SISTEM BERBASIS FPGA DENGAN METODE FUZZY PADA STUDI KASUS GAS BUANG MOTOR

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Mochamad Afief Hamidi
NIM: 115060900111015

PROGRAM STUDI TEKNIK INFORMATIKA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2016

PENGESAHAN

SISTEM BERBASIS FPGA DENGAN METODE FUZZY PADA STUDI KASUS GAS BUANG MOTOR

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Mochamad Afief Hamidi
NIM: 115060900111015

Skripsi ini telah diuji dan dinyatakan lulus pada
14 Januari 2016
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Eko Setiawan, S.T, M.Eng	M. Hannats Hanafi I, S.T, M.T
NIK. 201102 870610 1 001	NIK. 201405 881229 1 001

Mengetahui
Ketua Program Studi Teknik Informatika

Drs. Marji, M.T.
NIP: 19670801 199203 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 14 Januari 2016

Mochamad Afief Hamidi

NIM: 115060900111015

KATA PENGANTAR

Puji syukur kehadirat Allah SWT, atas limpahan Rahmat dan Karunia-Nya, sehingga penulis dapat merampungkan skripsi dengan judul: Sistem Berbasis FPGA dengan Metode Fuzzy pada Studi Kasus Gas Buang Motor. Ini untuk memenuhi salah satu syarat menyelesaikan studi serta dalam rangka memperoleh gelar Sarjana Pendidikan Strata Satu pada Program Studi Sistem Komputer Fakultas Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya. Penghargaan dan terima kasih yang setulus-tulusnya kepada Ayahanda tercinta Achmad Yani dan Ibunda yang kusayangi Ganti Choliwati semoga Allah SWT melimpahkan Rahmat, Kesehatan, Karunia dan keberkahan di dunia dan di akhirat atas budi baik yang telah diberikan kepada penulis.

Penulis menyadari bahwa skripsi ini tidak akan berhasil tanpa bantuan dari beberapa pihak. Oleh karena itu, penulis ingin menyampaikan rasa hormat dan terima kasih kepada:

1. Bapak Prof. Dr. Ir. Mohammad Bisri, MS selaku Rektor Universitas Brawijaya
2. Bapak Ir Sutrisno, M.T. selaku Dekan Fakultas Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
3. Bapak Eko Setiawan, S.T., M.T. M.Eng selaku pembimbing I yang telah banyak membantu dalam pengerjaan skripsi ini.
4. Bapak M. Hannats Hanafi I, S.S.T., M.T. selaku pembimbing II yang telah banyak membantu dalam pengerjaan skripsi ini.
5. Terima kasih kepada teman-teman kontrakan, Ahonk, Bagus, Labib, Alam, Yuda, Ghayuh yang membantu dalam pengerjaan skripsi.

Seluruh teman-teman Teknik Komputer 2011 yang telah menemani selama perkuliahan dan seluruh pihak yang tidak dapat disebutkan namanya satu persatu.

Malang, 14 Januari 2016

Mochamad Afief Hamidi

ABSTRAK

Dewasa ini berbagai macam alat digunakan menggunakan teknologi mikroprosesor. Sebagian besar alat yang ada menggunakan arduino maupun atmega. Sedangkan untuk FPGA sedikit pengguna yang memakai alat tersebut. FPGA menggunakan pemrograman bahasa VHDL. VHDL atau VHSIC (Very High Speed Integrated Circuit Hardware Description Language) merupakan salah satu jenis bahasa HDL yang digunakan untuk mendeskripsikan fungsi rangkaian digital seperti FPGA maupun gerbang logika. Logika *fuzzy* telah banyak diterapkan dalam bidang kendali otomatis dan juga system pendukung keputusan. Oleh sebab itu dimbuat sistem berbasis FPGA dengan metode fuzzy pada studi kasus gas buang pada motor agar dapat mengetahui nilai ambang batas emisi gas buang.

Untuk hasil dari sistem yang dibuat dengan perhitungan manual tingkat akurasi sebesar 52%. Dari hasil pengujian metode fuzzy yang diimplementasikan pada FPGA dihasilkan waktu eksekusi program sebesar 45000 ns dan memberikan keputusan termasuk dalam kategori aman, waspada atau bahaya berdasarkan gas HC dan CO.

Kata kunci: emisi gas buang motor, VHDL, Fuzzy, FPGA.

ABSTRACT

Today a variety of tools using microprocessor technology. Most of the existing tools are use arduino or atmega. FPGA there is a few users who use these tools. FPGA using VHDL language programming. VHDL (Very High Speed Integrated Circuit Hardware Description Language) is one type of HDL language used to describe the function of a digital circuit such as FPGA and logic gates. Fuzzy logic has been widely applied in the field of automatic control and decision support systems. Therefore use FPGA-based systems with fuzzy method in a case study on motorcycle exhaust gases in order to determine the threshold value of exhaust emissions.

The results of the system compared by manual calculation the accuracy rate is 52%. From the results, the testing methods that are implemented on an FPGA fuzzy generated program execution time of 45000 ns and provide decision are included in the category of safe, alert or danger based on HC and CO gas.

Keywords: motorcycle exhaust emissions, VHDL, Fuzzy, FPGA.

DAFTAR ISI

Contents

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Dewasa ini berbagai macam alat digunakan menggunakan teknologi mikroprosesor. Sebagian besar alat yang ada menggunakan arduino maupun atmega. Sedangkan untuk FPGA sedikit pengguna yang memakai alat tersebut. FPGA (*Field-programmable Gate Arrays*) merupakan sebuah IC digital yang digunakan untuk mengimplementasikan rangkaian-rangkaian digital. FPGA terdiri dari komponen elektronika dan semikonduktor yang terdiri atas komponen gerbang meliputi jenis gerbang logika biasa (AND, OR, NOT) maupun jenis fungsi matematis dan kombinasi yang lebih kompleks seperti decoder, adder, subtractor maupun multiplier. Keuntungan dari mengimplementasi menggunakan FPGA untuk meningkatkan efisiensi rancangan dengan cara mengurangi pemakaian pemrograman perangkat keras. FPGA mempunyai koreksi error yang sangat kecil dan merupakan teknologi yang bebas untuk diimplementasikan dengan berbagai algoritma.

FPGA menggunakan pemrograman bahasa VHDL. VHDL atau VHSIC Very High Speed Integrated Circuit Hardware Description Language merupakan salah satu jenis bahasa HDL yang digunakan untuk mendeskripsikan fungsi rangkaian digital seperti FPGA maupun gerbang logika. VHDL juga bisa digunakan sebagai bahasa pemrograman untuk simulasi rangkaian dari macam-macam komponen digital. VHDL untuk menangkap desain dan VHDL berbasis sintesis logika adalah metode yang efisien untuk merancang hardware yang kompleks. Untuk menggambarkan struktur tersebut, kami mempekerjakan Statecharts. Selain itu, alat komersial berdasarkan Statecharts menggabungkan fasilitas generasi VHDL untuk menghasilkan kode disintesis. Kami telah mempekerjakan Statecharts menangkap basis aturan sistem kontrol fuzzy. Menggunakan pendekatan tingkat tinggi ini, waktu desain berkurang secara signifikan, dan desain yang berbeda dari basis aturan dapat dieksplorasi dalam waktu singkat. Pada fuzzifikasi dan defuzzifikasi bagian dari sistem, yang menggabungkan banyak perhitungan matematis, dijelaskan dalam VHDL sebagai desain tangan-kode. Kami merakit sistem kontrol fuzzy, dan mensintesis deskripsi tingkat gerbang untuk field programmable gate array (FPGA) teknologi (Salapura,1996).

Logika *fuzzy* telah banyak diterapkan dalam bidang kendali otomatis dan juga system pendukung keputusan. Banyak sistem kendali dengan menggunakan logika *fuzzy*, karena proses kendalinya yang relatif mudah dan fleksibel dirancang dengan tidak melibatkan model matematis yang rumit dari sistem yang akan dikendalikan[3]. Pengambilan keputusan sebaiknya dilakukan berdasarkan pertimbangan-pertimbangan tertentu agar keputusan yang diambil dapat bersifat objektif, karena pengambilan keputusan yang dilakukan tanpa pertimbangan yang benar dapat mengakibatkan keputusan yang diambil menjadi kurang objektif. Pengambilan keputusan dalam teknik fuzzy dilakukan dalam beberapa tahap yaitu: *fuzzification*, penentuan *membership function*, *rule*

evaluation dan *defuzzification*. Pada rule evaluation akan dilakukan pencarian fuzzy output dari fuzzy input dengan proses dimana suatu nilai fuzzy input yang berasal dari proses fuzzification kemudian dimasukkan kedalam sebuah rule yang telah dibuat untuk dijadikan sebuah fuzzy output. Rule evaluation merupakan bagian utama dari fuzzy karena rule evaluation akan menjadi dasar untuk menentukan sistem menjadi pintar atau tidak. Rule yang sudah ditentukan kemudian diaplikasikan kedalam fungsi implikasi. Secara umum ada 2 fungsi implikasi yaitu: *Max-Min* dan *Max-Prod*, fungsi implikasi yang sering digunakan dalam pengambilan keputusan adalah *Max-Min* (Purnomo,2004).

Oleh sebab itu penulis ingin mengimplementasikan logika fuzzy pada VHDL pada studi kasus gas buang motor yang mana dapat menentukan apakah tingkat emisi pada kendaraan tersebut berbahaya atau tidak.

1.2 Rumusan masalah

Berdasarkan uraian latar belakang, dapat dirumuskan permasalahan pada penelitian sebagai berikut:

1. Bagaimana merancang simulasi emisi gas buang pada kendaraan bermotor menggunakan FPGA menggunakan bahasa vhdl.
2. Bagaimana mengimplementasikan metode fuzzy pada FPGA.
3. Berapa besar tingkat akurasi sistem yang dibuat dengan perhitungan matematis yang dilakukan secara manual.

1.3 Tujuan

Tujuan penulisan skripsi ini adalah:

1. Untuk mengimplementasikan sistem emisi kendaraan bermotor pada FPGA menggunakan bahasa vhdl
2. Untuk mengimplementasi metode fuzzy pada FPGA
3. Untuk mengetahui tingkat akurasi atau ketepatan sistem yang dibuat.

1.4 Manfaat

Manfaat yang didapat dari pembuatan skripsi ini adalah:

1. Untuk menambah wawasan tentang FPGA serta bahasa vhdl
2. Untuk referensi penelitian tentang FPGA maupun bahasa vhdl

1.5 Batasan masalah

Batasan masalah dari sistem yang akan di buat adalah sebagai berikut :

1. Bahasa pemrograman yang digunakan adalah VHDL.
2. Simulasi dilakukan menggunakan software Xilinx ISE Design Suite 13.4.
3. Menggunakan nilai sample keluaran gas buang kendaraan bermotor sebagai model.
4. Menggunakan 2 parameter sebagai parameter input (HC dan CO) dan 3 parameter output(aman, waspada, bahaya).

1.6 Sistematika pembahasan

Penulisan tugas akhir ini terdiri dari beberapa bab, yang dijelaskan sebagai berikut:

BAB I : PENDAHULUAN

Bab ini berisi latar belakang masalah, tujuan, dan manfaat pembuatan tugas akhir, permasalahan, batasan masalah, dan sistematika penusunan tugas akhir.

BAB II : TINJAUAN PUSTAKA DAN DASAR TEORI

Bab ini membahas yentang dasar teori yang terkait dengan topik penulisan skripsi yang diangkat dan menjadi acuan dasar dalam pembuatan sistem pengklasifikasian teks ini.

BAB III : METODOLOGI PENELITIAN

Pada bab ini dijelaskan metode-metode yang digunakan dalam menyelesaikan masalah, dan perancanga sistem yang akan dibuat.

BAB IV : PERANCANGAN DAN IMPLEMENTASI

Bab ini membahas perancangan dari sistem yang akan dibuat dan implementasi dari desain sistem disertai dengan potongan source code yang penting dalam sistem tersebut dan membahas uji coba dari sistem yang dibuat dengan melihat *output* yang dihasilkan oleh sistem.

BAB V : ANALISIS DAN PEMBAHASAN

Bab ini berisi hasil uji coba sistem apakah dapat diterima dengan baik oleh user dan analisis hasil pengujian apakah sistem yang dibuat berhasil.

