not*D.



Gambar 3.18 Diagram Alir *Multiplexer*

Multiplexer pada dasarnya adalah rangkaian logika berbentuk AND-OR. Berdasarkan tabel kebenarannya, maka dapat diperoleh persamaan *output* MUX 4 ke 1 adalah :

Y= ((*not*S1) *and* (*not*S0) *and* I0) *or* ((*not*S1) *and* S0 *and* I1) *or* (S1 *and* (*not*S0) *and* I2) *or* (S1 *and* S0 *and* I3).



Gambar 3.19 Diagram Alir *Demultiplexer*

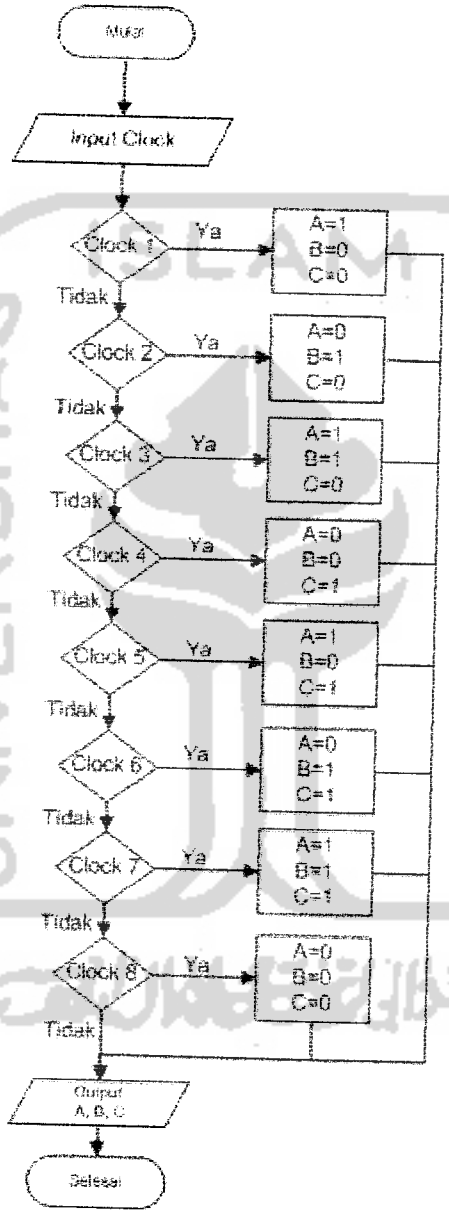
Demultiplexer pada dasarnya adalah kumpulan gerbang AND. Dengan demikian dapat diperoleh operasi AND untuk setiap *output* DEMUX 1 ke 4, sehingga untuk DEMUX 1 ke 4 persamaan *output*nya adalah:

Y0= (*not*S1) *and* (*not*S0) *and* I

Y1= (*not*S1) *and* S0 *and* I

Y2= S1 *and* (*not*S0) *and* I

Y3= S1 *and* S0 *and* I



Gambar 3.20 Diagram Alir Pencacah (*Counter*)

Cara kerja pencacah (*counter)* ini dapat dijelaskan sebagai berikut :

Anggap pada saat awal *output* pencacah ABC=000. Pada *clock* ke-1, flip-flop A terpicu sehingga A=1, dan *flip-flop* yang lain belum terpicu sehingga *output*nya masih bernilai 0. Pada keadaan ini *output* pencacah adalah ABC=100.

Pada *clock* ke-2 *output flip-flop* A akan membalik *output* sebelumnya sehingga menjadi rendah, dan perubahan ini akan memicu *flip-flop* B sehingga B=1, pada sisi lain *output* C tetap rendah karena belum terpicu. Untuk keadaan ini *output* pencacah menjadi ABC=010.

Selanjutnya pada *clock* ke-3 *flip-flop* A terpicu sehingga *output*nya berubah dari rendah ke tinggi menjadi A=1, *flip-flop* B tidak terpicu karena *output* A sebagai pemicunya berubah dari rendah ke tinggi sehingga B tetap tinggi. Demikian pula dengan *flip-flo*p C, karena belum terpicu *output*nya masih rendah sehingga untuk keadaan ini *output* pencacah menjadi ABC=110.

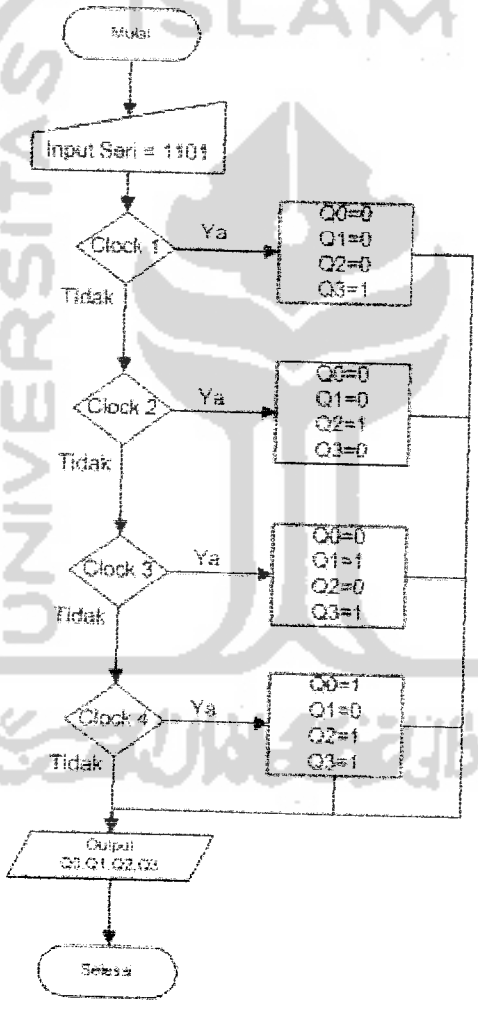
Pada *clock* ke-4 *flip-flop* A terpicu sehingga *output*nya terbalik menjadi A=0, dan perubahan ini memicu *flip-flop* B sehingga *output* B juga terbalik menjadi B=0. Karena B berubah dari tinggi ke rendah maka *output*nya memicu *flip-flop* C sehingga C berubah menjadi C=1. pada keadaan ini *output* pencacah menjadi ABC=001.

Pada *clock* ke-5, *flip-flop* A terpicu sehingga *output*nya berubah dari A=0 menjadi A=1, *flip-flop* B dan C tidak terpicu sehingga *output*nya tetap B=0 dan C=1. untuk keadaan ini *output* pencacah adalah ABC=101.

Pada *clock* ke-6, *flip-flop* A terpicu sehingga A=0, *output flip-flop* A memicu *flip-flop* B sehingga *output*nya berubah menjadi B=1, dan *flip-flop* C tidak terpicu sehingga C=1. Pada keadaan ini *output* pencacah menjadi ABC=011.

Pada *clock* ke-7, *flip-flop* A terpicu sehingga *output*nya A=1, B tetap, C tetap, dan *output* pencacah menjadi ABC=111.

Pada saat terjadinya *clock* ke-8, ketiga *flip-flop* terpicu, dan karena keadaan *output* awalnya tinggi maka akan berubah menjadi *reset*. Keadaan tersebut menyebabkan *output* pencacah menjadi ABC=000.



Gambar 3.21 Diagram Alir *Register* Geser 4-*bit*

Cara kerja *register* geser 4-*bit* dapat dijelaskan sebagai berikut :

Anggap data yang akan disimpan adalah 1101 dan keadaan mula-mula isi *register* masih kosong sehingga $Q_3Q_2Q_1Q_0$=0000. Mekanisme penyimpanan datanya dilakukan dengan memasukkan terlebih dahulu bit LSB (*Least Significant Bit*) dari data yang akan disimpan ke bagian elemen MSB (*Most Significant Bit*) *register*. Untuk memulai penyimpanan data dipasang terlebih dahulu data MSB yakni 1 pada *input* seri.

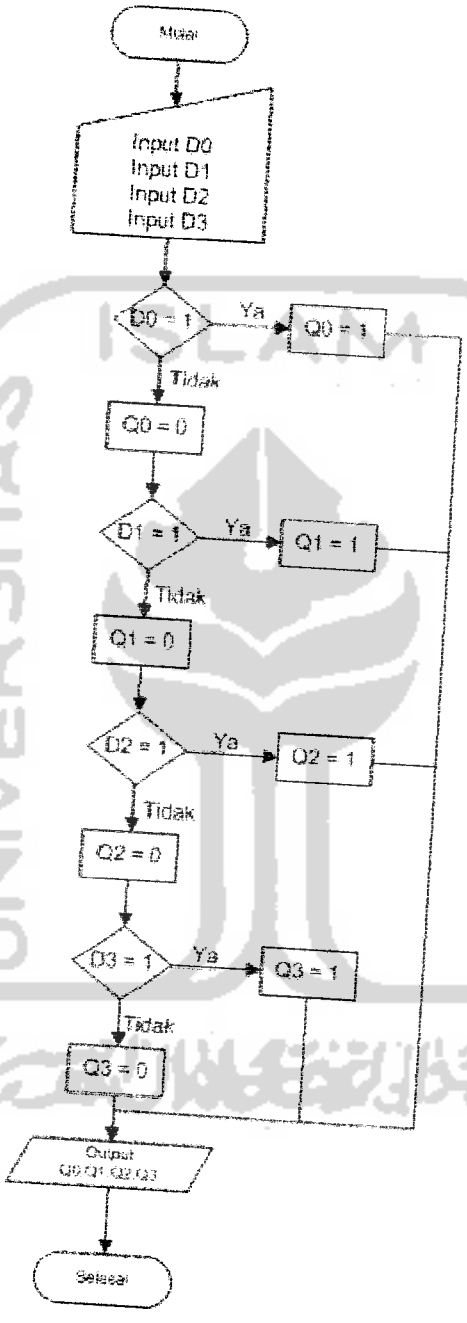
Pada pulsa *clock* ke-1 akan memicu semua *input flip-flop*. Oleh karena pada *input flip-flop* D_3 terpasang data 1 maka Q_3=1. Pada sisi lain *input flip-flop* D_2, D_1 dan D_0 bernilai 0 sehingga $Q_3Q_2Q_1Q_0$ = 1000. Sebelum ada *clock* ke-2 dipasang lagi data berikutnya yakni 0 pada *input* seri.

Saat *clock* ke-2, semua *input flip-flop* kembali terpicu. Oleh karena D_3=0 (berasal dari *input* seri), maka Q_3=0 dan Q_2=1. Pada sisi lain D_1 dan D_0 bernilai 0 sehingga $Q_3Q_2Q_1Q_0$= 0100. Keadaaan tersebut menyebabkan seolah-olah isi Q_3 digeser ke posisi Q_2.

Sebelum ada *clock* ke-3, data berikutnya yakni 1 dipasang pada *input* seri, dan saat ada *clock* ke-3 semua *input flip-flop* kembali terpicu menyebabkan Q_3=1 (berasal dari *input* seri). Q_2=0 dan Q_1=1. Pada sisi lain Q_0=0 sehingga $Q_3Q_2Q_1Q_0$=1010. Keadaan tersebut menyebabkan seolah-olah isi Q_3 digeser ke Q_2 dan isi Q_2 digeser ke Q_1.

Selanjutnya, sebelum *clock* ke-4, pada *input* seri dipasang data yang terakhir yakni 1. oleh karena keadaan sebelumnya Q_3=1, Q_2=0 dan Q_1=1 maka setelah *clock* ke-4 menjadikan Q_2=1, Q_1=0 dan Q_0=1. Pada sisi lain pemasangan data 1 pada *input* seri menyebabkan Q_3=1 sehingga $Q_3Q_2Q_1Q_0$= 1101. Keadaan ini menyebabkan seolah-olah isi Q_3 digeser ke Q_2, isi Q_2 digeser ke Q_1, dan isi Q_1 digeser ke Q_0.

Terlihat bahwa pada *register* geser 4-bit, penyimpanan data yang dilakukan memerlukan waktu sebanyak 4 siklus *clock*, sebab setelah *clock* ke-4 isi *register* sama dengan data yang dimasukkan yakni 1101.



Gambar 3.22 Diagram Alir *Register* Paralel

Cara kerja *register* paralel dapat dijelaskan sebagai berikut :

Pada *register* ini data dimasukkan ke dalamnya secara serempak melalui saluran $D_3D_2D_1D_0$. Demikian pula ketika *register* tersebut akan dibaca *output*nya, data dikeluarkan secara serempak melalui $Q_3Q_2Q_1Q_0$. Prinsip penyimpanan data pada *register* adalah memindahkan data yang ada pada *input*nya ke *output*nya. Penyimpanan data pada *register* paralel dilakukan dengan cara menempatkan data yang akan disimpan pada *input* paralel, dan untuk memindahkan data tersebut ke *output*nya dilakukan dengan memberikan sebuah pulsa *clock*.

Anggap saja *register* tersebut akan menyimpan data 1011. Mula-mula data tersebut ditempatkan pada saluran *input register* yakni $D_3D_2D_1D_0$=1011, dan saat terjadi pulsa *clock*, data dipindah ke *output* register sehingga $Q_3Q_2Q_1Q_0$ = 1011.

3.2.2.2 Perancangan Antarmuka (*interface*)

Perancangan antarmuka pada "Membangun Tabel Kebenaran Dengan Logika Matematika Untuk Aplikasi Gerbang Logika Pada Rangkaian Elektronika " ini dibuat sesederhana mungkin sehingga diharapkan pengguna dapat dengan mudah memahami berbagai *control* yang ada pada *form* tampilan. Berikut desain dari antarmuka sistem:

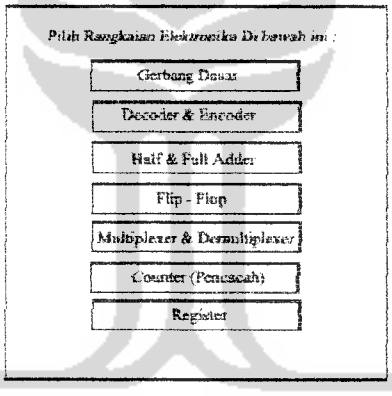
3.2.2.2.1 Antarmuka Halaman Utama

Rancangan antarmuka ini merupakan halaman utama dalam aplikasi ini dan merupakan antarmuka untuk memulai proses. Gambar 3.23 mengilustrasikan perancangan antarmuka halaman utama.

Gambar 3.23 Tampilan Antarmuka Halaman Utama

3.2.2.2.2 Antarmuka Halaman Menu

Rancangan antarmuka ini merupakan halaman dimana *user* dapat memilih rangkaian elektronika mana yang akan dituju. Gambar 3.24 mengilustrasikan perancangan antarmuka halaman menu

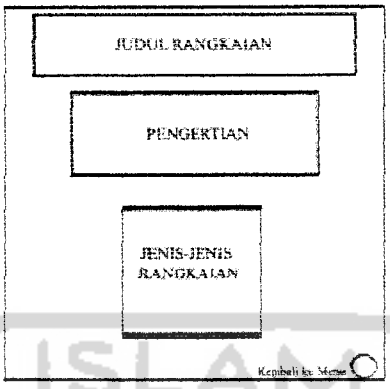


Gambar 3.24 Tampilan Antarmuka Halaman Menu

3.2.2.2.3 Antarmuka Halaman Sub Rangkaian Elektronika

Rancangan antarmuka ini merupakan halaman dimana *user* sebelum menuju halaman aplikasi, user akan disuguhkan sekilas pengertian tentang

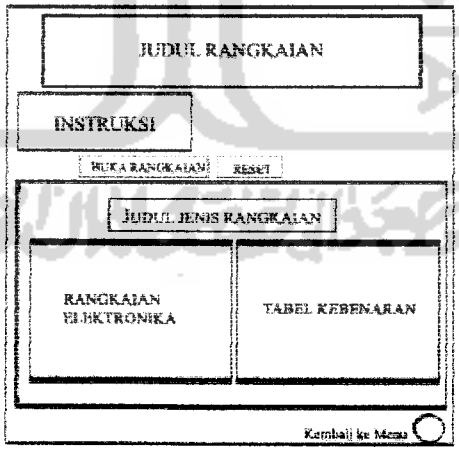
rangkaian elektronika yang dituju dan pilihan jenis-jenisnya . Gambar 3.25 mengilustrasikan perancangan antarmuka halaman sub rangkaian elektronika.



Gambar 3.25 Tampilan Antarmuka Halaman Sub Rangkaian Elektronika

3.2.2.2.4 Antarmuka Halaman Aplikasi

Rancangan antarmuka ini merupakan halaman dimana *user* berinteraksi langsung dengan rangkaian yang dipilih tadi. Disini *user* dapat mencoba mensimulasikan rangkaian elektronika yang dipilih serta dapat melihat hasilnya langsung ataupun dalam bentuk tabel kebenaran . Gambar 3.26 mengilustrasikan perancangan antarmuka halaman aplikasi.



Gambar 3.26 Tampilan Antarmuka Halaman Aplikasi

3.3 Implementasi Perangkat Lunak

Implementasi merupakan tahap dimana sistem siap dioperasikan pada keadaan sebenarnya, dari sini akan diketahui apakah sistem yang dibuat benar-benar dapat menghasilkan tujuan yang diinginkan.

Sebelum program diterapkan dan diimplementasikan, maka program harus *error free* (bebas kesalahan). Kesalahan program yang mungkin terjadi antara lain kesalahan penulisan bahasa, kesalahan waktu proses, atau kesalahan logikal. Setelah program bebas dari kesalahan, kemudian dapat dilakukan pengujian dengan menjalankan program dan memasukkan data yang akan diolah.

3.3.1 Batasan Implementasi

Pada bagian implementasi ini perangkat lunak yang sudah dibangun sebelumnya akan di uji cobakan untuk melihat apakah pembangunan perangkat lunak sudah sesuai dengan fungsi dan tujuan dari di kembangkannya perangkat lunak tersebut.

Didalam program "Membangun Tabel Kebenaran Untuk Aplikasi Gerbang Logika Pada Rangkaian Elektronika " ini pada kenyataannya terdapat beberapa batasan-batasan antara lain:

1. Rangkaian Elektronika yang kami tampilkan hanya mencakup rangkaian logika digital sederhana saja, seperti : gerbang-gerbang dasar, *decoder* dan *encoder*, *half adder* dan *full adder*, *flip-flop*, *multiplexer* dan *demultiplexer*, pencacah(*counter*) dan *register*.
2. Untuk rangkaian pencacah (counter) dan register geser, kami hanya menampilkan animasi gerakan *input* dan *output*nya saja, mengingat data yang ditampilkan bergantung pada keadaan *clock* nya.

3. Untuk rangkaian *register* baik *register* geser maupun paralel kami tidak menampilkan tabel kebenarannya karena tidak semua watak rangkaian logika bisa dijelaskan dengan tabel kebenaran, bisa juga menggunakan diagram waktu, persamaan, atau yang lainnya.

3.3.2 Implementasi Perangkat Lunak

Pada bagian implementasi perangkat lunak ini, memuat dokumentasi dan penjelasan tentang implementasi perangkat lunak yang meliputi antarmuka (*interface*) dari aplikasi yang digunakan.

3.3.2.1 Implementasi Antarmuka

Hasil dari implementasi antarmuka dari "Aplikasi Gerbang Logika Pada Rangkaian Elektronika" adalah sebagai berikut:

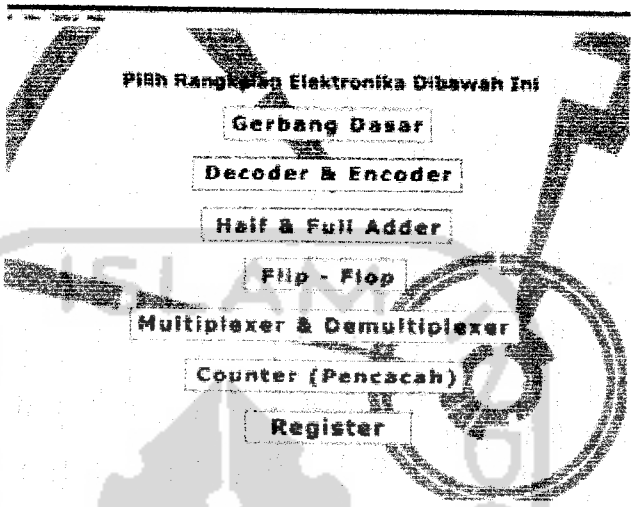
3.3.2.1.1 Halaman Utama

Halaman Utama, pada halaman ini *user* dapat melihat halaman pembuka dari program aplikasi gerbang logika ini, seperti yang tampak pada Gambar 3.27.

Gambar 3.27 Halaman Utama

3.3.2.1.2 Halaman Menu

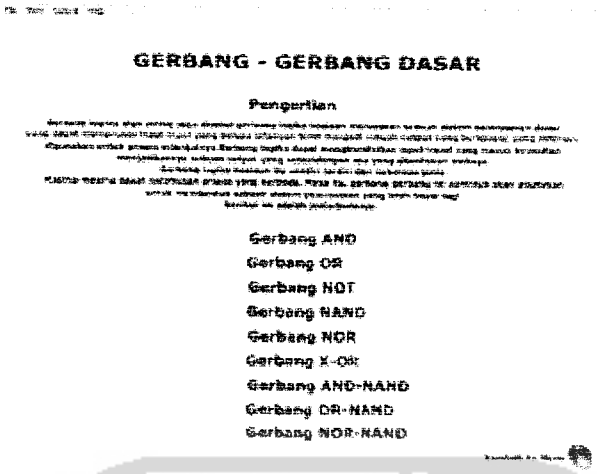
Halaman Menu, pada halaman ini *user* dapat mengakses pilihan menu-menu yang ada, seperti yang tampak pada Gambar 3.28.



Gambar 3.28 Halaman Menu

3.3.2.1.3 Halaman Gerbang – Gerbang Dasar

Halaman gerbang –gerbang dasar, pada halaman ini *user* dapat memilih jenis-jenis dari gerbang – gerbang dasar yang telah dipilih serta melihat sekilas pengertian tentang gerbang – gerbang dasar , seperti yang tampak pada Gambar 3.29.



Gambar 3.29 Halaman Gerbang-Gerbang Dasar

3.3.2.1.3.1 Gerbang AND

Setelah *user* mengakses gerbang AND maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.30

Gambar 3.30 Tampilan Halaman Gerbang AND

***Action Script* Gerbang AND :**

```
on (release) {
        Y = A & B & C ;
}
```

Penjelasan *Script* Gerbang AND:

Script diatas diletakkan pada tombol "Lihat Hasil", jika tombol "Lihat Hasil" ditekan dan dilepaskan maka *input* yang ada pada kotak A,B dan C akan di ANDkan oleh program sehingga *output* Y akan menampilkan hasil AND dari *input* A,B dan C.

3.3.2.1.3.2 Gerbang OR

Pada gerbang OR maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.31.

Gambar 3.31 Tampilan Halaman Gerbang OR

***Action Script* Gerbang OR :**

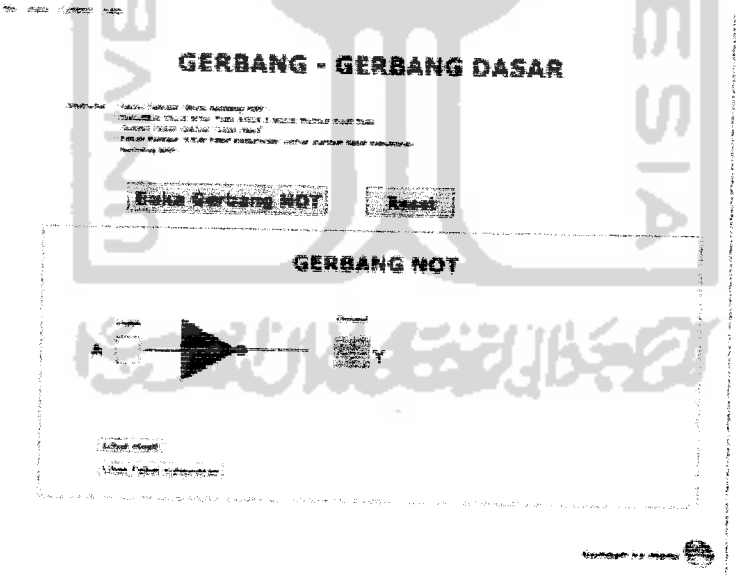
```
on (release) {
        Y = A | B | C;
}
```

Penjelasan *Script* Gerbang OR:

Script diatas diletakkan pada tombol "Lihat Hasil", jika tombol "Lihat Hasil" ditekan dan dilepaskan maka *input* yang ada pada kotak A,B dan C akan di OR-kan oleh program sehingga *output* Y akan menampilkan hasil OR dari *input* A,B dan C.

3.3.2.1.3.3 Gerbang NOT

Pada gerbang NOT maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.32.



Gambar 3.32 Tampilan Halaman Gerbang NOT

Action Script Gerbang NOT :

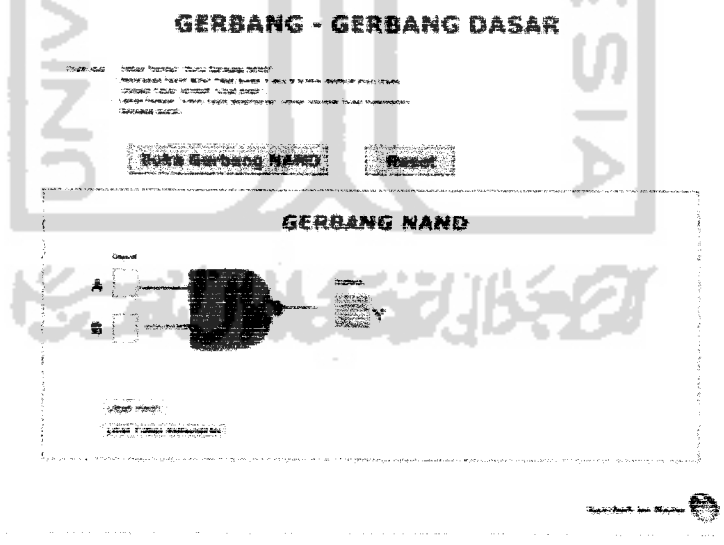
```
on (release) {
        if (A == "0") {
                Y = "1";
        } else {
                Y = "0";
        }
}
```

Penjelasan *Script* Gerbang NOT:

Script diatas diletakkan pada tombol "Lihat Hasil", Jika *input* A = 0 maka *output* Y = 1, selain dari *input* diatas *output* Y=0.

