BAB 2

LANDASAN TEORI

2.1 Motor DC

Sejak Michael Faraday menemukan motor elektrik awal 1800an, peralatan elektronik mengalami perkembangan yang sangat menakjubkan. Motor elektrik memiliki medan kumparan tetap yang menghasilkan medan elektromagnetik dan pembungkus putaran yang kuat yang terbuat dari banyak lilitan yang membungkus lempeng baja. Kesamaan yang terlihat antara motor (motor elektrik) dan generator, yang sama - sama untuk motor yang dilengkapi *brushes* dan *commutator*. *Commutator*, terbuat dari sepasang lempeng baja yang dilekatkan pada kumparan (*coil*) yang dililitkan (berputar / *armature*), yang secara kontan berhubungan dengan bagian *brushes*-nya. *Commutator* melayani perubahan dan pembalikan arah dari arus tetap secara periodik / berkala. Sewaktu tegangan yang dibutuhkan tersedia untuk terminal - terminal yang ada, motor akan memutar dan menyuplai kekuatan (*power*) mekanikal. Dengan kata lain, sewaktu *power* mekanikal disuplay untuk menjalankan tongkat dan medan kumparan dibangkitkan dari sumber DC, tenaga potensial DC akan tersedia dari terminal dan dapat digunakan untuk mengoperasikan peralatan elektrikal dan elektronik yang lain.

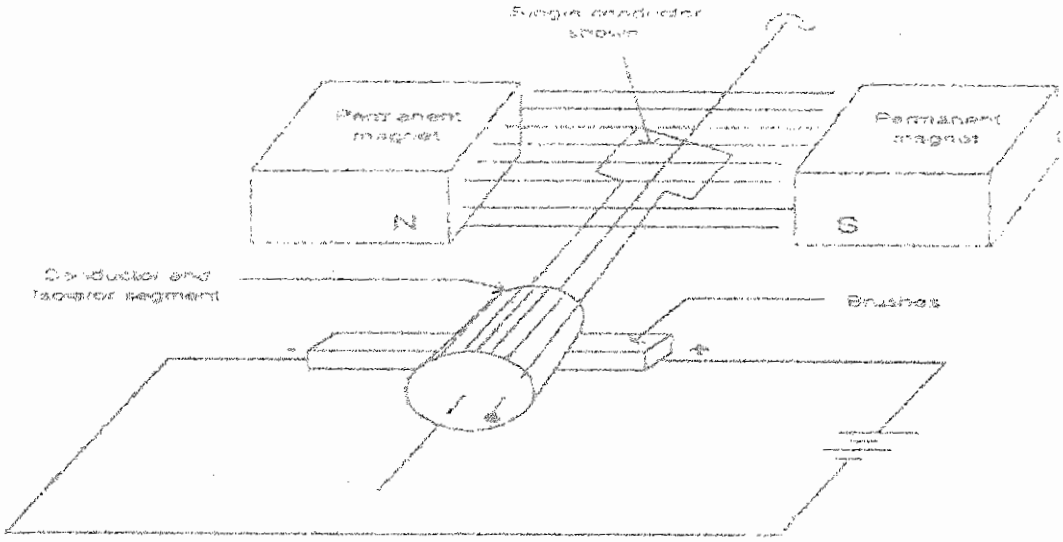
Hal lain yang menarik dari prinsip ini terlihat dari hubungan antara kumparan medan (*field winding*), berseri dengan lilitannya. Motor kemudian akan dioperasikan sumber AC dan atau DC. Motor yang seperti ini dinamakan

universal motor (motor secara umum) yang kebanyakan digunakan sebagai peralatan rumah tangga seperti *mixer*, *blender*, dan *vacuum cleaner*.

Motor DC pada dasarnya adalah sebuah torsi yang mengkonversikan energi listrik menjadi mekanik. Secara umum motor terdiri dari bagian – bagian :

- *Armature* : menghubungkan sumber energi listrik dengan motor. *Armature* biasanya berbentuk silinder, yang terdiri dari beberapa lilitan konduktor. *Armature* yang biasa berputar disebut *rotor*.
- *Magnet* : *fluks* medan magnet dari magnet akan dipotong oleh *rotor*. Bila magnet ini berupa lilitan maka arus yang akan menghasilkan medan magnet biasanya sama dengan arus untuk *armature* atau *rotor*. Magnet ini akan memiliki posisi yang tetap sehingga biasa disebut *stator*.
- *Brushes* atau sikat : merupakan penghubung antara *armature* dengan sumber tegangan.
- *Commutator* : mekanisme yang membuat arus sehingga motor dapat berputar.

Kompenen dasar dari sebuah motor DC sederhana digambarkan seperti dibawah ini :



Gambar 2.1 Komponen motor DC

Gambar diatas menunjukan sebuah *armature* dengan satu atau lebih lilitan berbentuk sayap yang saling berpisah dan berputar. Setiap lilitan berujung pada sebuah cincin terpisah (*commutator*), dimana energi (listrik) dialirkan menuju *Commutator* melalui sikat (*brushes*). Diantara *commutator* terdapat *isolator*, sehingga cincin ini berlaku sebagai saklar *Double-Pole Double-Throw*. Pada saat *armature* berputar, *commutator* akan men-*switch* arus terus menerus, sehingga medan magnet *armature* akan bernilai tetap. Putaran armature timbul karena medan magnet *armature* melawan medan elektromagnetik tetap akan disebut *field*. Pada motor dengan magnet tetap (permanen magnet), *field* ditimbulkan oleh magnet tetap.

2.2 FPGA

Field Programmable Gate Array (FPGA) adalah bagian dari *Programmable Logic Devices* (PLD), yang merupakan sirkuit dengan gerbang logika *internal* dan memiliki koneksi yang dapat berubah oleh proses pemogramannya. Keuntungan menggunakan PLD dalam sistem digital adalah dapat terprogram dengan menyertakan fungsi logika yang komplek dengan menggunakan satu IC. Aplikasi untuk fungsi yang lebih komplek, teknologi VLSI lebih diutamakan. Desain VLSI ditujukan untuk sistem digital yang memiliki 1000 sampai satu juta gerbang dengan satu IC.

