# Teknologi Integrated Circuit (IC) : Menuju Airforce Industry

Oleh :

**Kapten Lek Ir. Arwin D.W. Sumari, FSI, FSME, VDBM, SA**[1]

## Era Pra Teknologi IC

Evolusi teknologi elektronika diawali dari keinginan manusia untuk melakukan otomatisasi pekerjaannya baik di kantor maupun di rumah. Pada awalnya semua pekerjaan itu dilakukan secara manual dengan bantuan hewan atau alat-alat lainnya. Revolusi industri dunia yang dipelopori oleh negara Inggris telah mengubah wajah industri dari tenaga kasar ke tenaga mesin. Secara perlahan tetapi pasti tenaga mesin konvensional mulai ditinggalkan dengan ditemukannya komponen-komponen elektronika yang digunakan untuk membuat mesin yang lebih baik.

Salah satu penemuan besar manusia adalah peralatan elektronika "pintar" yang dinamakan dengan komputer. Komputer generasi pertama menggunakan komponen elektronika dalam bentuk tabung sehingga bentuknya sangat besar, memakan tempat dan tidak *portable*. Perkembangan teknologi elektronika yang sangat pesat seiring dengan kebutuhan manusia akan peralatan yang kompak dan dapat dibawa ke mana-mana memunculkan ide untuk memperkecil ukuran komponen-komponen elektronika tersebut. Ide ini akhirnya dapat direalisasikan setelah ditemukannya teknologi yang memungkinkan mereka mengimplementasikan hal ini yakni teknologi **Integrated Circuit (IC).**

## Teknologi IC

IC adalah suatu media yang berisi berbagai macam komponen elektronika yang terintegrasi dan terhubung satu dengan lainnya sedemikian rupa untuk melaksanakan suatu fungsi tertentu. IC umumnya berwarna hitam dengan kaki-kaki yang banyak sehingga kadang disebut dengan

---

[1] Kepala Urusan Operasi Faslat Wing – 3, Flight Simulator Instructor (FSI), Flight Simulator Maintenance Engineer (FSME), Visual Database Modeler (VDBM) dan System Administrator (SA) Full Mission Simulator F-16A Faslat Wing – 3, Lanud Iswahjudi

komponen "kaki seribu". Bila Anda pernah membuka *casing* komputer atau peralatan elektronik lainnya dan melongok ke dalamnya, Anda akan melihat banyak sekali benda segi panjang atau bujur sangkar berwarna hitam atau kadang abu-abu dengan tulisan kode-kode tertentu di punggungnya, itulah yang dinamakan dengan IC. Satu IC dapat berisi ribuan bahkan jutaan komponen elektronika seperti resistor, capacitor dan transistor. Prosesor Pentium IV berisi lebih dari 10 juta transistor di dalam IC-nya. Betapa kecilnya komponen-komponen elektronika tersebut hingga mampu berdesak-desakan dalam jumlah yang sangat banyak di dalam sebuah IC yang ukurannya tidak lebih dari 25 $cm^2$. Itulah kehebatan teknologi elektronika.

Pada awalnya jumlah komponen yang dapat dimasukkan ke dalam sebuah IC sangat terbatas. Perkembangan teknologi IC mampu mengatasi hal ini sehingga kita mengenal istilah LSI, VLSI dan ULSI. LSI adalah singkatan dari Large Scale Integration sedangkan VLSI adalah singkatan dari Very LSI dan ULSI dari Ultra LSI. ULSI mengandung jumlah transistor lebih banyak dibandingkan VLSI dan seterusnya. Pada tahun 1965, pendiri Intel Gordon Moore mengatakan bahwa jumlah transistor per inci persegi di dalam sebuah IC akan meningkat dua kali lipat setiap tahun dan ini menjadi kenyataan. Lalu bagaimana caranya memasukkan berjuta-juta transistor ke dalam sebuah IC seperti prosesor Pentium IV ? Hal ini dapat dilaksanakan dengan ditemukannya teknologi Nano yang memungkinkan untuk membuat transistor dengan ukuran 0,18 micron atau 0,18 x $10^{-6}$ m, betapa kecilnya.