3.3.2.1.3.4 Gerbang NAND

Pada gerbang NAND maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.33.



Gambar 3.33 Tampilan Halaman Gerbang NAND

Action Script Gerbang NAND :

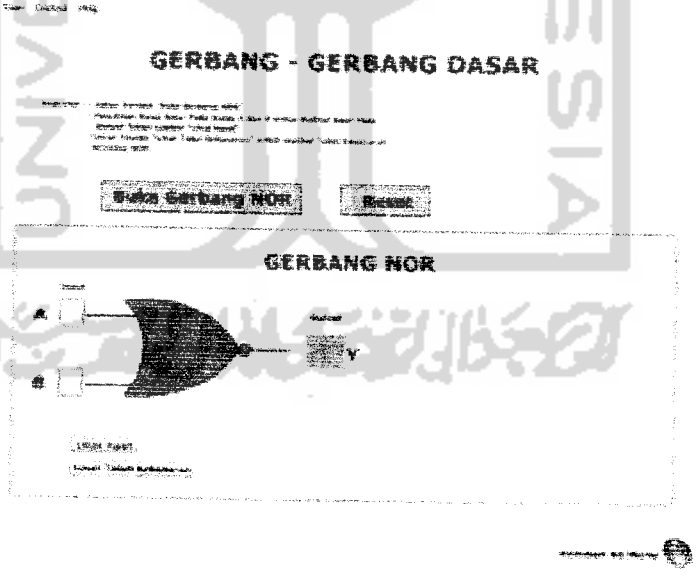
```
on (release) {
        if (A == "1" & B == "1") {
                Y = "0";
        } else {
                Y = "1";
        }
}
```

Penjelasan *Script* Gerbang NAND:

Script diatas diletakkan pada tombol "Lihat Hasil", Jika *input* A = 1 dan B=1 maka *output* Y = 0, selain dari *input* diatas *output* Y=1.

3.3.2.1.3.5 Gerbang NOR

Pada gerbang NOR maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.34.



Gambar 3.34 Tampilan Halaman Gerbang NOR

Action Script Gerbang NOR :

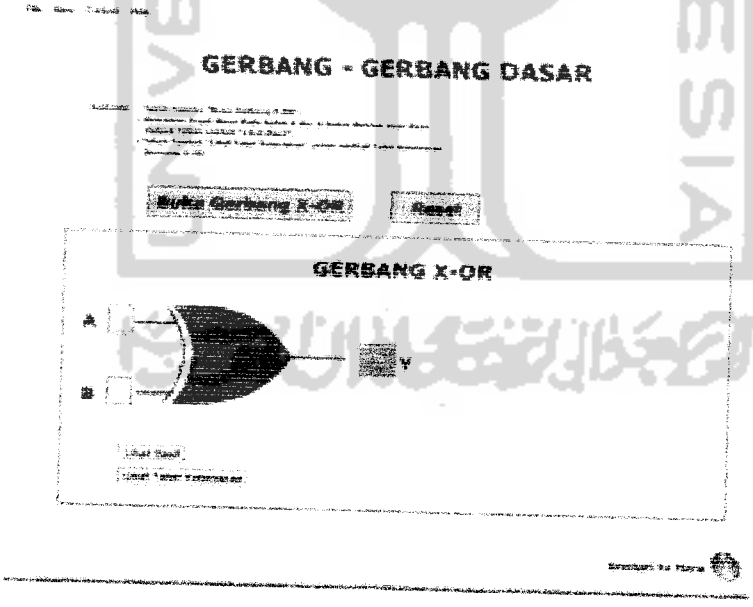
```
on (release) {
        if (A == "0" & B == "0") {
                Y = "1";
        } else {
                Y = "0";
        }
}
```

Penjelasan *Script* Gerbang NOR:

Script diatas diletakkan pada tombol "Lihat Hasil", Jika *input* A = 0 dan B=0 maka *output* Y = 1, selain dari *input* diatas *output* Y=0.

3.3.2.1.3.6 Gerbang XOR

Pada gerbang XOR maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.35.



Gambar 3.35 Tampilan Halaman Gerbang XOR

Action Script Gerbang XOR :

```
on (release) {
        Y = A ^ B ;
}
```

Penjelasan *Script* Gerbang XOR:

Script diatas diletakkan pada tombol "Lihat Hasil", jika tombol "Lihat Hasil" ditekan dan dilepaskan maka *input* yang ada pada kotak A dan B akan di XOR-kan oleh program sehingga *output* Y akan menampilkan hasil XOR dari *input* A dan B.

3.3.2.1.3.7 Gerbang AND-NAND

Pada gerbang AND-NAND maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.36.

Gambar 3.36 Tampilan Halaman AND-NAND

Action Script Gerbang AND-NAND:

```
on (release) {
        if (A=="0" & B== "0" & C== "0" & D== "0") {
                P="0", Q="0", Y="1";
                stop();
        }
}
//
on (release) {
        if (A=="0" & B== "0" & C== "0" & D== "1") {
                P="0", Q="0", Y="1";
                stop();
        }
}
//
on (release) {
        if (A=="0" & B== "0" & C== "1" & D== "0") {
                P="0", Q="0", Y="1";
                stop();
        }
}
//
on (release) {
        if (A=="0" & B== "0" & C== "1" & D== "1") {
                P="0", Q="1", Y="1";
```

```
            stop();
        }
}
//
on (release) {
        if (A=="0" & B== "1" & C== "0" & D== "0") {
            P="0", Q="0", Y="1";
            stop();
        }
}
//
on (release) {
        if (A=="0" & B== "1" & C== "0" & D== "1") {
            P="0", Q="0", Y="1";
            stop();
        }
}
//
on (release) {
        if (A=="0" & B== "1" & C== "1" & D== "0") {
            P="0", Q="0", Y="1";
            stop();
        }
}
//
on (release) {
        if (A=="0" & B== "1" & C== "1" & D== "1") {
            P="0", Q="1", Y="1";
```

```
            stop();
        }
    }
    //
    on (release) {
        if (A=="1" & B== "0" & C== "0" & D== "0") {
            P="0", Q="0", Y="1";
            stop();
        }
    }
    //
    on (release) {
        if (A=="1" & B== "0" & C== "0" & D== "1") {
            P="0", Q="0", Y="1";
            stop();
        }
    }
    //
    on (release) {
        if (A=="1" & B== "0" & C== "1" & D== "0") {
            P="0", Q="0", Y="1";
            stop();
        }
    }
    //
    on (release) {
        if (A=="1" & B== "0" & C== "1" & D== "1") {
            P="0", Q="1", Y="1";
```

```
            stop();
        }
}
//
on (release) {
        if (A=="1" & B== "1" & C== "0" & D== "0") {
            P="1", Q="0", Y="1";
            stop();
        }
}
//
on (release) {
        if (A=="1" & B== "1" & C== "0" & D== "1") {
            P="1", Q="0", Y="1";
            stop();
        }
}
//
on (release) {
        if (A=="1" & B== "1" & C== "1" & D== "0") {
            P="1", Q="0", Y="1";
            stop();
        }
}
//
on (release) {
        if (A=="1" & B== "1" & C== "1" & D== "1") {
            P="1", Q="1", Y="0";
```

```
            stop();
        }
}
```

Penjelasan *Script* Gerbang AND-NAND:

Script diatas diletakkan pada tombol "Lihat Hasil", jika *input* A=0, B=0, C=0 dan D=0 maka *output* Y = 1 sedangkan P dan Q adalah *output* dari 2 gerbang AND untuk selanjutnya dijadikan *input* pada gerbang NAND.

Untuk kombinasi *input-input* yang ada penjelasannya sama seperti untuk *input* ABCD=0000 diatas.

3.3.2.1.3.8 Gerbang OR-NAND

Pada gerbang OR-NAND maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.37.

Gambar 3.37 Tampilan Halaman OR-NAND

Action Script Gerbang OR-NAND:

```
on (release) {
	if (A=="0" & B== "0" & C== "0" & D== "0") {
		P="0", Q="0", Y="1";
		stop();
	}
}
on (release) {
	if (A=="0" & B== "0" & C== "0" & D== "1") {
		P="0", Q="1", Y="1";
		stop();
	}
}
on (release) {
	if (A=="0" & B== "0" & C== "1" & D== "0") {
		P="0", Q="1", Y="1";
		stop();
	}
}
on (release) {
	if (A=="0" & B== "0" & C== "1" & D== "1") {
		P="0", Q="1", Y="1";
		stop();
	}
}

on (release) {
```

```
        if (A=="0" & B== "1" & C== "0" & D== "0") {
                P="1", Q="0", Y="1";
                stop();
        }
}
on (release) {
        if (A=="0" & B== "1" & C== "0" & D== "1") {
                P="1", Q="1", Y="0";
                stop();
        }
}
on (release) {
        if (A=="0" & B== "1" & C== "1" & D== "0") {
                P="1", Q="1", Y="0";
                stop();
        }
}
on (release) {
        if (A=="0" & B== "1" & C== "1" & D== "1") {
                P="1", Q="1", Y="0";
                stop();
        }
}
on (release) {
        if (A=="1" & B== "0" & C== "0" & D== "0") {
                P="1", Q="0", Y="1";
                stop();
        }
```

```
}
on (release) {
      if (A=="1" & B== "0" & C== "0" & D== "1") {
            P="1", Q="1", Y="0";
            stop();
      }
}
on (release) {
      if (A=="1" & B== "0" & C== "1" & D== "0") {
            P="1", Q="1", Y="0";
            stop();
      }
}
on (release) {
      if (A=="1" & B== "0" & C== "1" & D== "1") {
            P="1", Q="1", Y="0";
            stop();
      }
}
on (release) {
      if (A=="1" & B== "1" & C== "0" & D== "0") {
            P="1", Q="0", Y="1";
            stop();
      }
}
on (release) {
      if (A=="1" & B== "1" & C== "0" & D== "1") {
            P="1", Q="1", Y="0";
```

```
            stop();
        }
}
on (release) {
        if (A=="1" & B== "1" & C== "1" & D== "0") {
            P="1", Q="1", Y="0";
            stop();
        }
}
on (release) {
        if (A=="1" & B== "1" & C== "1" & D== "1") {
            P="1", Q="1", Y="0";
            stop();
        }
}
```

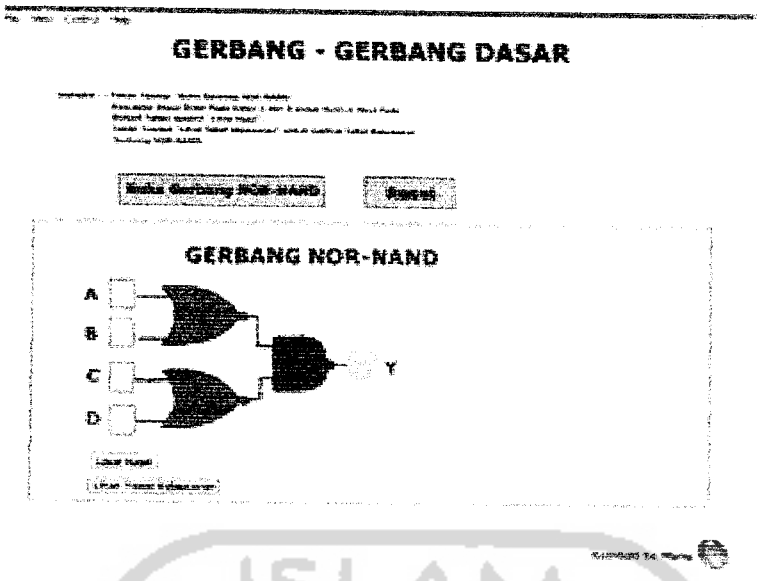
Penjelasan *Script* Gerbang OR-NAND :

Script diatas diletakkan pada tombol "Lihat Hasil", jika *input* A=0, B=0, C=0 dan D=0 maka *output* Y = 1 sedangkan P dan Q adalah *output* dari 2 gerbang OR untuk selanjutnya dijadikan *input* pada gerbang NAND.

Untuk kombinasi *input-input* yang ada penjelasannya sama seperti untuk *input* ABCD=0000 diatas.

3.3.2.1.3.9. Gerbang NOR-NAND

Pada gerbang OR-NAND maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.38.



Gambar 3.38 Tampilan Halaman NOR-NAND

Action Script Gerbang NOR-NAND:

```
on (release) {
      if (A=="0" & B== "0" & C== "0" & D== "0") {
            P="1", Q="1", Y="0";
            stop();
      }
}
on (release) {
      if (A=="0" & B== "0" & C== "0" & D== "1") {
            P="1", Q="0", Y="1";
            stop();
      }
}
on (release) {
      if (A=="0" & B== "0" & C== "1" & D== "0") {
            P="1", Q="0", Y="1";
            stop();
```

```
        }
}
on (release) {
        if (A=="0" & B== "0" & C== "1" & D== "1") {
                P="1", Q="0", Y="1";
                stop();
        }
}

on (release) {
        if (A=="0" & B== "1" & C== "0" & D== "0") {
                P="0", Q="0", Y="1";
                stop();
        }
}
on (release) {
        if (A=="0" & B== "1" & C== "0" & D== "1") {
                P="0", Q="0", Y="1";
                stop();
        }
}
on (release) {
        if (A=="0" & B== "1" & C== "1" & D== "0") {
                P="0", Q="0", Y="1";
                stop();
        }
}
on (release) {
```

```
        if (A=="0" & B== "1" & C== "1" & D== "1") {
                P="0", Q="0", Y="1";
                stop();
        }
}
on (release) {
        if (A=="1" & B== "0" & C== "0" & D== "0") {
                P="0", Q="1", Y="1";
                stop();
        }
}
on (release) {
        if (A=="1" & B== "0" & C== "0" & D== "1") {
                P="0", Q="0", Y="1";
                stop();
        }
}
on (release) {
        if (A=="1" & B== "0" & C== "1" & D== "0") {
                P="0", Q="0", Y="1";
                stop();
        }
}
on (release) {
        if (A=="1" & B== "0" & C== "1" & D== "1") {
                P="0", Q="0", Y="1";
                stop();
        }
```

```
}
on (release) {
        if (A=="1" & B== "1" & C== "0" & D== "0") {
                P="0", Q="1", Y="1";
                stop();
        }
}
on (release) {
        if (A=="1" & B== "1" & C== "0" & D== "1") {
                P="0", Q="0", Y="1";
                stop();
        }
}
on (release) {
        if (A=="1" & B== "1" & C== "1" & D== "0") {
                P="0", Q="0", Y="1";
                stop();
        }
}
on (release) {
        if (A=="1" & B== "1" & C== "1" & D== "1") {
                P="0", Q="0", Y="1";
                stop();
        }
}
```

Penjelasan *Script* Gerbang NOR-NAND :

Script diatas diletakkan pada tombol "Lihat Hasil", jika *input* A=0, B=0, C=0 dan D=0 maka *output* Y = 1 sedangkan P dan Q adalah *output* dari 2 gerbang NOR untuk selanjutnya dijadikan *input* pada gerbang NAND.

Untuk kombinasi *input-input* yang ada penjelasannya sama seperti untuk *input* ABCD=0000 diatas.

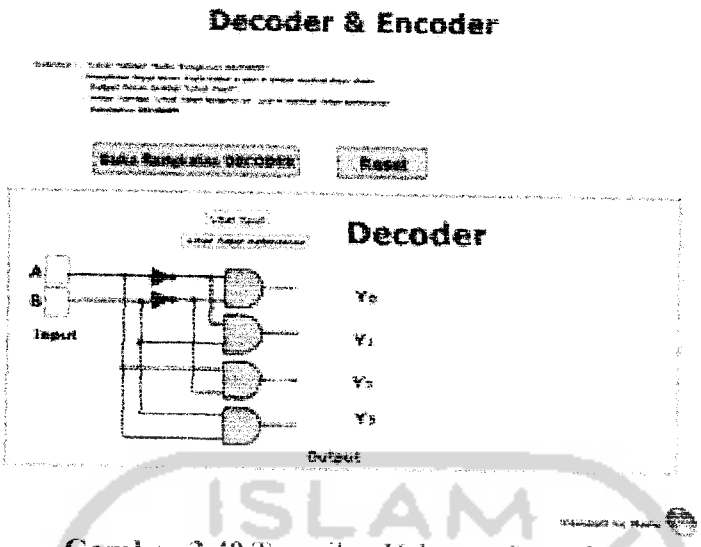
3.3.2.1.4 Halaman *Decoder* dan *Encoder*

Halaman *Decoder* dan *Encoder*, pada halaman ini *user* dapat memilih jenis-jenis dari *Decoder* dan *Encoder* yang telah dipilih serta melihat sekilas pengertian tentang *Decoder* dan *Encoder*, seperti yang tampak pada Gambar 3.39.

Gambar 3.39 Tampilan Halaman Menu *Decoder* dan *Encoder*

3.3.2.1.4.1 *Decoder*

Pada rangkaian *decoder* maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.40.



Gambar 3.40 Tampilan Halaman *Decoder*

Action Script Rangkaian *Decoder* :

```
on (release) {
        if (A=="0" & B== "0" ) {
                N1="1", N2="1", A1="1",A2="1", A3="1", A4="0",
                A5="0", A6="1", A7="0",A8="0", Y0="1", Y1="0",
                Y2="0", Y3="0";
                stop();
        }
}
//
on (release) {
        if (A=="0" & B== "1" ) {
                N1="1", N2="0", A1="1",A2="0", A3="1", A4="1",
                A5="0", A6="0", A7="1",A8="0", Y0="0", Y1="1",
                Y2="0", Y3="0";
                stop();
        }
```

```
}
//
on (release) {
      if (A=="1" & B== "0" ) {
            N1="0", N2="1", A1="0",A2="1", A3="0", A4="0",
            A5="1", A6="1", A7="0",A8="1", Y0="0", Y1="0",
            Y2="1", Y3="0";
            stop();
      }
}
//
on (release) {
      if (A=="1" & B== "1" ) {
            N1="0", N2="0", A1="0",A2="0", A3="0", A4="1",
            A5="1", A6="0", A7="1",A8="1", Y0="0", Y1="0",
            Y2="0", Y3="1";
            stop();      }
}
```

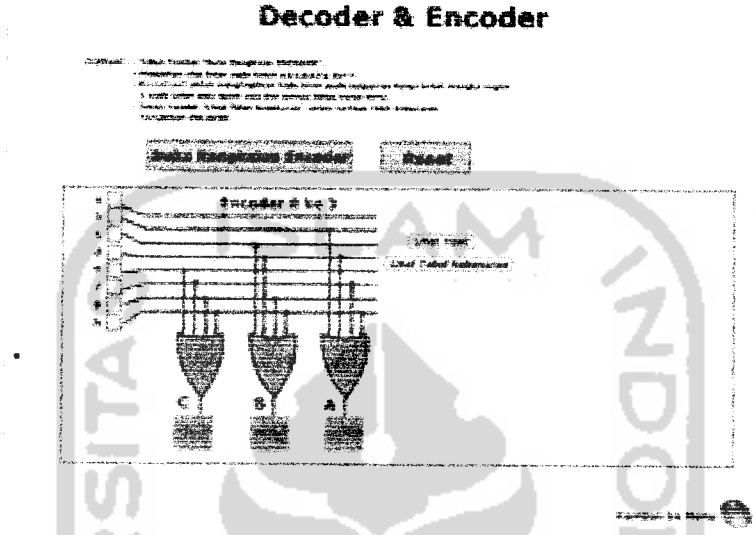
Penjelasan *Script* Rangkaian *Decoder* :

Script diatas diletakkan pada tombol "Lihat Hasil", jika *input* A=0 dan B=0, maka *output* Y0=1, Y1=0, Y3=0 dan Y4=0 sedangkan N1 dan N2 adalah *output* dari 2 gerbang NOT, dan A1-A8 adalah alir angka biner yang keluar dari *input* AB untuk selanjutnya dijadikan *input* pada 4 gerbang AND.

Untuk kombinasi *input-input* yang ada, penjelasannya sama seperti untuk *input* AB=0000 diatas.

3.3.2.1.4.2 *Encoder*

Pada rangkaian *encoder* maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.41.



Gambar 3.41 Tampilan Halaman *Encoder*

Action Script Rangkaian *Encoder* :

```
        on (release) {
        if (a=="1" & b== "0" & c=="0" & d== "0"& e=="0" &
            f== "0" & g=="0"& h=="0" ) {
              C1="0", C2="0", C3="0",C4="0", B1="0", B2="0",
              B3="0", B4="0", A1="0",A2="0", A3="0", A4="0",
              output1="0", output2="0", output3="0";
              stop();
              }
        }
//
on (release) {
```

```
        if (a=="0" & b== "1" & c=="0" & d== "0"& e=="0" & f== "0" &
        g=="0"& h=="0" ) {

                C1="0", C2="0", C3="0",C4="0", B1="0", B2="0",

                B3="0", B4="0", A1="1",A2="0", A3="0", A4="0",

                output1="0", output2="0", output3="1";

                stop();

        }

}

//

on (release) {

        if (a=="0" & b== "0" & c=="1" & d== "0"& e=="0" & f== "0" &
g=="0"& h=="0" ) {

                C1="0", C2="0", C3="0",C4="0", B1="1", B2="0",

                B3="0", B4="0", A1="0",A2="0", A3="0", A4="0",

                output1="0", output2="1", output3="0";

                stop();

        }

}

//

on (release) {

        if (a=="0" & b== "0" & c=="0" & d== "1"& e=="0" & f== "0" &
g=="0"& h=="0" ) {

                C1="0", C2="0", C3="0",C4="0", B1="0", B2="1",

                B3="0", B4="0", A1="0",A2="1", A3="0", A4="0",

                output1="0", output2="1", output3="1";

                stop();

        }

}

//
```

```
on (release) {

      if (a=="0" & b== "0" & c=="0" & d== "0"& e=="1" & f== "0" &
g=="0" & h=="0") {

            C1="1", C2="0", C3="0",C4="0", B1="0", B2="0",

            B3="0", B4="0", A1="0",A2="0", A3="0", A4="0",

            output1="1", output2="0", output3="0";

            stop();

      }

}

//

on (release) {

      if (a=="0" & b== "0" & c=="0" & d== "0"& e=="0" & f== "1" &
g=="0" & h=="0") {

            C1="0", C2="1", C3="0",C4="0", B1="0", B2="0",

            B3="0", B4="0", A1="0",A2="0", A3="1", A4="0",

            output1="1", output2="0", output3="1";

            stop();

      }

}

//

on (release) {

      if (a=="0" & b== "0" & c=="0" & d== "0"& e=="0" & f== "0" &
g=="1" & h=="0") {

            C1="0", C2="0", C3="1",C4="0", B1="0", B2="0",

            B3="1", B4="0", A1="0",A2="0", A3="0", A4="0",

            output1="1", output2="1", output3="0";

            stop();

      }

}
```

```
//
on (release) {
      if (a=="0" & b== "0" & c=="0" & d== "0"& e=="0" & f== "0" &
g=="0" & h=="1" ) {
            C1="0", C2="0", C3="0",C4="1", B1="0", B2="0",
            B3="0", B4="1", A1="0",A2="0", A3="0", A4="1",
            output1="1", output2="1", output3="1";
            stop();
      }
}
```

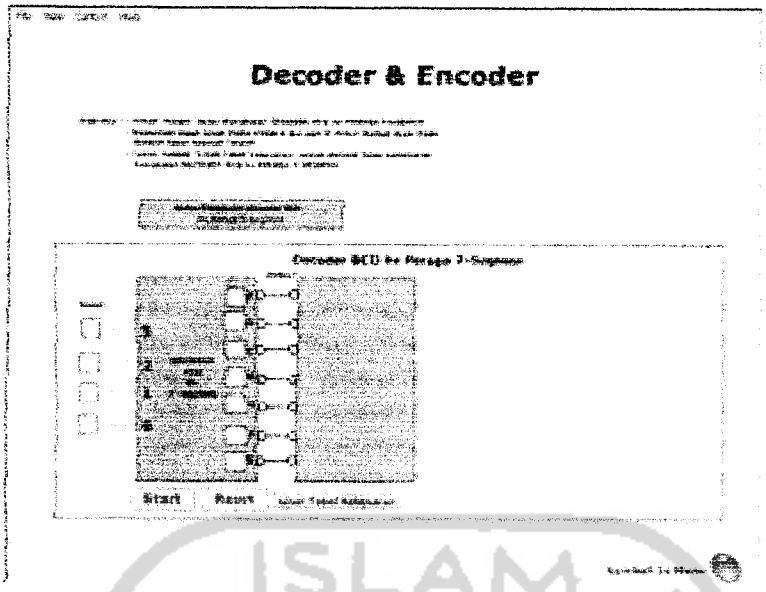
Penjelasan *Script* Rangkaian *Encoder* :

Script diatas diletakkan pada tombol "Lihat Hasil", jika *input* a=0,b=0,c=0, d=0,e=0,f=0,g=0,h=0 maka *output1*(A)=0, *output2*(B)=0, dan *output3*(C)=0 sedangkan A1-A4, B1-B4 dan C1-C4 adalah alir angka biner yang keluar dari *input* abcdefgh untuk selanjutnya dijadikan *input* pada 3 gerbang OR yang masing-masing mempunyai 4 kaki.

Untuk kombinasi *input-input* yang ada, penjelasannya sama seperti untuk *input* abcdefgh=00000000 diatas.

3.3.2.1.4.3 *Decoder* BCD ke Peraga 7 Segmen

Pada rangkaian *Decoder* BCD ke Peraga 7 Segmen maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.42.