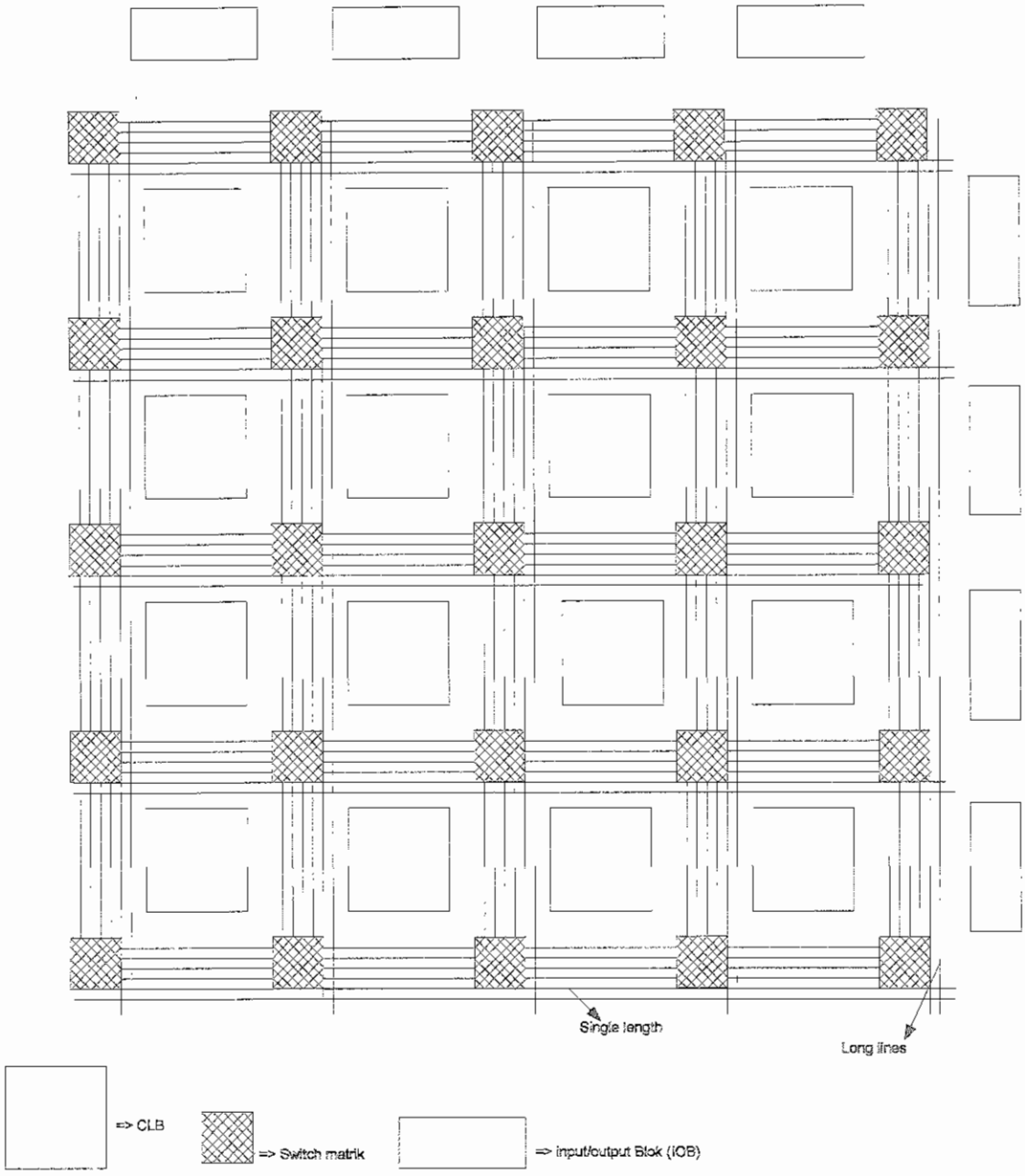
Pada saat ini, pendekatan VLSI lebih dikembangkan untuk perkembangan PLD yang mana untuk mengatasi desain yang lama untuk diimplementasikan dengan banyak chip yang kecil atau yang memiliki gerbang dari 1000 sampai satu juta gerbang. Hasil dari pendekatan tersebut adalah *high-capacity* PLD, yang sekarang lebih dikenal dengan sebutan *Field-programmable gate array* (FPGAs). Adapun tipe dari struktur desain ini adalah sebagai berikut :

- Jumlah substansial dari logika kombinasional yang tak terikat,
- Preimplementasi dari *flip-flops*, dan
- Interkoneksi programmable antara logika kombinasional, *flip-flops*, dan *chip input* dan *output*.

Untuk menggambarkan sistem ini, akan diberikan contoh salah satu tipe dari FPGA, yaitu :

Xilinx XC4000

Struktur dari FPGA ini dapat di lihat pada gambar di bawah ini :



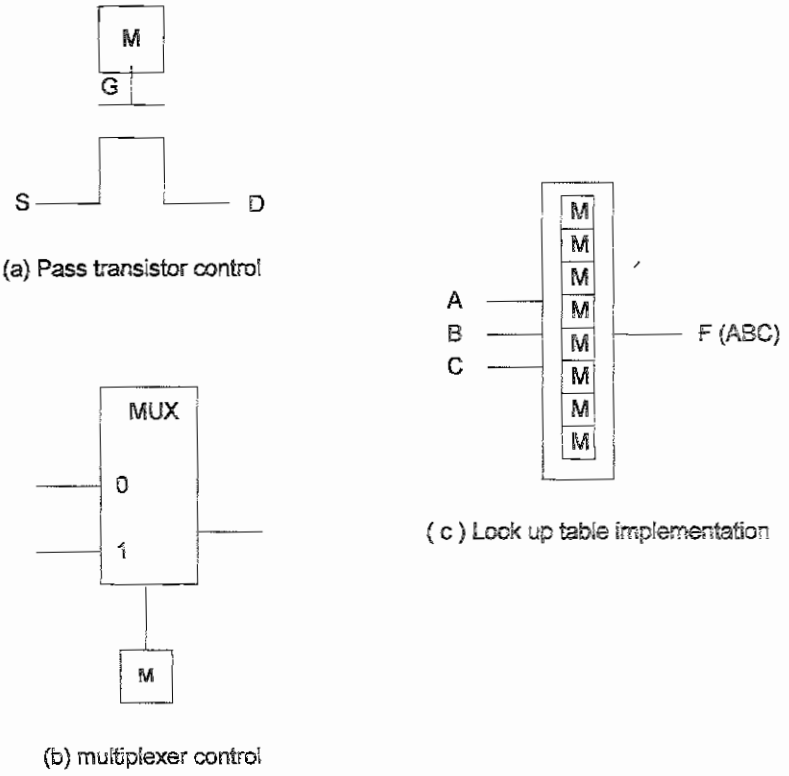
Gambar 2.2 Xilinnx XC4000 FPGA *Structure*[1]