BAB VI : KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dari hasil ujicoba yang dilakukan serta saran untuk pengembangan sistem selanjutnya.

BAB 2 KAJIAN PUSTAKA DAN LANDASAN TEORI

Bab ini berisi tinjauan pustaka yang meliputi kajian pustaka dan dasar teori yang diperlukan.Kajian pustaka membahas penelitian-penelitian yang telah ada dan yang diusulkan.Dasar teori membahas teori yang diperlukan untuk menyusun penelitian yang diusulkan.

2.1 Kajian Pustaka

Pada penelitian sebelumnya yang dilakukan Velentina Sapalura pada percobaan tersebut menggunakan fuzzy pada VHDL. Percobaan tersebut menghasilkan output besarnya katup gas yang membuka. Pada penelitian tersebut menggunakan fuzzy sugeno untuk menentukan output (Valentina,1996). Pada penelitian tersebut menggunakan 2 input 1 output yang mana setiap input memiliki 3 kondisi. Sedangkan output memiliki 5 kondisi. Input yang digunakan dalam penelitian tersebut dari sensor temperatur dan sensor gas. Pada sensor temperatur dibagi menjadi 3 keanggotaan yaitu *cold, timid dan warm*. Sedangkan untuk sensor gas dibagi menjadi 3 keanggotaan yaitu *poor*, *medium*, *high.* Dari 2 input tersebut dibuat 9 rule. Untuk output dibagi menjadi lima keanggotaan yaitu *tiny, small,half,large, dan huge*. Pada penelitian tersebut waktu delay untuk tiap proses fuzzy yaitu 4,75ns untuk fuzzifikasi 6,24ns untuk rule dan 4,75ns untuk defuzzifikasi.

Penelitian lainnya dilakukan oleh zati azmiana pada penelitian tersebut menggunakan fuzzy mamdani untuk menentukan jurusan di SMA Negeri 1 Bireuen. Peneliti menggunakan nilai raport kelas 10 untuk dijadikan input dengan cara mengurutkan nilai kemudian membentuk himpunan fuzzy dan kemudian menentukan rule dan dilanjutkan dengan defuzzifikasi berupa jurusan apa yang ditentukan (Azmiana,2013). Pada penelitian tersebut terdapat 5 input yaitu nilai IPA, nilai IPS, IQ, minat dan kapasitas. Sedangkan untuk output ada 2 yaitu IPA atau IPS. Setiap input terdapat 3 keanggotaan. Untuk proses implikasi rule menggunakan metode min. Kemudian menggunakan metode max untuk mencari komposisi aturan. Pada defuzzifikasi menggunakan metode centoid. Untuk keputusan diambil dari membandingkan nilai output IPA dengan IPS.

2.2 Dasar Teori

Pada dasar teori ini akan membahas tentang dasar-dasar penulis dalam membuat sistem. Dasar teori yang digunakan oleh penulis antara lain: dasar logika fuzzy.

2.2.1 Logika *Fuzzy*

Pada logika fuzzy ini membahas tentang unsur-unsur yang terdapat didalam logika fuzzy. Dengan mempelajari setiap unsure yang ada didalam logika fuzzy maka akan mempermudah untuk mengaplikasikan logika fuzzy tersebut.

2.2.1.1 Dasar Logika Fuzzy

Logika adalah ilmu yang mempelajari secara sistematis aturan-aturan penalaran yang absah (*valid*). Logika fuzzy ini merupakan sebuah metode yang tepat untuk memetakan ruang lingkup input dan juga output.

Logika fuzzy sudah banyak diterapkan kedalam sistem otomatis dan juga system pendukung keputusan. Logika yang digunakan adalah logika dalam kehidupan sehari-hari sehingga mudah untuk dipahami.

2.2.1.2 Variabel Numeris dan Linguistik

Variabel adalah suatu lambang atau kata yang menunjukkan kepada sesuatu yang tidak tertentu dalam semesta pembicaraannya. Variabel merupakan unsur yang akan menjadi pembicaraan yang ada di logika fuzzy Suatu variabel dapat diganti oleh unsur-unsur dalam semesta pembicaraannya, ada dua macam variabel di logika *fuzzy* antara yaitu variabel numerik dan variabel linguistic.

Variabel numerik adalah suatu variabel yang semesta pembicaraannya berupa himpunan bilangan-bilangan. Misalnya *x* adalah bilangan yang habis dibagi 4.Variable "*x*" adalah suatu variabel numerik karena menunjuk sesuatu yang tidak tentu dalam semesta pembicaraannya yaitu himpunan bilangan. Variabel linguistik adalah suatu yang pembicaraanya berupa himpunan kata atau istilah yang di pakai sehari hari. Misalnya: sedikit, sedang, banyak, pelan, sedang, cepat dan seterusnya.

2.2.2 Himpunan Fuzzy

Himpunan adalah suatu kumpulan atau koleksi objek-objek yang mempunyai kesamaan sifat tertentu. Himpunan *fuzzy* adalah besar rentang nilai, yang setiap nilainya mempunyai derajat keanggotaan antara 0 sampai 1. Suatu himpunan *fuzzy A* dalam semesta pembicaraan U dinyatakan dengan fungsi keanggotaan *A*, yang nilainya berada dalam interval [0,1], dapat dinyatakan pada persamaan (2.1).

$$A : U \rightarrow [0,1] \qquad (2.1)$$

Didalam himpunan fuzzy terdapat beberapa bagian antara lain variabel fuzzy, semesta pembicaraan, domain, dan fungsi keanggotaan. Variabel *fuzzy* merupakan suatu lambang atau kata yang menunjuk kepada suatu yang tidak tertentu dalam sistem *fuzzy*. Contoh: suhu, kelembaban, penyiraman, dan lainnnya. Semesta pembicaraan adalah rentang total dari nilai yang diizinkan untuk diolah dalam suatu variabel *fuzzy*. Domain himpunan *fuzzy* adalah rentang nilai yang diijinkan dalam semesta pembicaraan dan boleh diolah dalam suatu himpunan *fuzzy*. Fungsi keanggotaan (*membership function*) adalah suatu kurva yang menunjukkan pemetaan titik-titik *input* data kedalam nilai keanggotaan yang memiliki interval antara 0 sampai 1. Ada beberapa fungsi yang bisa digunakan diantaranya :

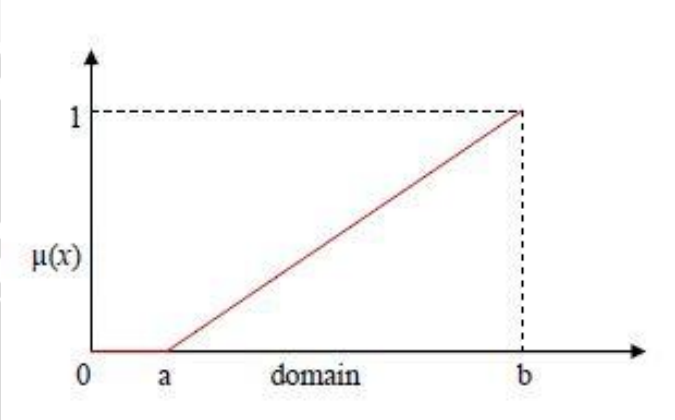
a.) Linear.

b.) Kurva Segitiga.

c.) Kurva Trapesium.

Pada linear, pemetaan input ke derajat keanggotaannya dapat digambarkan sebagai suatu garis lurus.. Ada 2 keadaan himpunan *fuzzy* yang linear:

1. linear naik, yaitu kenaikan himpunan dimulai dari nilai domain yang memiliki nilai keanggotaan nol [0] menuju arah kanan dan ke nilai domain yang memiliki derajat keanggotaan yang lebih tinggi (Gambar 2.1).

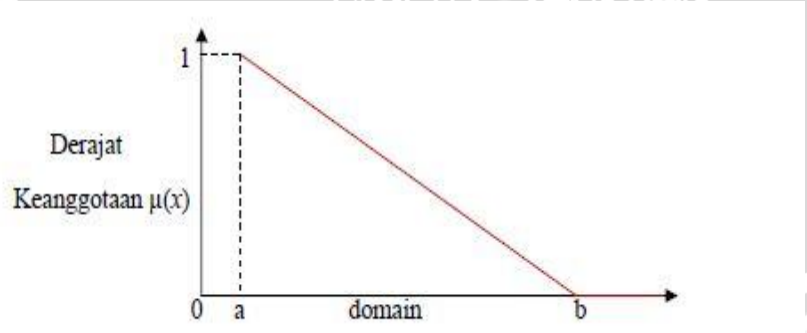


Gambar 2.1 Representasi linier Naik

Fungsi keanggotaan dapat dilihat pada Persamaan (2.2).

$$\mu(x)=\begin{cases} 0 & : x \le a \\ \frac{(x-a)}{(b-a)} & : a < x \le b \end{cases} \quad (2.2)$$

2. linear turun, yaitu garis lurus yang dimulai dari nilai domain dengan derajat keanggotaan tertinggi menuju keanggotaan yang lebih rendah.(Gambar 2.2).

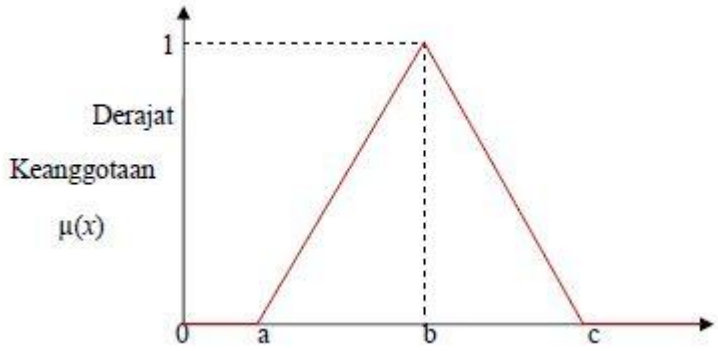


Gambar 2.2 Representasi Linear Turun

Fungsi keanggotaan keanggotaan dapat dilihat pada Persamaan (2.2).

$$(x)=\begin{cases} \frac{(x-a)}{(b-a)} & : a \le x < b \\ 0 & : x \ge b \end{cases} \quad (2.3)$$

3. Representasi kurva segitiga, adalah gabungan dari linear naik dan linear turun seperti terlihat pada Gambar 2.3.

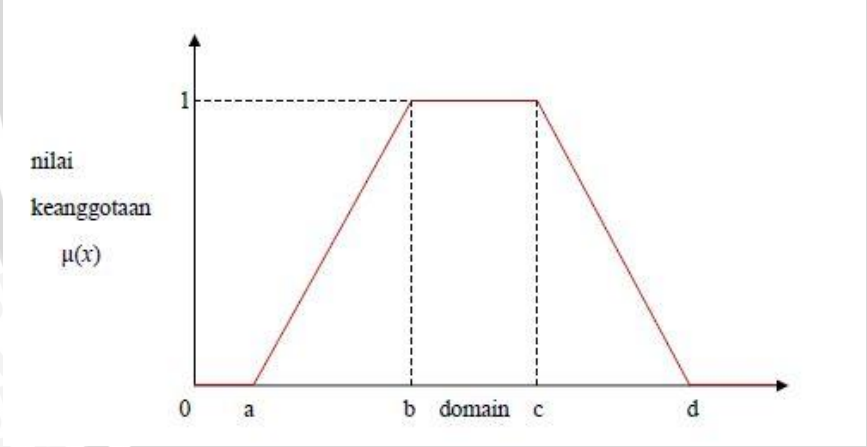


Gambar 2.3 Representasi Linear Naik dan Turun

Fungsi keanggotaan keanggotaan dapat dilihat pada Persamaan (2.4).

$$\mu(x)=\begin{cases}0 & : x \le a \; dan \; x \ge c \\ \frac{(x-a)}{(b-a)} & : \; a < x \le b \\ \frac{(c-x)}{(c-b)} & : \; b < x < c\end{cases} \quad (2.4)$$

4. Kurva trapesium bentuknya mirip seperti kurva segitiga, hanya saja ada beberapa titik yang memiliki nilai keanggotaan 1 (satu), seperti pada Gambar 2.4.