Gambar 3.42 Tampilan Halaman *Decoder* BCD ke Peraga 7 Segmen

Action Script Rangkaian *Decoder* BCD ke Peraga 7 Segmen :

```
on (release) {

      if (input1 == "0" & input2 == "0" & input3 == "0" & input4 ==
"0") {

            output1="1", output2="1", output3="1", output4="1",
            output5="1", output6="1", output7="0";

            loadMovie("nol.swf", "sasaran");

            stop();

      }

}

//

on (release) {

      if (input1 == "1" & input2 == "0" & input3 == "0" & input4 ==
"0") {

            output1="1", output2="1", output3="1", output4="1",
            output5="1", output6="1", output7="1";

            loadMovie("delapan.swf", "sasaran");

            stop();

      }
```

```
}

//

on (release) {

      if (input1 == "0" & input2 == "1" & input3 == "1" & input4 ==
"1") {

            output1="1", output2="1", output3="1", output4="0",
            output5="0", output6="0", output7="0";

            loadMovie("tujuh.swf", "sasaran");

            stop();

      }

}

//

on (release) {

      if (input1 == "1" & input2 == "0" & input3 == "0" & input4 ==
"1") {

            output1="1", output2="1", output3="1", output4="0",
            output5="0", output6="1", output7="0";

            loadMovie("sembilan.swf", "sasaran");

            stop();

      }

}

//

on (release) {

      if (input1 == "0" & input2 == "1" & input3 == "1" & input4 ==
"0") {

            output1="0", output2="0", output3="1", output4="1",
            output5="1", output6="1", output7="1";

            loadMovie("enam.swf", "sasaran");

            stop();

      }

}
```

```
//
on (release) {
      if (input1 == "0" & input2 == "1" & input3 == "0" & input4 ==
"1") {
            output1="1", output2="0", output3="1", output4="1",
            output5="0", output6="1", output7="1";
            loadMovie("lima.swf", "sasaran");
            stop();
      }
}
//
on (release) {
      if (input1 == "0" & input2 == "1" & input3 == "0" & input4 ==
"0") {
            output1="0", output2="1", output3="1", output4="0",
            output5="0", output6="1", output7="1";
            loadMovie("empat.swf", "sasaran");
            stop();
      }
}
//
on (release) {
      if (input1 == "0" & input2 == "0" & input3 == "1" & input4 ==
"1") {
            output1="1", output2="1", output3="1", output4="1",
            output5="0", output6="0", output7="1";
            loadMovie("tiga.swf", "sasaran");
            stop();
      }
}
//
```

```
on (release) {

      if (input1 == "0" & input2 == "0" & input3 == "1" & input4 ==
"0") {

            output1="1", output2="1", output3="0", output4="1",
            output5="1", output6="0", output7="1";

            loadMovie("dua.swf", "sasaran");

            stop();

      }

}

//

on (release) {

      if (input1 == "0" & input2 == "0" & input3 == "0" & input4 ==
"1") {

            output1="0", output2="1", output3="1", output4="0",
            output5="0", output6="0", output7="0";

            loadMovie("satu.swf", "sasaran");

            stop();

      }

}

//

on (release) {

      if (input1 == "1" & input2 == "0" & input3 == "1" & input4 ==
"0") {

            output1="0", output2="0", output3="0", output4="1",
            output5="1", output6="0", output7="1";

            loadMovie("gakjelas1.swf", "sasaran");

            stop();

      }

}

//

on (release) {
```

```
        if (input1 == "1" & input2 == "0" & input3 == "1" & input4 ==
"1") {

                output1="0", output2="0", output3="1", output4="1",
                output5="0", output6="0", output7="1";

                loadMovie("gakjelas2.swf", "sasaran");

                stop();

        }

}

//

on (release) {

        if (input1 == "1" & input2 == "1" & input3 == "0" & input4 ==
"0") {

                output1="0", output2="1", output3="0", output4="0",
                output5="0", output6="1", output7="1";

                loadMovie("gakjelas3.swf", "sasaran");

                stop();

        }

}

//

on (release) {

        if (input1 == "1" & input2 == "1" & input3 == "0" & input4 ==
"1") {

                output1="1", output2="0", output3="0", output4="1",
                output5="0", output6="1", output7="1";

                loadMovie("gakjelas4.swf", "sasaran");

                stop();

        }

}

//

on (release) {
```

```
        if (input1 == "1" & input2 == "1" & input3 == "1" & input4 ==
"0") {

                output1="0", output2="0", output3="0", output4="1",
                output5="1", output6="1", output7="1";

                loadMovie("gakjelas5.swf", "sasaran");

                stop();

        }

}

//

on (release) {

        if (input1 == "1" & input2 == "1" & input3 == "1" & input4 ==
"1") {

                output1="0", output2="0", output3="0", output4="0",
                output5="0", output6="0", output7="0";

                loadMovie("kosong.swf", "sasaran");

                stop();

        }

}
```

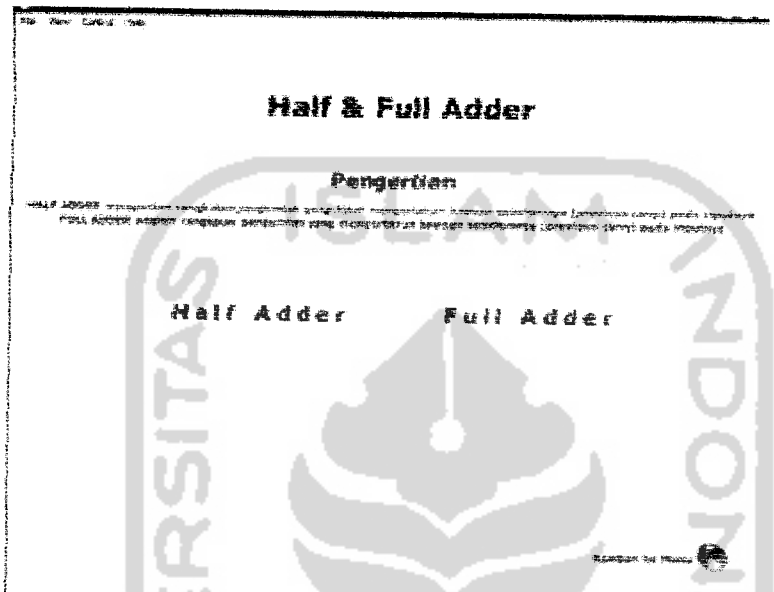
Penjelasan *Script* Rangkaian *Decoder* BCD ke Peraga 7 Segmen :

Script diatas diletakkan pada tombol "Start", jika *input*1=0, *input*2=0, *input*3=0, dan *input*4=0 maka *output*1(a)=1, *output*2(b)=1, *output*3(c)=1, *output*4(d)=1, *output*5(e)=1, *output*6(f)=1 dan *output*7(g)=0 lalu program akan memanggil movie "nol.swf"

Untuk kombinasi *input-input* yang ada, penjelasannya sama seperti untuk *input* 1234=0000diatas.

3.3.2.1.5 Halaman *Half* dan *Full Adder*

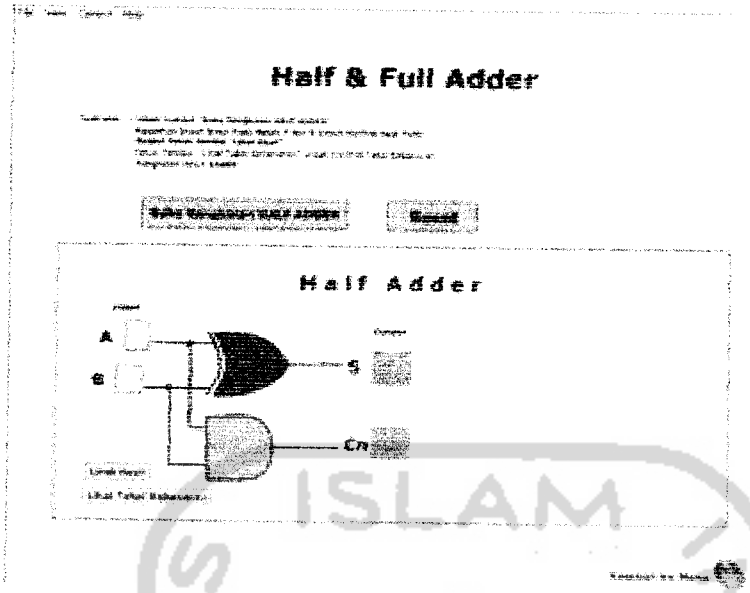
Halaman *half dan full adder*, pada halaman ini *user* dapat mengakses rangkaian dari *half dan full adder* yang telah dipilih serta melihat sekilas pengertian tentang *half dan full adder*. Seperti yang ditunjukkan pada gambar 3.43.



Gambar 3.43 Tampilan Halaman Menu *Half* dan *Full Adder*

3.3.2.1.5.1 *Half Adder*

Pada rangkaian *half adder* maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.44.



Gambar 3.44 Tampilan Halaman *Half Adder*

Action Script Rangkaian *Half Adder* :

```
on (release) {
        if (A=="0" & B== "0" ) {
                P="0", Q="0", R="0",S="0", output1="0", output2="0";
                stop();
        }
}
//
on (release) {
        if (A=="0" & B=="1" ) {
                P="0", Q="1", R="0",S="1", output1="1", output2="0";
                stop();
        }
}
//
on (release) {
```

```
        if (A=="1" & B== "0" ) {
                P="1", Q="0", R="1",S="0", output1="1", output2="0";
                stop();
        }
}
//
on (release) {
        if (A=="1" & B== "1" ) {
                P="1", Q="1", R="1",S="1", output1="0", output2="1";
                stop();
        }
}
```

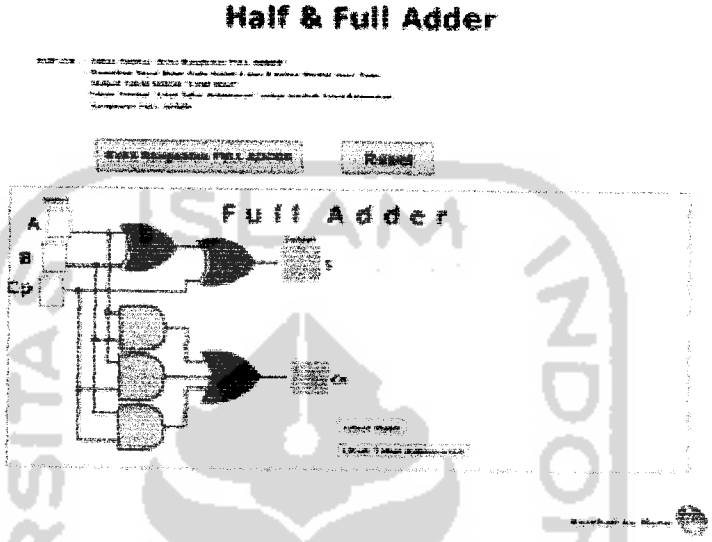
Penjelasan *Script* Rangkaian *Half Adder* :

Script diatas diletakkan pada tombol "Lihat Hasil", jika *input* A=0 dan B=0 maka *output1*(Cn)=0, *output2*(S)=0 sedangkan P, Q, R dan S adalah alir angka biner yang keluar dari *input* A dan B untuk selanjutnya dijadikan *input* pada gerbang XOR dan AND.

Untuk kombinasi *input-input* yang ada, penjelasannya sama seperti untuk *input* AB=00 diatas.

3.3.2.1.5.2 *Full Adder*

Pada rangkaian *full adder* maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.45.



Gambar 3.45 Tampilan Halaman *Full Adder*

Action Script Rangkaian *Full Adder* :

```
on (release) {
      if (A=="0" & B== "0" & C=="0" ) {
            X1="0", X2="0", X3="0",X4="0", A1="0", A2="0",
            A3="0", A4="0", A5="0",A6="0", O1="1", O2="1",
            O3="1",      output1="0", output2="0";
            stop();
      }
}
//
on (release) {
      if (A=="0" & B== "0" & C=="1" ) {
```

```
            X1="0", X2="0", X3="0",X4="1", A1="0", A2="0",
            A3="1", A4="0", A5="1",A6="0", O1="1", O2="1",
            O3="1",     output1="0", output2="1";
            stop();
      }
}
//
on (release) {
      if (A=="0" & B== "1" & C=="0" ) {
            X1="0", X2="1", X3="0",X4="0", A1="0", A2="1",
            A3="0", A4="1", A5="0",A6="0", O1="1", O2="1",
            O3="1",     output1="0", output2="1";
            stop();
      }
}
//
on (release) {
      if (A=="0" & B== "1" & C=="1" ) {
            X1="0", X2="1", X3="0",X4="1", A1="0", A2="1",
            A3="1", A4="1", A5="1",A6="0", O1="1", O2="0",
            O3="1",     output1="1", output2="0";
            stop();
      }
}
//
on (release) {
      if (A=="1" & B== "0" & C=="0" ) {
            X1="1", X2="0", X3="0",X4="0", A1="1", A2="0",
```

```
            A3="0", A4="0", A5="0",A6="1", O1="1", O2="1",
            O3="1",     output1="0", output2="1";
            stop();
        }
}
//
on (release) {
        if (A=="1" & B== "0" & C=="1" ) {
            X1="1", X2="0", X3="0",X4="1", A1="1", A2="0",
            A3="1", A4="0", A5="1",A6="1", O1="1", O2="1",
            O3="0",     output1="1", output2="0";
            stop();
        }
}
//
on (release) {
        if (A=="1" & B== "1" & C=="0" ) {
            X1="1", X2="1", X3="0",X4="0", A1="1", A2="1",
            A3="0", A4="1", A5="0",A6="1", O1="0", O2="1",
            O3="1",     output1="1", output2="0";
            stop();
        }
}
//
on (release) {
        if (A=="1" & B== "1" & C=="1" ) {
            X1="1", X2="1", X3="1",X4="1", A1="1", A2="1",
            A3="1", A4="1", A5="1",A6="1", O1="0", O2="0",
```

```
            O3="0",      output1="1", output2="1";
            stop();
        }
}
```

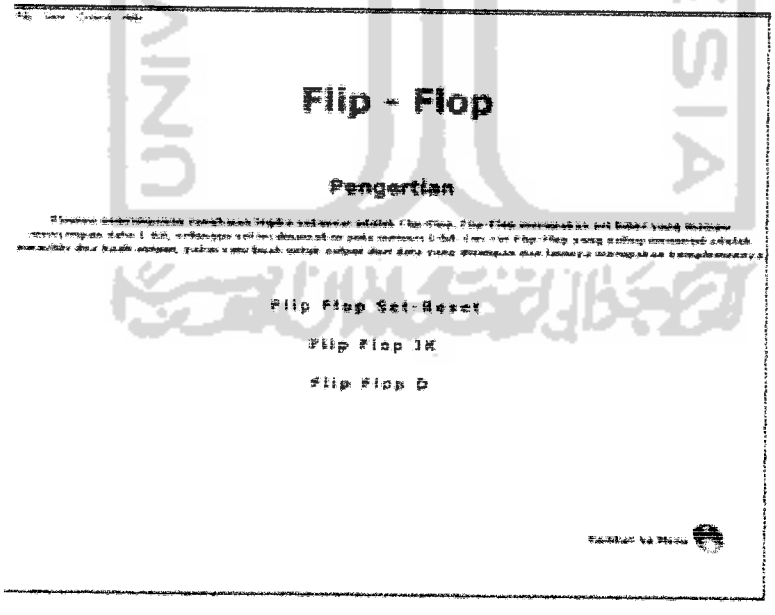
Penjelasan *Script* Rangkaian *Full Adder* :

Script diatas diletakkan pada tombol "Lihat Hasil", jika *input* A=0, B=0 dan C=0 maka *output1*(Cn)=0, *output2*(S)=0 sedangkan X1-X4, A1-A5, dan O1-O2 adalah alir angka biner yang keluar dari *input* A, B dan C untuk selanjutnya dijadikan *input* pada gerbang XOR, AND dan OR.

Untuk kombinasi *input-input* yang ada, penjelasannya sama seperti untuk *input* ABC=000 diatas.

3.3.2.1.6 Halaman *Flip - Flop*

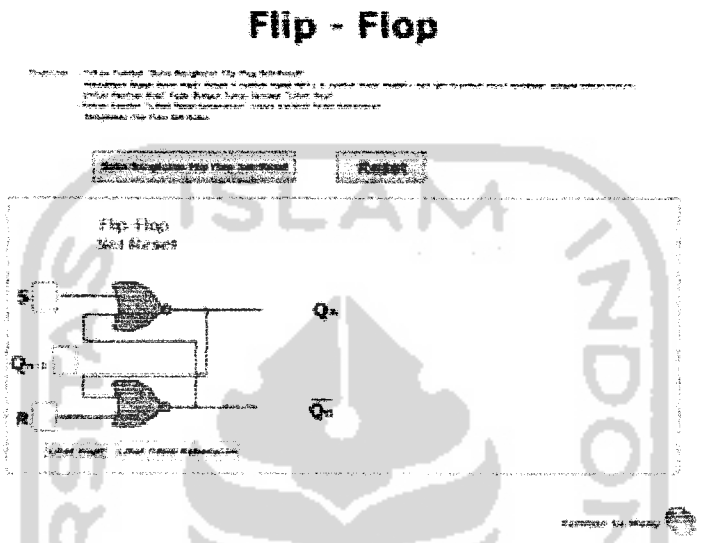
Halaman *flip - flop*, pada halaman ini *user* dapat memilih jenis-jenis dari *flip - flop* serta melihat sekilas pengertian tentang *flip - flop*. Seperti yang ditunjukkan pada gambar 3.46.



Gambar 3.46 Tampilan Halaman Menu *Flip – Flop*

3.3.2.1.6.1 *Flip – Flop Set Reset*

Pada rangkaian *flip-flop set reset* maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.47.



Gambar 3.47 Tampilan Halaman *Flip – Flop Set Reset*

Action Script Rangkaian *Flip-Flop* SR :

```
on (release) {
      if (s == "0" & r == "0" & preq == "0" ) {
            output="0", output1="1";
            loadMovie("TETAP.swf", "destination");
            stop();
      }
}
//
on (release) {
      if (s == "0" & r == "0" & preq == "1" ) {
```

```
            output="1", output1="0";
            loadMovie("TETAP.swf", "destination");
            stop();
        }
}
//
on (release) {
        if (s == "1" & r == "1" & preq == "0" ) {
            output="?", output1="?";
            loadMovie("TERLARANG.swf", "destination");
            stop();
        }
}
//
on (release) {
        if (s == "1" & r == "1" & preq == "1" ) {
            output="?", output1="?";
            loadMovie("TERLARANG.swf", "destination");
            stop();
        }
}
//
on (releaso) {
        if (s == "1" & r == "0" & preq == "0" ) {
            output="1", output1="0";
            loadMovie("SET.swf", "destination");
            stop();
        }
```

```
}
//
on (release) {
      if (s == "1" & r == "0" & preq == "1" ) {
            output="1", output1="0";
            loadMovie("SET.swf", "destination");
            stop();
      }
}
//
on (release) {
      if (s == "0" & r == "1" & preq == "0" ) {
            output="0", output1="1";
            loadMovie("RESET.swf", "destination");
            stop();
      }
}
//
on (release) {
      if (s == "0" & r == "1" & preq == "1" ) {
            output="0", output1="1";
            loadMovie("RESET.swf", "destination");
            stop();
      }
}
```

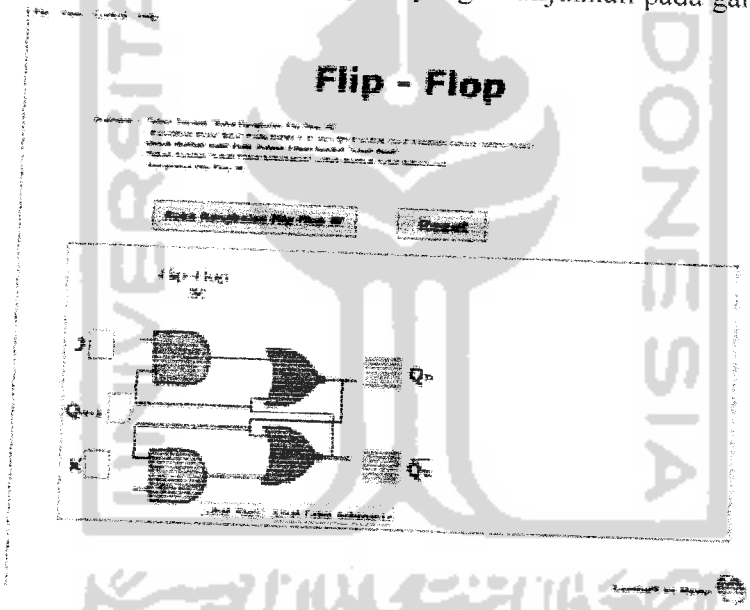
Penjelasan *Script* Rangkaian *Flip-Flop SR* :

Script diatas diletakkan pada tombol "Lihat Hasil", jika *input* S=0, R=0 dan Qn-1=0 maka *output* (Q)=0, *output*1(notQ)=1 lalu program akan memanggil *movie* TETAP.swf ke dalam kotak *movie* klip "destination".

Untuk kombinasi *input-input* yang ada, penjelasannya sama seperti untuk *input* SR=00 diatas.

3.3.2.1.6.2 *Flip – Flop JK*

Pada rangkaian *flip-flop JK* maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.48.



Gambar 3.48 Tampilan Halaman *Flip – Flop JK*

Action Script Rangkaian *Flip-Flop* JK :

```
on (release) {
      if (j == "0" & k == "0" & preq == "0" ) {
            output="0", output1="1";
            loadMovie("TETAP.swf", "destination");
```

```
            stop();
        }
}
//
on (releaseo) {
        if (j == "0" & k == "0" & preq == "1" ) {
            output="1", output1="0";
            loadMovie("TETAP.swf", "destination");
            stop();
        }
}
//
on (release) {
        if (j == "1" & k == "1" & preq == "0" ) {
            output="1", output1="0";
            loadMovie("KOMPLEMEN.swf", "destination");
            stop();
        }
}
//
on (release) {
        if (j == "1" & k == "1" & preq == "1" ) {
            output="0", output1="1";
            loadMovie("KOMPLEMEN.swf", "destination");
            stop();
        }
}
//
```

```
on (release) {
        if (j == "1" & k == "0" & preq == "0" ) {
                output="1", output1="0";
                loadMovie("SET.swf", "destination");
                stop();
        }
}
//
on (release) {
        if (j == "1" & k == "0" & preq == "1" ) {
                output="1", output1="0";
                loadMovie("SET.swf", "destination");
                stop();
        }
}
//
on (release) {
        if (j == "0" & k == "1" & preq == "0" ) {
                output="0", output1="1";
                loadMovie("RESET.swf", "destination");
                stop();
        }
}
//
on (release) {
        if (j == "0" & k == "1" & preq == "1" ) {
                output="0", output1="1";
                loadMovie("RESET.swf", "destination");
```

```
            stop();
        }
}
```

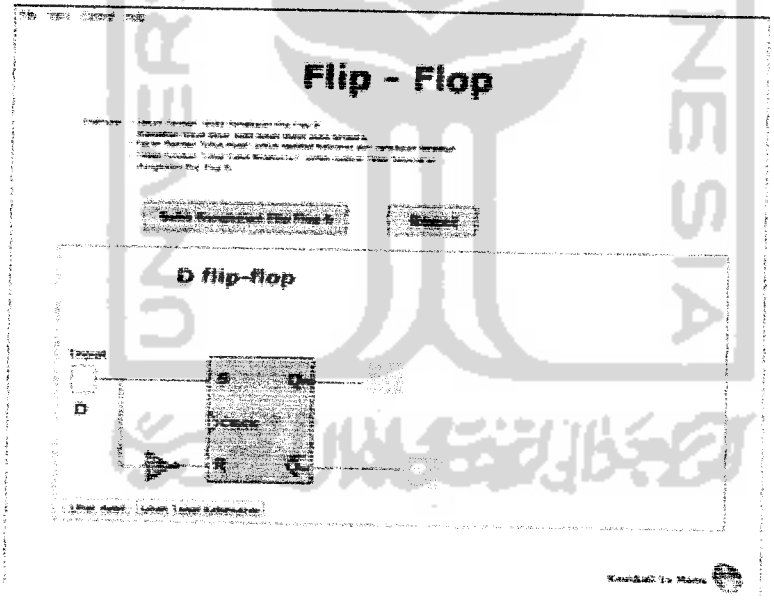
Penjelasan *Script* Rangkaian *Flip-Flop JK* :

Script diatas diletakkan pada tombol "Lihat Hasil", jika *input* J=0, K=0 dan Qn-1=0 maka *output* (Q)=0, *output*1(notQ)=1 lalu program akan memanggil *movie* TETAP.swf ke dalam kotak *movie* klip "destination".

Untuk kombinasi *input-input* yang ada, penjelasannya sama seperti untuk *input* JK=00 diatas.

3.3.2.1.6.3 *Flip- Flop D*

Pada rangkaian *flip-flop D* maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.49.



Gambar 3.49 Tampilan Halaman *Flip Flop D*

Action Script Rangkaian *Flip-Flop* D :

```
on (release) {
      if (D == "0" ) {
            Q="0", NQ="1";
            loadMovie("RESET.swf", "destination");
            stop();
      }
}
//

on (release) {
      if (D == "1" ) {
            Q="1", NQ="0";
            loadMovie("SET.swf", "destination");
            stop();
      }
}
```

Penjelasan *Script* Rangkaian *Flip-Flop* D :

Script diatas diletakkan pada tombol "Lihat Hasil", jika *input* D=0 maka *output* (Q)=0, *output*1(notQ)=1 lalu program akan memanggil *movie* RESET.swf ke dalam kotak *movie* klip "destination". Sedangkan jika *input* D=1 maka *output* (Q)=1, *output*1(notQ)=0 lalu program akan memanggil *movie* SET.swf ke dalam kotak *movie* klip "destination"

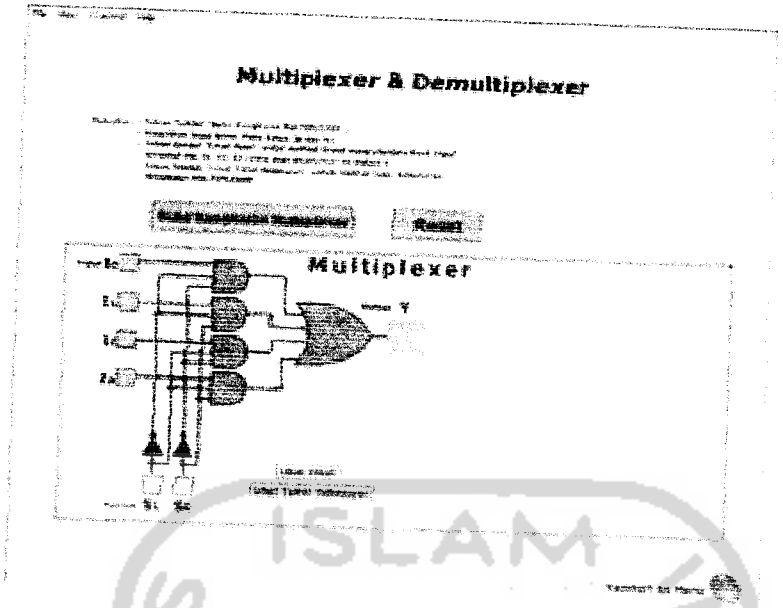
3.3.2.1.7 Halaman *Multiplexer* dan *Demultiplexer*

Halaman *multiplexer* dan *demultiplexer*, pada halaman ini *user* dapat mengakses rangkaian dari *multiplexer* dan *demultiplexer* serta melihat sekilas pengertian tentang *multiplexer* dan *demultiplexer*. Seperti yang ditunjukkan pada gambar 3.50.