[1] Logic and Computer Design Fundamentals 2nd edition updated.p328

Logika dengan menggunakan FPGA diimplementasikan dalam sebuah *array* dari blok-blok yang terprogram dari logika yang dinamakan *Configurable Logics Blocks* (CLBs). Masukan ke dan keluaran dari *array* ditangani oleh *input* dan *output blocks* (IOBs) sepanjang rusuk dari *array*. CLBs dan IOBs terkoneksi oleh sebuah variasi dari struktur interkoneksi program. Koneksi ke dan dari CLBs dan IOBs dapat diprogram, dan menggunakan *segment* yang terkoneksi, dari jalur yang terformat dari blok menuju bagian lainnya, dengan menggunakan sebuah *array* dari blok koneksi program yang disebut *switch matrix.*

Xilinx menggunakan teknologi SRAM untuk memberikan informasi programming. Setelah tenaganya tersalurkan ke sirkuit, data program akan menemukan konfigurasi logika yang harus disimpan ke dalam SRAM. Itu adalah salah satu dari berbagai cara untuk menyimpan informasi. Pada kenyataannya, Sebuah FPGA benar-benarr memiliki logika untuk menyimpan informasi dengan sendirinya dari PROM. Sekali informasi dalam programming tersimpan, FPGA akan memilih dari programming mode ke *operational mode* jika logikanya tersedia. Logika dari FPGA manapun jika terreprogram atau dimatikan. Kemampuan untuk mereprogram dari FPGA memberikan logika yang yang berbeda pada setiap implementasinya dalam sistem oleh FPGA yang sama pada saat yang berbeda.

Logika implementasi bit kontrol SRAM dalam sebuah FPGA *Xilinx* digunakan dalam tiga cara, yaitu *pass transistor control, multiplexer control,* dan *lookup table implementation.* Ketiga cara tersebut tergambar di bawah ini :



Gambar 2.3 Logika Implementasi FPGA Xilinx[2]

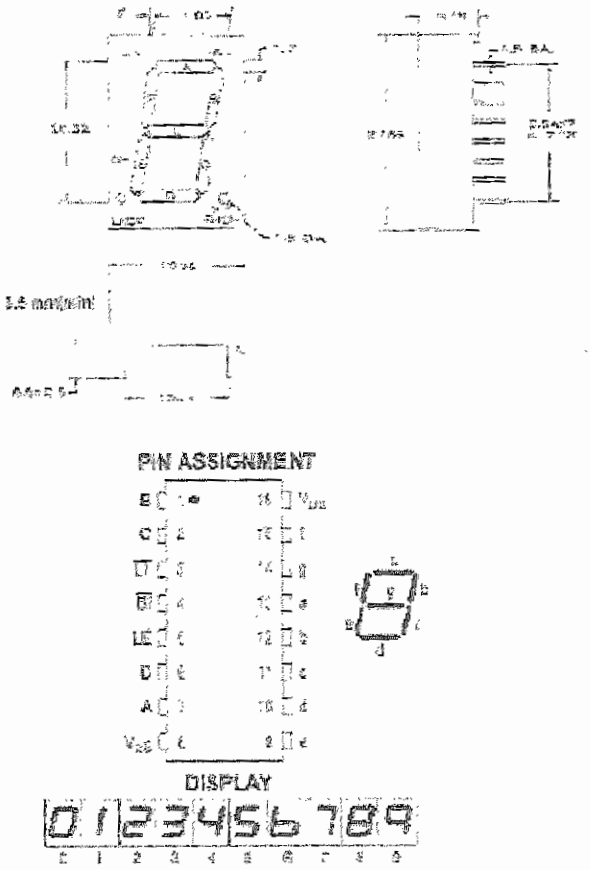
Interkoneksi Xilinx

Koneksi antara CLBs dan IOBs dibuat dengan menggunakan *segment* yang terpakai oleh kedua *channel horizontal* dan *vertikal* yang terbaring antara berbagai blok.Beberapa dari *segment* tersebut sangat panjang, memutar pada keseluruhan panjangnya. Dua *segment lagi*, yang satu disebut sebagai *long lines*, sedangkan yang satunya lagi disebut sebagai single *length*. (untuk lebih jelasnya lihat gambar 2.2).

[2] Logic and Computer Design Fundamentals 2nd edition updated.p330

2.2.1 *Seven Segment*

Display *seven segment* mempunyai 7, 8 atau 9 lampu dalam satu chip. Biasanya berupa angka desimal. Gambar di bawah ini dapat dilihat komponen dan *pin out* dari *seven segment*.



Gambar 2.4 Komponen dan *pin out* dari *seven segment*[3]

Seven segment sebenarnya adalah tujuh LED yang terbentuk persegi empat (dengan tanda a sampai g). Masing-masing LED disebut suatu segmen karena merupakan suatu bagian dari huruf (karakter) yang ditampilkan itu.

[3] http://hibp.ecse.rpi.edu/~condore/education/IEE/Lectures/lecture_7_digital_display.PDF

2.3 Sensor

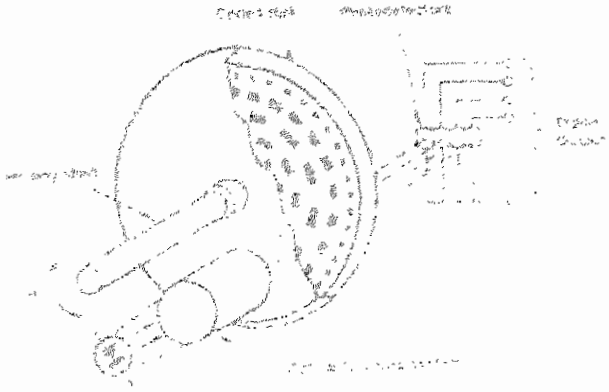
Sensor merupakan alat yang mengubah besaran fisik menjadi besaran listrik yang proposional yang digunakan untuk memberikan umpan balik kepada *mikrokontroller*. Sensor memberikan peran penting dalam perancangan sistem ini, karena memberikan informasi berupa umpan balik (*feedback*) dari sistem pengendalian yang berjalan. Dapat dikatakan lebih lanjut bahwa sensor adalah sebuah "mata" dari sistem pengendali (Goodwin, 2001,p202).