Proses pembuatan IC diawali dari merancang rangkaian elektronika sesuai dengan fungsi yang diharapkan. Rancangan ini kemudian dituangkan ke bentuk IC melalui proses yang cukup panjang dan hati-hati. Wadah yang digunakan untuk mengimplementasikan rangkaian elektronika ini dinamakan dengan *wafer*. Satu lembar *wafer* dapat berharga berjuta-juta rupiah sehingga agar *break event point* atau kembali modal plus keuntungan, pabrik pembuat IC akan memproduksinya dalam jumlah besar. Pengerjaan IC ini harus dilakukan dalam ruangan dengan suhu di bawah $18^o$ C dengan konsentrasi debu satu pro mil atau 1/1000 dan pekerjanya harus menggunakan pakaian khusus seperti dokter bedah. Mengapa demikian ? Aturan ini diberlakukan untuk mencegah debu yang menempel di baju, di tangan atau yang melayang di udara menempel pada rancangan rangkaian elektronika di wafer. Pada kadar tertentu, debu yang menempel dapat mempengaruhi karakteristik IC yang dibuat dan dapat menggagalkan produksi. Penulis pernah melaksanakan sendiri kegiatan ini sewaktu kuliah di Teknik Elektro ITB dan sangat mengasyikkan.

Kegiatan pembuatan IC seperti di atas tidak sulit untuk rangkaian-rangkaian sederhana dengan jumlah komponen yang minim. Namun pada saat rangkaian yang akan diimplementasikan sangat kompleks dengan komponen yang ribuan jumlahnya diperlukan cara lain agar

perancangannya lebih cepat dan meminimisasi kemungkinan kesalahan yang dapat berakibat fatal. Dengan tujuan agar IC yang dibuat telah benar-benar memenuhi persyaratan yang diberikan, dirancanglah suatu perangkat lunak (*software*) untuk tugas-tugas peracangan IC yang dinamakan dengan **Very High Speed Integrated Circuit Hardware Description Language (VHSIC HDL)** yang lebih dikenal dengan **VHDL**.

**Sekilas VHDL**

VHDL adalah *software* yang digunakan untuk menerangkan tingkah laku dan struktur rancangan perangkat keras (*hardware*) sistem digital elektronika baik rangkaian digital konvensional maupun **Applied Specific Integrated Circuit (ASIC)** dan **Field Programmable Gate Array (FPGA)**. Pada dasarnya VHDL dirancang untuk mengisi sejumlah kebutuhan dalam proses perancangan sistem digital elektronika. Pertama, ia mengijinkan penjelasan struktur suatu rancangan yakni bagaimana rancangan itu didekomposisi ke dalam sub-sub rancangan dan bagaimana sub-sub rancangan tersebut berhubungan. Kedua, ia mengijinkan penspesifikasian fungsi suatu rancangan menggunakan bentuk-bentuk bahasa pemrograman yang sudah dikenal. Ketiga, sebagai hasil akhir, ia mengijinkan suatu rancangan disimulasikan sebelum diproduksi sehingga para perancangnya dapat dengan cepat membandingkan alternatif-alternatif dan menguji ketepatan fungsinya tanpa penundaan dan penambahan biaya untuk mem-*prototype hardware* lagi.

Simulasi dan sintesa adalah dua perangkat (*tool*) utama VHDL. Fasilitas simulasi digunakan untuk mensimulasikan program rangkaian untuk menguji ketepatan fungsinya. Fasilitas sintesa digunakan untuk menguji kelayakan rangkaian untuk diimplementasikan ke bentuk *hardware*-nya karena dalam beberapa kasus rangkaian yang berhasil disimulasikan dengan sempurna tidak menjamin lulus dari pengujian sintesa. Bila ini yang terjadi harus ada modifikasi pada program rangkaian dan siklus perancangan dimulai lagi dari awal. Penulis pernah mengalami masalah yang sama ketika mensimulasikan rangkaian digital fungsi XOR. Ternyata tidak mudah karena harus mempunyai *programming sense* yang tinggi agar rangkaian yang dirancang dengan pengkodean (*coding*) tersebut dapat melewati fase sintesa dengan mulus untuk kemudian diproduksi dalam jumlah besar.