Gambar 3.50 Tampilan Halaman Menu *Multiplexer* dan *Demultiplexer*

3.3.2.1.7.1 *Multiplexer*

Pada rangkaian *multiplexer* maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.51.



Gambar 3.51 Tampilan Halaman *Multiplexer*

Action Script Rangkaian *Multiplexer* :

```
on (release) {
        if (S1=="0" & S0=="0"){
                N1="0",N2="0",N3="1", N4="1",Y="I0"
                A1="1",A2="1",A3="1",A4="0",A5="1"
                A6="0",A7="0",A8="0",A9="1",A10="0"
                A11="0",A12="0",O1="1",O2="0"
                O3="0",O4="0";
                loadMovie("I0.swf", "destination");
                loadMovie("merah.swf", "incaran1");
                unloadMovie("incaran3");
                unloadMovie("incaran2");
                unloadMovie("incaran4");
                stop();
        }
}
```

```
//
on (release) {
      if (S1=="0" & S0=="1"){
            N1="0",N2="1",N3="1", N4="0",Y="I1"
            A1="0",A2="1",A3="0",A4="1",A5="1"
            A6="1",A7="0",A8="0",A9="0",A10="0"
            A11="0",A12="1",O1="0",O2="1"
            O3="0",O4="0";
            loadMovie("I1.swf", "destination");
            loadMovie("merah.swf", "incaran2");
            unloadMovie("incaran3");
            unloadMovie("incaran4");
            unloadMovie("incaran1");
            stop();
      }
}
//

on (release) {
      if (S1=="1" & S0=="0"){
            N1="1",N2="0",N3="0", N4="1",Y="I2"
            A1="0",A2="0",A3="1",A4="0",A5="0"
            A6="0",A7="1",A8="1",A9="1",A10="0"
            A11="1",A12="0",A13="0",O1="0",O2="0"
            O3="1",O4="0";
            loadMovie("I2.swf", "destination");
            loadMovie("merah.swf", "incaran3");
            unloadMovie("incaran4");
```

```
            unloadMovie("incaran2");
            unloadMovie("incaran1");
            stop();
        }
}
//
on (release) {
      if (S1=="1" & S0=="1"){
            N1="1",N2="1",N3="0", N4="0",Y="I3"
            A1="0",A2="0",A3="0",A4="0",A5="0"
            A6="1",A7="0",A8="1",A9="0",A10="1"
            A11="1",A12="1",A13="0",O1="0",O2="0"
            O3="0",O4="1";
            loadMovie("I3.swf", "destination");
            loadMovie("merah.swf", "incaran4");
            unloadMovie("incaran3");
            unloadMovie("incaran2");
            unloadMovie("incaran1");
            stop();
      }
}
```

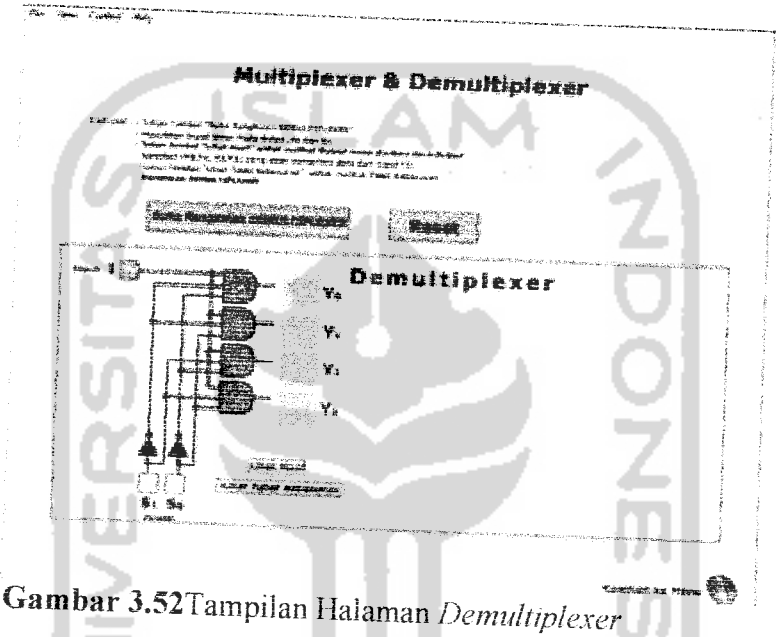
Penjelasan *Script* Rangkaian *Multiplexer* :

Script diatas diletakkan pada tombol "Lihat Hasil", jika *input* S1=0 dan *input* S2=0 maka *output* Y= I0 sedangkan N1-N4, A1-A12, dan O1-O4 adalah alir angka biner yang keluar dari *input* S1 dan S0 untuk selanjutnya dijadikan *input* pada gerbang NOT, AND dan OR.

Untuk kombinasi *input-input* yang ada, penjelasannya sama seperti untuk *input* S1,S2=00diatas.

3.3.2.1.7.2 *Demultiplexer*

Pada rangkaian de*multiplexer* maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.52.



Gambar 3.52 Tampilan Halaman *Demultiplexer*

Action Script Rangkaian *Demultiplexer* :

```
on (release) {
      if (S1 == "0" & S0 == "0" ) {
            Y0="1", Y1="0", Y2="0", Y3="0",
            N1="0",N2="0",N3="1", N4="1",A1="1",
            A2="1",A3="1",A4="1",A5="1",A6="0",
            A7="1",A8="0",A9="1",A10="1",A11="0",A12="0";
            loadMovie("merah.swf", "incaran");
```

```
            stop();
        }
    }
    //
    on (release) {
        if (S1 == "0" & S0 == "1" ) {
            Y0="0", Y1="I", Y2="0", Y3="0",
            N1="0",N2="1",N3="1", N4="0",A1="1",
            A2="1",A3="0",A4="1",A5="1",A6="1",
            A7="1",A8="0",A9="0",A10="1",A11="0",A12="1";
            loadMovie("merah.swf", "incaran");
            stop();
        }
    }
    //
    on (release) {
        if (S1 == "1" & S0 == "0" ) {
            Y0="0", Y1="0", Y2="I", Y3="0",
            N1="1",N2="0",N3="0", N4="1",A1="1",
            A2="0",A3="1",A4="1",A5="0",A6="0",
            A7="1",A8="1",A9="1",A10="1",A11="1",A12="0";
            loadMovie("merah.swf", "incaran");
            stop();
        }
    }
    //
    on (release) {
        if (S1 == "1" & S0 == "1" ) {
```

```
            Y0="0", Y1="0", Y2="0", Y3="I",
            N1="1",N2="1",N3="0", N4="1",A1="1",
            A2="0",A3="0",A4="1",A5="0",A6="1",
            A7="1",A8="1",A9="0",A10="1",A11="1",A12="1";
            loadMovie("merah.swf", "incaran");
            stop();
        }
}
```

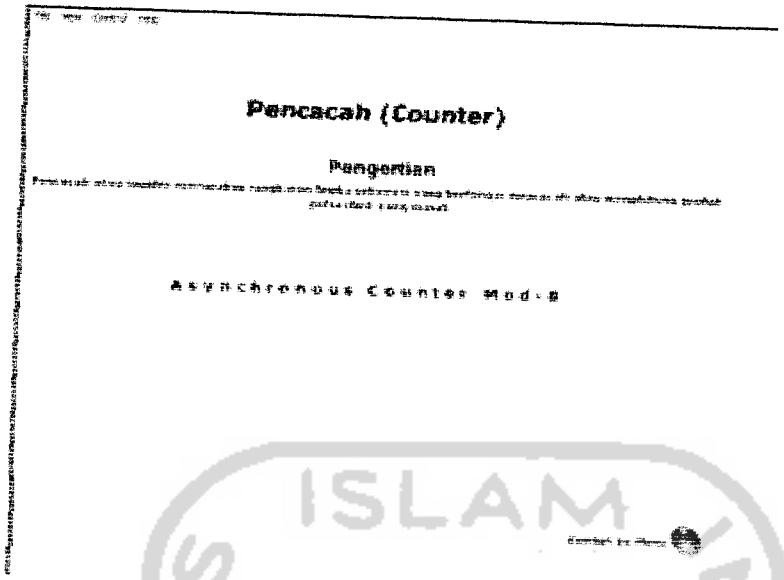
Penjelasan *Script* Rangkaian *Demultiplexer* :

Script diatas diletakkan pada tombol "Lihat Hasil", jika *input* S1=0 dan *input* S2=0 maka *output* Y0= I, Y1=0, Y2=0 dan Y3=0 sedangkan N1-N4 dan A1-A12 adalah alir angka biner yang keluar dari *input* S1 dan S0 untuk selanjutnya dijadikan *input* pada gerbang NOT dan AND.

Untuk kombinasi *input-input* yang ada, penjelasannya sama seperti untuk *input* S1,S2=00diatas.

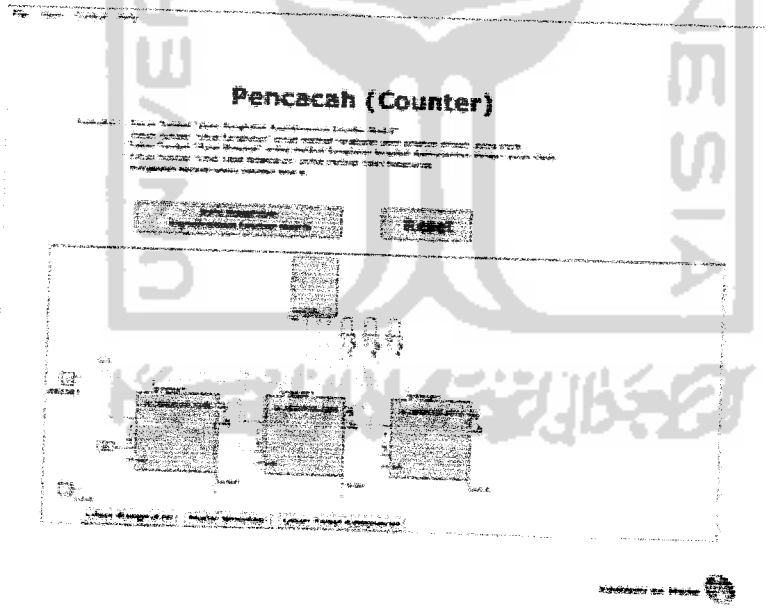
3.3.2.1.8 Halaman Pencacah (*Counter*)

Halaman pencacah (*counter*), pada halaman ini *user* dapat mengakses rangkaian dari pencacah (*counter*)serta melihat sekilas pengertian tentang pencacah (*counter*). Seperti yang ditunjukkan pada gambar 3.53.



Gambar 3.53 Tampilan Halaman Menu Pencacah (*Counter*)

Pada rangkaian pencacah (*counter*) maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.54.



Gambar 3.54 Tampilan Halaman Pencacah (*Counter*)

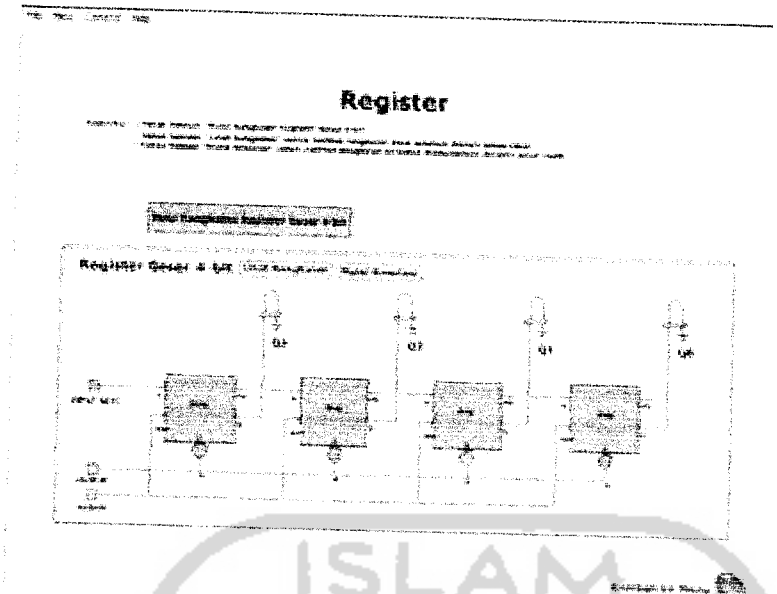
3.3.2.1.9 Halaman *Register*

Halaman *register*, pada halaman ini *user* dapat mengakses rangkaian dari *register* serta melihat sekilas pengertian tentang *register*. Seperti yang ditunjukkan pada gambar 3.55.

Gambar 3.55 Tampilan Halaman Menu *Register*

3.3.2.1.9.1 *Register* Geser

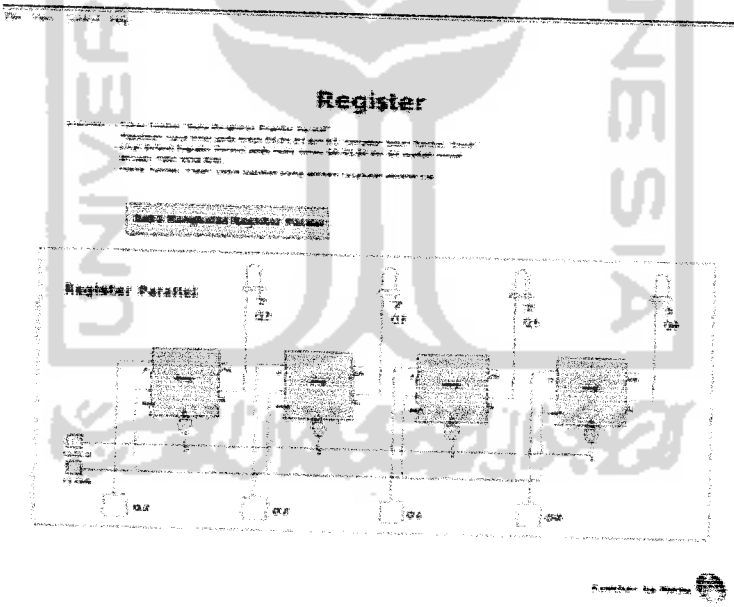
Pada rangkaian *register* geser maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.56.



Gambar 3.56 Tampilan Halaman *Register* Geser

3.3.2.1.9.2 *Register* Paralel

Pada rangkaian *register* paralel maka *user* akan mendapatkan tampilan serta instruksi cara menjalankannya. Seperti yang ditunjukkan pada gambar 3.57.



Gambar 3.57 Tampilan Halaman *Register* Paralel

***Action Script* Rangkaian *Register* Paralel :**

```
on (release) {
        if (input1=="1"){
                loadMovie ("lampu.swf", "target1");
        }if (input1=="0"){
                unloadMovie ("target1");
                stop();
        }
}
//
on (release) {
        if (input2=="1"){
                loadMovie ("lampu.swf", "target2");
        }if (input2=="0"){
                unloadMovie ("target2");
                stop();
        }
}
//
on (release) {
        if (input3=="1"){
                loadMovie ("lampu.swf", "target3");
        }if (input3=="0"){
                unloadMovie ("target3");
                stop();
        }
}
```

```
//
on (release) {
      if (input4=="1"){
            loadMovie ("lampu.swf", "target4");
      }if (input4=="0"){
            unloadMovie ("target4");
                  stop();
      }
}
```

Penjelasan *Script* Rangkaian *Register* Paralel :

Script diatas diletakkan pada tombol "Clock", jika *input*1=0 maka program akan memanggil movie "lampu.swf" ke dalam kotak *movie* klip "target1" dan jika *input* 1=1 maka program akan menghapus *movie* yang ada dalam kotak *movie* klip "target1" begitu juga seterusnya terhadap *input*2, *input*3 dan *input* 4.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Pengujian Aplikasi

Pengujian aplikasi dilakukan untuk menganalisis kinerja perangkat lunak. Dari hasil pengujian akan diketahui apakah fungsi-fungsi yang ada dalam sistem ini dapat berjalan dengan baik dan memenuhi kebutuhan. Pengujian dilakukan dengan menjalankan proses-proses yang ada dalam sistem dengan memasukkan data sesuai kebutuhan. Hasil dari pengujian ini kemudian dianalisis untuk mengetahui sejauh mana aplikasi dapat berjalan, apakah sesuai dengan yang diharapkan. Kekurangan-kekurangan yang ada akan menjadi masukan untuk kemudian diterapkan pada implementasi aplikasi selanjutnya.

4.2 Pengujian dan Analisis

Pada tahap pengujian, penulis akan mencoba membandingkan kesesuaian antara *input* dari *user* dengan kebutuhan *input* aplikasi. Pengujian akan dilakukan dengan memasukkan *input* yang dianggap sesuai dengan kebutuhan dari aplikasi "Membangun Tabel Kebenaran Untuk Aplikasi Gerbang Logika Pada Rangkaian Elektronika ". Hal ini dilakukan untuk menganalisis kinerja perangkat lunak yang

telah dibuat. Hasil analisis ini akan sangat bermanfaat dalam pengembangan aplikasi ini dikemudian hari.

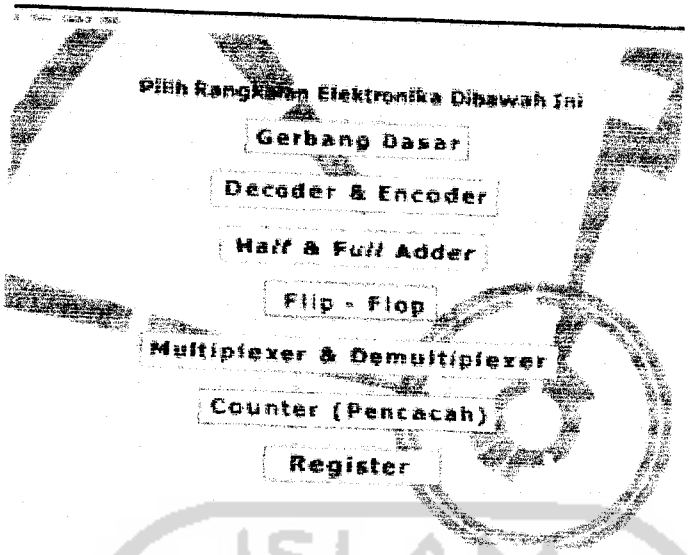
4.2.1 Pengujian Normal

Pengujian normal dilakukan dengan memberikan *input* yang sesuai dengan prosedur atau aturan yang telah ditetapkan dalam peng*input*an data, sehingga proses yang akan dijalankan oleh aplikasi dapat berjalan dengan sebagaimana mestinya.

4.2.1.1 Pengujian Normal Aplikasi Gerbang Logika

Langkah pertama aplikasi gerbang logika pada rangkaian elektronika adalah *user* cukup mengakses aplikasi.*exe* dari program ini. Sehingga akan muncul tampilan seperti yang ditunjukkan pada gambar 4.1.

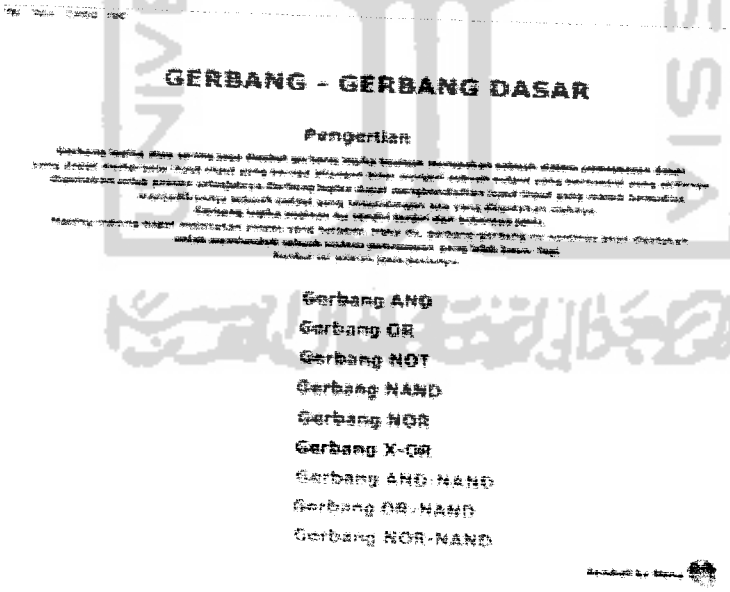
Gambar 4.1 Tampilan Halaman Utama



Gambar 4.2 Tampilan Halaman Menu

4.2.1.1.1 Pengujian Pada Gerbang – Gerbang Dasar

Setelah memilih salah satu menu rangkaian (anggap saja *user* memilih menu gerbang dasar), kemudian user akan melihat pengertian rangkaian yang dia pilih serta jenis-jenis rangkaian tersebut. Seperti yang ditunjukkan pada gambar 4.3.



Gambar 4.3 Tampilan Halaman Menu Gerbang-Gerbang Dasar

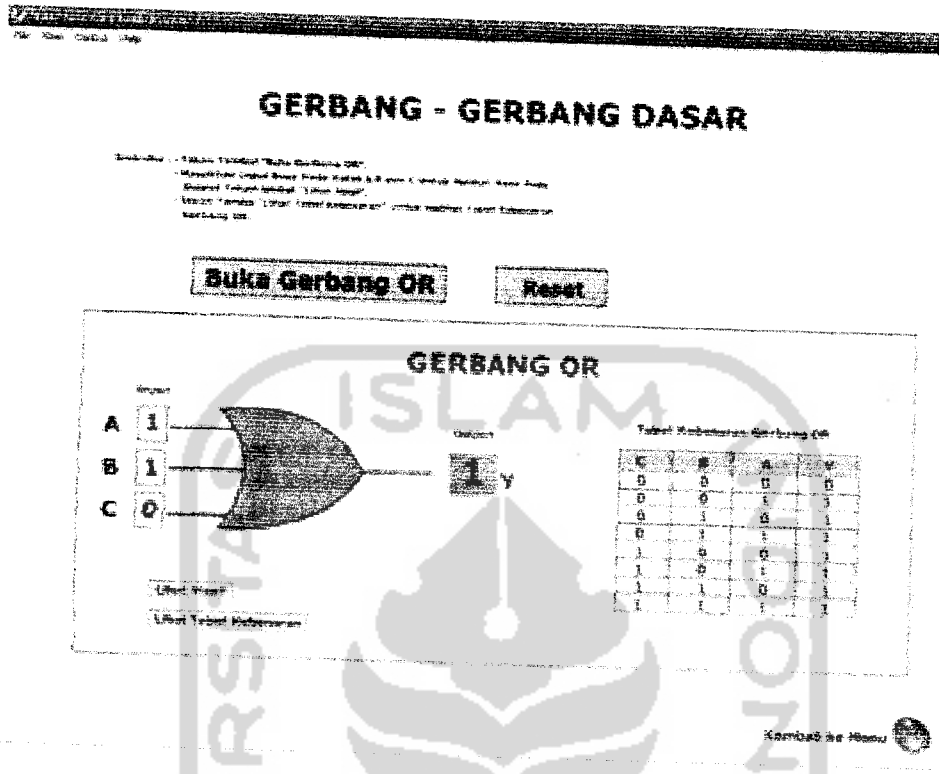
4.2.1.1.1.1 Pengujian Pada Gerbang AND

Pada gerbang AND akan diuji dengan memasukkan *input* A=0, B=1 dan C=0, maka seperti yang terlihat pada gambar 4.4 dibawah ini, *output* yang keluar dari gerbang AND adalah Y=0. Begitu juga hasil yang terlihat pada tabel kebenarannya.

Gambar 4.4 Pengujian Pada Gerbang AND

4.2.1.1.1.2 Pengujian Pada Gerbang OR

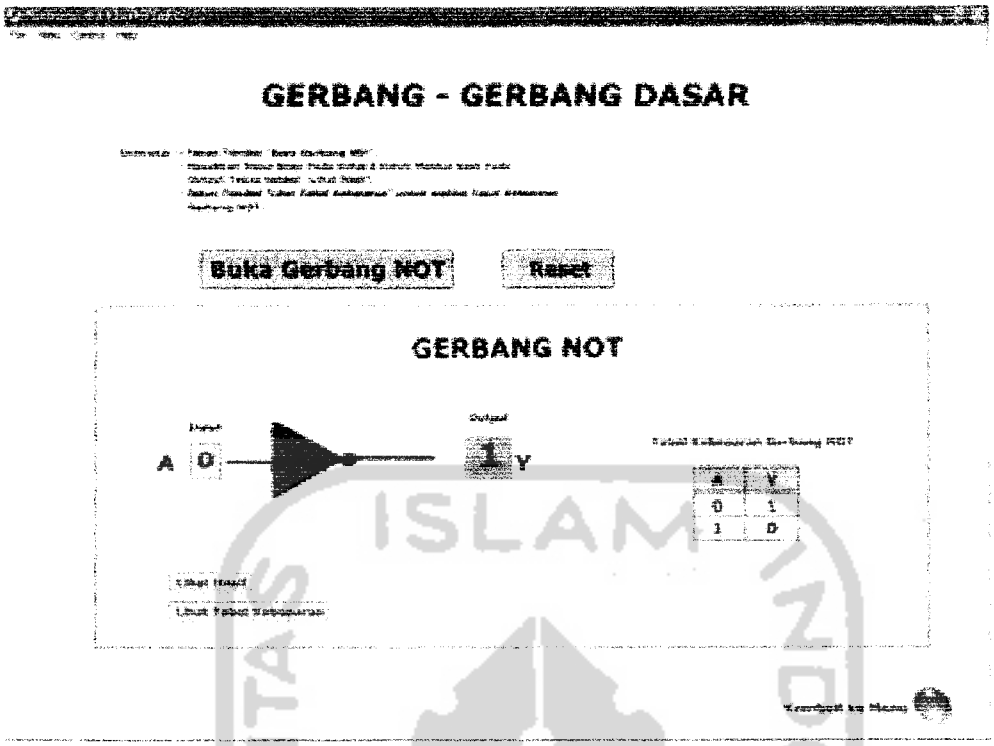
Pada gerbang OR akan diuji dengan memasukkan *input* A=1, B=1 dan C=0, maka seperti yang terlihat pada gambar 4.5 dibawah ini, *output* yang keluar dari gerbang OR adalah Y=1. Begitu juga hasil yang terlihat pada tabel kebenarannya.