Gambar 2.4 Representasi Kurva Trapesium

Fungsi Keanggotaan keanggotaan dapat dilihat pada Persamaan (2.5).

$$\mu(x)=\begin{cases}0 & : x \le a \; atau \; x \ge d \\ \frac{(x-a)}{(b-a)} & : \; a < x \le b \\ \frac{(d-x)}{(d-c)} & : \; c < x < d \\ 1 & : b < x \le c\end{cases} \quad (2.5)$$

2.2.3 Sistem Inteferensi *Fuzzy*

Salah satu aplikasi logika *fuzzy* yang telah berkembang amat luas dewasa ini adalah sistem inferensi *fuzzy*, yaitu sistem komputasi yang bekerja dengan dasar prinsip penalaran *fuzzy*, sama seperti manusia melakukan penalaran dengan nalurinya. Dalam subbab ini akan dibahas salah satu dari proses semacam itu, yaitu penentuan pemberian pupuk AB mix dan juga pemberian larutan pH. Sistem ini berfungsi untuk mengambil keputusan melalui proses tertentu dengan mempergunakan aturan inferensi berdasarkan logika *fuzzy*. Sistem inferensi fuzzy terdiri dari tiga unit, yaitu :

1. Unit fuzzifikasi *(fuzzification unit)*
2. Unit penalaran logika *fuzzy* (*fuzzy logic reasoning unit)*
3. Unit defuzzifikasi (*defuzzification unit* / unit penegasan)

Pada sistem inferensi fuzzy, nilai-nilai masukan tegas dikonversikan oleh unit fuzzifikasi ke nilai fuzzy yang sesuai. Hasil pengukuran yang telah difuzzikan kemudian diproses oleh unit penalaran. Langkah terakhir dikerjakan oleh unit defuzzifikasi yaitu menerjemahkan himpunan keluaran itu kedalam nilai yang tegas. Nilai tegas inilah yang kemudian diproses dalam bentuk suatu tindakan yang akan dilaksanakan.

2.2.3.1 Unit Fuzzifikasi

Proses fuzzyfikasi merupakan proses mengubah variabel non *fuzzy* (*variable numerik*) menjadi variabel *fuzzy* (*variabel linguistik*). Karena sistem inferensi *fuzzy* bekerja dengan aturan dan masukan dari fuzzy, maka langkah pertama yaitu mengubah input tegas yang diterima, menjadi masukan *fuzzy*,Itulah yang dilakukan oleh unit fuzzifikasi.

Untuk masing–masing variabel masukan, ditentukan suatu fungsi fuzzifikasi yang akan mengubah variabel masukan yang tegas (yang biasa dinyatakan dalam bilangan real) menjadi nilai pendekatan *fuzzy*. Fungsi fuzzifikasi ditentukan berdasarkan kriteria di bawah ini :

1. Fungsi fuzzifikasi diharapkan mengubah suatu nilai tegas, misalnya *a* elemen R, ke suatu himpunan *fuzzy* $\tilde{A}$ dengan nilai keanggotaan *a* terletak pada selang tertutup [0,1].
2. Bila nilai masukannya cacat karena gangguan (derau), diharapkan fungsi fuzzifikasi dapat menekan sejauh mungkin gangguan itu.
3. Fungsi fuzzifikasi diharapkan dapat membantu menyederhanakan komputasi yang harus dilakukan oleh sistem tersebut dalam proses inferensinya.

2.2.3.2 Unit Penalaran

Penalaran *fuzzy* sering di sebut juga dengan penalaran hampiran adalah suatu cara penarikan kesimpulan berdasarkan seperangkat implikasi *fuzzy* dan

suatu fakta yang diketahui (sering disebut premis). Penarikan kesimpulan dalam logika klasik didasarkan pada tautologi, yaitu proposisi yang selalu benar, tidak bergantung pada nilai kebenaran proposisi penyusunnya.

2.2.3.3 Unit Defuzzifikasi

Unit defuzzifikasi digunakan untuk menghasilkan nilai variabel solusi yang diinginkan dari suatu daerah keluaran *fuzzy*. Karena sistem inferensi hanya dapat membaca nilai logika tegas, maka diperlukan suatu mekanisme untuk mengubah nilai keluaran *fuzzy* itu menjadi nilai logika tegas. Itulah yang dilakukan unit defuzzifikasi yang memuat fungsi-fungsi penegasan dalam sistem itu.

2.2.4 Logika Fuzzy Dalam Pengambilan Keputusan

Penalaran metode ini hampir sama dengan penalaran Mamdani, yang membedakan adalah keluaran sistem tidak berupa himpunan *fuzzy*, tetapi berupa nilai konstanta atau persamaan linear.

2.2.4.1 Metode Min Max

Pemilihan metode implikasi fuzzy adalah masalah yang signifikan dalam pengembangan teori himpunan fuzzy. Fungsi implikasi Max-Min dan Max-Prod merupakan metode yang paling umum digunakan karena mudah diimplementasikan dan bila diagregasikan dengan fungsi lain akan menghasilkan bentuk yang mudah di-defuzzification[3]. Metode Max (Maximum) mengambil solusi himpunan fuzzy diperoleh dengan cara mengambil nilai maksimum aturan, kemudian menggunakannya untuk memodifikasi daerah fuzzy, dan mengapilasikannya ke output dengan menggunakan operator OR (union). Jika semua proposisi telah dievaluasi, maka output akan berisi suatu himpunan fuzzy yang merefleksikan kontribusi dari tiap-tiap proporsi.

Metode MinMax menggunakan konstanta atau fungsi matematika dari variabel *input*:

Jika a adalah Ai dan b adalah Bi maka c adalah Ci = f(a,b)

Dengan *a, b* dan *c* adalah variabel linguistik; *Ai* dan *Bi* himpunan *fuzzy* ke-*I* untuk *a* dan *b*, dan *f(a,b)* adalah fungsi matematik. Untuk mendapatkan *output* (hasil), maka terdapat 4 langkah / tahapan sebagai berikut:

1. Pembentukan himpunan *fuzzy*

Langkah ini merupakan langkah pertama *fuzzy* yaitu membentuk himpunan keanggotaan masing-masing *input fuzzy*.

2. Aplikasi fungsi implikasi

Menyusun basis aturan, yaitu aturan-aturan berupa implikasi *fuzzy* yang menyatakan relasi antara variabel *input* dengan variabel *output*. Pada metode Sugeno, fungsi implikasi yang digunakan adalah *Min*. Bentuk umumnya adalah sebagai berikut:

Jika a adalah Ai dan b adalah Bi maka c adalah Ci = f(a,b)

3. Komposisi aturan

Apabila sistem terdiri dari beberapa aturan, maka inferensi diperoleh dari kumpulan dan korelasi antar aturan. Metode yang digunakan dalam melakukan inferensi sistem *fuzzy* adalah metode *centroid*.

Pada metode ini, solusi himpunan *fuzzy* diperoleh dengan cara mengambil nilai minimum aturan, kemudian menggunakan nilai tersebut untuk memodifikasi daerah *fuzzy* dan mengaplikasikannya ke *output* dengan menggunakan operator AND (irisan). Jika semua proporsi telah dievaluasi, maka *output* akan berisi suatu himpunan *fuzzy* yang merefleksikan kontribusi dari tiap-tiap proporsi. Secara umum dapat dituliskan pada persamaan 2.10.

$$\mu(xi) = max\ (\mu sf(xi), \mu kf(xi)) \quad (2.10)$$

dengan:

μsf (*x*i) = nilai keanggotaan solusi *fuzzy* sampai aturan ke-I μkf (*x*i) = nilai keanggotaan konsekuen *fuzzy* aturan ke-i.

4. Defuzzifikasi dengan metode centroid

Masukan dari proses penegasan adalah suatu himpunan *fuzzy* yang diperoleh dari komposisi aturan-aturan *fuzzy*, sedangkan *output* yang dihasilkan merupakan suatu bilangan *real* yang tegas. Sehingga jika diberikan suatu himpunan *fuzzy* dalam range tertentu, maka dapat diambil suatu nilai tegas tertentu sebagai *output.* Apabila komposisi aturan menggunakan metode Sugeno maka defuzzifikasi (Z*) dilakukan dengan cara mencari nilai rata-rata terpusatnya. Seperti yang dituliskan pada persamaan 2.11.

$$c \leftarrow \max(a, b) \quad (2.11)$$

dengan *di* adalah nilai keluaran pada aturan ke-*i* adalah derajat keanggotaan nilai keluaran pada aturan ke-*i* sedangkan *n* adalah banyaknya aturan yang digunakan.

2.2.5 FPGA

FPGA (*Field-programmable Gate Arrays*) merupakan sebuah IC digital yang digunakan untuk mengimplementasikan rangkaian-rangkaian digital. FPGA terdiri dari komponen gerbang logika.

2.2.5.1 Pengertian FPGA

FPGA merupakan komponen silikon seperti halnya IC yang fungsi dan rancangannya dapat dikonfigurasi oleh pengguna atau perancang dengan menggunakan bahasa HDL. FPGA dapat dikonfigurasikan untuk mengimplementasikan fungsi logika apapun yang dapat dilakukan oleh suatu (ASIC). Karakteristik dari FPGA antara lain adalah dapat dirancang sesuai dengan keinginan dan kebutuhan pemakai tanpa melalui tahap "*burn*" di laboratorium atau di "*hardwire*" oleh perusahaan piranti, hal tersebut mungkin dilakukan karena FPGA terdiri atas sekumpulan *Configurable Logic Blocks* (CLB) yang terhubung melalui *Programmable Interconnects* (PI). Pada umumnya setiap CLB terdiri dari terdiri dari beberapa *Logic Cells* yang kadang disebut juga dengan

Slice, yang masing-masing terdiri dari 4-input *Lookup Table* (LUT), D Flip-Flop dan 2-to-1 Mux, seperti pada gambar 2.1 dibawah ini.

Konfigurasi CLB dalam FPGA dapat berbeda-beda, tergantung dari manufaktur dan varian FPGA yang digunakan, perbedaannya termasuk jumlah *inputs* dan *outputs*, kompleksitas rangkaian CLB dan jumlah transistor yang digunakan. Jadi kemampuan untuk mengimplementasikan fungsi logika disediakan oleh CLB ini. Setiap *Logic Cell* dapat dihubungkan dengan *Logic Cell* lainnya melalui *Programmable Interconnect* (PI) yang akan membentuk suatu fungsi logika yang compleks, ilustrasi dari PI ditunjukkan pada gambar 2.2 dibawah ini. [8]

Setiap *Logic Cell* dapat dihubungkan dengan *Logic Cell* lainnya melalui *Programmable Interconnect* (PI) yang akan membentuk suatu fungsi logika yang compleks, ilustrasi dari PI ditunjukkan pada gambar 2.2 dibawah ini. [8]

2.2.5.2 Arsitektur FPGA

Sebuah IC pada FPGA terdiri dari 3 komponen pendukung utama yaitu Configurable Logic Block, Input Output Block dan Programmable Interconnect.