**VHDL Milestone**

Pengembangan VHDL diawali tahun 1980 oleh Departemen Pertahanan Amerika Serikat (US DoD) untuk mengatasi krisis *life cycle hardware*. Alasan mendasar perlunya pemerintah AS melaksanakan program ini adalah tidak seragamnya perangkat simulasi yang digunakan untuk membuat IC karena setiap pabrik (*manufacturer*) mempunyai perangkat simulasi yang berbeda karakteristiknya. Oleh karena diperlukan satu bahasa standar yang menjelaskan struktur dan fungsi IC, maka dibangunlah VHDL. Tak berapa lama VHDL diadopsi sebagai standar oleh **Institute for Electrical and Electronic Engineers (IEEE)** di Amerika Serikat.

Proses standardisasi VHDL cukup unik dalam hal partisipasi dan umpan balik dari industri pada awal pengembangannya. Bahasa awal (versi 7.2) dipublikasikan dua tahun sebelum versi standar. DoD Military Standard (Mil Std) 454 memandatkan agar ASIC yang dikirim ke DoD disuplai dengan penjelasan VHDL-nya. Maka cara terbaik untuk menyediakan level penjelasan yang tepat adalah menggunakan VHDL dalam proses perancangan IC. Sebagai standar IEEE, VHDL harus melaksanakan proses *review* setiap 5 tahun sekali untuk meyakinkan kesinambungan relevansinya dengan industri. Revisi pertama dilakukan pada bula September 1993 dan saat ini yang dipakai adalah **VHDL IEEE 1076-1993**. Jejak langkah VHDL dirangkum sebagai berikut :

| | | |
|---|---|---|
| Awal 1970an | - | Diskusi awal mengenai perlunya bahasa standar IC. |
| Akhir 1970an | - | Pendefinisian persyaratan-persyaratan VHDL. |
| 1981 | - | Dipelopori oleh US DoD untuk mengatasi krisis *life cycle hardware*. |
| 1983-85 | - | Pembangunan bahasa awal oleh Intermetrics, IBM dan TI. |
| 1986 | - | Semua hak diserahkan ke IEEE. |
| 1987 | - | Publikasi standar VHDL IEEE. DoD mengadopsi standar IEEE 1076. |
| 1987 | - | Mil Std 454 mensyaratkan kelengkapan VHDL bersama dengan ASIC yang dikirimkan. |
| 1988 | - | Mendapat dukungan dari CAE. |
| 1991 | - | Revisi VHDL. |

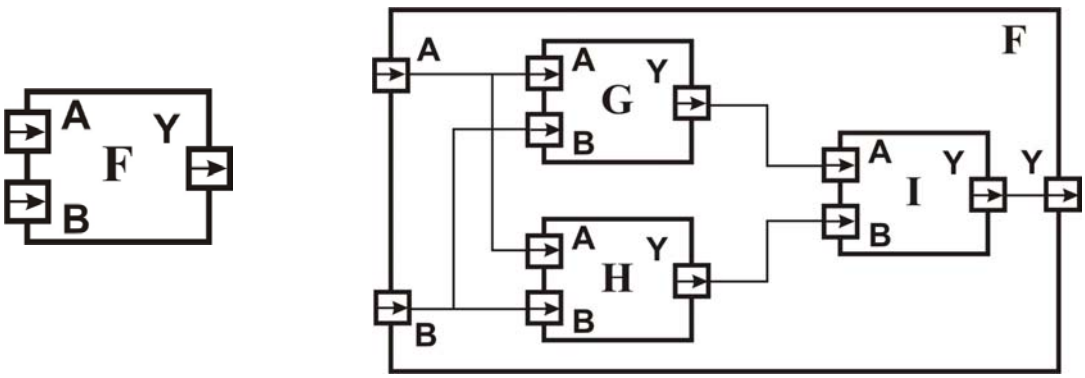
1994 - Standar yang telah direvisi diberi nama VHDL 1076-1993.