Gambar 4.5 Pengujian Pada Gerbang OR

4.2.1.1.1.3 Pengujian Pada Gerbang NOT

Pada gerbang NOT akan diuji dengan memasukkan *input* A=0, maka seperti yang terlihat pada gambar 4.6 dibawah ini, *output* yang keluar dari gerbang NOT adalah Y=1. Begitu juga hasil yang terlihat pada tabel kebenarannya.



Gambar 4.6 Pengujian Pada Gerbang NOT

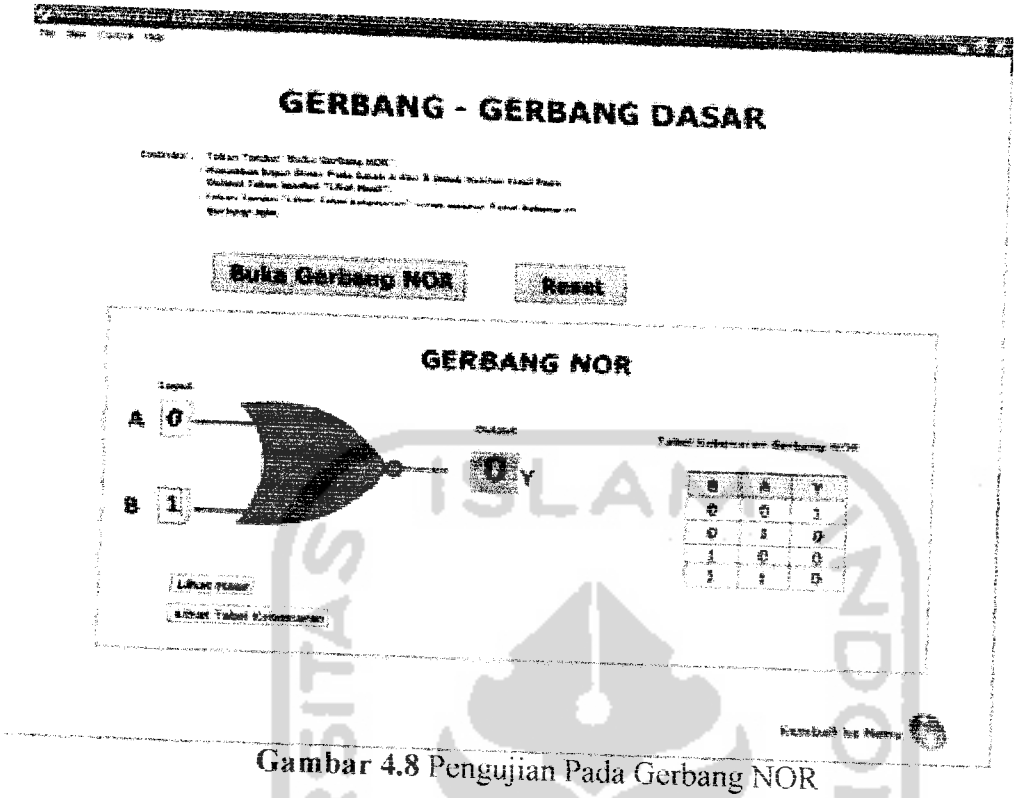
4.2.1.1.1.4 Pengujian Pada Gerbang NAND

Pada gerbang NAND akan diuji dengan memasukkan *input* A=1 dan B=0, maka seperti yang terlihat pada gambar 4.7 dibawah ini, *output* yang keluar dari gerbang NAND adalah Y=1. Begitu juga hasil yang terlihat pada tabel kebenarannya.

Gambar 4.7 Pengujian Pada Gerbang NAND

4.2.1.1.1.5 Pengujian Pada Gerbang NOR

Pada gerbang NOR akan diuji dengan memasukkan *input* A=0 dan B=1, maka seperti yang terlihat pada gambar 4.8 dibawah ini, *output* yang keluar dari gerbang NOR adalah Y=0. Begitu juga hasil yang terlihat pada tabel kebenarannya.



Gambar 4.8 Pengujian Pada Gerbang NOR

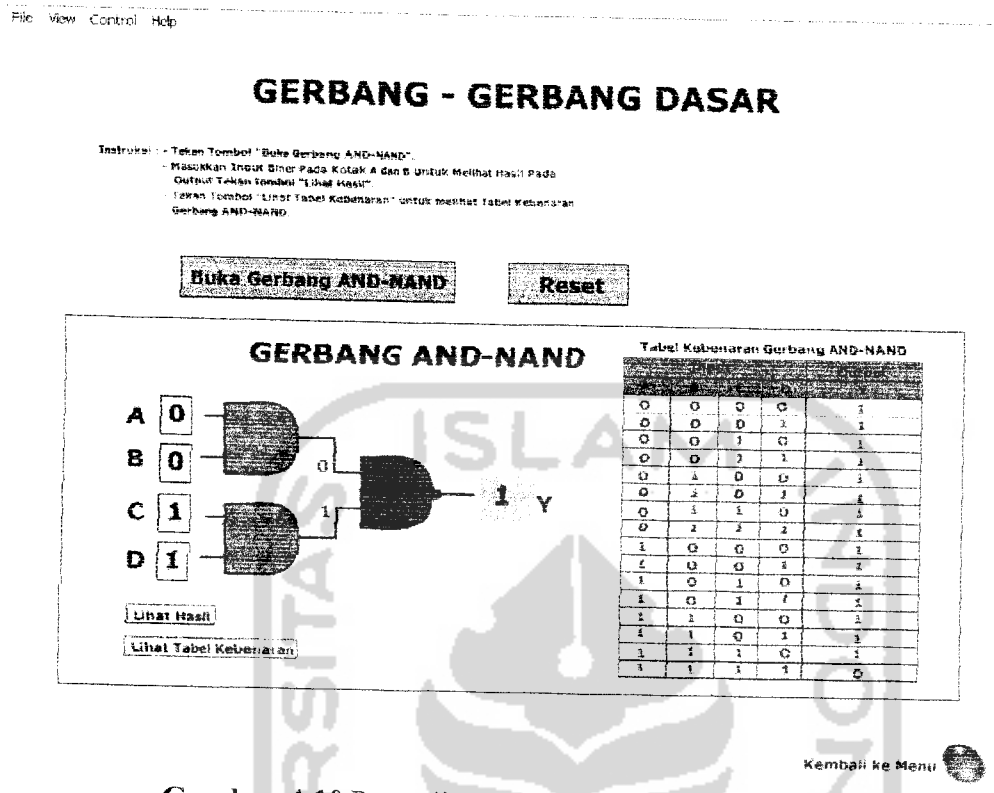
4.2.1.1.1.6 Pengujian Pada Gerbang XOR

Pada gerbang XOR akan diuji dengan memasukkan *input* A=1 dan B=1, maka seperti yang terlihat pada gambar 4.9 dibawah ini, *output* yang keluar dari gerbang XOR adalah Y=0. Begitu juga hasil yang terlihat pada tabel kebenarannya.

Gambar 4.9 Pengujian Pada Gerbang XOR

4.2.1.1.1.7 Pengujian Pada Rangkaian AND-NAND

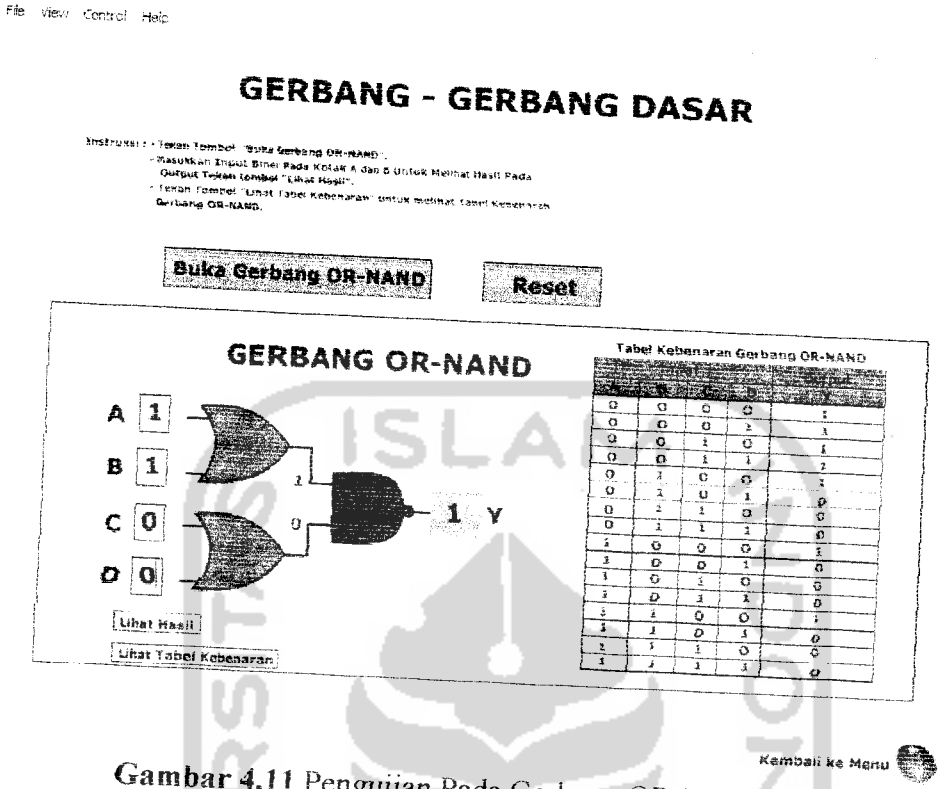
Pada gerbang AND-NAND akan diuji dengan memasukkan *input* A=0, B=0, C=1 dan D=1, maka seperti yang terlihat pada gambar 4.10dibawah ini, *output* yang keluar dari rangkaian AND-NAND adalah Y=1. Begitu juga hasil yang terlihat pada tabel kebenarannya.



Gambar 4.10 Pengujian Pada Rangkaian AND-NAND

4.2.1.1.1.8 Pengujian Pada Rangkaian OR-NAND

Pada gerbang OR-NAND akan diuji dengan memasukkan *input* A=1, B=1, C=0 dan D=0, maka seperti yang terlihat pada gambar 4.11 dibawah ini, *output* yang keluar dari rangkaian OR-NAND adalah Y=1. Begitu juga hasil yang terlihat pada tabel kebenarannya.



Gambar 4.11 Pengujian Pada Gerbang OR-NAND

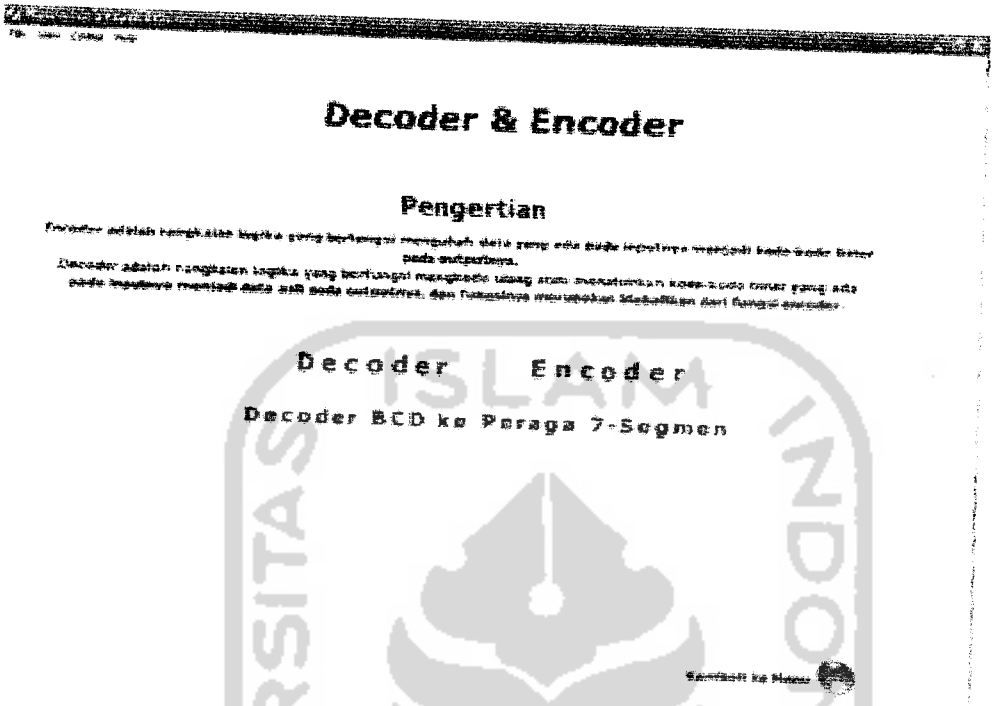
4.2.1.1.1.9 Pengujian Pada Rangkaian NOR-NAND

Pada gerbang NOR-NAND akan diuji dengan memasukkan *input* A=1, B=0, C=1 dan D=0, maka seperti yang terlihat pada gambar 4.12 dibawah ini, *output* yang keluar dari rangkaian NOR-NAND adalah Y=1. Begitu juga hasil yang terlihat pada tabel kebenarannya.

Gambar 4.12 Pengujian Pada Gerbang NOR-NAND

4.2.1.1.2 Pengujian Pada *Decoder* dan *Encoder*

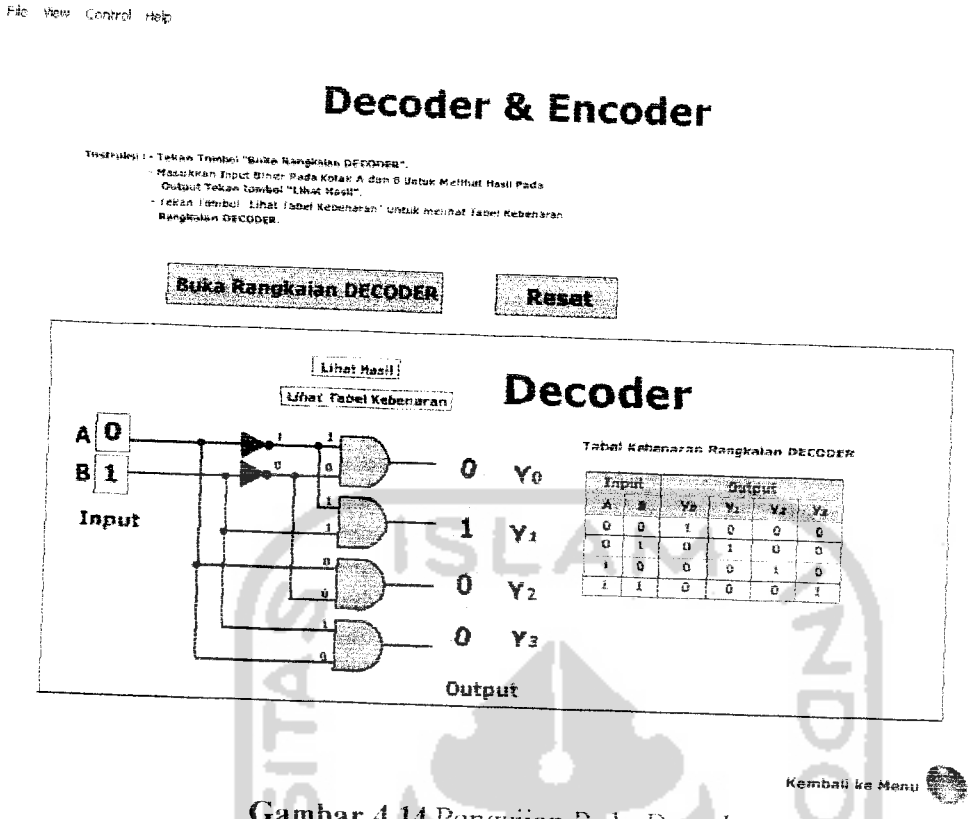
Selanjutnya untuk pengujian *decoder* dan *encoder*, user akan ditampilkan halaman menu *decoder* dan *encoder*, user dapat melihat sekilas pengertian *decoder* dan *encoder* kemudian user dapat mengakses rangkaian yang dipilih. Seperti yang ditunjukkan pada gambar 4.13.



Gambar 4.13 Halaman Menu *Decoder* dan *Encoder*

4.2.1.1.2.1 Pengujian Pada *Decoder*

Pada *decoder* akan diuji dengan memasukkan *input* A=0 dan B=1 maka seperti yang terlihat pada gambar 4.14 dibawah ini, *output* yang keluar dari *decoder* adalah Y1=1. Itu artinya, *decoder* menafsirkan kode 01 biner sebagai 1 desimal. Begitu juga hasil yang terlihat pada tabel kebenarannya.

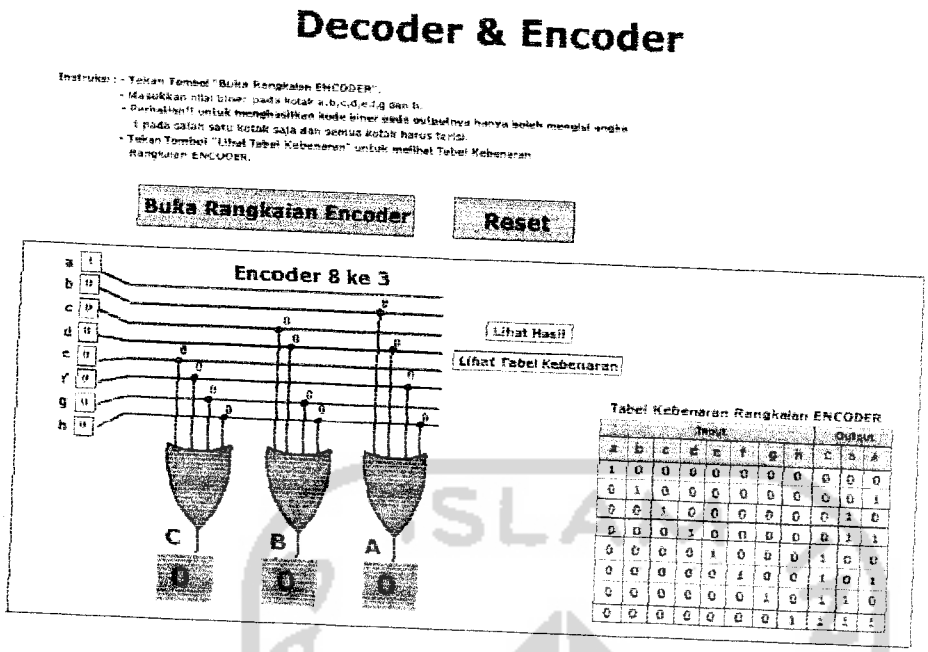


Gambar 4.14 Pengujian Pada *Decoder*

4.2.1.1.2.2 Pengujian Pada *Encoder*

Pada *encoder* akan diuji dengan memasukkan *input* a=1, b=0,c=0, d=0, e=0, f=0, g=0 maka seperti yang terlihat pada gambar 4.15 dibawah ini, *output* yang keluar dari *encoder* adalah A=0, B=0 dan C=0. Itu artinya, jika a = 0 oktal maka *encoder* menafsirkan kode 0 oktal menjadi kode biner 3-*bit* pada *output*nya. Begitu juga hasil yang terlihat pada tabel kebenarannya.

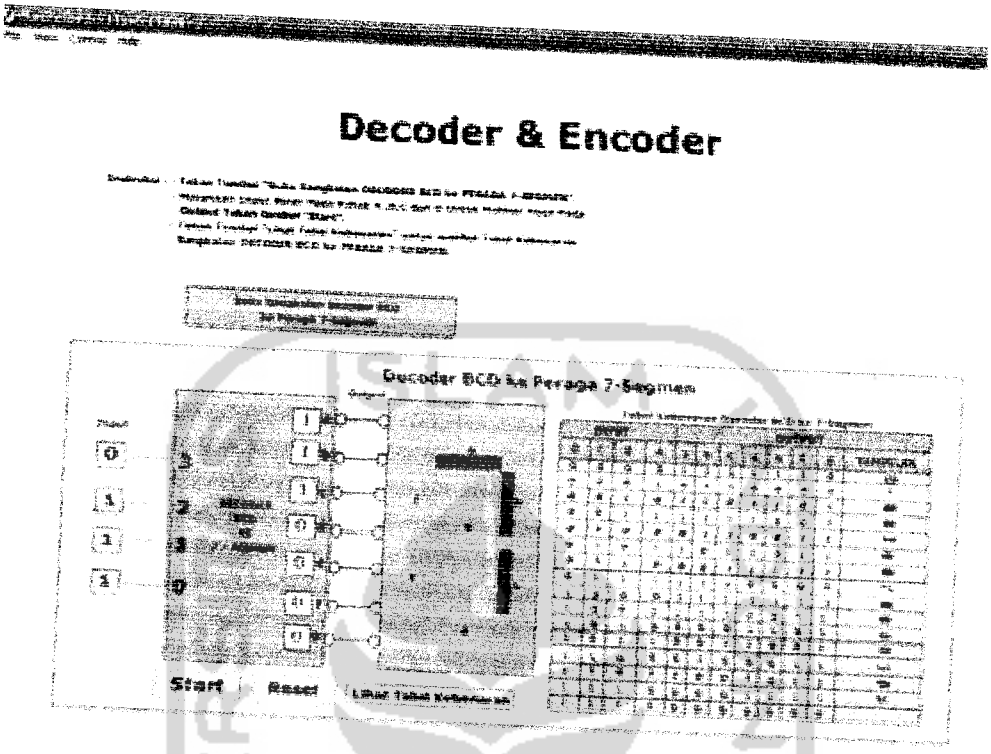
File View Control Help



Gambar 4.15 Pengujian Pada *Encoder*

4.2.1.1.2.3 Pengujian Pada *Decoder* BCD ke Peraga 7 - Segmen

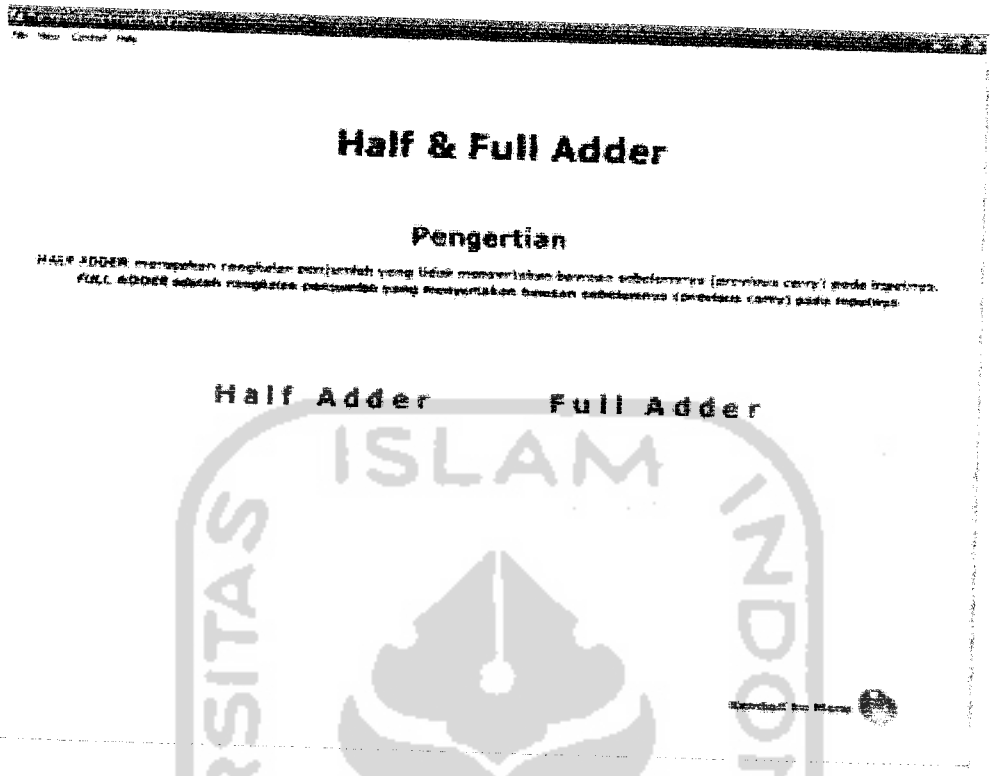
Pada *decoder BCD* ke peraga 7-segmen akan diuji dengan memasukkan *input* A=0, B=1, C=1 dan D=1 maka seperti yang terlihat pada gambar 4.16 dibawah ini, *output* yang keluar dari *decoder BCD* ke peraga 7-segmen adalah angka desima 7. Itu artinya, *decoder BCD* ke peraga 7-segmen memerlukan sinyal *decoder* dengan *input* 0111 untuk menghasilkan sinyal-sinyal penggerak peraga 7-segmen. Setiap segmen dari peraga tersebut berupa LED (*Light Emitting Diode*) yang susunannya membentuk suatu konfigurasi tertentu Begitu juga hasil yang terlihat pada tabel kebenarannya.