Sensor yang digunakan dalam perancangan sistem ini adalah *optical encoder*. *Optical encoder* menghasilkan *output* dalam bentuk pulsa digital. Sensor ini juga tidak mudah terkena pengaruh gelombang elektromagnetik dan oleh karena itu, dapat digunakan untuk proses informasi dan menampilkan ukuran atau posisi dan sistem kontrol. *Digital encoder* merupakan alat yang mengkonversi gerakan menjadi sebuah pulsa digital yang sekuensial. Dengan menghitung sebuah bit /men-*decode* satu set bit, pulsa tersebut dapat dikonversikan menjadi sebuah pengukuran posisi yang *relative* maupun *absolute*. *Encoder* mempunyai konfigurasi *linear* dan *rotary*, yang paling banyak digunakan adalah tipe *rotary*. *Rotary encoder* dibuat dalam dua bentuk dasar :

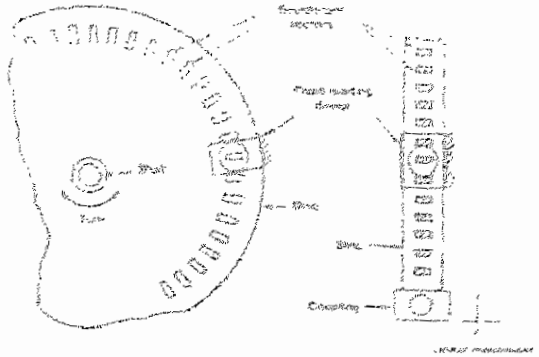
- Bentuk *Encoder Absolute*

  Dimana sebuah *data digital word* berkorespondensi dengan tiap posisi rotasi dari *shaft* dan kenaikan *encoder* yang menghasilkan pulsa digital pada saat *shaft* berputar, mebuat pengukuran *relative* terhadap posisi *shaft* seperti pada gambar di bawah ini. Kebanyakan *rotary encoder* terbuat dari kaca

ataupun plastik yang ber-*code* dengan sebuah pola fotografi radial yang tersusun pada putaran *track*-nya. Pada tiap garis – garis radikalnya tiap *track* akan meng-*interrupt* cahaya diantara pasangan foto *emitter- detector*, pulsa digital dihasilkan.



Gambar 2.5 Komponen *Optical Encoder*



Gambar 2.6 Komponen Piringan *Encoder*

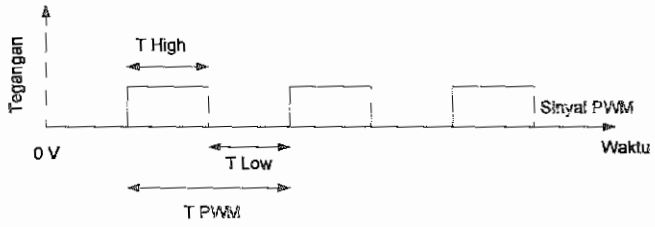
Piringan *optical* dari sebuah *encoder absolute* didesain untuk menghasilkan sebuah data digital yang berbentuk *word* yang membedakan posisi N pada posisi dari shaft.

- *Incremental encoder* kadang disebut dengan *relative encoder* dimana lebih mudah dibandingkan *absolute encoder*. *Incremental encoder* terdiri dari dua *track* dan dua sensor yang *output*-nya disebut dengan *channel* A dan *channel* B. karena perputaran *shaft*, terjadilah urutan pulsa pada kedua *channel* pada frekuensi proporsional ke kecepatan *shaft* dan hubungan fasenya antara sinyal menghasilkan arah perputaran.

2.4 *Pulse Width Modulation (PWM)*

Pada dasarnya PWM (*Pulse Width Modulation*) adalah suatu cara untuk mengubah besaran digital ke besaran analog. Pada metode PWM suatu sinyal kotak frekuensi tinggi di *generate* untuk menghasilkan suatu besaran analog yang diinginkan. Sinyal kotak tesebut dapat dihasilkan dengan cara mematikan dan menyalakan kembali suatu alat(misalnya motor DC) dengan periode tertentu.

Hal yang terpenting pada metode PWM ini adalah pengaturan unsur waktu (t) dimana *unsure* waktu inilah yang mempengaruhi besaran analog yang diberikan kepada motor DC. Yang dimaksud dengan mengatur unsur waktu pemberian tegangan kepada motor DC adalah memberikan pulsa – pulsa yang mempunyai lebar waktu yang berbeda pada saat *on (high)* dan *off (low)* yang berbeda, seperti pada gambar



Gambar 2.6 *Timing* Diagram PWM

2.5 Logika *Fuzzy*

Fuzzy logic pertama kali diperkenalkan oleh Dr. Loffi A.Zadeh professor *Computer Science University of California* di Berkeley pada tahun 1965 dan berhasil diaplikasikan dalam bidang kontrol oleh E.H. Mamdani. Sejak itu aplikasi dari *fuzzy logic* berkembang pesat. *Fuzzy logic* pada dasarnya merupakan logika bernilai banyak (*Multivalued logic*) yang dapat mendefenisikan nilai antara keadaan yang biasa kita kenal seperti ya – tidak, hitam –putih, benar- salah dan nol- satu. *Fuzzy Logic* menirukan cara manusia mengambil keputusan dengan kemampuannya bekerja dari data yang samar atau tidak rinci dan menemukan penyelesaian yang tepat.

Fuzzy logic berangkat dari kenyataan bahwa dunia nyata adalah sangat kompleks. Komplektisitas ini muncul dari ketidakpastian dalam bentuk ke tidak telitian (*imprecision)* informasi. Mengapa komputer yang dibuat manusia tidak mampu menangani persoalan yang kompleks dan tidak presisi ini sedangkan manusia biasa Jawabnya adalah manusia yang mempunyai kemampuan untuk menalar (*Reasoming*) dengan baik yaitu kemampuan yang komputer tidak mempunyainya.

Pada suatu sistem jika kompleksitasnya berkurang, maka persamaan metematik dapat digunakan dan ketelitian menjadi sangat berguna dalam pemodelan sistem tetapi jika kompleksitasnya bertambah dimana persamaan matematik tidak dapat digunakan, *fuzzy logic* menjadi salah satu alternatif penyelesaian. *Fuzzy logic* merupakan alternatif cara berpikir yang dapat memodelkan kekompleksan suatu system dengan menggunakan pengetahuan dan pengalaman yang kita miliki.

Dalam penggunaan *fuzzy logic*, yang perlu diperhatikan adalah *truth values* (nilai kebenaran) dan *membership values* (nilai keanggotaan yang menghubungkan suatu elemen pada suatu himpunan dengan tingkat keanggotaanya), yang nilainya antara 0.0 sampai 1.0. dimana nilai 0.0 menunjukan kesalahan mutlak dan 1.0 menunjukan kebenaran mutlak.