1999 - VHDL-AMS *extension*.

**Bekerja dengan VHDL**

Salah satu fitur istimewa VHDL adalah pola pemrogramannya tidak jauh berbeda dengan bahasa pemrograman C/C++ dan sifat konkurennya mengikuti bahasa pemrograman standar US DoD yakni ADA. VHDL dibuat khusus untuk konsumsi perguruan tinggi, pabrik-pabrik pembuat IC dan untuk kepentingan militer sehingga *software* VHDL tidak akan ditemukan di pasaran umum apalagi versi bajakannya seperti yang sering kita temui pada *software-software* games atau Windows. Selain itu *software* VHDL pada umumnya tidak dijalankan pada Personal Computer (PC) tapi pada *workstation* berbasis *operating system* UNIX.

Suatu sistem elektronika digital dapat dijelaskan sebagai suatu modul dengan *input* dan atau *output*. Nilai-nilai elektrik pada *output* merupakan suatu fungsi dari nilai-nilai *input*-nya. Dalam VHDL ada dua cara untuk menjelaskan fenomena ini yakni secara struktural (*structural*) dan secara tingkah laku (*behavioral*).

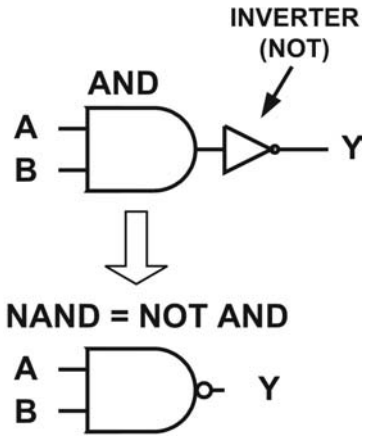



Gambar 1. (a) Sistem digital dengan rangkaian sederhana, (b) Sistem digital dengan modul yang berisi beberapa rangkaian digital sederhana.

Gambar 1 (a) memperlihatkan contoh sistem digital yang dilambangkan dalam bentuk sebuah modul F dengan dua *input* A dan B serta sebuah *output* Y. Dalam terminologi VHDL, modul F ini

disebut dengan sebuah *entity* rancangan (*design entity*) dan *input*/*output*-nya dinamakan dengan *port*. Penjelasan fungsi suatu modul secara searah adalah untuk menjelaskan bagaimana ia dibentuk oleh beberapa sub modul. Setiap sub modul adalah suatu *instance* dari beberapa entitas dan *port-port instance* tersebut dihubungkan menggunakan sinyal (*signal*). Gambar 1 (b) memperlihatkan bagaimana modul F dibentuk oleh beberapa *instance* entitas G,H dan I. Perhatikan, masing-masing entitas mempunyai *port input* dan *port output* yang menghubungkan satu entitas ke entitas lainnya yang membentuk fungsi modul F.

Dalam banyak kasus, menjelaskan suatu modul secara struktural tidak selalu tepat. Sebagai contoh jika Anda merancang suatu system menggunakan paket IC yang dibeli dari toko komponen, Anda tidak perlu menjelaskan struktur internalnya. Dalam kasus seperti ini penjelasan fungsi yang dilakukan oleh modul tersebut lebih diperlukan tanpa harus bereferensi pada struktur internalnya. Penjelasan seperti ini disebiut dengan penjelasan fungsional atau tingkah laku (*functional, behavioral*). Untuk mengilustrasikan hal ini, anggap fungsi *entity* F pada gambar 1 (a) adalah fungsi **not-and (NAND)** dan gambar 2 menampilkan bentuk rangkaian logika **NAND gate**. Maka penjelasan tingkah laku F dapat berupa fungsi Boolean $Y = \overline{A.B} = \overline{A} + \overline{B}$. VHDL menyelesaikan masalah ini dengan memberikan fasilitas untuk penjelasan tingkah laku ini dalam bentuk suatu program yang dapat dieksekusi langsung (*executable program*).