1. Configurable Logic Block
   CLB merupakan komponen dasar yang membentuk IC FPGA berupa matrik-matrik yang saling terhubung oleh PI, yang dapat dikonfigurasikan untuk melakukan rangkaian logika kombinasional, *shift register,* atau *Random Access Memory* (RAM).
2. Input Output Block
   IOB merupakan penghubung atau sebagai antarmuka antara pin-pin terminal IC FPGA dengan kawat penghubung atau jalur-jalur koneksi di luar IC, IOB dikelompokkan ke dalam beberapa *I/O Bank* yang sesuai dengan standar I/O.
3. Programmable Interconncect

PI merupakan komponen yang berperan sebagai kawat atau sakelar penghubung yang dapat dikonfigurasikan dan mengelilingi blok-blok CLB, yang akan menghubungkan antar blok-blok CLB maupun dengan IOB

2.2.5.3 Konfigurasi FPGA

Tanpa memperhatikan metode interkoneksi yang digunakan pada FPGA, dapat dilihat jelas bahwa untuk membayangkan sakelar mana yang harus dibuka dan yang ditutup untuk membuat suatu rangkaian logika merupakan pekerjaan yang sangat berat, karenanya perusahaan pembuat chip menyediakan perangkat lunak yang melakukan deskripsi dari disain logika sebagai masukannya dan kemudian akan menghasilkan file biner yang akan mengkonfigurasi sakelar-sakelar didalam chip CPLD atau FPGA sehingga menjadi seperti disain yang dibuat. Perusahaan pembuat chip pada umumnya memberikan perangkat lunak secara cuma-cuma atau gratis. Perangkat lunak ini digunakan untuk mendukung proses design entry, simulation, synthesis and place-androute, dan Programming through special cables (JTAG). Perusahaan Xilinx terkenal dengan software miliknya yang bernama ISE WebPack sedangkan perusahaan Altera terkenal dengan software miliknya yang bernama Quartus II Web Edition. Implementasi

sebuah disain logika dengan perangkat lunak tersebut biasanya terdiri dari beberapa langkah :

1. Mendeskripsikan rangkaian logika dari sistem yang dirancang dengan menggunakan hardware description language (HDL) seperti VHDL atau Verilog atau dengan menggambarkannya dengan schematic editor.
2. Deskripsi atau skematik rangkaian logika dari sistem dikonversikan menjadi sebuah Netlist menggunakan program logic synthesizer. Netlist merupakan sebuah deskripsi dari bermacam-macam gerbang logika dalam disain dan bagaimana mereka dihubungkan.
3. Implementasi rangkaian dengan melakukan pemetaan (mapping) gerbang logika dan interkoneksi (routing) netlist kedalam FPGA menggunakan implementation tools, yakni CLB didalam FPGA selanjutnya disusun kedalam look-up tables (LUT) yang melakukan operasi logika. CLB dan LUT terjalin dengan berbagai routing resources. Program pemetaan dan interkoneksi menggumpulkan netlis gerbang logika kedalam kelompok yang muat dalam LUT kemudian menentukan kumpulan gate ke CLB tertentu sambil membuka atau menutup sakelar pada matriks interkoneksi untuk menghubungkan gerbang-gerbang logika tersebut
4. Setelah fase implementasi selesai, aplikasi akan membuat sebuah file bitstream yang berisi nilai logika ‘1’ dan ‘0’ yang sebagai representasi konfigurasi rangkaian digital yang akan diimplementasikan ke dalam FPGA untuk menunjukkan terbuka atau tertutupnya sakelar pada IC FPGA.

File bitstream kemudian digugah (upload) kedalam IC FPGA atau IC memori yang tersedia melalui kabel JTAG sesuai dengan jenis dan tipe development board yang digunakan. Lambang, satuan, dan singkatan

Penulisan lambang atau simbol sebaiknya menggunakan fasilitas simbol atau jenis huruf Symbol yang ada pada program komputer pengolah kata untuk membedakannya dengan huruf biasa. Sebagai contoh untuk tanda perkalian tidak menggunakan huruf x tetapi “×” dari symbol. Untuk rumus matematika diusahakan ditulis dalam satu baris. Bila hal ini tidak memungkinkan maka harus diatur sedemikian rupa agar mudah dimengerti.

Satuan dan singkatan yang digunakan adalah yang lazim dipakai dalam disiplin ilmu terkait, misalnya 25°C; 10 ppm; H_2O; dan sebagainya. *Superscript* dan *subscript* sebaiknya digunakan ketika diperlukan.

2.2.6 VDHL

VHDL (VHSIC *Hardware Description Languange*)merupakan bahasa pemrograman yang digunakan pada FPGA.

2.2.6.1 Pengertian VHDL

VHDL merupakan salah satu bahasa yang digunakan untuk mendeskripsikan suatu perangkat keras. VHDL merupakan kependekan dari VHSIC *Hardware Description Languange*, sedangkan VHSIC kependekan dari *Very*

High Speed Integrated Circuit. Pada awalnya VHDL merupakan standard dokumentasi perangkat keras yang disponsori oleh Departemen Pertahanan Amerika Serikat sekitar tahun 1980, yang kemudian dokumentasi tersebut dikirimkan ke *Institute of Electrical and Electronic Engineers* (IEEE) dan diratifikasi oleh IEEE dengan standard IEEE 1076 dan IEEE 1164 dengan nama VHDL-87. [13] Setelah peluncuran pertama, berbagai varian VHDL dikembangkan untuk memfasilitasi berbagai macam kebutuhan perancangan dan pemodelan perangkat keras, varian-varian tersebut juga mendapatkan standard dari IEEE, yakni sebagai berikut :

1. Standard IEEE 1076.1-1999, VHDL Analog and Mixed Signal Exensions (VHDL AMS), menggambarkan pemodelan sinyal analog dan campuran.
2. Standard IEEE 1076.2-1996, VHDL Mathematical Packages, menggambarkan fungsi matematika untuk bilangan real dan kompleks.
3. Standard IEEE 1076.3-1997, Synthesis Packages, menggambarkan fungsi aritmatika dengan manipulasi bit.
4. Standard IEEE 1076.4-1995, VHDL Initiative Towards ASIC Lybraries (VITAL), menggambarkan mekanisme penambahan informasi detil pewaktuan sel-sel pada ASIC.
5. Standard IEEE 1076.6-1999, VHDL Register Transfer Level (RTL) Synthesis, menggambarkan VHDL yang kompatibel dengan proses Synthesis.
6. Standard IEEE 1164-1993, Multivalue Logic System For VHDL Model Interoperability (std_logic_1164), menggambarkan tipe data yang baru untuk memodelkan logika dengan beragam nilai.
7. Standard IEEE 1029.1-1998, VHDL Waveform and Vector Exchange to Support Design and Test Verification (WAVES), menggambarkan penggunaan VHDL dalam pertukaran informasi pada lingkungan simulasi HDL.

2.2.6.2 Struktur Dasar

Secara umum struktur bahasa pemrograman VHDL terdiri dari tiga bagian pokok yaitu deklarasi Library, Entity, dan Architecture.

Deklarasi *Library* merupakan kumpulan potongan kode yang sering digunakan, Sebuah *entity* berisi deklarasi *Input/Output* dari suatu sistem, sedangkan deklarasi *arsitektur* berisi deskripsi dari suatu rangkaian atau fungsi logika yang akan diimplemntasikan. Di dalam arsitektur dapat menangani baik rangkaian sekuensial maupun rangkaian kombinasional dan dalam 1 (satu) modul VHDL dimungkinkan untuk menggunakan lebih dari 1 (satu) *arsitektur.*

2.2.6.3 Metode Verifikasi

Metode verifikasi yang sering digunakan untuk mengetahui hasil implementasi rancangan dengan VHDL adalah dengan melakukan simulasi. Simulasi merupakan proses rekonstruksi sistem yang telah dideskripsikan dengan VHDL kemudian mengeksekusi sistem tersebut dengan data masukan dengan pola tertentu pada perangkat komputer yang kemudian menguji dan menganalisa respon dari sistem melalui keluarannya. Keluaran yang diuji dan dianalisa bisa berdasarkan sistem yang sesungguhnya (sesuai dengan datasheet)

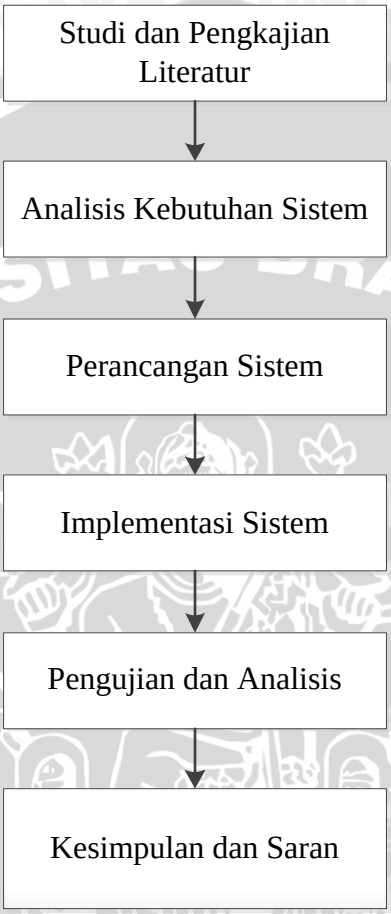
atau berdasarkan nilai ekspektasi (sesuai dengan teori atau hipotesis dari keluarannya) yang menyertakan informasi fungsional dan pewaktuannya. Simulasi merupakan proses yang serbaguna karena dapat diaplikasikan pada level abstraksi apapun baik pada level skematik amupun kode VHDL. Manfaat utama dari simulasi adalah memberikan fasilitas kepada pengguna untuk menguji dan mendeteksi kesalahan-kesalahan pada rancangan tanpa benar-benar merekonstruksi rancangan. Tetapi simulasi hanya memberikan sebagian potongan informasi dari keseluruhan operasi pada rancangan, yang digambarkan oleh serangkaian input stimulus yang tidak dijamin mencakup seluruh kemungkinan stimulus yang dibutuhkan oleh sistem, sehingga simulasi hanya mampu mendeteksi kesalahankesalahan utama saja. Keterbatasan simulasi lainnya adalah kompleksitas komputasi yang dibutuhkan, perangkat keras dapat bekerja secara kombinasional dan parallel sehingga membutuhkan cukup banyak waktu jika dioperasikan pada komputer yang melakukan komputasi secara sekuensial, yang akan menjadi masalah serius ketika harus melakukan simulasi dengan sistem yang terdiri dari ratusan hingga ribuan komponen level rendah. Metode verifikasi lainnya selain simulasi adalah analisa waktu, verifikasi formal dan emulasi perangkat keras. Analisa waktu hanya berfokus pada perhitungan waktu yang dibutuhkan pada setiap proses dan semua kemungkinan delay agar waktu yang dibutuhkan oleh sistem sesuai dengan spesifikasi waktu yang dibutuhkan. Verifikasi formal adalah memverifikasi sistem berdasarkan kesesuaian hasil operasi pada sistem dari sisi perhitungan matematika, apakah nilai keluarannya sudah sesuai dengan rancangan yang diimplementasikan. Emulasi perangkat keras adalah melakukan rekonstruksi sistem ke dalam FPGA atau menggugah konfigurasi yang dihasilkan oleh VHDL ke dalam FPGA dan memverifikasi fungsional FPGA seperti halnya memverifikasi FPGA sebagai ASIC.

2.2.6.4 Metode Verifikasi VHDL

Deskripsi dengan VHDL dapat dilakukan dengan beberapa metode, baik digunakan dengan salah satu metode atau campuran dari beberapa metode. Metode deskripsi VHDL antara lain metode struktural, aliran data dan perilaku sistem. Metode struktural mendeskripsikan sistem sebagai sebuah struktur yang terbentuk dari komponen-komponen yang dibutuhkan dan bagaimana masing-masing komponen tersebut saling terhubung, atau kurang lebih seperti membuat skematik dari sistem. Metode aliran data menitikberatkan pada bagaimana data itu mengalir dari satu komponen ke komponen lain, dan bagaimana data tersebut berubah setelah melewati komponen demi komponen. Komponen-komponen yang dilewati baik rangkaian logika maupun rangkaian memori atau register, sehingga lebih sering disebut sebagai *Register Transfer Level.* Sedangkan metode perilaku sistem mendeskripsikan sistem hanya dari sisi perilaku sistem atau kesesuaian antara input dan output yang dihasilkan, metode ini tidak memperhatikan komponen atau rangkaian logika serta memori yang digunakan, sehingga jika digunakan untuk rancangan dengan kompleksitas tinggi dapat menyebabkan pemborosan atau ketidaksesuaian penggunaan *source* yang tersedia.

BAB 3 METODOLOGI

Pada bab ini akan dijelaskan tentang langkah langkah yang akan ditempuh dalam penyusunan skripsi, meliputi studi kasus, studi literature, analisis kebutuhan sistem, perancangan sistem, pengujian dan analisis, kesimpulan dan saran.