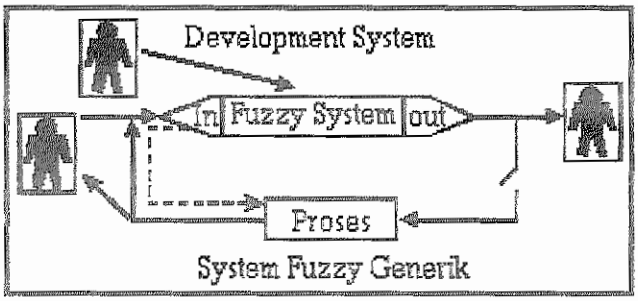
2.5.1 Konsep Logika *Fuzzy*

Seperti dikatakan sebelumnya dalam menggunakan *fuzzy logic*, ada beberapa hal pokok yang perlu diperhatikan yaitu *truth values* dan *membership values*, *fuzzy set, membership degree* dan *membership function*. Himpunan *fuzzy* adalah semua penggambaran tentang keadaan suatu masalah. *Membership degree* dan ,*membership function* adalah termasuk dalam himpunan *fuzzy*. *Membership function* memberi hasil yang disebut *membership values*. Pada masalah yang lebih luas, akan ditemukan sub himpunan *fuzzy* yang satu akan berpotongan dengan sub himpunan *fuzzy* yang lain dan himpunan *fuzzy* akan lebih dari satu himpunan. Hal

ini menyebabkan perlunya suatu operasi himpunan *fuzzy*, yang berguna untuk menentukan tingkat derajat keanggotaan.

Aplikasi yang menggunakan logika *fuzzy*, selalu identik dengan pengendalian *fuzzy*. Walaupun sebenarnya aplikasi itu digolongkan dalam klasifikasi *fuzzy* atau diagnosis *fuzzy*. Kejadian ini bukanlah masalah yang dominan dan pelik dalam sistem *sistem fuzzy*, karena istilah "*fuzzy*" sebenarnya sudah kabur dan sering disamakan dengan istilah – istilah yang ada pada teori himpunana *fuzzy*, topologi *fuzzy* atau dalam pengertian yang lebih sempit lagi sering disebut sebagai *appromaxite reasoning* dalam logika keputusan. Dengan cara pandang yang sama system kendali *fuzzy* sering sekali dinyatakan sebagai bagian teori himpunan *fuzzy* yang digunakan pada aplikasi – aplikasi dalam bentuk sistem lingkar tertutup(*close loop system*). Membedakan antara sistem kendali *fuzzy* dengam sistem klasifikasi *fuzzy* dan sistem diagnosis *fuzzy*. Pada ruang lingkup yang lebih luas lagi, masih ada sistem lainnya yang cukup sukses digunakan seperti sistem pakar *fuzzy*, system analisa data *fuzzy*, sistem pengolahan citra *fuzzy*, dan berbagai ragam aplikasi sistem *fuzzy* yang sudah ada. Dibawah ini akan dijelaskan mengenai perbedaan antara sistem kendali *fuzzy* dengan sistem klasifikasi *fuzzy* dan sistem diagnosis *fuzzy*.

1. Sistem *Fuzzy* Secara Umum

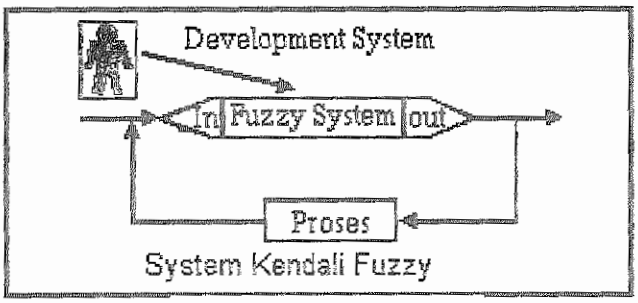


Gambar 2.8 Sistem *Fuzzy Generik*[4]

Sistem *fuzzy* secara umum dapat dilihat pada gambar diatas. Pada gambar tersebut terdapat blok proses, sistem *fuzzy*, dan sistem pengembangan (*development system*). Pihak *developer* diletakkan paling atas pada gambar ini. Selain itu, terdapat dua *operator*, yaitu seorang yang bertanggung jawab atas masukan untuk sistem *fuzzy* dan keluaran dari proses, dan seorang lagi bertugas membawa masukan ke dalam prosese dan menentukan keluaran dari sistem *fuzzy*. *Operator* ini sebenarnya tidak mesti seorang *operator* manusia, biasanya sistem *fuzzy* atau *non fuzzy* yang berfungsi mengantarkan masukan atati keluaran sinyal proses. Dari gambar ini dapat diturunkan beberapa sistem *fuzzy* seperti pengendali *fuzzy*, *klasifikator fuzzy* dan sistem pendiagnosaan *fuzzy*

[4] http://www.elektroindonesia.com/elektro/no6b.html

2. Sistem Kendali *Fuzzy*

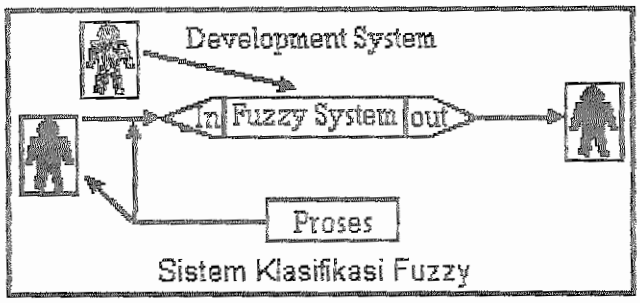


Gambar 2.9 Sistem Kendali *Fuzzy*[5]

Sebuah kendali *fuzzy* yang digambarkan pada gambar diatas merupakan suatu sistem lingkar tertutup, dimana tidak terdapat *operator* yang menjadi bagian dari sistem lingkar kendali (*control loop*). Contoh dari sistem kendali ini adalah *vaccum cleaner*. Sistem pada alat ini mengatur daya motor penghisap tergantung pada banyaknya debu di lantai atau karpet. Contoh lain dari sistem kendali *fuzzy* adalah optimasi torsi dalam sistem anti slip yang digunakan kereta listrik dan sistem kereta bawah tanah. Masukan sistem kendali berupa kecepatan kereta dan koefisien resistansi rel.