Gambar 2. Rangkaian logika NAND *gate*.

**Discrete Event Time Model**

Setelah struktur dan tingkah laku suatu modul telah dispesifikasikan, simulasi modul dapat dilakukan dengan mengeksekusi penjelasan tingkah lakunya. Ini dilakukan dengan mensimulasikan

urutan waktu secara diskrit atau diskontinu. Pada beberapa waktu simulasi, *input* modul dapat distimulasi dengan merubah nilainya pada *input port*. Modul bereaksi dengan menjalankan kode penjelasan tingkah lakunya dan menjadwalkan nilai-nilai baru untuk diletakkan pada sinyal-sinyal yang terhubung pada *output port*-nya pada waktu simulasi lainnya. Kegiatan ini dinamakan dengan penjadwalan suatu transaksi (*transaction*) pada sinyal tersebut. Jika nilai baru tersebut berbeda dari nilai pada sinyal sebelumnya, maka suatu kejadian (*event*) terjadi dan modul-modul lainnya dengan *input port* terhubung pada sinyal dapat diaktifkan.

Simulasi dimulai dari fase inisialisasi (*initialization*) dan kemudian dilanjutkan dengan mengulang siklus simulasi dua-tahap (*two-stage simulation cycle*). Pada fase inisialisasi semua sinyal diberi nilai awal (*initial values*), waktu simulasi (*simulation time*) diatur ke 0 dan setiap program tingkah laku modul dijalankan. Ini biasanya mengakibatkan transaksi dijadwalkan pada sinyal *output* beberapa waktu kemudian. Pada tahap pertama siklus simulasi, waktu simulasi dimajukan pada waktu tercepat suatu transaksi yang telah dijadwalkan. Semua transaksi yang dijadwalkan pada waktu tersebut dieksekusi dan ini dapat menyebabkan beberapa kejadian terjadi pada beberapa sinyal. Pada tahap kedua, program tingkah laku semua modul yang bereaksi terhadap kejadian-kejadian yang terjadi pada tahap pertama dijalankan. Program-program ini biasanya akan menjadwal transaksi berikutnya pada sinyal-sinyal *output*-nya. Ketika semua program tingkah laku telah selesai dijalankan, siklus simulasi berulang. Jika sudah tidak ada lagi transaksi-transaksi yang dijadwalkan, proses simulasi telah lengkap secara keseluruhan. Tujuan simulasi adalah untuk mengumpulkan informasi mengenai perubahan-perubahan di dalam keadaan sistem menurut waktu. Ini dapat dilakukan dengan menjalankan simulasi di bawah kendali suatu monitor simulasi. Monitor mengijinkan sinyal-sinyal dan informasi keadaan lain diperlihatkan atau disimpan di dalam suatu *file* pelacak untuk analisa berikutnya. Selain itu, monitor juga mengijinkan pentahapan interaktif dari suatu proses simulasi, mirip dengan program *debugger* interaktif pada bahasa pemrograman di PC.

**Implementasi Perancangan VHDL**

Agar dapat mengambil filosofi atau ide dasar VHDL, akan diberikan satu contoh cara mengimplementasikan suatu rangkaian digital ke dalam bahasa VHDL secara struktur maupun tingkah laku. Kita akan membuat sebuah rangkaian elektronika NAND (not-and). Implementasi dengan VHDL dimulai dengan mendeskripsikan entitas rangkaian tersebut dengan menspesifikasikan antarmuka (*interface*) eksternalnya yang mengikut sertakan penjelasan *port-port*-nya. Maka rangkaian ini dapat didefinisikan sebagai berikut :

```
-- file name: nand_gate.vhd

library IEEE;
use IEEE.std_logic_1164.all;

-- NAND gate in VHDL

entity nand_gate is
port (
        A : in std_logic;
        B : in std_logic;
        Y : out std_logic;
);
end nand_gate;
```