Gambar 3.1 Flowchart metode penelitian

3.1 Studi Literatur

Dalam perancangan dan implementasi penelitian ini, perlu diadakan studi literatur.Literatur digunakan sebagai teori penguat dan landasan dasar dalam penelitian.Teori pendukung tersebut didapat dari buku, jurnal, paper dan internet. Literatur yang digunakan meliputi:

1. PC
2. Logika fuzzy
   a. *Knowledge Base* (Basis Pengetahuan)
   b. Fuzzifikasi
   c. Penalaran
   d. Defuzzifikasi
3. Software xilinx

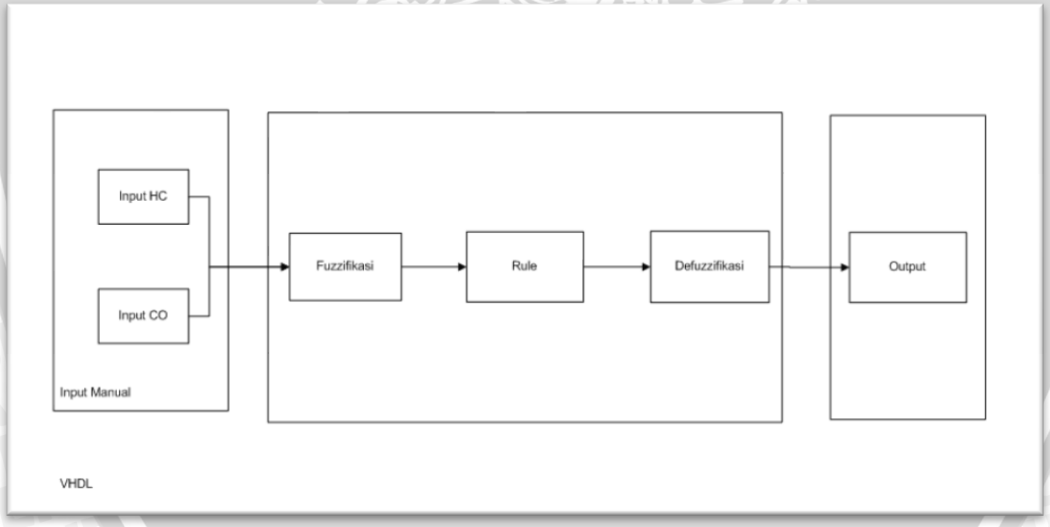
3.2 Analisis Kebutuhan Sistem

Analisis kebutuhan bertujuan untuk menganalis semua kebutuhan yang diperlukan oleh sistem yang akan dibangun. Analisis kebutuhan dilakukan dengan mengidentifikasi kebutuhan sistem. Analisis kebutuhan tersebut meliputi kebutuhan perangkat lunak

a) Sistem Operasi, sistem operasi yang digunakan untuk menjalankan perangkat lunak adalah Windows 7.
b) Xilinx ISE Design Suite 13.4 sebuah perangkat lunak yang digunakan untuk membuat program yang dibutuhkan pada sistem yang akan dibuat,
c) Matlab 7.6.0 (R2008a), sebuah perangkat lunak yang digunakan untuk membuat simulasi program *fuzzy*.
d) Microsoft Visio, sebuah perangkat lunak yang digunakan untuk membuat gambar flowchart.

3.3 Blok Diagram

Pada blok diagram ini akan menjelaskan tentang sistem kerja alat yang akan dibuat secara keseluruhan. Blok diagram ini gunakan untuk mempermudah pembaca dalam memahami alur kerja sistem. Blok diagram sistem dapat dilihat pada Gambar 3.1.



Gambar 3.2 Diagram Blok

Dari gambar diatas dapat diketahui input yang digunakan menggunakan input manual yang didapat dari nilai output sensor pada penelitian sebelumnya. Nilai input menggunakan 8bit pada vhdl. Dari input kemudian menuju fuzzifikasi untuk menentukan nilai derajat keanggotaan. Setelah itu nilai dari fuzzifikasi akan dicocokkan dengan rule yang sudah dibuat. Terdapat 9 rule yang dibuat dari 2 input karena setiap input mempunyai 3 fungsi keanggotaan. Setelah proses rule kemudian menuju proses defuzzifikasi untuk menentukan keputusan. Setelah defuzzifikasi maka keluarlah output yang menunjukkan bahwa nilai dari 2 input tersebut aman, waspada atau bahaya.

3.4 Perancangan Sistem

Perancangan dalam pembuatan sistem ini meliputi perancangan perangkat lunak. Pada perancangan perangkat lunak ini berupa simulasi fuzzy yang disimulasikan pada aplikasi Xilinx ISE untuk melihat apakah aturan-aturan yang dibuat berjalan dengan baik atau tidak.

3.5 Pengujian dan Analisis

Pengujian sistem pada penelitian ini dilakukan agar dapat menunjukkan bahwa sistem telah mampu bekerja sesuai dengan spesifikasi dari keutuhan yang melandasinya. Pengujian yang dilakukan meliputi:

1. Pengujian logika *fuzzy*.
2. Pengujian *output*.
3. Pengujian secara keseluruhan.

Setelah mendapatkan data dari serangkaian pengujian sistem maka akan dilakukan analisis kembali. Analisis ini bertujuan untuk mengukur kinerja sistem yang telah dibuat. Dengan dilakukan pengujian maka dapat ditarik sebuah kesimpulan dari penelitian yang dilakukan. Kesimpulan tersebut akan digunakan untuk mengetahui seberapa handal dan layak sistem yang telah dibuat.

3.6 Kesimpulan dan Saran

Kesimpulan dan Saran yang dimana merupakan tahap terakhir dari penelitian yang dilakukan.Tahap ini penelitian yang dilakukan haruslah memiliki hasil analisis data yang telah dilakukan serta dapat ditarik sebuah kesimpulan dari sistem yang dibuat. Dengan adanya analisis data dan kesimpulan maka dapat saran kedepannya untuk menyempurnakan alat yang dibuat.

BAB 4 PERANCANGAN DAN IMPLEMENTASI

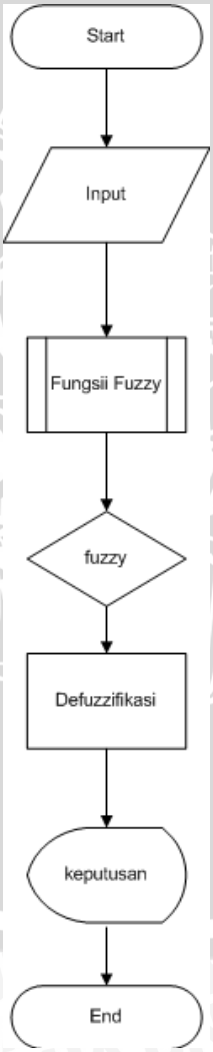
Bab ini menjelaskan tentang perancangan sistem keseluruan dan pengimplementasi pada sistem.

4.1 Diagram Alir

Diagram alir sistem keseluruhan menggambarkan sistem kerja dari alat yang akan dibuat. Diagram alir sistem keseluruhan dapat dilihat pada Gambar 4.1.

4.1.1 Diagram Alir Sistem Keseluruhan

Diagram alir sistem keseluruhan menggambarkan sistem kerja dari alat yang akan dibuat. Diagram alir sistem keseluruhan dapat dilihat pada Gambar 4.1.



Gambar 4.1 Diagram Alir

4.2 Perancangan Perangkat Lunak

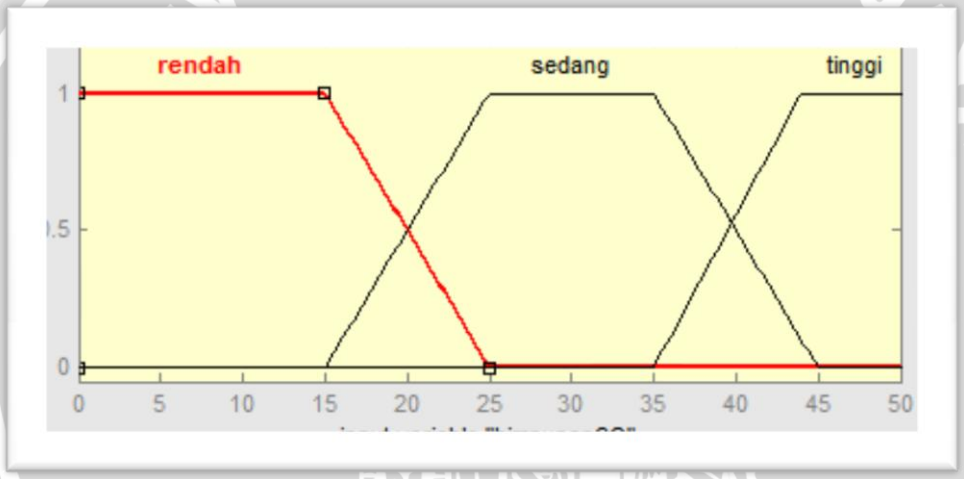
Pada perancangan perangkat lunak ini meliputi perancangan pemodelan fuzzy menggunakan 2 input dan 1 output dan juga rule-rule yang akan digunakan yang lakukan di matlab.

4.2.1 Pemodelan Fuzzy Matlab

Pemodelan fuzzy di matlab ini memodelkan fuzzifikasi dari variabel masukan yang diperlukan dalam sistem penulis yaitu C0 dan HC. Pemodelan variabel masukan tersebut yaitu himpunan CO dan himpunan HC.

4.2.1.1 Himpunan CO (Variabel *Input*)

Pada himpunan CO ini memiliki range yang berdasarkan dengan batas emisi kendaraan. Dari himpunan CO tersebut hanya ada 3 anggota yaitu Rendah,Sedang dan Tinggi. Berikut ini adalah pemodelan himpunan CO di matlab yang dapat dilihat pada Gambar 4.8.

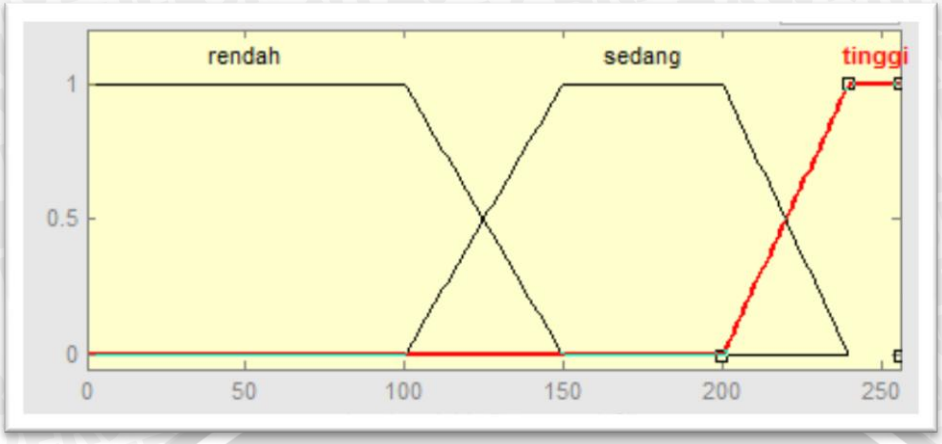


Gambar 4.2 Himpunan CO

Dari Gambar 4.3 dapat dilihat bahwa penulis memasukkan Domain untuk fungsi keanggotaan rendah adalah [0 0 15 25] penulis memberikan nilai 25 pada nilai maksimum karena pembacaan nilai sensor yang tidak 100% sesuai dengan nilai asli. Untuk fungsi keanggotaan sedang adalah [15 25 35 45]. Untuk fungsi keanggotaan tinggi adalah [35 45 5 6] nilai ini diperoleh penulis dari penelitian yang dilakukan pada pembuangan gas Knalpot.

4.2.1.2 Himpunan HC (variabel *input*)

Pada himpunan HC ini memiliki range yang berdasarkan dengan batas emisi kendaraan. Dari himpunan HC tersebut hanya ada 3 anggota yaitu Rendah,Sedang dan Tinggi. Berikut ini adalah pemodelan himpunan HC di matlab yang dapat dilihat pada Gambar 4.9.



Gambar 4.3 Himpunan HC

Dari Gambar 4.5 dapat dilihat bahwa penulis memasukkan Domain untuk fungsi keanggotaan rendah adalah [0 0 100 150]. Untuk fungsi keanggotaan sedang [110 150 200 240]. Untuk fungsi keanggotaan tinggi adalah [200 240 256 256] nilai ini diperoleh penulis dari penelitian yang dilakukan pada pembuangan gas Knalpot.

4.2.2 *Rule Evaluation*

Pada rule fuzzy ini akan memberikan aturan aturan dalam fuzzy sistem yang akan dibuat dengan menggunakan perintah "if" dan "and". Berikut adalah rule fuzzy yang dimodelkan :

1. IF CO rendah AND HC rendah THEN baik
2. IF CO rendah AND HC sedang THEN baik
3. IF CO rendah AND HC tinggi THEN waspada
4. IF CO sedang AND HC rendah THEN baik
5. IF CO sedang AND HC sedang THEN waspada
6. IF CO sedang AND HC tinggi THEN bahaya
7. IF CO tinggi AND HC rendah THEN waspada
8. IF CO tinggi AND HC sedang THEN bahaya
9. IF CO tinggi AND HC tinggi THEN bahaya

Pada rule diatas adalah rule untuk fuzzy sugeno yang keluaran berupa peringatan. Pada setiap *output* memiliki nilainya masing-masing tergantung pada berapa nilai gas buang yang dihasilkan pada knalpot kendaraan.