Gambar 4.16 Pengujian Pada *Decoder BCD* ke Peraga 7-Segmen

4.2.1.1.3 Pengujian Pada *Half* dan *Full Adder*

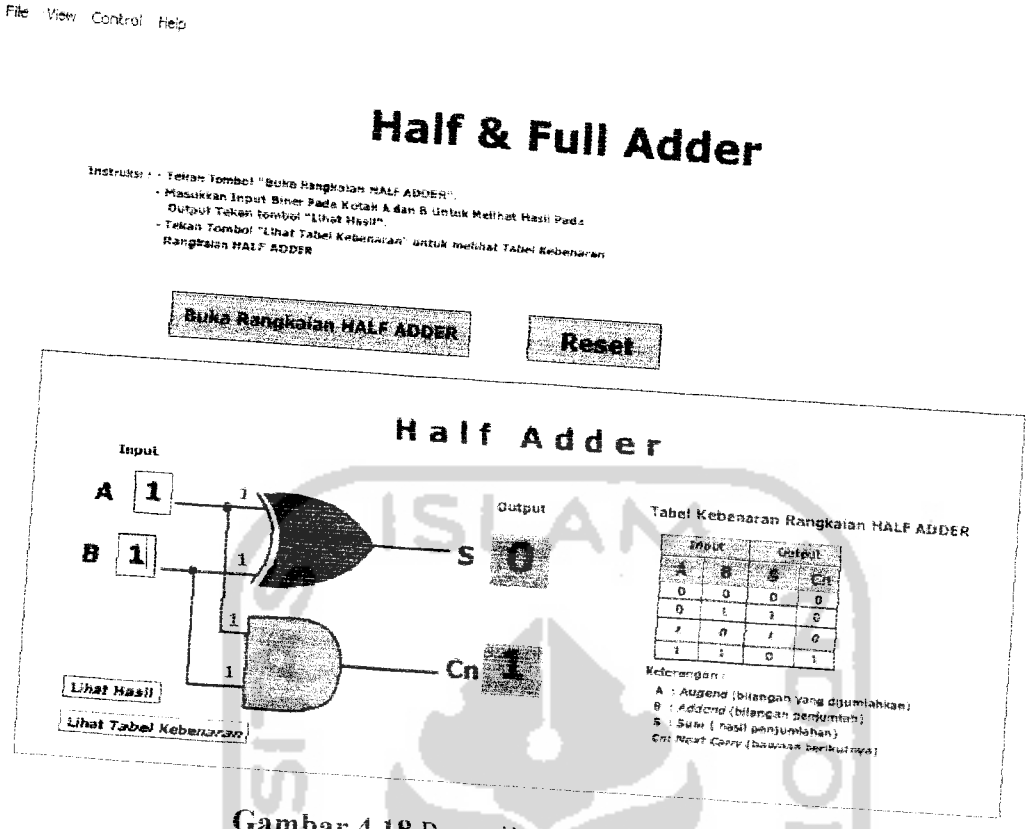
Selanjutnya untuk pengujian *half* dan *full adder*, user akan ditampilkan halaman menu *half* dan *full adder*, user dapat melihat sekilas pengertian *half* dan *full adder* kemudian user dapat mengakses rangkaian yang dipilih. Seperti yang ditunjukkan pada gambar 4.17.



Gambar 4.17 Halaman Menu *Half* dan *Full Adder*

4.2.1.1.3.1 Pengujian Pada *Half Adder*

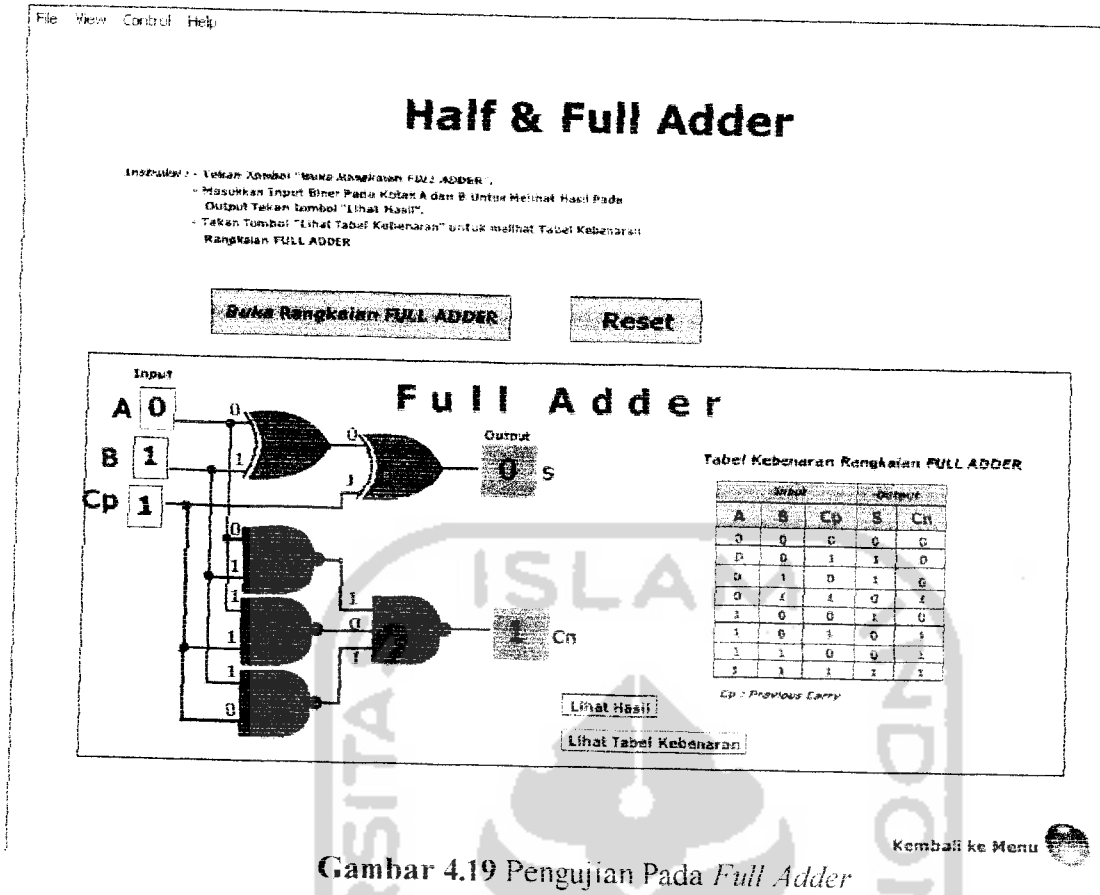
Pada *half adder* akan diuji dengan memasukkan *input* A=1 dan B=1 maka seperti yang terlihat pada gambar 4.18 dibawah ini, *output* yang keluar dari *half adder* adalah S atau Sum = 0 dan Cn atau *next carry* = 1. Itu artinya, *half adder* menjumlahkan *input* A dengan *input* B menggunakan operasi XOR atau 1 XOR 1 = 0 dengan *next carry* 1. Begitu juga hasil yang terlihat pada tabel kebenarannya.



Gambar 4.18 Pengujian Pada *Half Adder*

4.2.1.1.3.2 Pengujian Pada *Full Adder*

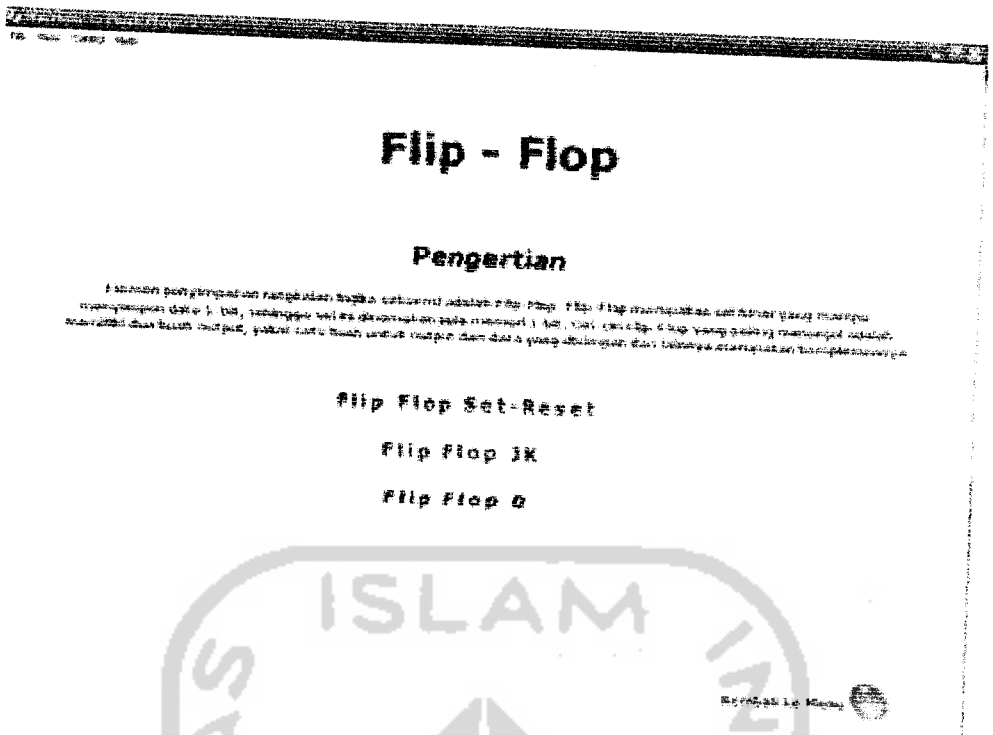
Pada *full adder* akan diuji dengan memasukkan *input* A=0, B=1 dan Cp atau *previous carry* = 1 maka seperti yang terlihat pada gambar 4.19 dibawah ini, *output* yang keluar dari *half adder* adalah S atau Sum = 0 dan Cn atau *next carry* = 1. Itu artinya, *full adder* menjumlahkan *input* A dengan *input* B dengan menyertakan *previous carry* atau bawaan sebelumnya menggunakan operasi XOR atau 0 XOR 1 XOR 1 = 0 dengan *next carry* 1. Begitu juga hasil yang terlihat pada tabel kebenarannya.



Gambar 4.19 Pengujian Pada *Full Adder*

4.2.1.1.4 Pengujian Pada *Flip - Flop*

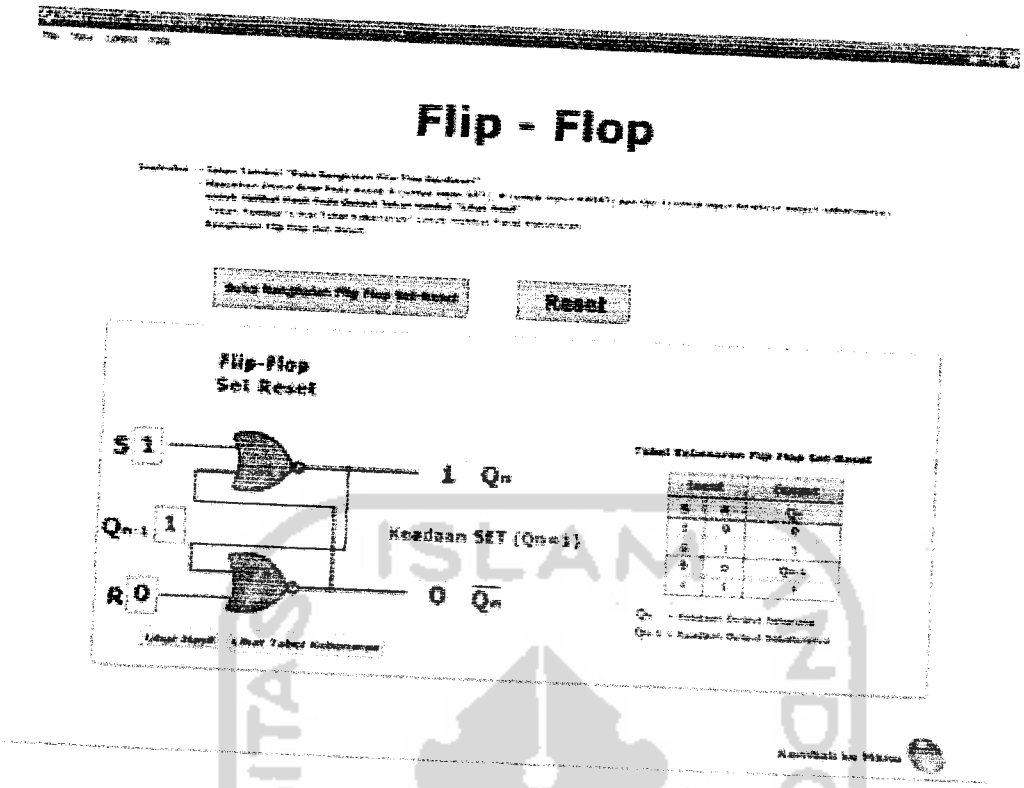
Selanjutnya untuk pengujian *flip - flop*, user akan ditampilkan halaman menu *flip- flop*, user dapat melihat sekilas pengertian *flip - flop* kemudian user dapat mengakses rangkaian yang dipilih. Seperti yang ditunjukkan pada gambar 4.20.



Gambar 4.20 Halaman Menu *Flip – Flop*

4.2.1.1.4.1 Pengujian Pada *Flip – Flop Set Reset*

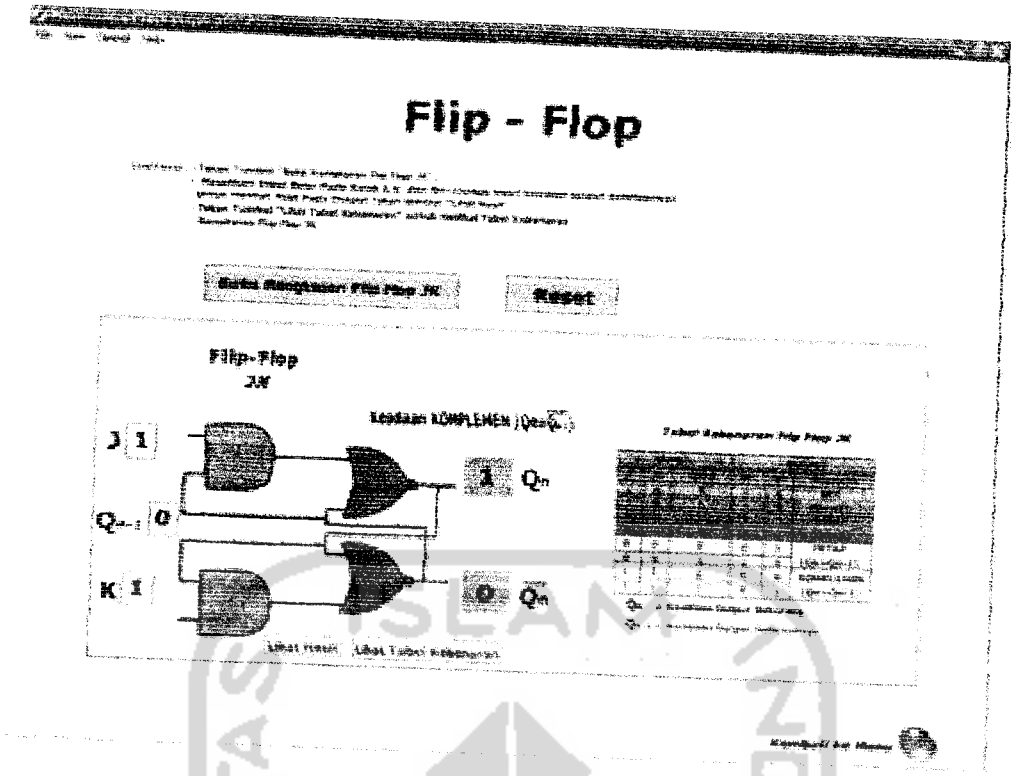
Pada *flip-flop set reset* akan diuji dengan memasukkan *input* S=1 dan R=0 dengan keadaan sebelumnya atau Qn-1 = 1 maka seperti yang terlihat pada gambar 4.21 dibawah ini, *output* yang keluar dari *flip-flop set reset* adalah Qn = 1 dan komplemennya notQn= 0. Itu artinya, dalam melakukan penyimpanan data 1 *flip-flop set reset* memberikan sinyal tinggi pada S dan sinyal rendah pada R (S=1 dan R=0) dan keadaan ini dinamakan *set* yang berarti flip-flop menyimpan data 1. Begitu juga hasil yang terlihat pada tabel kebenarannya.



Gambar 4.21 Pengujian Pada *Flip – Flop Set Reset*

4.2.1.1.4.2 Pengujian Pada *Flip – Flop* JK

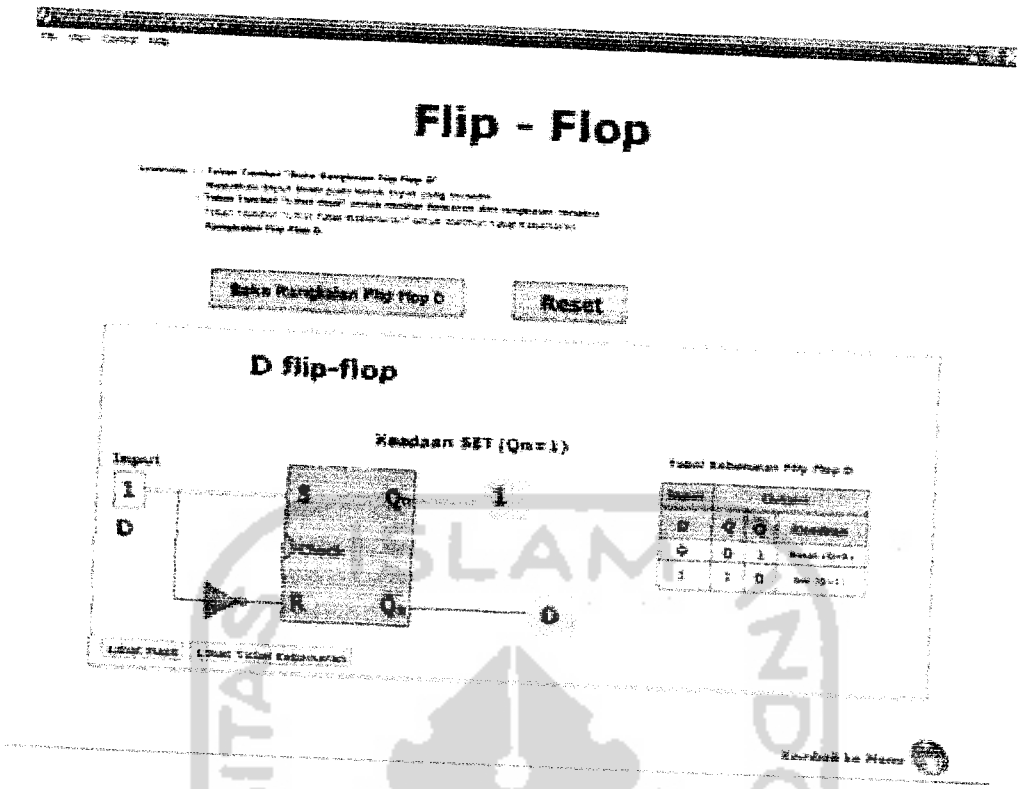
Pada *flip-flop* JK akan diuji dengan memasukkan *input* J=1 dan K=1 dengan keadaan sebelumnya atau Qn-1 = 0 maka seperti yang terlihat pada gambar 4.22 dibawah ini, *output* yang keluar dari *flip-flop* JK adalah Qn = 1 dan komplemennya notQn= 0. Itu artinya, kelemahan pada *flip-flop* SR yaitu munculnya *output* yang tidak dapat didefinisi ketika *input* S dan R tinggi untuk jenis NOR dapat diatasi oleh *flip-flop* JK dengan menjadikan *ouput flip-flop* melakukan pembalikan terhadap keadaan sebelumnya (Qn-1). Begitu juga hasil yang terlihat pada tabel kebenarannya.



Gambar 4.22 Pengujian Pada *Flip – Flop* JK

4.2.1.1.4.3 Pengujian Pada *Flip – Flop* D

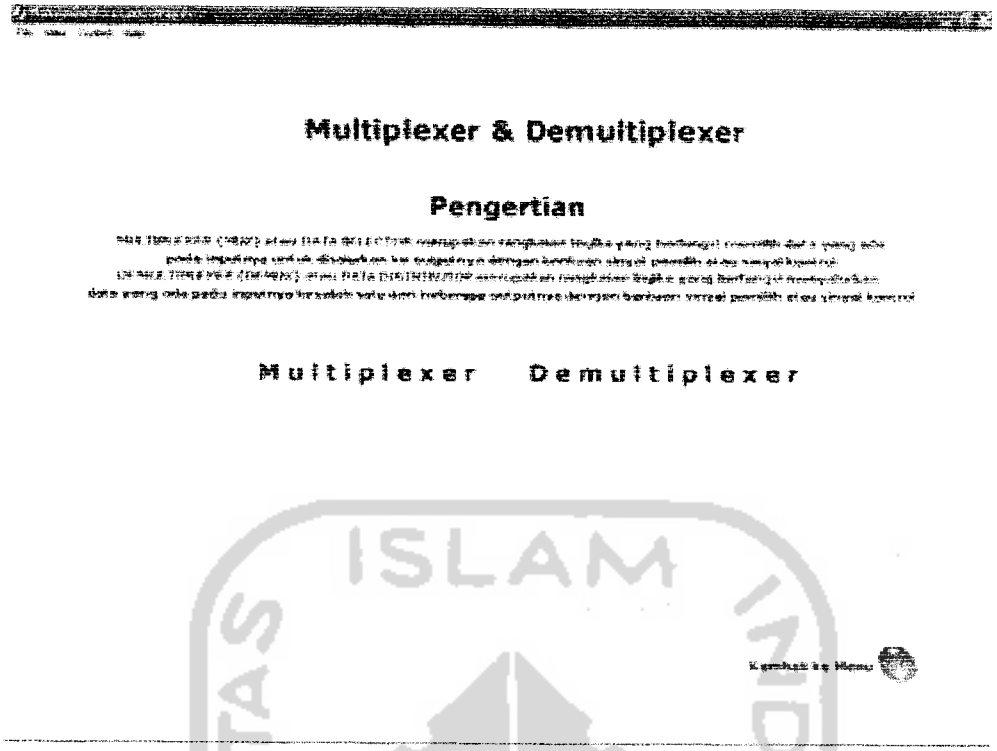
Pada *flip-flop* D akan kita uji dengan memasukkan *input* D=1 maka seperti yang terlihat pada gambar 4.23 dibawah ini, *output* yang keluar dari *flip-flop* D adalah Qn = 1 dan komplemennya notQn = 0. Itu artinya dengan adanya gerbang NOT yang masuk ke *input* R, maka setiap *input* yang diumpankan ke D akan memberikan keadaan yang berbeda pada *input* S dan R. Dengan demikian hanya akan terdapat dua keadaan dari S dan R yakni S=0 dan R=1 atau S=1 dan R=0. jadi, *output flip-flop* D juga hanya memiliki dua keadaan yakni keadaan *set* atau keadaan *reset*. Begitu juga hasil yang terlihat pada tabel kebenarannya.



Gambar 4.23 Pengujian Pada *Flip – Flop* D

4.2.1.1.5 Pengujian Pada *Multiplexer* dan *Demultiplexer*

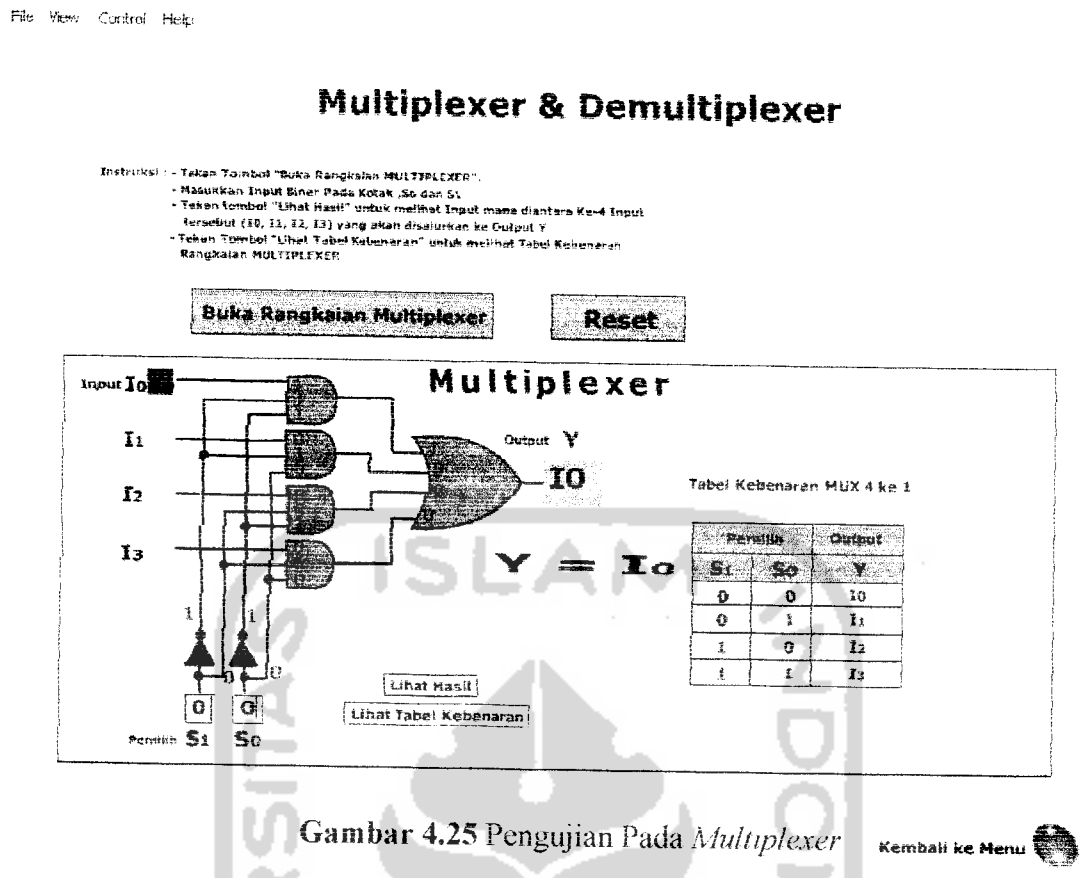
Selanjutnya untuk pengujian *multiplexer* dan *demultiplexer*, user akan ditampilkan halaman menu *multiplexer* dan *demultiplexer*, user dapat melihat sekilas pengertian *multiplexer* dan *demultiplexer* kemudian user dapat mengakses rangkaian yang dipilih. Seperti yang ditunjukkan pada gambar 4.24.