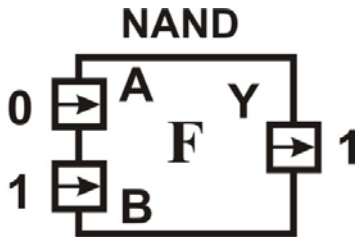
Urutan kode ini menspesifikasikan bahwa entitas **nand_gate** mempunyai dua *input* dan satu *output* yang semuanya bernilai **std_logic**. Deklarasi library IEEE memberitahu compiler VHDL mengenai pustaka yang digunakan pada *source code* ini. Sebagian besar VHDL baik itu simulator maupun *logic synthesizer*-nya mempunyai pustaka IEEE ini secara *built-in* atau dalam satu paket. Deklarasi **use** mempunyai fungsi yang sama dengan *statement* **#include** pada bahasa pemrograman ADA atatu C/C++ yakni memberitahu VHDL lokasi paket-paket penyedia fungsi-fungsi atau tipe-tipe tertentu yang diperlukan dalam pemrosesan *source code* menjadi *executable file*. Paket **std_logic_1164** mengandung *statement* yang berisi tipe *std_logic* yang digunakan di dalam rancangan di atas. Tipe ini mewakili sistem lo2gika sembilan-nilai (*nine-valued logic*). Sinyal-sinyal tipe ini dapat menggunakan salah satu dari sembilan nilai logika yang disediakan. Dalam contoh di atas rancangan NAND *gate* hanya menggunakan dua nilai saja yakni 0 (logika 0) dan 1 (logika 1). **entity** adalah tempat definisi *interface* rancangan dengan "dunia luar". Pendeklarasian input dan ouput dibangkitkan dilakukan di dalam port. Setiap pin menyatakan arah gerakan beserta jenis datanya – **A** dan **B** di-*assign* sebagai *input* (**in**) dengan tipe data **std_logic** dan dan **Y** di-*assign* sebagai *output* (**out**) juga dengan tipe **std_logic** untuk data hasil olahan dari *input*.

Implementasi suatu **entity** dilakukan pada badan **architecture** yang juga menjelaskan tingkah laku **entity** tersebut. Ada kemungkinan terdapat lebih dari satu badan **architecture** 2yang berkaitan dengan spesifikasi **entity** tunggal, masing-masing menjelaskan **entity** tersebut dari sudut pandang berbeda. Karena NAND *gate* adalah salah satu tipe rangkaian logika sederhana maka penjelasan tingkah lakunya dideklarasikan sebagai berikut :

```
architecture nand_gate_arch of nand_gate is

begin

        Y <= A nand B;

end nand_gate_arch;
```

Pada *architecture* di atas dijelaskan bahwa **Y** menerima hasil NAND data pada **A** dan **B**. Untuk rangkaian logika NAND *gate* dengan dua *input* terdapat empat kemungkinan *output* yakni :

| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

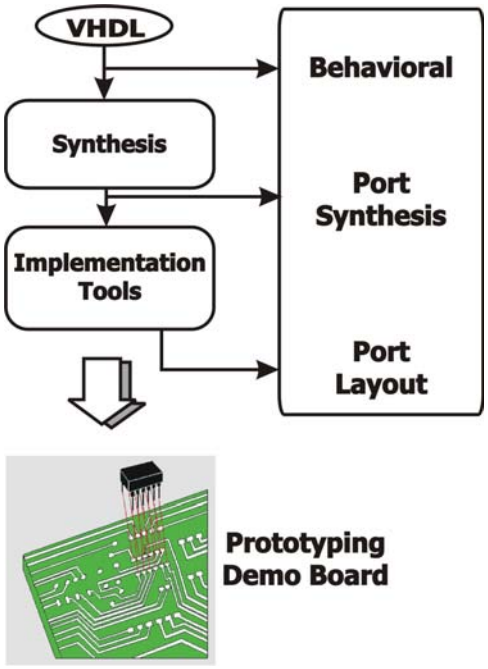



Gambar 2. NAND *gate*.