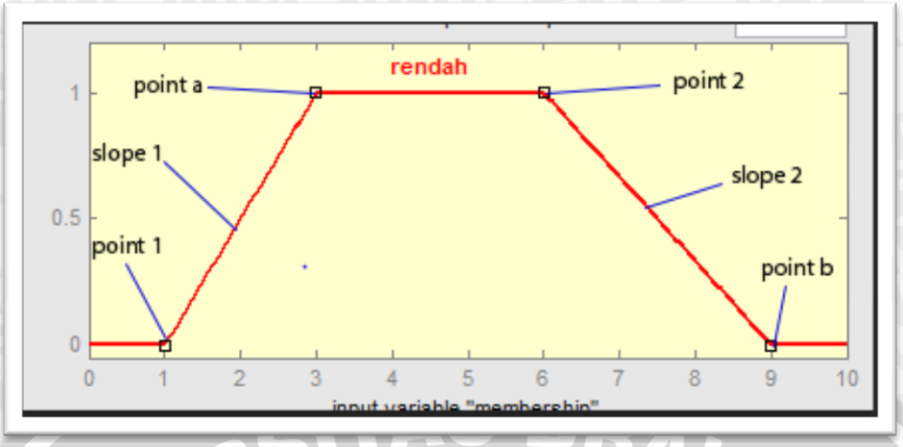
4.3 Implementasi

Pada implementasi akan mengimplementasikan perancangan perangkat lunak yang mencangkup tentang himpunan CO, himpunan HC ,rule dan implementasi sistem keseluruhan pada Isim.

4.3.1 Implementasi Himpunan CO dan HC

Pada proses fuzzifikasi mendefinisikan derajat keanggotaan. Derajat keanggotaan didapat dari setiap himpunan input. Derajat keanggotaan dari 0 ke

1 maka di penelitian ini diubah ke 00H ke FFH. Sebagai contoh terdapat sebuah himpunan dibawah ini



Gambar 4.4 *Membership* tipe trapesium

Dari Gambar 4.2 dapat dilihat terdiri dari 2 slope. Point merupakan titik pada tiap trapesium keanggotaan. Sedangkan slope merupakan sisi miring pada trapesium keanggotaann. Untuk menentukan slope didapat dari rumus:

$$(y2 - y1) \div (x2 - x1) \quad (3.0)$$

Dikarenakan menggunakan 8 bit maka μ = 1 sama dengan 255 atau $FF (tanda $ mengindikasikan pada penulisan hexadesimal). Sebagai contoh point1 bernilai 1 dan point 2 bernilai 6 maka slope 1 dan slope 2 menggunakan rumus (3.0) maka perhitungannya sebagai berikut (3.1) :

$$\text{Slope1} = 1 / (3-1) = \$FF/2 = 255/2 = 127 = \$7F \quad (3.1)$$

$$\text{Slope2} = 1 / (9-6) = \$FF/3 = 255/3 = 85 = \$55 \quad (3.2)$$

Kalkulasi derajat keanggotaan, μ dapat dibagi menjadi 4 bagian yaitu:

1. Sebelum point 1 dan setelah point b: μ = 0. (3.3)
2. Setelah point 1 dan sebelum point a: μ = 255-(input-point1)x slope1 (3.4)
3. Setelah point a dan sebelum point 2: μ = 1 (3.5)
4. Setelah point 2 dan sebelum point b: μ = 255-(input-point2)x slope2 (3.6)

Pada gambar membership terlihat bahwa rangenya adalah 0 sampai 50 karena nilai diskalakan dan di vhdl menggunakan input 8-bit. Pada vhdl himpunan CO didefinisikan sebagai berikut:

```
architecture Behavioral of vhdlfuz is
   type input is (corendah,cosedang,cotinggi);
   type membership is record term: input;
   point1: std_logic_vector (7 downto 0);
   slope1: std_logic_vector (7 downto 0);
   point2: std_logic_vector (7 downto 0);
   slope2: std_logic_vector (7 downto 0);
   end record;

   type membership_functions is array(natural range<>) of membership;
   constant mfs: membership_functions:=
   ((term => corendah, point1=> x"00", slope1=> x"00", point2 => x"0F", slope2 => x"19"),
   (term => cosedang, point1=> x"0F", slope1=> x"19", point2 => x"23", slope2 => x"19"),
   (term => cotinggi, point1=> x"23", slope1=> x"19", point2 => x"2D", slope2 => x"19"));
```

Gambar 4.5 Code VHDL untuk Himpunan CO

Pada gambar membership terlihat bahwa rangenya adalah 0 sampai 256 dikarenakan input diskalakan dikarenakan di vhdl menggunakan input 8-bit, untuk fungsi keanggotaan tinggi memiliki nilai maksimal 256. Pada vhdl himpunan HC didefinisikan sebagai berikut:

```
architecture Behavioral of vhdlfuz is
   type input is (hcrendah,hcsedang,hctinggi);
   type membership is record term: input;
   point1: std_logic_vector (7 downto 0);
   slope1: std_logic_vector (7 downto 0);
   point2: std_logic_vector (7 downto 0);
   slope2: std_logic_vector (7 downto 0);
   end record;

   type membership_functions is array(natural range<>) of membership;
   constant mfs: membership_functions:=
   ((term => hcrendah, point1 => "00000000", slope1 => x"00", point2 => x"64", slope2 => x"05"),
   (term => hcsedang, point1=> x"6E", slope1=> x"06", point2 => x"C8", slope2 => x"06"),
   (term => hctinggi, point1=> x"C8", slope1=> x"06", point2 => x"F0", slope2 => x"FF"));
```

Gambar 4.6 Code VHDL untuk himpunan HC

4.3.2 Implementasi Rule

Setelah menentukan derajat keanggotaan dalam tahap fuzzifikasi. Langkah selanjutnya adalah membuat aturan untuk memutuskan tindakan apa yang harus diambil dalam himpunan fungsi keanggotaan. Ada standar operator fuzzy yang dapat digunakan untuk mendefinisikan rule yaitu “AND”, “OR” dan “NOT”. Untuk operator “AND” digunakan Di bawah ini adalah contoh dari pelaksanaan aturan dan minimum dan maksimum.

Untuk "OR" => C = maksimum (A, B).

Untuk "AND" => C = minimum (A, B).

Fungsi minimum dan maksimum yang digunakan untuk memperoleh hasil dari setiap rule evaluasi

coding vhdl untuk rule:

```
function minimum (a,b: std_logic_vector) RETURN std_logic_vector IS
   variable min: std_logic_vector(7 downto 0):= (others => '0');

   begin
      if a<b then min := a;
      elsif b<a then min := b;

      end if;
   RETURN min;
end minimum;
```

Gambar 4.7 Fungsi Minimum

Dari gambar diatas dapat diketahui pada baris pertama merupakan deklarasi untuk fungsi minimum. Pada baris kedua merupakan tipe data yang digunakan yaitu std_logic_vector. Pada baris kelima dan enam merupakan main program fungsi minimum. Pada baris selanjutnya penutup progran fungsi minimum.

```
function maximum(a,b: std_logic_vector) RETURN std_logic_vector IS
   variable max: std_logic_vector(7 downto 0):= (others => '0');

   begin
         if a>b then max := a;
      elsif b>a then max := b;

      end if;
   RETURN max;
end maximum;
```

Gambar 4.8 Fungsi Maksimum

Dari gambar diatas dapat diketahui pada baris pertama merupakan deklarasi untuk fungsi maksimum. Pada baris kedua merupakan tipe data yang digunakan yaitu std_logic_vector. Pada baris kelima dan enam merupakan main program fungsi maksimum. Pada baris selanjutnya penutup progran fungsi maksimum.

```
--RULE BASE
rule1 : position(0)<= minimum(u1(0),u4(3));

rule2 : position(1)<= minimum(u1(0),u5(4));

rule3 : position(2)<= minimum(u1(0),u6(5));

rule4 : position(3)<= minimum(u2(1),u4(3));

rule5 : position(4)<= minimum(u2(1),u5(4));

rule6 : position(5)<= minimum(u2(1),u6(5));

rule7 : position(6)<= minimum(u3(2),u4(3));

rule8 : position(7)<= minimum(u3(2),u5(4));

rule9 : position(8)<= minimum(u3(2),u6(5));
```

Gambar 4.9 *Rule Evaluation*

Dari gambar diatas dapat diketahui terdapat 9 rule yang dapat dibuat dari 3 kondisi untuk setiap input.

Setelah mengetahui keluaran dari rule maka selanjutnya adalah mengkombinasikannya menjadi satu output. Proses defuzzifikasi menggunakan metode min max untuk menentukan keluaran output dengan nilai terbesar. Code vhdl untuk proses defuzzifikasi ditunjukkan pada gambar:

```
-- UNTUK INFERENSI

aman <= maximum(product9b,maximum(product9d,product9a));
waspada <= maximum(product9c,maximum(product9g,product9e));
bahaya <= maximum(product9f,maximum(product9h,product9i));
```

Gambar 4.10 Code VHDL untuk Defuzzifikasi

Dari gambar dapat diketahui bahwa ouput aman berasal dari rule a.b dan d. Sedangkan output waspada berasal dari rule c,e dan g. Untuk output bahaya berasal dari rule f,h dan i. Setelah proses defuzzifikasi selesai maka nilai output yang dijadikan keputusan akan keluar.

BAB 5 PENGUJIAN DAN ANALISIS

Bab ini membahas mengenai tahapan pengujian dan analisis dari sistem yang sudah dibuat.

5.1 Pengujian

Pada pengujian sistem ini akan membahas tentang pengujian hasil keluaran input sistem.

5.1.1 Pengujian pada simulator

a. Tujuan :

Menguji waktu yang diperlukan untuk menghasilkan output.

b. Peralatan :

a. Komputer

b. Simulator Xilinx

c. Langkah Pengujian :

a. Buka aplikasi simulator xilinx

b. Buka file fuzzy vhdl secara keseluruhan

c. Membuat testbench dari file fuzzy vhdl

d. Set nilai untuk masing-masing input

e. Check syntax dari testbench

f. Simulasikan jika tidak terjadi error pada check syntax

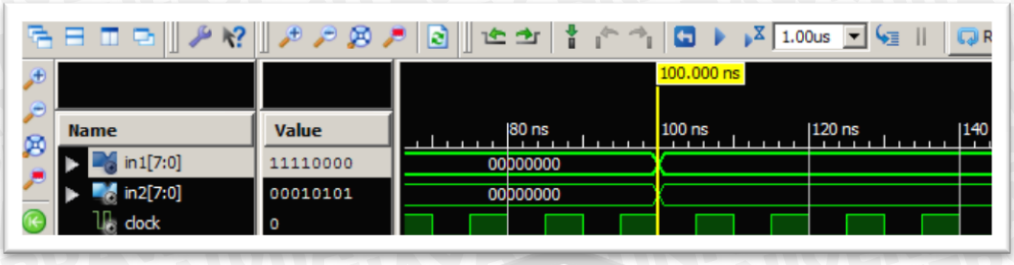
Potongan code program testbench untuk menentukan nilai input terdapat pada Gambar 5.1.

```
-- Stimulus process
stim_proc: process
begin
   -- hold reset state for 100 ns.
   wait for 100 ns;

   -- insert stimulus here
   in1 <= "11110000"; --240
   in2 <= "00010101"; --21
```

Gambar 5.1 Code Input VHDL

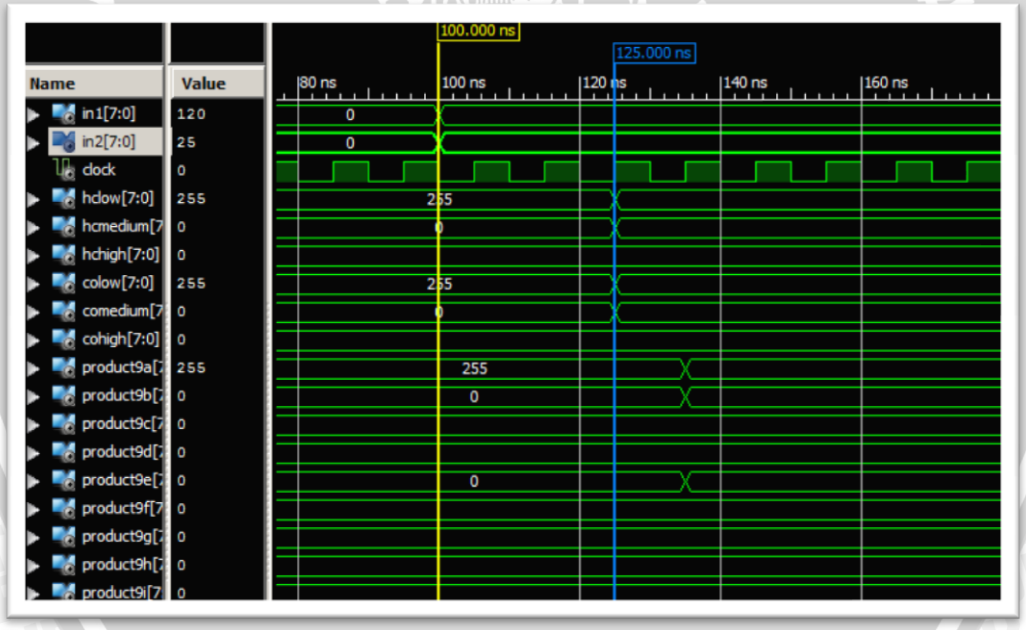
Dari potongan source code diatas dapat diketahui bahwa input akan masuk pada detik ke 100ns. In1 merupakan nilai untuk HC sedangkan in2 nilai untuk CO. Nilai in1 dan in2 dapat diubah sesuai yang diinginkan.