[5] http://www.elektroindonesia.com/elektro/no6b.html

3. Sistem Klasifikasi *Fuzzy*



Gambar 2.10 Sistem Klasifikasi *Fuzzy*[6]

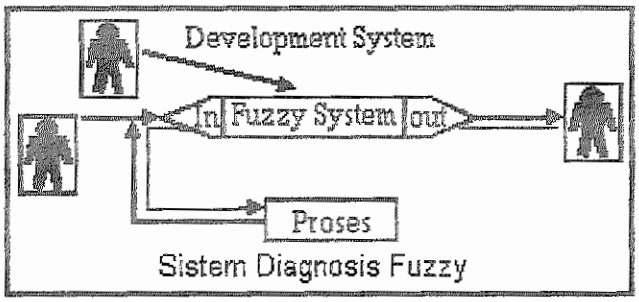
Pada sistem klasifikasi *fuzzy* diatas tidak terdapat *loop* tertutup. Sistem ini hanya menerima masukan dan keluaran dari proses untuk selanjutnya memberikan informasi berupa kondisi (*state*) dari proses tadi. Informasi kondisi ini dapat digunakan untuk mengendalikan sistem atau memeberikan tanggung jawab kendali kepada *operator*. Secara matematis, sistem klasifikasi lebih dekat pada teori himpunan dari pada teori fungsi. Pada sistem ini, sifat kesamaan (*Vagueness*) sering ditemui pada opini pakar dan jarang menggunakan model relasi *fuzzy*.

Contoh dari sistem klasifikasi *fuzzy* adalah mesin cuci *fuzzy*. Beberapa *variable/* parameter mesin cuci ditentukan berdasarkan jumlah dan jenis pakaian. Keluaran atau informasi dari sistem klasifikasi ini digunakan untuk menentukan jenis *spin-dry* serta lembut atau kasar gesekan pakaian yang optimal. Contoh kedua dari sistem *fuzzy* ini adalah sistem transmisi otomatik *fuzzy*. Sistem ini menggunakan beberapa sensor

[6] http://www.elektroindonesia.com/elektro/no6b.html

yang ditaruh pada sistem ABS, sistem *power steering*, sistem kendali motor, dan bagian penting lainnya. Selama kendaraan berjalan, sistem ini akan terus memantau dan menilai kondisi mobil tersebut, sepeerti beban kendaraan, kondisi mobil saat melewati jalan yang menanjak atau menurun dan kondisi – kondisi lainya. Pada gambar diatas, gambar *operator* manusia pada kiri dan kanan sistem klasifikasi *fuzzy*, biasanya merupakan suatu sistem khusus yang bertugas memberikan informasi yang diperlukan untuk kemudian di proses.

4. **Sistem diagnosis *Fuzzy***



Gambar 2.11 Sistem Diagnosis *Fuzzy*[7]

Pada sistem diagnosis *fuzzy* gambar diatas peranan manusia/ *operator* lebih dominan. Pengiriman data dilaksanakan oleh *operator* ke dalam sistem, ketika sistem memerlukan data tambahan. Selain itu *operator* dapat meminta atau menanyakan informasi dari sistem diagnosis berupa hasil konklusi diagnosis atau *prosedur detail* hasil diagnosis oleh sistem. Dari sifat sistem ini, sistem diagnosis *fuzzy* dapat digolongkan

[7] http://www.elektroindonesia.com/elektro/no6b.html

pada sistem pakar *fuzzy*. Sistem pakar *fuzzy* adalah sistem pakar yang menggunakan notasi *fuzzy* pada aturan – aturan dan proses inferensi (logika keputusan). Salah satu kelebihan sistem pakar *fuzzy* dibandingkan sistem pakar konvensional adalah jumlah aturan lebih sedikit, sehiungga sistem lebih transparan untuk dianalisa. Kekurangannya adalah kehandalan sistem sangat tergantung pada baik buruknya proses pengumpulan aturan seperti prosedur pertanyaan dan komponen – komponen kuisioner, serta sering terjadi kesulitan untuk menyimpulkan suatu pernyataaan tertentu oleh *operator*.

Bidang aplikasi sistem diagnosis ini biasanya suatu proses yang besar dan kompleks, sehingga sangat sulit dianalisa menggunakan algoritma eksak dan dimodelkan dengan model matematika biasa. Pada permulaan persiapan sistem, jumlah aturan akan lebih sedikit dan mudah dibaca. Ini merupakan sifat sistem pakar *fuzzy*, seperti yang dikatakan oleh Prof. Zadeh, bahwa sistem pakar konvensional sehingga mudah dibaca dalam membantu menghindarkan inkonsistansi dan inkomplit sistem pengendali. Yang perlu diperhatikan pada sistem diagnostik ini adalah, tidak berlakunya proses defuzzifikasi, karena sistem ini hanya menghasilakn sifat keluaran berupa aproksimasi linguistik yang merupakan suatu pernyataan atau jawaban yang mudah dipahami oleh *operator*.

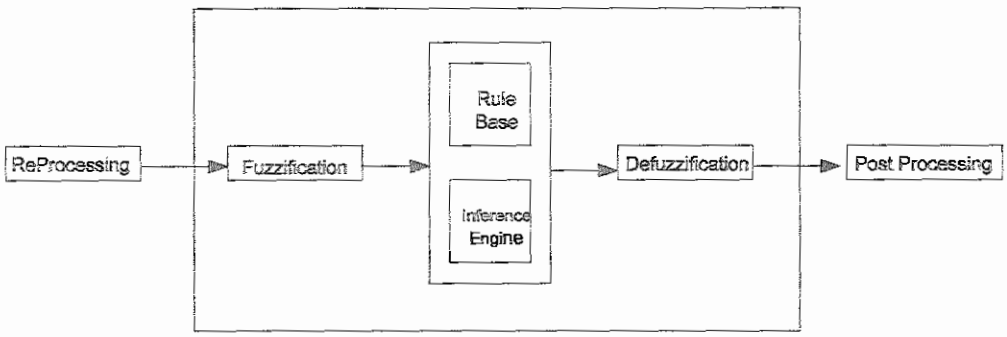
2.5.2 *Fuzzy Logic Controller*

Sistem kontrol berbasiskan mikroprosessor digital sudah sangat banyak digunakan dalam proses kontrol suatu sistem. Bila dibandingkan dengan sistem kontrol secara analog, sistem kontrol digital jauh lebih fleksibel dalam responnya. Hal tersebut disebabkan karena banyaknya variasi algoritma yang bias diterapkan dalam sistem kontrol digital. Sistem kontrol digital mempunyai keunggulan lebih baik untuk menangani ketidak – linearan dari sistem, perubahan keadaan pada sistem seperti perubahan beban, *supply system* dan juga perubahan parameter – parameter lainnya.