*Statement architecture* yang berada di antara deklarasi **begin** dan **end** ditangani dengan cara yang jauh berbeda dari bahasa pemrograman umum. Semua perintah yang berada di antara kedua statement tersebut dieksekusi secara bersamaan (*concurrently*) karena obyek yang dimodelkan adalah hardware dan mempunyai derajat paralelisme yang besar. Di sinilah faktor utama yang membedakan VHDL dengan bahasa pemrograman sekuensial yang banyak dipakai di *personal computer* (PC).

*Source code* ini selanjutnya dieksekusi atau dijalankan pada VHDL untuk diuji kelayakannya. Ada dua jenis eksekusi yakni Simulasi (*simulation*) dan Sintesis Logika (*logic synthesis*). Simulasi digunakan untuk melihat rancangan dari segi tingkah laku apakah sudah sesuai dengan yang

diharapkan dan ditunjukkan dalam bentuk gelombang (*waveform*) di layar simulator VHDL. Bila di dalam simulasi ini ditemukan kejanggalan, maka *source code* harus dicek ulang dan kesalahan diperbaiki sebelum disimulasikan kembali. Dalam pelaksanaan simulasi nilai *input* dapat diubah-ubah untuk melihat *output*-nya. Bila sudah benar, baru dilanjutkan ke tahap berikutnya yakni sintesa. Dalam proses sintesa dilakukan pengujian kelayakan rancangan untuk diimplementasikan ke bentuk *hardware*-nya yakni IC. Banyak ditemui kasus rancangan yang lulus simulasi namun tidak lulus sintesa. Artinya secara tingkah laku atau fungsional suatu rancangan dapat bekerja dengan sempurna tetapi secara *hardware* rancangan tersebut tidak dapat diimplementasikan karena dalam implementasi *hardware* ada aturan-aturan yang harus dipatuhi seperti *routing* atau jalur penghubung antara satu *gate* dengan *gate* lainnya di dalam IC, *routing* yang tidak efisien akan menambah biaya produksi. Bila rancangan telah lulus sintesa, ia harus diuji lagi dalam simulator VHDL untuk memeriksa ketepatan proses dengan pewaktuan (*timing*) yang diberikan dan fungsionalitas. Setelah lulus prosedur perancangan, selanjutnya rancangan dibawa ke *clean room* untuk dibuat *prototype*-nya dan ditawarkan ke publik untuk diuji coba pada rancangan sistem elektronika mereka.




Gambar 4. *Design flow* IC menggunakan VHDL.

Beberapa keuntungan penggunaan VHDL dalam *softening of hardware design* ini dirangkum sebagai berikut :

- Simulasi rancangan terlebih dulu sebelum *hardware* dibuat sehingga memudahkan pengelolaan rancangan yang kompleks.

- Memungkinkan alternatif rancangan yang beragam.

- Umpan balik hasil tes rancangan sehingga rancangan berikutnya akan lebih baik.

- Sintesa otomatis sehingga memudahkan *hardware layout* dalam proses pembuatannya.

- Meningkatkan produktivitas dengan mempersingkat *time-to-market*.

- Data rancangan yang *portable*.

**Bobot Teknologi Padat Materiil**

TNI AU adalah salah satu pengguna sistem dan alutsista udara berbasis teknologi tinggi khususnya teknologi elektronika seperti RADAR, sistem avionik pada pesawat terbang, sistem komputer pada jaringan informasi eksekutif dan manajemen serta pada peralatan pemeliharaan alutsista udara. Sudah banyak pengalaman hanya karena sulitnya mendapatkan komponen IC menyebabkan alutsistaud atau peralatan tersebut harus "istirahat" atau malah di-"pensiun dini" karena suku cadangnya sudah tidak diproduksi lagi (*out of production*). Kesulitan ini timbul karena komponen yang digunakan untuk membuat sistem dan peralatan elektronika standar militer (MIL-STD) mempunyai karakteristik khusus dan tidak ada di pasaran umum. DI samping karena sifatnya untuk aplikasi militer, komponen-komponen ini dirancang khusus untuk digunakan pada peralatan tertentu dalam bentuk **Applied Specific Integrated Circuits (ASICs)** sehingga harga per satuannya sangat mahal. Beberapa karakteristik IC aplikasi militer adalah :