Gambar 5.2 Gambar Tampilan Input Pengujian

5.1.2 Pengujian waktu eksekusi fuzzy pada simulator

Pada pengujian fuzzy min max ini penulis menguji waktu eksekusi program fuzzy yang sudah diimplementasikan kedalam Xilinx simulator. Dengan menggunakan fungsi simulate behavioral model yang ada pada xilinx simulator maka dapat mengetahui waktu yang dibutuhkan dari awal sampai selesai eksekusi. Terdapat beberapa proses pada sistem yaitu proses fuzzifikasi, rule evaluation, dan defuzzifikasi. Pada proses fuzzifikasi dibutuhkan waktu 25000ns untuk menentukan keanggotaan. Hasil waktu eksekusi dapat dilihat pada Gambar 5.3

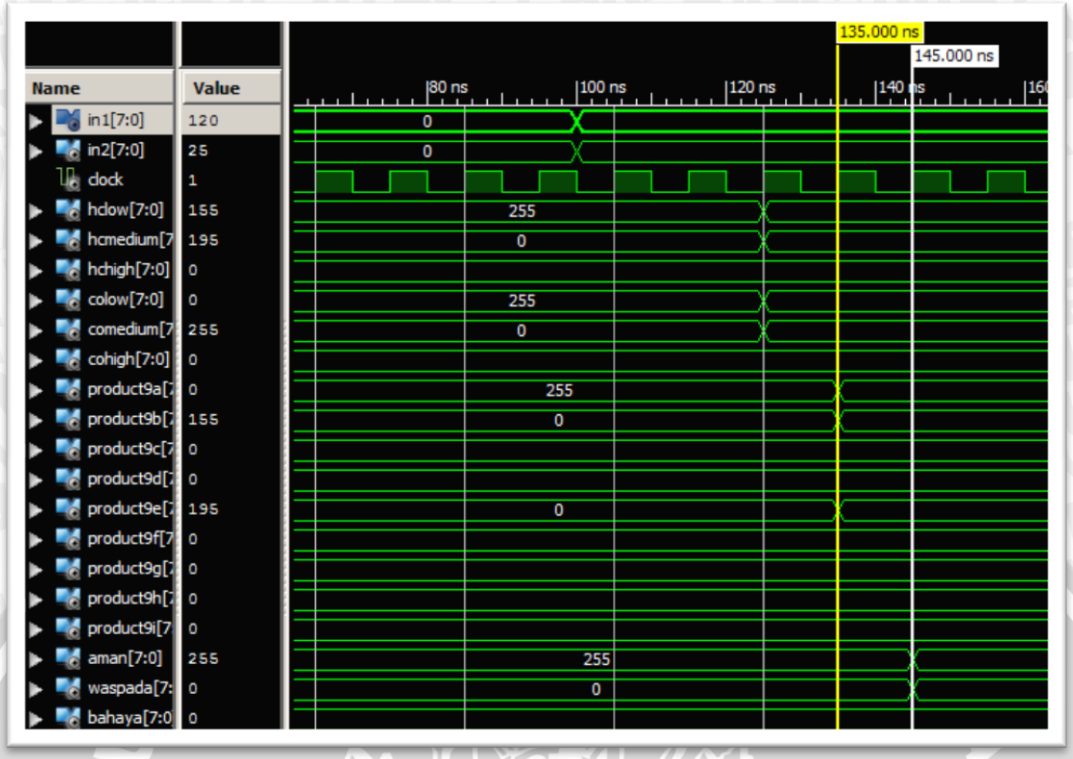


Gambar 5.3 proses Fuzzifikasi

Dari gambar diatas dapat diketahui garis warna kuning merupakan masuknya input pada sistem, sedangkan garis warna biru merupakan output fungsi keanggotaan yang diperoleh dari input. Untuk proses selanjutnya yaitu rule evaluation dibutuhkan waktu 10000ns untuk keluar nilai output. Hasil waktu eksekusi dapat dilihat pada Gambar 5.4

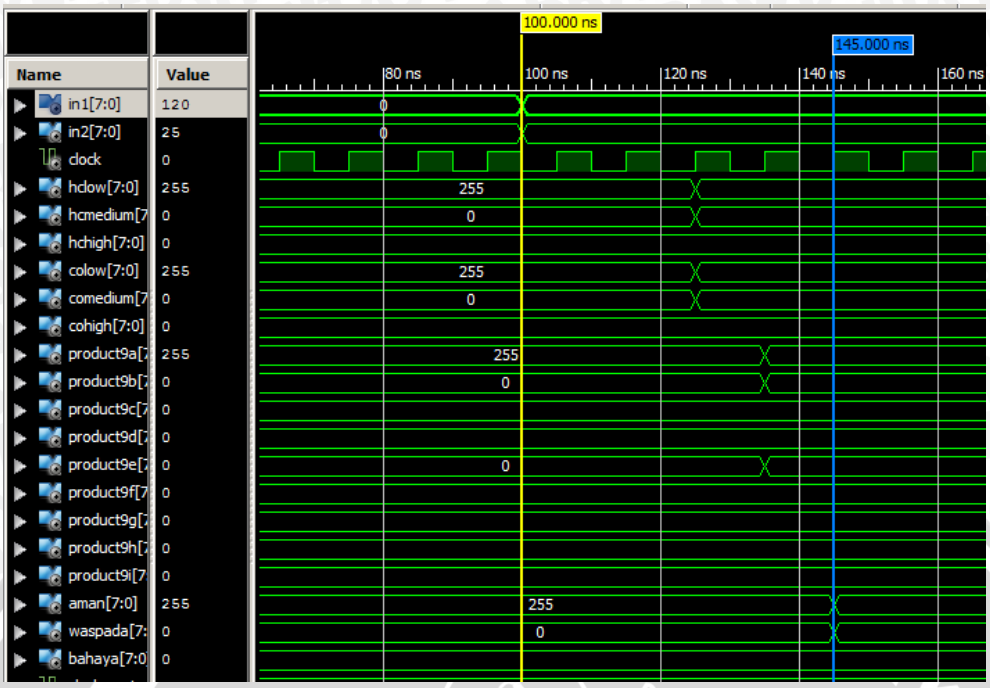
Gambar 5.4 Proses *Rule Evaluation*

Dari gambar diatas dapat diketahui garis warna putih merupakan hasil output dari rule evaluation. Setelah proses rule evaluation dilanjutkan pada proses defuzzifikasi yaitu menentukan output akhir dari sistem tersebut. Hasil waktu eksekusi dapat dilihat pada Gambar 5.5



Gambar 5.5 proses Defuzzifikasi

Dari gambar diatas dapat diketahui garis kuning merupakan hasil output dari rule evaluation, sedangkan garis berwarna putih merupakan hasil output dari defuzzifikasi. Dari semua proses tersebut dapat diketahui berapa waktu yang dibutuhkan sistem dari awal input sampai hasil akhir keluar. Berikut adalah hasil waktu eksekusi dapat dilihat pada Gambar 5.6.



Gambar 5.6 Proses Eksekusi Keseluruhan

5.2 Analisis

Pada pengujian keseluruhan untuk perhitungan manual dilakukan sesuai rumus fuzzy pada bab IV. Pada pengujian akhir ini nilai input telah diskalakan. Pada pengujian akhir perhitungan secara manual akan dicocokkan dengan output dari ise simulator yang ditunjukkan pada tabel pada Tabel 5.1

Tabel 5.1 Hasil Pengujian

no	input		manual		xilinx	
	hc	co				
1	80	10	255	aman	255	aman
2	80	20	130	aman	130	aman
3	80	30	255	aman	255	aman
4	80	40	130	waspada	130	aman waspada
5	80	50	255	waspada	255	waspada
6	120	10	195	aman	195	aman
7	120	20	130	waspada	130	aman waspada
8	120	30	195	waspada	195/155	aman waspada
9	120	40	195	waspada	130	semua
10	120	50	195	waspada	195/155	waspada bahaya
11	160	10	255	aman	255	aman
12	160	20	130	waspada	130	aman waspada
13	160	30	255	waspada	255	waspada
14	160	40	130	waspada	130	waspada bahaya
15	160	50	255	bahaya	255	bahaya
16	210	10	195	waspada	195/179	aman waspada

no	input		manual		xilinx	
	hc	co				
17	210	20	130	waspada	130	semua
18	210	30	195	bahaya	195/179	waspada bahaya
19	210	40	130	bahaya	130	waspada bahaya
20	210	50	195	bahaya	195	bahaya
21	240	10	255	waspada	255	waspada
22	240	20	130	waspada	130	waspada bahaya
23	240	30	255	bahaya	255	bahaya
24	240	40	130	bahaya	130	bahaya
25	240	50	255	bahaya	255	bahaya

BAB 6 PENUTUP

6.1 Kesimpulan

Berdasarkan rumusan masalah yang ada, hasil perancangan, implementasi dan pengujian yang dilakukan, maka diambil kesimpulan sebagai berikut :

1. Perancangan fuzzy pada FPGA dalam studi kasus emisi gas buang kendaraan bermotor bisa dilakukan menggunakan.
2. Metode fuzzy dapat dimplementasikan pada FPGA dengan menggunakan coding yang sangat kompleks agar output yang dihasilkan tepat dan akurat.
3. tingkat akurasi sistem yang dibuat dengan perhitungan matematis yang dilakukan secara manual sebesar 90% dengan kecepatan eksekusi sebesar 0,000045s.

6.2 Saran

Saran yang dapat diberikan untuk pengembangan sistem kedepannya adalah:

1. Perancangan ini bisa di lanjutkan ke tahap implementasi pada xilinx spartan.
2. Untuk meningkatkan kecepatan eksekusi dapat dilakukan dengan penulisan coding yang lebih pendek.

DAFTAR PUSTAKA

Anon., 1988. IEEE Standart VHDL Language Reference Manual.

H, Y. Z., n.d. Fuzzy Systems and Knowledge Discovery. Volume 4, pp. 154-158.

Nasrullah, E., 2009. Rancang Bangun Alat Pemantau Kualitas Udara Sekitar Berbasis Mikrokontroler AVR ATMega8 Dengan Penampil Dot Matri.

Purnomo, K., 2004. Aplikasi Logika Fuzzy untuk Pendukung Keputusan.

Salapura, V. & Hamann, V., 1996. Implementing Fuzzy Control Systems Using VHDL and Statecharts.

Zati, A., 2013. Penggunaan Sistem Inferensi Fuzzy Untuk Penentuan Jurusan di SMA Negeri 1 Bireuen. Volume 1.