Gambar 4.24 Halaman Menu *Multiplexer* dan *Demultiplexer*

4.2.1.1.5.1 Pengujian Pada *Multiplexer*

Pada *multiplexer* akan diuji dengan memasukkan sinyal pemilih S_0=0 dan S_1=1 maka seperti yang terlihat pada gambar 4.25 dibawah ini, *output* yang keluar dari *multiplexer* adalah Y = I_0. Itu artinya, S_0S_1= 00 yang bernilai 0 menyebabkan *input* yang bersesuaian dengan nilai pemilihnya yakni I_0 akan dipilih untuk disalurkan ke *output*nya sehingga Y=I_0. Begitu juga hasil yang terlihat pada tabel kebenarannya.



Gambar 4.25 Pengujian Pada *Multiplexer*

4.2.1.1.5.2 Pengujian Pada *Demultiplexer*

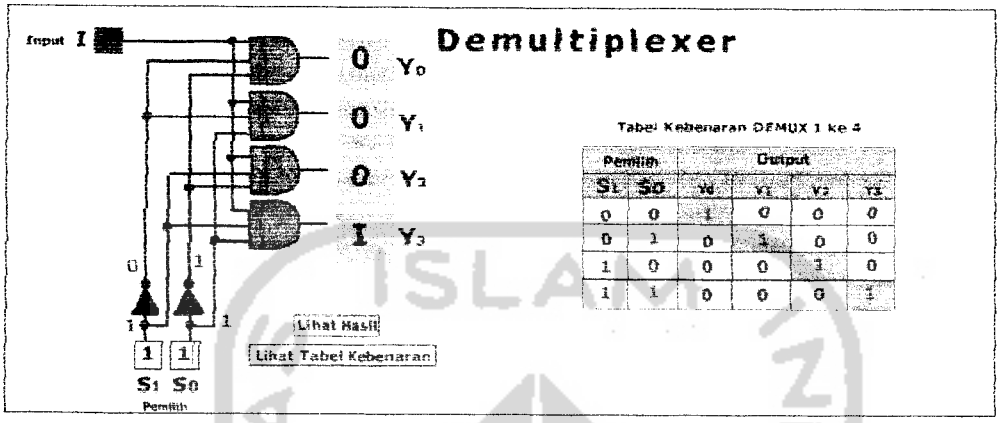
Pada de*multiplexer* akan diuji dengan memasukkan sinyal pemilih S_0=1 dan S_1=1 maka seperti yang terlihat pada gambar 4.26 dibawah ini, *output* yang keluar dari de*multiplexer* adalah Y_3 = I. Itu artinya, pemberian sinyal pemilih S_0S_1= 11 menyebabkan data I pada *input demultiplexer* disalurkan ke *output* 3 sehingga Y_3 = I dan *output* lainnya bernilai 0. Begitu juga hasil yang terlihat pada tabel kebenarannya.

File View Control Help

Multiplexer & Demultiplexer

Instruksi : - Tekan Tombol "Buka Rangkaian DEMULTIPLEXER"
- Masukkan Input Biner Pada Kotak S0 dan S1
- Tekan tombol "Lihat Hasil" untuk melihat Output mana diantara Ke-4 Output tersebut (Y0,Y1, Y2,Y3) yang akan menerima data dari Input (I)
- Tekan Tombol "Lihat Tabel Kebenaran" untuk melihat Tabel Kebenaran Rangkaian DEMULTIPLEXER

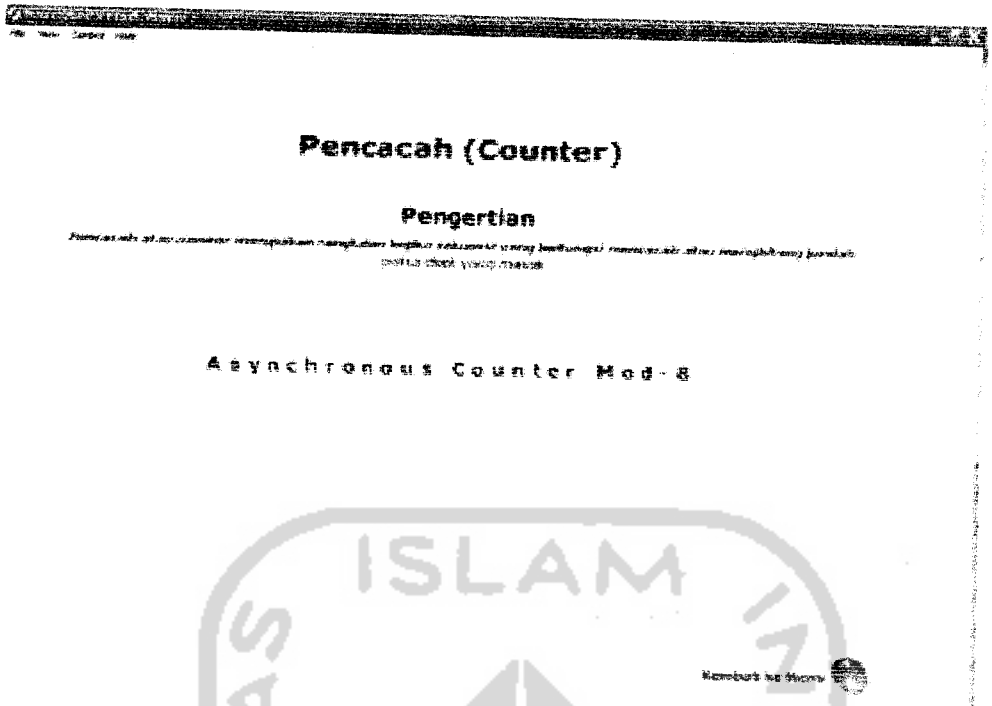
Buka Rangkaian DEMULTIPLEXER Reset



Gambar 4.26 Pengujian Pada *Demultiplexer*

4.2.1.1.6 Pengujian Pada Pencacah (*Counter*)

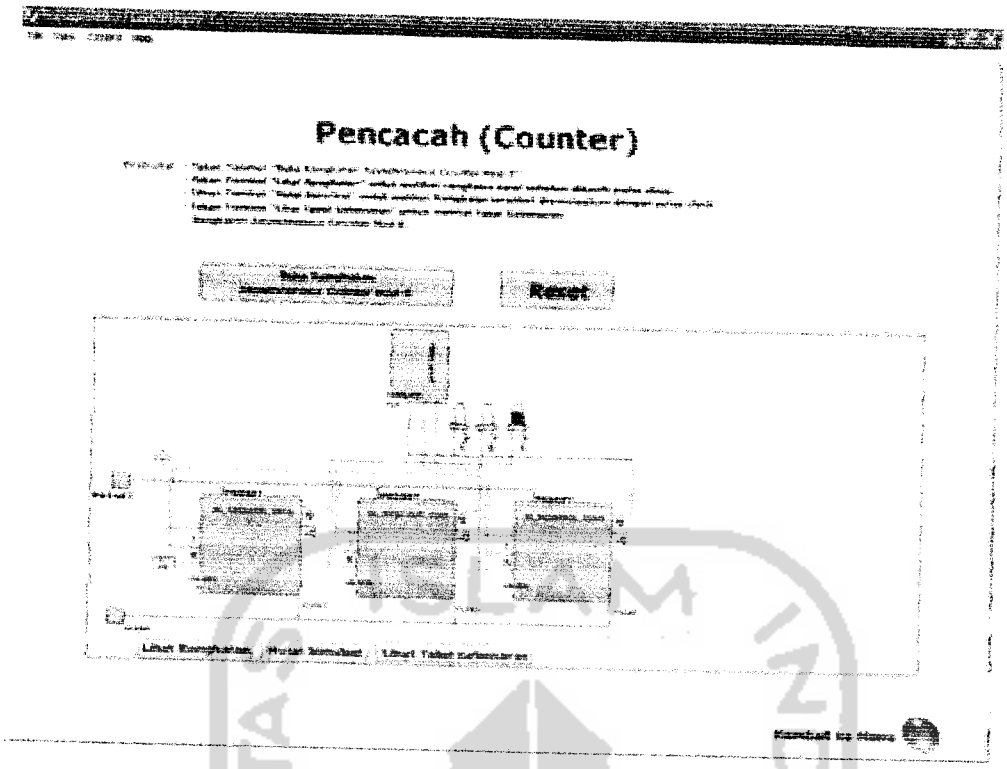
Selanjutnya untuk pengujian pencacah (*counter*), user akan ditampilkan halaman menu pencacah (*counter*), user dapat melihat sekilas pengertian pencacah (*counter*) kemudian user dapat mengakses rangkaian yang dipilih. Seperti yang ditunjukkan pada gambar 4.27.



Gambar 4.27 Halaman Menu Pencacah (*Counter*)

4.2.1.1.6.1 Pengujian Pada Pencacah (*Counter*) *Clock* ke-1

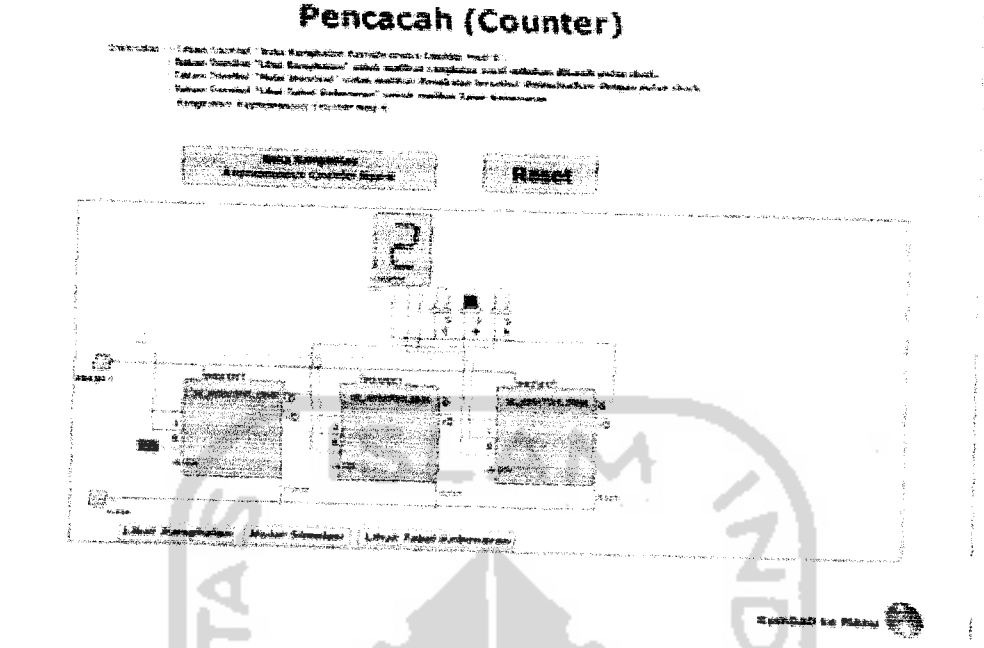
Anggap pada saat awal *output* pencacah ABC=000. Pada *clock* ke-1, flip-flop A terpicu sehingga A=1, dan *flip-flop* yang lain belum terpicu sehingga *output*nya masih bernilai 0. Pada keadaan ini *output* pencacah adalah ABC=100. Begitu juga hasil yang terlihat pada tabel kebenarannya. Seperti yang ditunjukkan pada gambar 4.28.



Gambar 4.28 Pengujian Pada Pencacah (*Counter*) Clock ke-1

4.2.1.1.6.2 Pengujian Pada Pencacah (*Counter*) *Clock* ke-2

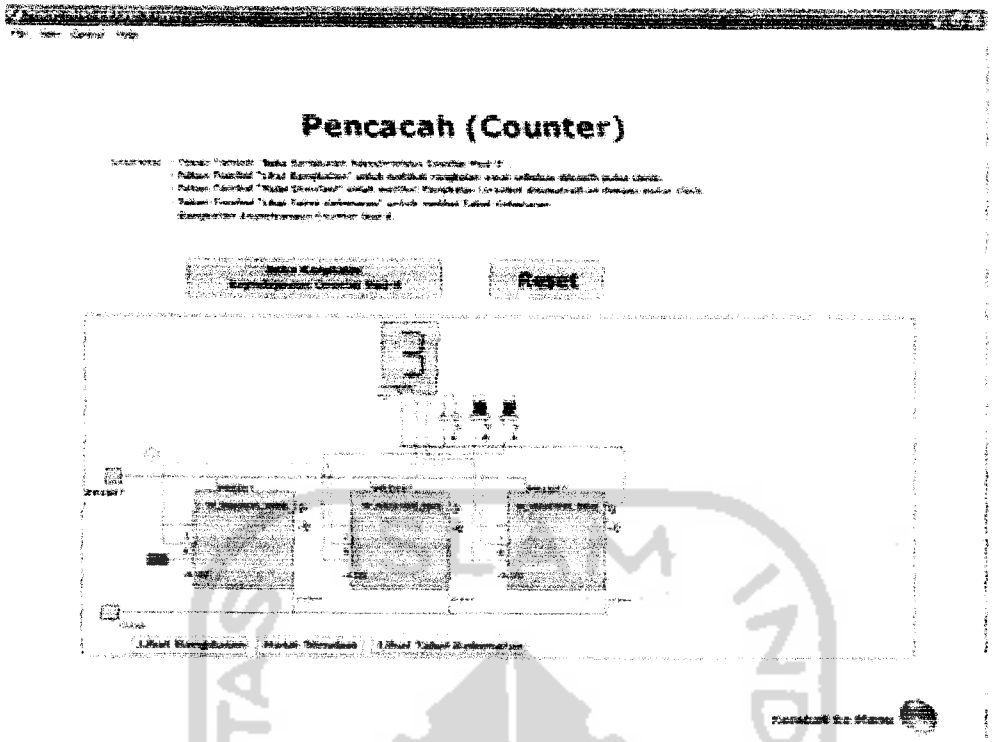
Pada *clock* ke-2 *output flip-flop* A akan membalik *output* sebelumnya sehingga menjadi rendah, dan perubahan ini akan memicu *flip-flop* B sehingga B=1, pada sisi lain *output* C tetap rendah karena belum terpicu. Untuk keadaan ini *output* pencacah menjadi ABC=010. Seperti yang ditunjukkan pada gambar 4.29.



Gambar 4.29 Pengujian Pada Pencacah (*Counter*) Clock ke-2

4.2.1.1.6.3 Pengujian Pada Pencacah (*Counter*) *Clock* ke-3

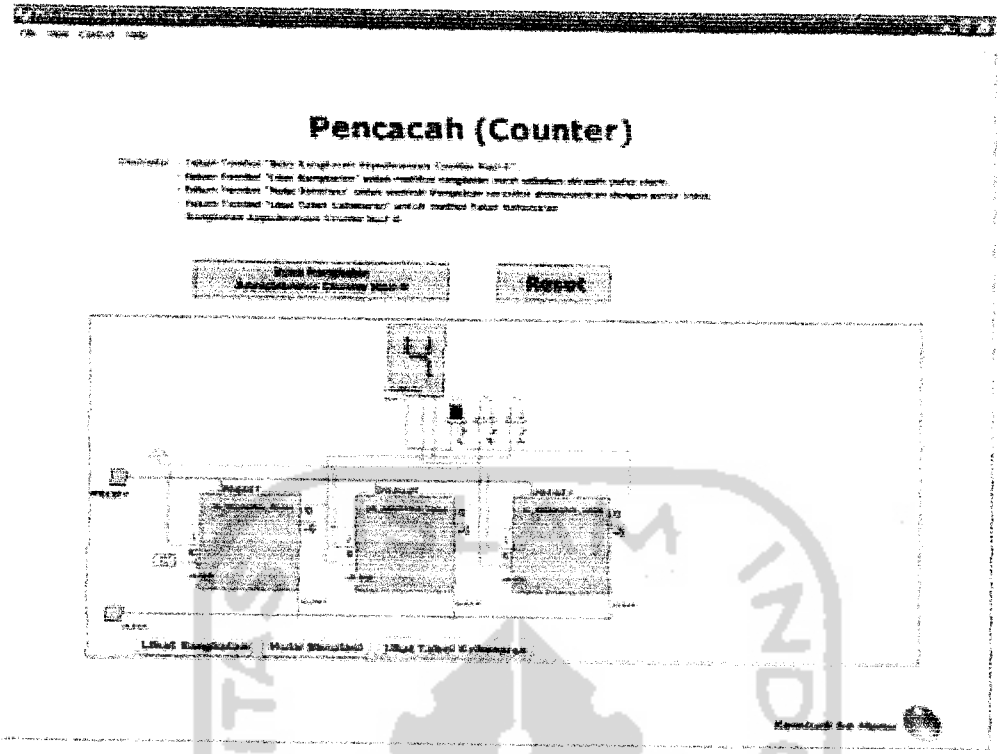
Selanjutnya pada *clock* ke-3 *flip-flop* A terpicu sehingga *output*nya berubah dari rendah ke tinggi menjadi A=1, *flip-flop* B tidak terpicu karena *output* A sebagai pemicunya berubah dari rendah ke tinggi sehingga B tetap tinggi. Demikian pula dengan *flip-flo*p C, karena belum terpicu *output*nya masih rendah sehingga untuk keadaan ini *output* pencacah menjadi ABC=110. Seperti yang ditunjukkan pada gambar 4.30.



Gambar 4.30 Pengujian Pada Pencacah (*Counter*) Clock ke-3

4.2.1.1.6.4 Pengujian Pada Pencacah (*Counter*) *Clock* ke-4

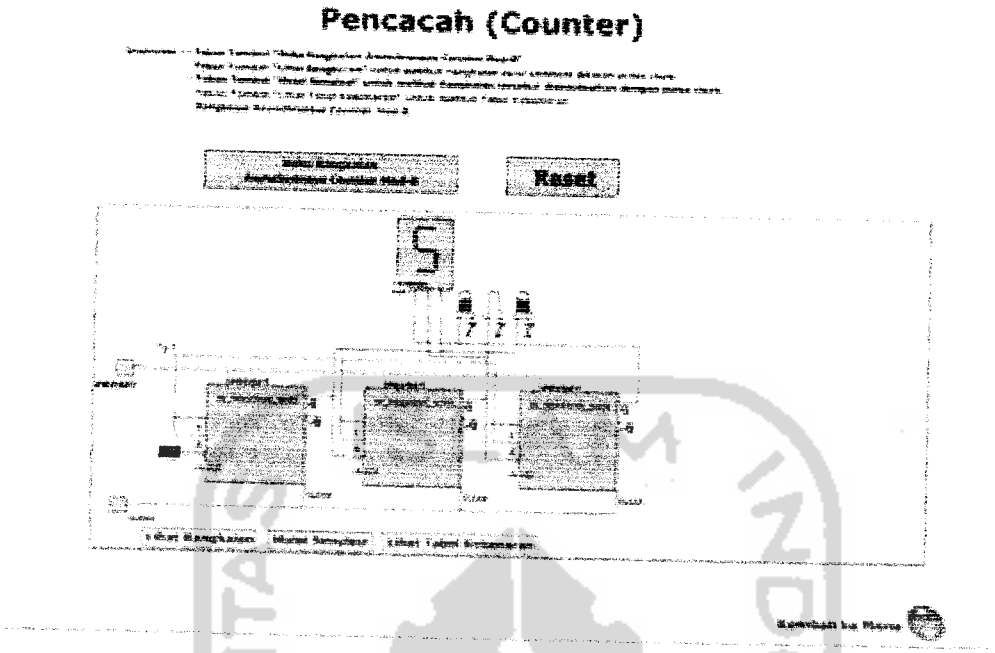
Pada *clock* ke-4 *flip-flop* A terpicu sehingga *output*nya terbalik menjadi A=0, dan perubahan ini memicu *flip-flop* B sehingga *output* B juga terbalik menjadi B=0. Karena B berubah dari tinggi ke rendah maka *output*nya memicu *flip-flop* C sehingga C berubah menjadi C=1. pada keadaan ini *output* pencacah menjadi ABC=001. Seperti yang ditunjukkan pada gambar 4.31.



Gambar 4.31 Pengujian Pada Pencacah (*Counter*) Clock ke-4

4.2.1.1.6.5 Pengujian Pada Pencacah (*Counter*) *Clock* ke-5

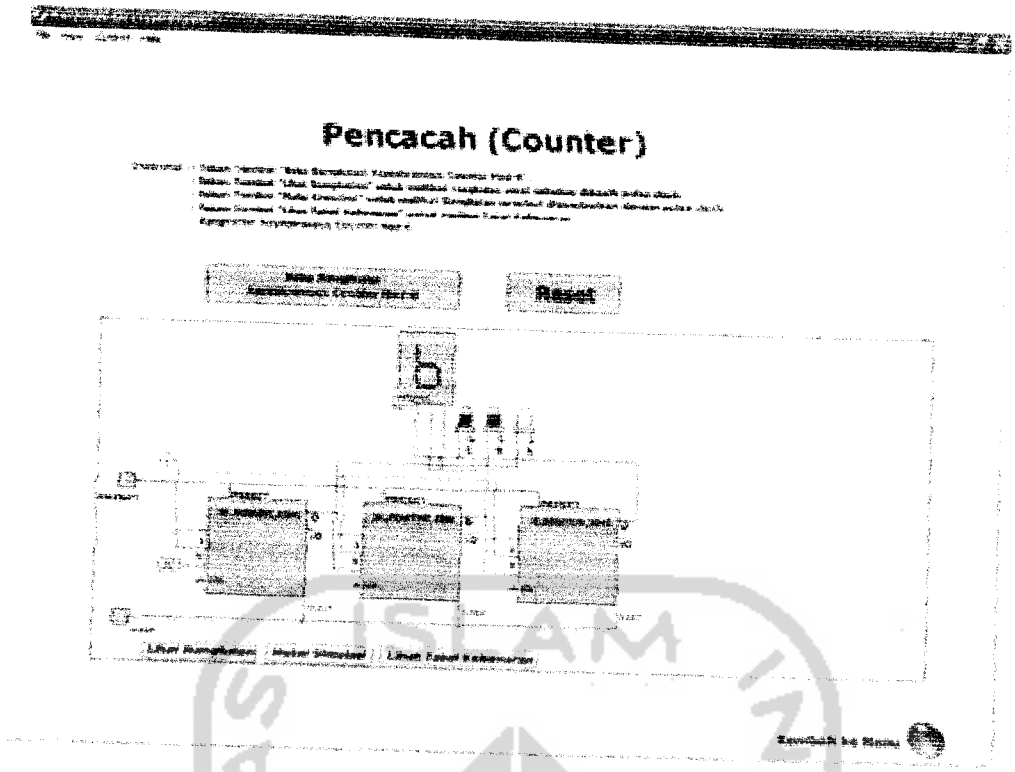
Pada *clock* ke-5, *flip-flop* A terpicu sehingga *output*nya berubah dari A=0 menjadi A=1, *flip-flop* B dan C tidak terpicu sehingga *output*nya tetap B=0 dan C=1. untuk keadaan ini *output* pencacah adalah ABC=101. Seperti yang ditunjukkan pada gambar 4.32.



Gambar 4.32 Pengujian Pada Pencacah (*Counter*) Clock ke-5

4.2.1.1.6.6 Pengujian Pada Pencacah (*Counter*) *Clock* ke-6

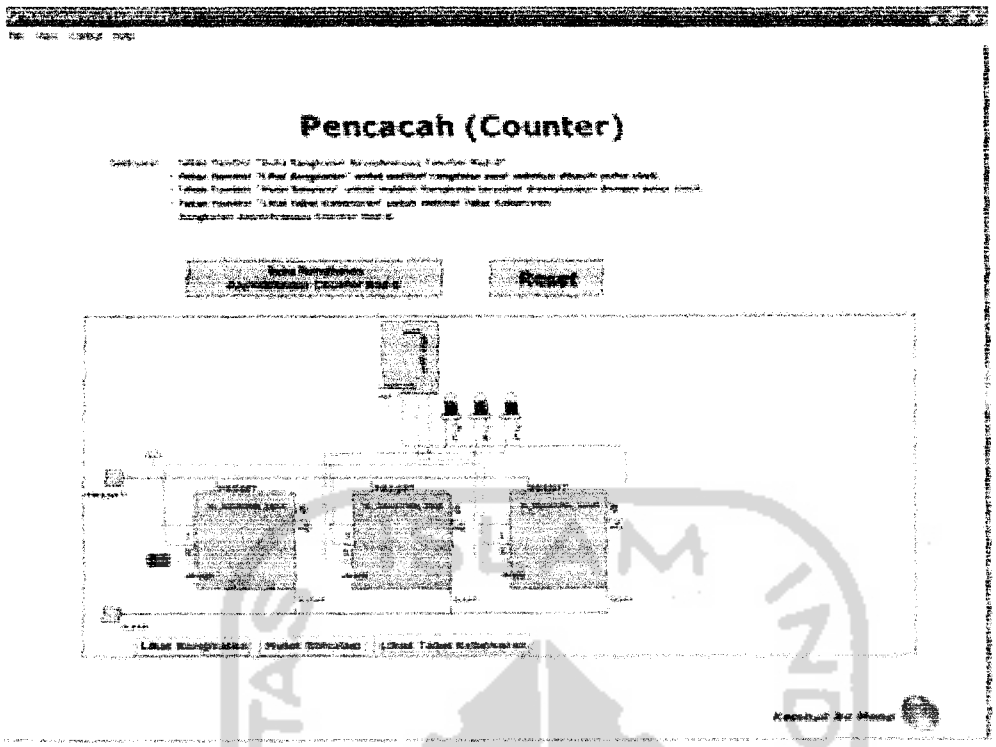
Pada *clock* ke-6, *flip-flop* A terpicu sehingga A=0, *output flip-flop* A memicu *flip-flop* B sehingga *output*nya berubah menjadi B=1, dan *flip-flop* C tidak terpicu sehingga C=1. Pada keadaan ini *output* pencacah menjadi ABC=011. Seperti yang ditunjukkan pada gambar 4.33.