- *Temperature range* sangat tinggi hingga di atas $100^0$C sehingga tetap dapat bekerja dengan normal pada suhu yang ekstrim sekalipun.

- Tidak cepat panas sehingga peralatan dapat dibuat sekecil dan sekompak mungkin.

- Dimensi minimal dengan fungsi optimal.

- Mean Time Between Failure (MTBF) tinggi. MTBF adalah interval waktu rata-rata yang biasanya diekspresikan dalam ribuan atau puluhan ribu jam (kadang disebut dengan *power-on hours* atau *POH*), yang berjalan sebelum suatu komponen *hardware* rusak atau memerlukan perbaikan misal : 100.000 jam baru dilakukan penggantian.

- Mengikuti trend teknologi militer.

Menyongsong rencana pembelian beberapa pesawat tempur Sukhoi-27 dan Sukhoi-30 dari Rusia oleh Pemerintah RI untuk memperkuat kekuatan udara TNI AU ditambah dengan adanya rencana untuk menjadikan Indonesia sebagai negara pemasok suku cadang peralatan tempur dari Rusia untuk kawasan Asia, pemahaman dan pendalaman teknologi IC ini akan sangat membantu. Dengan adanya **Airforce Industry** ini diharapkan kontribusi TNI AU dalam bentuk perbantuan tenaga ahli maupun sumber daya lainnya akan mengangkat TNI AU dari sekedar **end-user** menjadi **valuable contributing factor** yang diperhitungkan di dalam industri tersebut. Dengan menyimak kembali pengalaman pembelian peralatan tempur dari AS yang berakhir dengan embargo sejak tahun 1999, *the worst case* tetap harus dipegang siapa tahu industri penerbangan Rusia tiba-tiba runtuh karena suatu hal. TNI AU yang paling berkompeten di dalam Airforce Industry ini harus menyiapkan antisipasi dengan menyiapkan sumber daya yang ada sehingga dukungan suku cadang khususnya yang berkaitan dengan sistem elektronika pesawat terbang tetap dapat dipelihara. IC memang kecil (warnanya hitam lagi !) tetapi justru yang kecil ini dapat menjadi awal rontoknya kekuatan udara TNI AU. *It's better late to realize than never* …….

**DAFTAR PUSTAKA**

*A Designer's Guide to VHDL*, [Online], http://www.doulos.com/fi/desguidevhdl/desguidevhdl.html, download tanggal 25 Juni 2002.

Ashenden, Peter J., *The VHDL Cookbook*, 1990, 1st Edition, Dept. Computer Science University of Adelaide, South Australia.

Microsoft Corp., *Microsoft Press Computer Dictionary, 3rd Edition on CD* [CD], 1997, Microsoft Corp., USA.

Oxford University Press, *The Pocket Oxford Dictionary* [CD], 1994, Oxford University Press, UK.

Padeffke, Martin, *Multimedia-Einsatz bei der Lelure von VHDL*, [Online], Universität Erlangen-Nümberg, Lehrstuhlfür, Germany, http://www.vhdl-online.de/vhdl_online_cbt.pdf, download tanggal 12 Desember 2002.

Reese, Bob, *Logic Synthesis with VHDL Combination Logic*, 1995, Electrical Engineering Dept., Mississippi State University, USA.

Sumari, Lettu Lek Arwin D.W., *Perancangan Sistem Elektronika (EL-419)* [unpub], 1995, Bandung.

Tomson, Phil, *VHDL for Hardware Design*, 1996, Embedded Systems, Dr. Dobb's Journal, Juni, USA.