LAMPIRAN A

Source code program utama

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity vhdlfuz is
        Port (  in1 : in STD_LOGIC_VECTOR (7 downto 0);
                in2 : in STD_LOGIC_VECTOR (7 downto 0);
                clock : in STD_LOGIC;

                hclow : out STD_LOGIC_VECTOR (7 downto 0);
                hcmedium : out STD_LOGIC_VECTOR (7 downto 0);
                hchigh : out STD_LOGIC_VECTOR (7 downto 0);
                colow : out STD_LOGIC_VECTOR (7 downto 0);
                comedium : out STD_LOGIC_VECTOR (7 downto 0);
                cohigh : out STD_LOGIC_VECTOR (7 downto 0);

                product9a : inout STD_LOGIC_VECTOR (7 downto 0);
                product9b : inout STD_LOGIC_VECTOR (7 downto 0);
                product9c : inout STD_LOGIC_VECTOR (7 downto 0);
                product9d : inout STD_LOGIC_VECTOR (7 downto 0);
                product9e : inout STD_LOGIC_VECTOR (7 downto 0);
                product9f : inout STD_LOGIC_VECTOR (7 downto 0);
                product9g : inout STD_LOGIC_VECTOR (7 downto 0);
                product9h : inout STD_LOGIC_VECTOR (7 downto 0);
                product9i : inout STD_LOGIC_VECTOR (7 downto 0);

                aman : inout STD_LOGIC_VECTOR (7 downto 0);
                waspada : inout STD_LOGIC_VECTOR (7 downto 0);
                bahaya : inout STD_LOGIC_VECTOR (7 downto 0));
```

```
end vhdlfuz;

architecture Behavioral of vhdlfuz is
        type input is (hcrendah,hcsedang,hctinggi,corendah,cosedang,cotinggi);
        type membership is record term: input;
        point1: std_logic_vector (7 downto 0);
        slope1: std_logic_vector (7 downto 0);
        point2: std_logic_vector (7 downto 0);
        slope2: std_logic_vector (7 downto 0);
        end record;

type membership_functions is array(natural range<>) of membership;
constant mfs: membership_functions:=
        ((term => hcrendah, point1 => "00000000", slope1 => x"00", point2 =>
x"64", slope2 => x"05"),
        (term => hcsedang, point1=> x"6E", slope1=> x"06", point2 => x"C8",
slope2 => x"06"),
        (term => hctinggi, point1=> x"C8", slope1=> x"06", point2 => x"F0",
slope2 => x"FF"),
        (term => corendah, point1=> x"00", slope1=> x"00", point2 => x"0F",
slope2 => x"19"),
        (term => cosedang, point1=> x"0F", slope1=> x"19", point2 => x"23",
slope2 => x"19"),
        (term => cotinggi, point1=> x"23", slope1=> x"19", point2 => x"2D",
slope2 => x"FF"));

-- FOR MEMBERSHIP FUNCTION
type input_array is array (0 to 7) of std_logic_vector(15 downto 0);
signal u: input_array;
type input_array1 is array (0 to 7) of std_logic_vector(7 downto 0);
signal u1: input_array1;
type input_array2 is array (0 to 7) of std_logic_vector(7 downto 0);
signal u2: input_array2;
```

```
type input_array3 is array (0 to 7) of std_logic_vector(7 downto 0);
signal u3: input_array3;
type input_array4 is array (0 to 7) of std_logic_vector(7 downto 0);
signal u4: input_array4;
type input_array5 is array (0 to 7) of std_logic_vector(7 downto 0);
signal u5: input_array5;
type input_array6 is array (0 to 7) of std_logic_vector(7 downto 0);
signal u6: input_array6;

-- FOR OUTPUT DEFUZZYFIKASI
type output_array is array (0 to 15) of std_logic_vector(7 downto 0);
signal position: output_array;

-- FOR MAX MIN
function minimum (a,b: std_logic_vector) RETURN std_logic_vector IS
        variable min: std_logic_vector(7 downto 0):= (others => '0');

        begin
                if a<b then min := a;
                else min := b;

                end if;
        RETURN min;
end minimum;

function maximum(a,b: std_logic_vector) RETURN std_logic_vector IS --
,c,d,e,f,g,h,i,j,k,l,m,n,o
        variable max: std_logic_vector(7 downto 0):= (others => '0');

        begin
                if a>b then max := a;
                else max := b;
```

```
                end if;
        RETURN max;
end maximum;

begin
        process (clock)
        begin
                if clock'EVENT and clock='1' then
--HC
--HC RENDAH
                if in1 < "01100100" then
                        u(0) <= "0000000011111111";
                elsif in1 >= "01100100" and in1 < "10010110" then
                        u(0) <= (255-(in1-mfs(0).point2)*mfs(0).slope2);
                else u(0) <= "0000000000000000";
                end if;
                u1(0) <= u(0)(7 downto 0);

--HC SEDANG
                if in1 > "01101110" and in1 < "10010110" then
                        u(1) <= (255-(in1-mfs(1).point1)*mfs(1).slope1);
                elsif in1 >= "10010110" and in1 < "11001000" then
                        u(1) <= "0000000011111111";
                elsif in1 >= "11001000" and in1 < "11110000" then
                        u(1) <= (255-(in1-mfs(1).point2)*mfs(1).slope2);
                else u(1) <= "0000000000000000";
                end if;
                u2(1) <= u(1)(7 downto 0);

--HC TINGGI
                if in1 > "11001000" and in1 < "11110000" then
                        u(2) <= (255-(in1-mfs(2).point2)*mfs(2).slope1);
                elsif in1 >= "11110000" then
```

```
                u(2) <= "0000000011111111";
        else u(2) <= "0000000000000000";
        end if;
        u3(2) <= u(2)(7 downto 0);

--CO
--CO RENDAH
        if in2 < "00001111" then
                u(3) <= "0000000011111111";
        elsif in2 >= "00001111" and in2 < "00011001" then
                u(3) <= (255-(in2-mfs(3).point2)*mfs(3).slope2);
        else u(3) <= "0000000000000000";
        end if;
        u4(3) <= u(3)(7 downto 0);

--CO SEDANG
        if in2 > "00001111" and in2 < "00011001" then
                u(4) <= (255-(in2-mfs(4).point1)*mfs(4).slope1);
        elsif in2 >= "00011001" and in2 < "00100011" then
                u(4) <= "0000000011111111";
        elsif in2 >= "00100011" and in2 < "00101101" then
                u(4) <= (255-(in2-mfs(4).point2)*mfs(4).slope2);
        else u(4) <= "0000000000000000";
        end if;
        u5(4) <= u(4)(7 downto 0);

--CO TINGGI
        if in2 > "00100011" and in2 < "00101101" then -->35 dan < 45
                u(5) <= (255-(in2-mfs(5).point1)*mfs(5).slope1);
        elsif in2 >= "00101101" then
                u(5) <= "0000000011111111";
        else u(5) <= "0000000000000000";
```

```
                end if;
                u6(5) <= u(5)(7 downto 0);

hclow <= u1(0);
hcmedium <= u2(1);
hchigh <= u3(2);
colow <= u4(3);
comedium <= u5(4);
cohigh <= u6(5);

--RULE BASE
rule1 : position(0)<= minimum(u1(0),u4(3));

rule2 : position(1)<= minimum(u1(0),u5(4));

rule3 : position(2)<= minimum(u1(0),u6(5));

rule4 : position(3)<= minimum(u2(1),u4(3));

rule5 : position(4)<= minimum(u2(1),u5(4));

rule6 : position(5)<= minimum(u2(1),u6(5));

rule7 : position(6)<= minimum(u3(2),u4(3));

rule8 : position(7)<= minimum(u3(2),u5(4));

rule9 : position(8)<= minimum(u3(2),u6(5));

-- UNTUK INFERENSI
product9a <= position(0);
product9b <= position(1);
```

```
product9c <= position(2);
product9d <= position(3);
product9e <= position(4);
product9f <= position(5);
product9g <= position(6);
product9h <= position(7);
product9i <= position(8);

aman <= maximum(product9b,maximum(product9d,product9a));
waspada <= maximum(product9c,maximum(product9g,product9e));
bahaya <= maximum(product9f,maximum(product9h,product9i));

            end if;
        end process;
end Behavioral;
```

LAMPIRAN B

Source Code pengujian

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tesfuzzy IS
END tesfuzzy;

ARCHITECTURE behavior OF tesfuzzy IS

  -- Component Declaration for the Unit Under Test (UUT)

  COMPONENT vhdlfuz
  PORT(
    in1 : IN  std_logic_vector(7 downto 0);
    in2 : IN  std_logic_vector(7 downto 0);
    clock : IN  std_logic;
                    hclow : inout STD_LOGIC_VECTOR (7 downto 0);
                    hcmedium : out STD_LOGIC_VECTOR (7 downto 0);
                    hchigh : out STD_LOGIC_VECTOR (7 downto 0);
                    colow : out STD_LOGIC_VECTOR (7 downto 0);
                    comedium : out STD_LOGIC_VECTOR (7 downto 0);
                    cohigh : out STD_LOGIC_VECTOR (7 downto 0);

                    product9a : inout STD_LOGIC_VECTOR (7 downto 0);
                    product9b : inout STD_LOGIC_VECTOR (7 downto 0);
                    product9c : inout STD_LOGIC_VECTOR (7 downto 0);
                    product9d : inout STD_LOGIC_VECTOR (7 downto 0);
                    product9e : inout STD_LOGIC_VECTOR (7 downto 0);
                    product9f : inout STD_LOGIC_VECTOR (7 downto 0);
                    product9g : inout STD_LOGIC_VECTOR (7 downto 0);
                    product9h : inout STD_LOGIC_VECTOR (7 downto 0);
                    product9i : inout STD_LOGIC_VECTOR (7 downto 0);
                    aman : inout  std_logic_vector(7 downto 0);
                    waspada : inout  std_logic_vector(7 downto 0);
                    bahaya : inout  std_logic_vector(7 downto 0)

    );
  END COMPONENT;


  --Inputs
  signal in1 : std_logic_vector(7 downto 0) := (others => '0');
  signal in2 : std_logic_vector(7 downto 0) := (others => '0');
  signal clock : std_logic := '0';
```

```
        --Outputs
        signal hclow : STD_LOGIC_VECTOR (7 downto 0);
        signal hcmedium : STD_LOGIC_VECTOR (7 downto 0);
        signal hchigh : STD_LOGIC_VECTOR (7 downto 0);
        signal colow : STD_LOGIC_VECTOR (7 downto 0);
        signal comedium : STD_LOGIC_VECTOR (7 downto 0);
        signal cohigh : STD_LOGIC_VECTOR (7 downto 0);
        --
        signal product9a : STD_LOGIC_VECTOR (7 downto 0);
        signal product9b : STD_LOGIC_VECTOR (7 downto 0);
        signal product9c : STD_LOGIC_VECTOR (7 downto 0);
        signal product9d : STD_LOGIC_VECTOR (7 downto 0);
        signal product9e : STD_LOGIC_VECTOR (7 downto 0);
        signal product9f : STD_LOGIC_VECTOR (7 downto 0);
        signal product9g : STD_LOGIC_VECTOR (7 downto 0);
        signal product9h : STD_LOGIC_VECTOR (7 downto 0);
        signal product9i : STD_LOGIC_VECTOR (7 downto 0);
        signal aman : std_logic_vector(7 downto 0);
        signal waspada : std_logic_vector(7 downto 0);
        signal bahaya : std_logic_vector(7 downto 0);

   -- Clock period definitions
   constant clock_period : time := 10 ns;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: vhdlfuz PORT MAP (
          in1 => in1,
          in2 => in2,
          clock => clock,
                        hclow => hclow,
                        hcmedium => hcmedium,
                        hchigh  => hchigh,
                        colow  => colow,
                        comedium => comedium,
                        cohigh => cohigh,
                        product9a => product9a,
                        product9b => product9b,
                        product9c => product9c,
                        product9d => product9d,
                        product9e => product9e,
                        product9f => product9f,
                        product9g => product9g,
```

```
                product9h => product9h,
                product9i => product9i,
                aman => aman,
                waspada => waspada,
                bahaya => bahaya
        );

    -- Clock process definitions
    clock_process :process
    begin
                clock <= '0';
                wait for clock_period/2;
                clock <= '1';
                wait for clock_period/2;
    end process;


    -- Stimulus process
    stim_proc: process
    begin
      -- hold reset state for 100 ns.
      wait for 100 ns;

          -- insert stimulus here
          in1 <= "11110000"; --240
          in2 <= "00010101"; --21

      wait;
    end process;

END;
```