Algoritma yang digunakan dalam sistem kontrol digital dapat dipilih berdasarkan beberapa metode tertentu seperti *Proportional – Integral – Derivative* (PID), Model *Reference Adaptive Control* (MRAC). Metoda kontrol PID cukup efektif untuk menangani sistem dengan respon tetap namun bila system menjadi tidak linear maka sistem PID akan mengalami kesulitan. Metoda MRAC dapat menangani ketidak – linearan sistem dan lingkungan kontrol yang berubah – ubah dengan cara membandingkan hasil proses *output actual* dengan model referensi. Namun metode ini membutuhkan model metematika dari proses yang berlangsung untuk mensimulasikan relasi *input-output*. Untuk menciptakan model matematika dari sistem tentunya merupakan kesulitan sendiri.

FLC adalah merupakan pengembangan dari teori *fuzzy logic* dan digunakan untuk pengontrolan suatu alat atau sistem. Seperti halnya pola pikir manusia yang membuat keputusan (*making decision*), didasarkan atas aturan-aturan yang ada maka FLC akan bertindak sama. Pola pikir manusia dalam

membuat keputusan bila diterapkan pada logika komputer merupakan kalimat *if-then*, begitu pula pada FLC, FLC akan bekerja berdasarkan aturan yang ada. Contohnya *if* A *then* Y, atau *If* B *then* Z. dibawah ini adalah bagan FLC.



Gambar 2.12 *Fuzzy Logic Controller*[8]

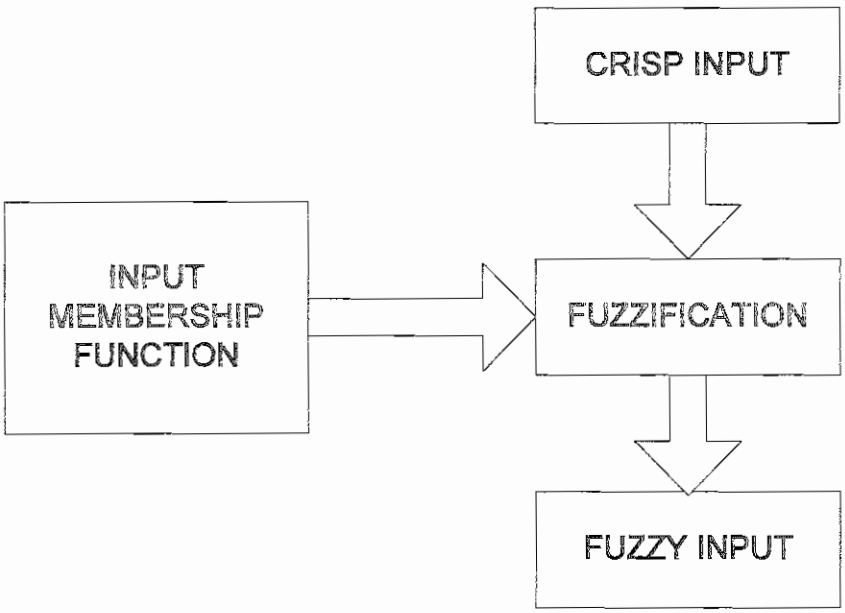
Aturan pada *fuzzy logic (fuzzy rules)* sangatlah penting didalam FLC, untuk menerapkan teori *fuzzy logic*.oleh sebab itu diperlukan langkah – langkah seperti bagian diatas tersebut yang meliputi :

- *Fuzzification*, yaitu proses penentuan masukan (*input*) yang akan diubah nilainya oleh *membership function.*
- *Rule Evaluation*, yaitu proses dimana nilai hasil dari *fuzzification* dibandingkan dengan aturan-aturan yang telah dibuat.
- *Defuzzification*, mengubah hasil dari *rule evaluation* menjadi nilai yang dapat dijalankan (*crisp output*).

2.5.2.1 Fuzzifikasi

Proses ini berfungsi untuk merubah suatu besaran *analog/ digital* menjadi *fuzzy input*. Secara diagram blok dapat dilihat pada gambar dibawah ini :

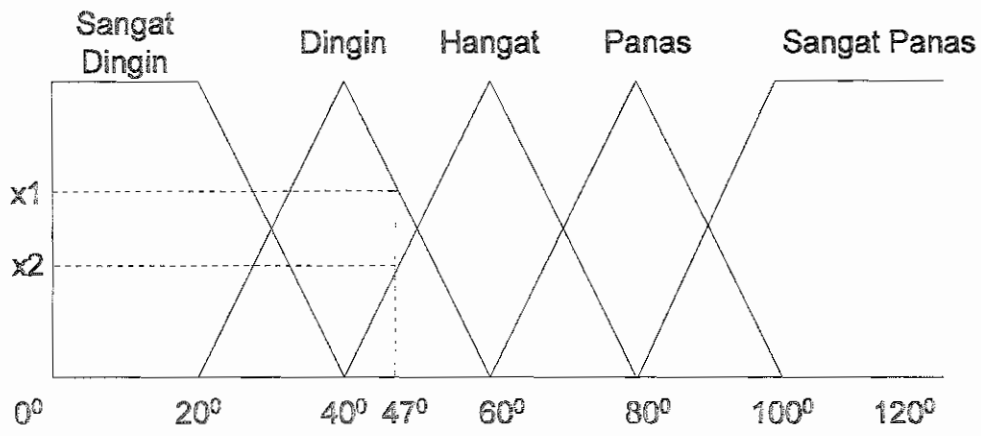
[8] http://www.elektroindonesia.com/elektro/no6b.html



Gambar 2.13 Fuzzifikasi

Prosesnya dapat dijelaskan sebagai berikut :

Suatu besaran analog dimasukan sebagai *input* (*crisp input*), lalu *input* tersebut dimasukan pada batas *scope*/dominan (merupakan suatu batas dari kumpulan *input* tertentu, misalnya panas antara 60-100`C) sehingga *input* tersebut dapat dinyatakan dengan label (dingin, hangat, panas) dari *membership fuction input*. Dari *membership function* kita bisa mengetahui berapa -nya (memberikan bobot pada suatu input yang telah kita berikan sehingga *input* tadi dapat dinyatakan dengan suatu nilai, biasanya antara 0.0-1.0). contoh dari proses *fuzzification* adalah sbb:

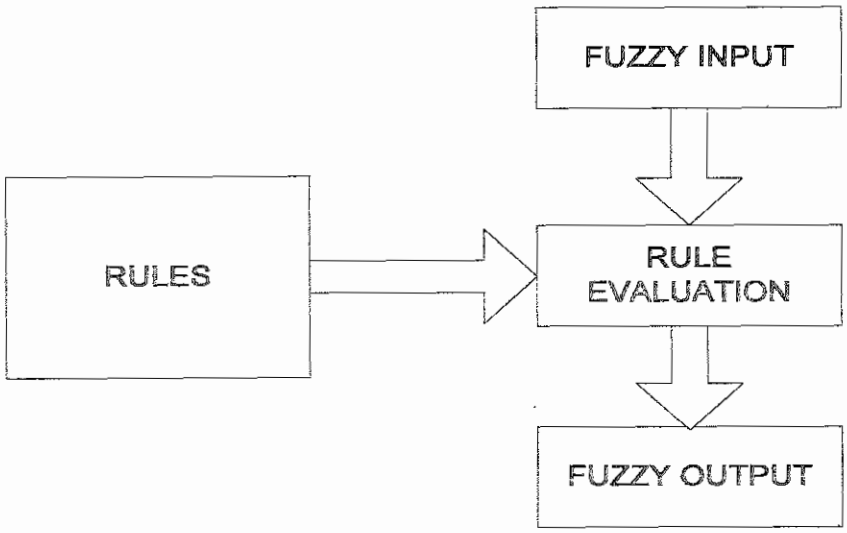


Gambar 2.14 *Membership Function*

Dari gambar diatas diperoleh crisp input adalah 47`C. sehingga didapat 2 *fuzzy input* yang masing- masing adalah : dingin (X2)dana hangat (X1), nilai X1 dan X2 dapat dicari dengan rumus persamaan garis. Yang menentukan sistem tersebut *sensitive* atau tidak adalah *membership function* ini. Jika *membership function*-nya banyak maka sistem akan menjadi sangat *sensitive*. Yang dimakasud *sensitive* disini adalah jika *input*-nya berubah sedikit saja maka sistem akan cepat merespon dan menghasilkan suatu *output* lain. *Output* dari *fuzzification* ini adalah sebuah nilai input fuzzy.

2.5.2.2 *Rule Evaluation*

Proses ini berfungsi untuk mencari suatu nilai *fuzzy – output* dari *fuzzy – input*. Suatu nilai *fuzzy input* yang bersala dari proses *fuzzification* kemudian dimasukkan kedalam sebuah *rule* yang telah dibuat untuk dijadikan sebuah *fuzzy output*, dapat digambarkan sbb:



Gambar 2.15 *Rule Evaluation*

Gambar diatas merupakan bagian utama dari *fuzzy*, karena disinilah sistem akan belajar menjadi pintar atau tidak. Jika sistem tersebut tidak pintar dalam mengatur *rule* maka sistem yang akan dikontrol menjadi kacau. Format dari rule adalah sebagai berikut :

"If antecedent1 operator antecedent2 then consequent1 operator consequent2"

Contoh :

" *If* suhu panas *and* kelembapan *is* kering *then* penyemprot sangat lama"

ada beberapa *operator* yang digunakan dalam *And, Or. Not*.jika *operator* yang digunakan adalah *And* maka *input* yang terkecil diambil. Misalnya :

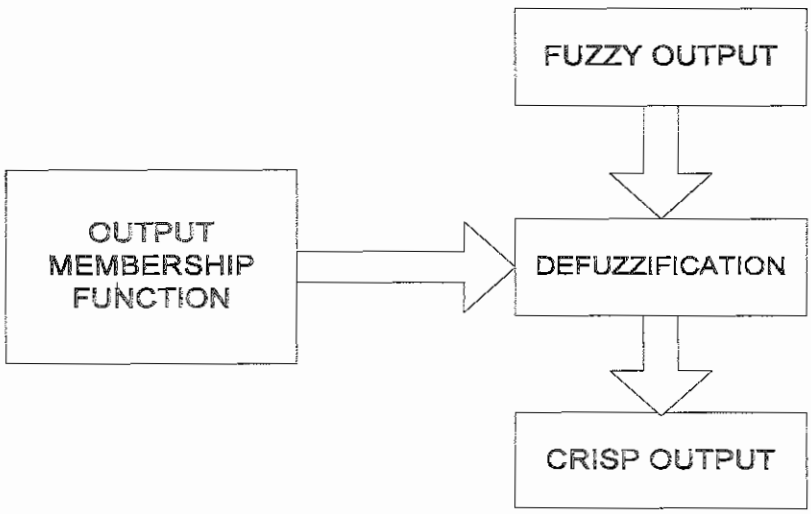
"*If* suhu *is* panas (0.15) *and* kelembapan (0.19) *then* penyemprot sangat lama":

Nilai *fuzzy output* dari pernyataan tesebut adalah 0.15. nilai 0.15 dan 0.19 dari contoh diatas diambil dari dua *membership function input* dengan cara

menarik garis lurus *vertical* dari nilai yang diinginkan.jika *operator* yang digunakan adalah *Or*, maka *fuzzy output*-nya diambil dari nilai yang terbesar. Jika operasi yang digunakan adalah *operator Not* maka *fuzzy output*-nya adalah kebalikannya. Misalnya *Not* 0.9 maka akan menghasilkan 0.1 dan *Not* 0.8 akan menghasilkan 0.2

2.5.2.3 Defuzzifikasi

Proses defuzzifikasi merupakan proses untuk memetakan data dari lingkungan fuzzy kembali menjadi data yang bisa diaplikasikan untuk kontrol sistem. Proses defuzzifikasi harus suatu nilai *fuzzy output* yang berasal dari *rule evaluation* diambil kemudian dimasukan kedalam suatu *membership function output*. Besarnya nilai *fuzzy output* dinyatakan sebagai *degree of membership function output*.



Gambar 2.16 Defuzzifikasi

Nilai tersebut dimasukan kedalam suatu rumus yang dinamakan COG (*center Of Grafity*) untuk mendapatkan hasil akhir yang disebut *crisp output*. *Crisp output* adalah suatu nilai digital yang akan dibutuhkan oleh sistem untuk mengolah data pada sistem yang telah dirancang. Rumus *Center Of Grafity* dapat ditulis pada persamaan di bawah ini :

$$\text{Crisp\_Output} = \frac{\sum_1 (\text{Fuzzy\_Output}_1) \times (\text{Point\_Center\_Position})}{\sum_1 (\text{Fuzzy\_Output}_1)}$$