Gambar 4.33 Pengujian Pada Pencacah (*Counter*) Clock ke-6

4.2.1.1.6.7 Pengujian Pada Pencacah (*Counter*) *Clock* ke-7

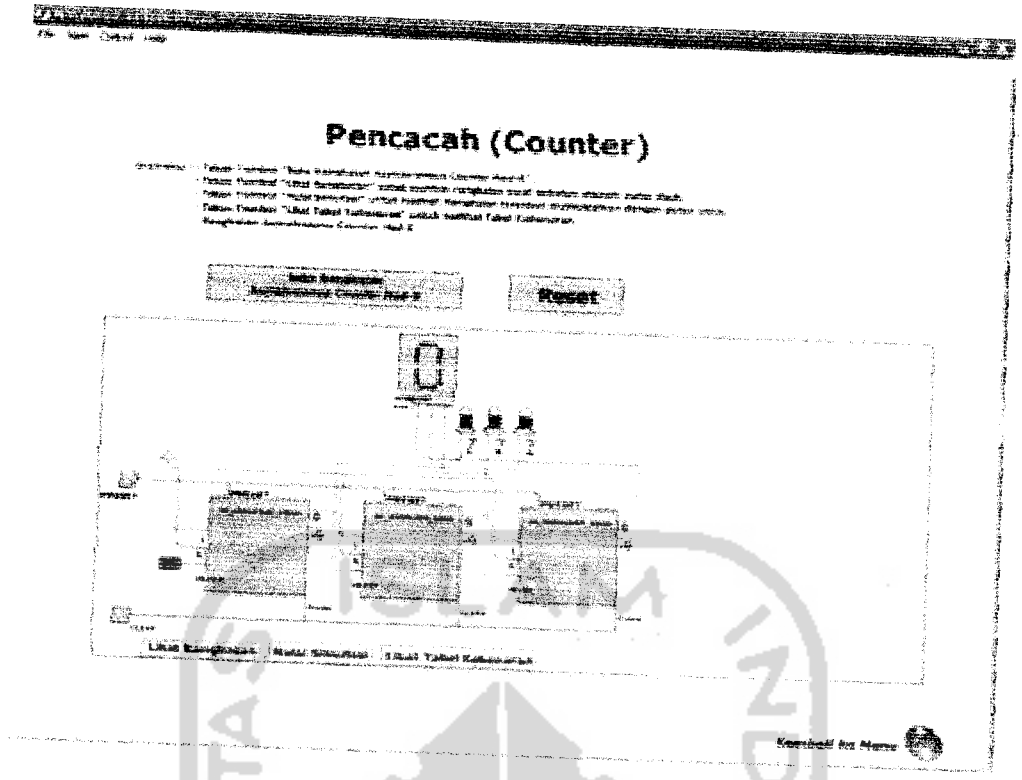
Pada *clock* ke-7, *flip-flop* A terpicu sehingga *output*nya A=1, B tetap, C tetap, dan *output* pencacah menjadi ABC=111. Seperti yang ditunjukkan pada gambar 4.34.



Gambar 4.34 Pengujian Pada Pencacah (*Counter*) Clock ke-7

4.2.1.1.6.8 Pengujian Pada Pencacah (*Counter*) *Clock* ke-8

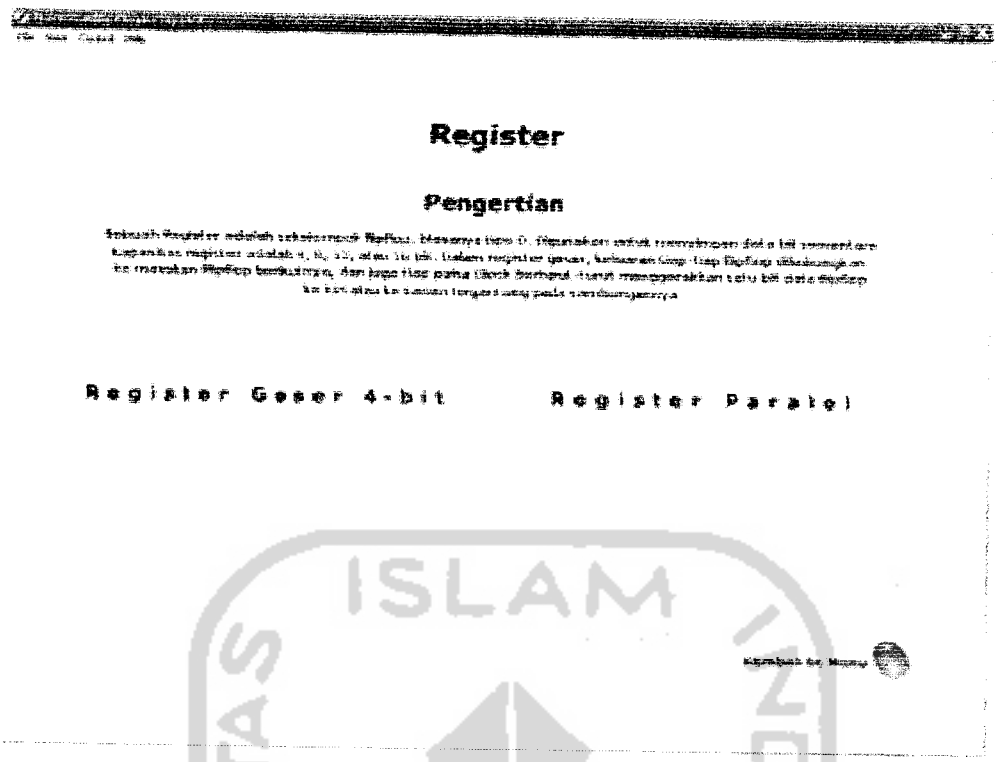
Pada saat terjadinya *clock* ke-8, ketiga *flip-flop* terpicu, dan karena keadaan *output* awalnya tinggi maka akan berubah menjadi *reset*. Keadaan tersebut menyebabkan *output* pencacah menjadi ABC=000. Seperti yang ditunjukkan pada gambar 4.35.



Gambar 4.35 Pengujian Pada Pencacah (*Counter*) Clock ke-8

4.2.1.1.7 Pengujian Pada *Register*

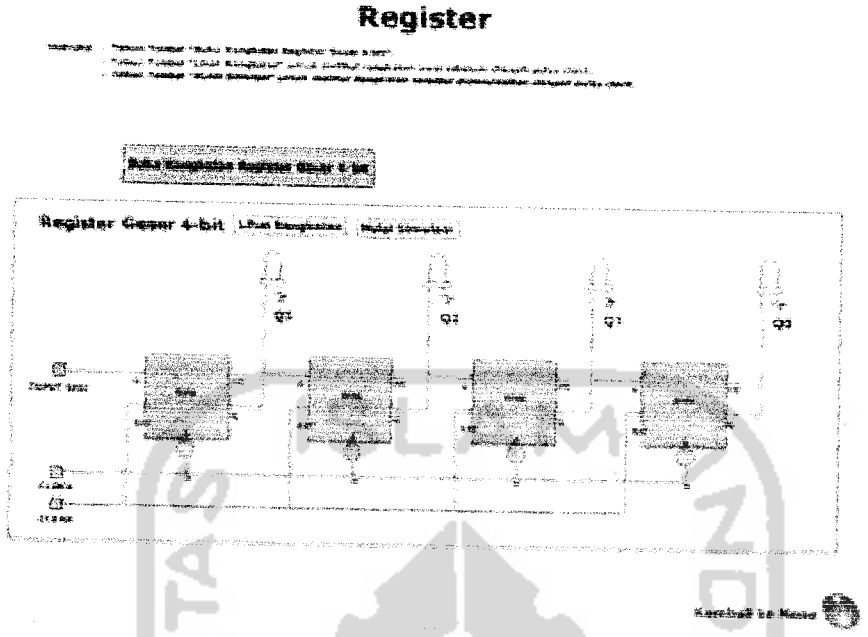
Selanjutnya untuk pengujian *register*, user akan ditampilkan halaman menu *register*, user dapat melihat sekilas pengertian *register* kemudian user dapat mengakses rangkaian yang dipilih. Seperti yang ditunjukkan pada gambar 4.36.



Gambar 4.36 Halaman Menu *Register*

4.2.1.1.7.1 Pengujian Pada *Register* Geser

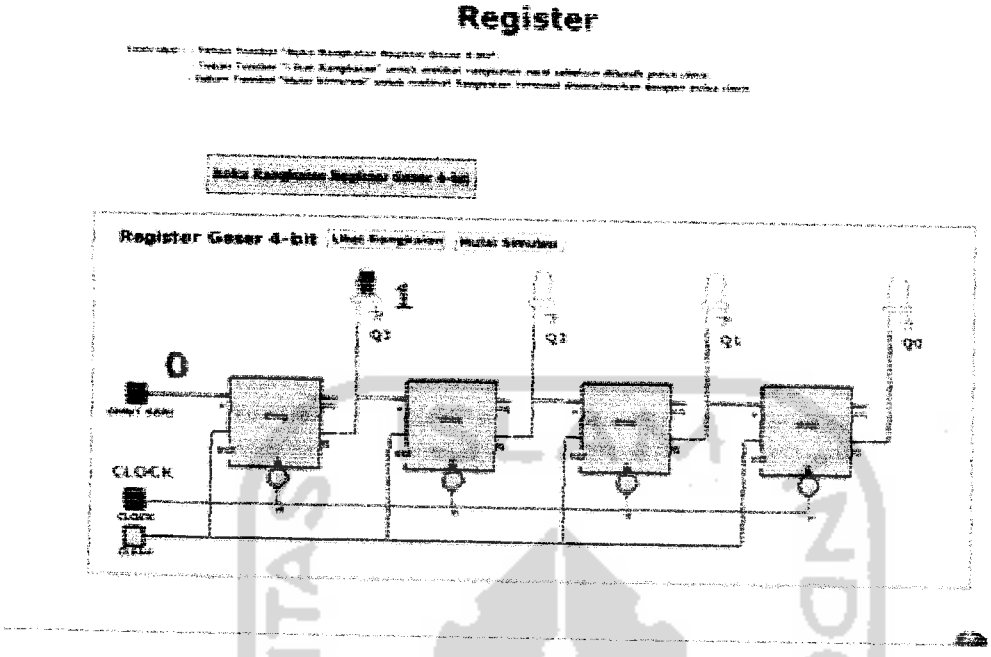
Anggap data yang akan disimpan adalah 1101 dan keadaan mula-mula isi *register* masih kosong schingga $Q_3Q_2Q_1Q_0$ = 0000. Mekanisme penyimpanan datanya dilakukan dengan memasukkan terlebih dahulu bit LSB (*Least Significant Bit*) dari data yang akan disimpan ke bagian elemen MSB (*Most Significant Bit*) *register*. Untuk memulai penyimpanan data dipasang terlebih dahulu data MSB yakni 1 pada *input* seri. Seperti yang ditunjukkan pada gambar 4.37.



Gambar 4.37 Rangkaian *Register* Geser

4.2.1.1.7.1.1 Pengujian Pada *Register* Geser *Clock* ke-1

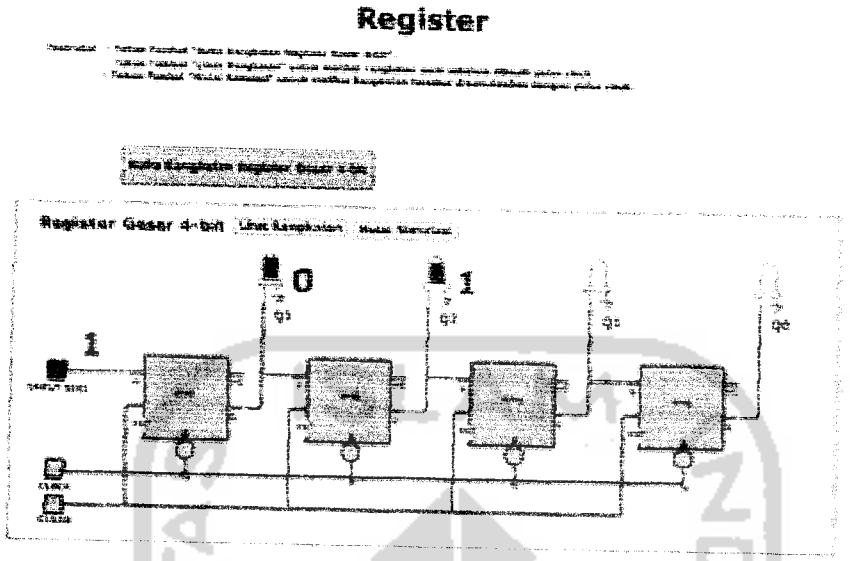
Pada pulsa *clock* ke-1 akan memicu semua *input flip-flop*. Oleh karena pada *input flip-flop* D_3 terpasang data 1 maka Q_3=1. Pada sisi lain *input flip-flop* D_2, D_1 dan D_0 bernilai 0 sehingga $Q_3Q_2Q_1Q_0$ = 1000. Sebelum ada *clock* ke-2 dipasang lagi data berikutnya yakni 0 pada *input* seri. Seperti yang ditunjukkan pada gambar 4.38.



Gambar 4.38 Pengujian Pada *Register* Geser *Clock* ke-1

4.2.1.1.7.1.2 Pengujian Pada *Register* Geser *Clock* ke-2

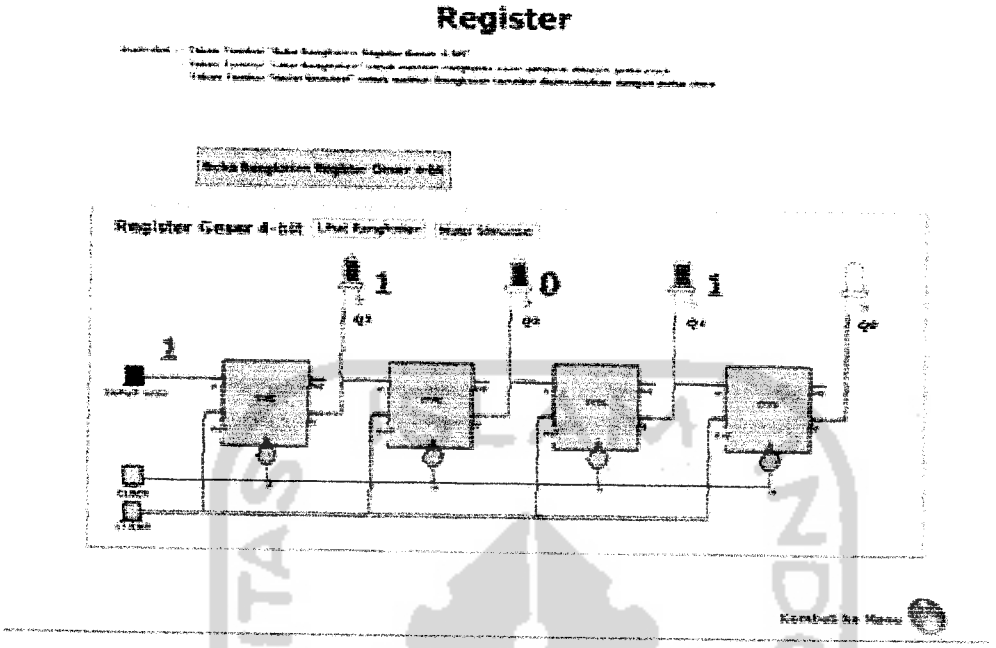
Saat *clock* ke-2, semua *input flip-flop* kembali terpicu. Oleh karena D_3=0 (berasal dari *input* seri), maka Q_3=0 dan Q_2=1. Pada sisi lain D_1 dan D_0 bernilai 0 sehingga $Q_3Q_2Q_1Q_0$= 0100. Keadaaan tersebut menyebabkan seolah-olah isi Q_3 digeser ke posisi Q_2. Seperti yang ditunjukkan pada gambar 4.39



Gambar 4.39 Pengujian Pada *Register* Geser *Clock* ke-2

4.2.1.1.7.1.3 Pengujian Pada *Register* Geser *Clock* ke-3

Sebelum ada *clock* ke-3, data berikutnya yakni 1 dipasang pada *input* seri, dan saat ada *clock* ke-3 semua *input flip-flop* kembali terpicu menyebabkan Q_3=1 (berasal dari *input* seri). Q_2=0 dan Q_1=1. Pada sisi lain Q_0=0 sehingga $Q_3Q_2Q_1Q_0$=1010. Keadaan tersebut menyebabkan seolah-olah isi Q_3 digeser ke Q_2 dan isi Q_2 digeser ke Q_1.Seperti yang ditunjukkan pada gambar 4.40.

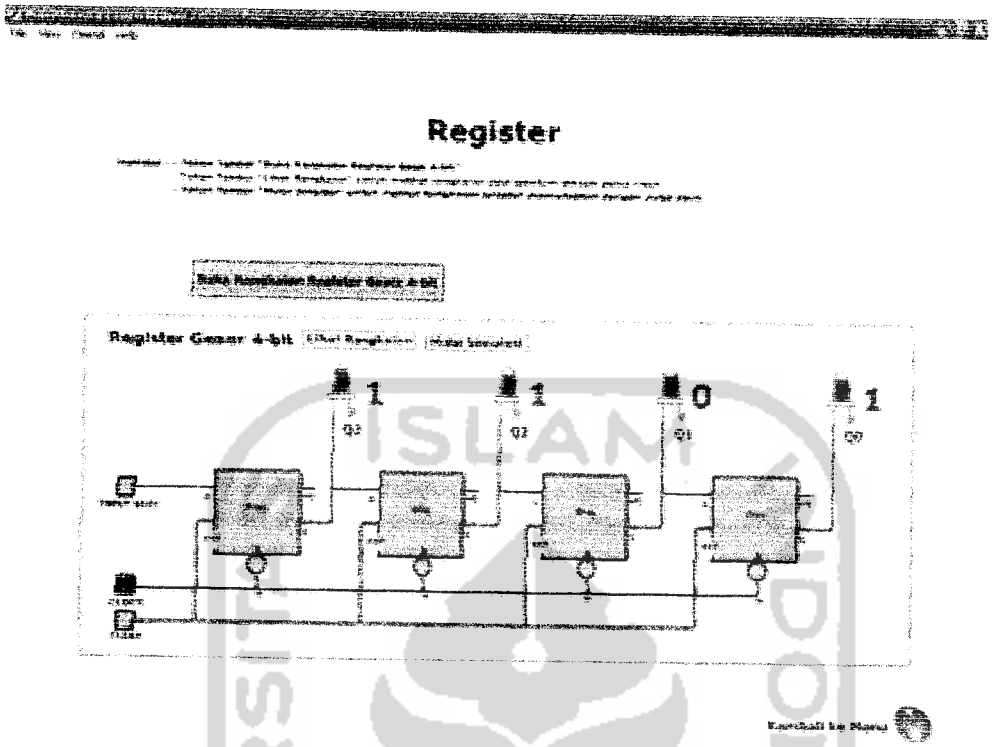


Gambar 4.40 Pengujian Pada *Register* Geser *Clock* ke-3

4.2.1.1.7.1.3.1 Pengujian Pada *Register* Geser *Clock* ke-4

Selanjutnya, sebelum *clock* ke-4, pada *input* seri dipasang data yang terakhir yakni 1. oleh karena keadaan sebelumnya Q_3=1, Q_2=0 dan Q_1=1 maka setelah *clock* ke-4 menjadikan Q_2=1, Q_1=0 dan Q_0=1. Pada sisi lain pemasangan data 1 pada *input* seri menyebabkan Q_3=1 sehingga $Q_3Q_2Q_1Q_0$= 1101. Keadaan ini menyebabkan seolah-olah isi Q_3 digeser ke Q_2, isi Q_2 digeser ke Q_1, dan isi Q_1 digeser ke Q_0.

Terlihat bahwa pada *register* geser 4-bit, penyimpanan data yang dilakukan memerlukan waktu sebanyak 4 siklus *clock*, sebab setelah *clock* ke-4 isi *register* sama dengan data yang dimasukkan yakni 1101. .Seperti yang ditunjukkan pada gambar 4.41.



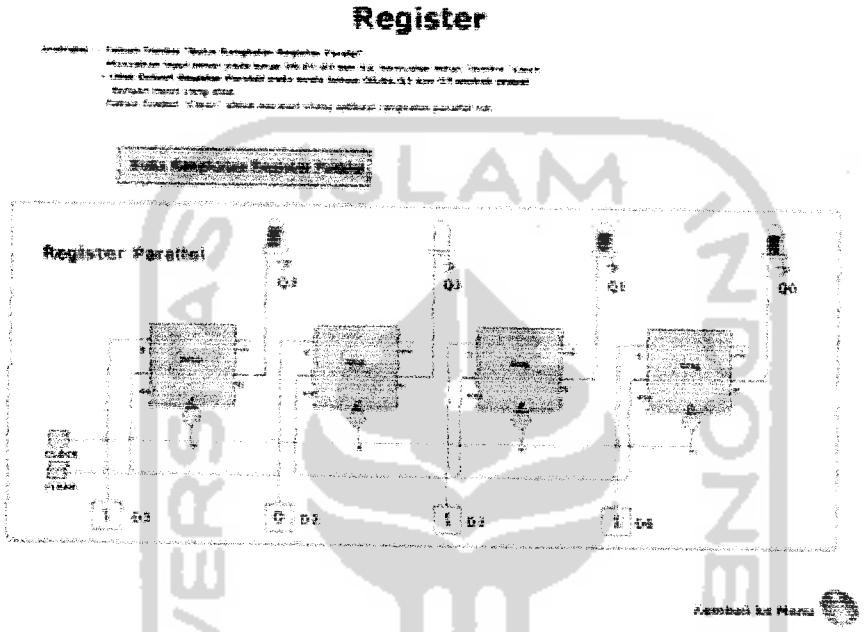
Gambar 4.41 Pengujian Pada *Register* Geser *Clock* ke-4

4.2.1.1.7.2 Pengujian Pada *Register* Paralel

Pada *register* ini data dimasukkan ke dalamnya secara serempak melalui saluran $D_3D_2D_1D_0$. Demikian pula ketika *register* tersebut akan dibaca *output*nya, data dikeluarkan secara serempak melalui $Q_3Q_2Q_1Q_0$. Prinsip penyimpanan data pada *register* adalah memindahkan data yang ada pada *input*nya ke *output*nya. Penyimpanan data pada *register* paralel dilakukan dengan cara menempatkan data yang akan disimpan pada *input* paralel, dan untuk memindahkan data tersebut ke *output*nya dilakukan dengan memberikan sebuah pulsa *clock*.

Anggap saja *register* tersebut akan menyimpan data 1011. Mula-mula data tersebut ditempatkan pada saluran *input register* yakni $D_3D_2D_1D_0$=1011, dan saat terjadi

pulsa *clock*, data dipindah ke *output* register sehingga $Q_3Q_2Q_1Q_0$ = 1011. Seperti yang ditunjukkan pada gambar 4.42.



Gambar 4.42 Pengujian Pada *Register* Paralel

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan hasil analisis dan implementasi program, maka dapat diambil kesimpulan sebagai berikut :

1. Telah berhasil dibangun suatu program membangun tabel kebenaran dengan logika matematika untuk aplikasi gerbang logika pada rangkaian elektronika..
2. Pada pengujian gerbang-gerbang logika dasar dapat disimpulkan bahwa gerbang-gerbang logika dasar merupakan elemen rangkaian digital dan rangkaian digital merupakan kesatuan dari gerbang-gerbang logika dasar yang membentuk fungsi pemrosesan sinyal digital.
3. Pada pengujian rangkaian *decoder* dan *encoder* dapat disimpulkan bahwa *decoder* dan *encoder* sangat penting fungsinya untuk menterjemahkan data baik oktal ke biner maupun biner ke bilangan desimal.
4. Pada pengujian rangkaian *half* dan *full adder* dapat disimpulkan bahwa *half* dan *full adder* berperan penting dalam operasi penjumlahan bilangan biner.
5. Pada pengujian rangkaian *flip –flop* dapat disimpulkan bahwa *flip-flop* berfungsi sebagai rangkaian penyimpan data 1-*bit*.
6. Pada pengujian *multiplexer* dan *demultiplexer* dapat disimpulkan bahwa rangkaian ini berperan sebagai penyaluran data pada rangkaian elektronika.
7. Pada pengujian rangkaian pencacah (*counter*) dapat disimpulkan bahwa rangkaian ini merupakan gabungan dari beberapa *flip-flop* yang berfungsi menghitung jumlah pulsa *clock* yang masuk.

8. Pada pengujian rangkaian *register* dapat disimpulkan bahwa rangkaian ini baik *register* geser maupun paralel berfungsi menyimpan data yang lebih dari 1-*bit*

5.2 Saran

Berdasarkan pada pengujian yang telah dilakukan pada perangkat lunak yang dibuat, masih terdapat banyak kekurangan dan kelemahan sehingga perlu dikembangkan lagi agar kinerja dan fungsi dari sistem dapat bertambah. Contoh yang nyata adalah rumitnya link-link tombol yang ada pada sistem, mungkin untuk *user* yang pertama kali mencoba akan kebingungan. Maka disarankan *user* setiap akan melakukan proses perhitungan untuk mambaca instruksi yang tersedia di setiap halaman aplikasi gerbang logika pada rangkaian elektronika ini.

DAFTAR PUSTAKA

[ARY04] Ary Maulana, Syarif. *Mastering Action Script Macromedia Flash MX 2004*, Elex Media Computindo, 2004

[CHA04] Chandra, *7 Jam Belajar Flash MX 2004 Untuk Orang Awam*, Maxicom :2004

[CHA06] Chandra, *Action Script Flash MX 2004 Untuk Profesional*, Maxicom :2006

[HAK04] Hakim, Lukmanul. *Cara Ampuh Menguasai Macromedia Flash MX 2004*, Elex Media Computindo, 2004

[MUC05] Muchlas, *Rangkaian Digital*, Yogyakarta : Penerbit Gava Media, 2005

[SOE06] Soesianto, F. *Logika Matematika untuk Ilmu Komputer*, Yogyakarta : Penerbit Andi, 2006.

[SUG04] Sugiri, *Elektronika Dasar & Peripheral Komputer*, Yogyakarta: Penerbit ANDI, 2004

[TRI05] Trikasjono, Toto. *Petunjuk Praktikum Elektronika Digital*, Yogyakarta : Sekolah Tinggi Teknologi Nuklir 2005

[ZEM04] Zembry, *Tip & Trik Action Script Flash MX 2004*, Elex Media Computindo,2004