RANCANG BANGUN PERANGKAT LUNAK ELECTRONIC DESIGN AUTOMATION UNTUK INDUSTRI SEMIKONDUKTOR INDONESIA

Heni Rachmawati [1], **Edmond Febrinicko Armay** [2]
[1] Politeknik Caltex Riau, Pekan Baru, Indonesia
Email: henni@pcr.ac.id
[2] Universitas Islam Negeri Sultan Syarif Kasim Riau, Pekan Baru, Indonesia
Email: nicko@uin-suska.ac.id

Abstract
Recently, designing of electronic circuits becomes more complex, so the requirements for Electronic Design Automation (EDA) software are also getting bigger. Current software only capable to design an electronic circuit on a small scale, and do not accommodate to the new synthesis and analysis equipment ones. In addition, available software also does not utilize yet electronic component layout geometry and connectivity. To meet these needs, this research creates software using integration and database approaches. Integration approach is used because it has advantages in terms of ease, collection of Input/output (I/O) handlers can be used, flexible, and functional, thus user interaction becomes more uniform and stronger. Database approach, particularly object-oriented database, has advantages in terms of unlimited objects and user can add attributes on any object. Using both approaches, the software shows the tools can interact each other, changing in control can very easily managed, and users can build constraint systems, thus allowing database manipulation.

Keywords: *Electronic Design Automation, Object Oriented Database*

Rancang Bangun Perangkat Lunak Electronic Design Automation
Heni Rachmawati, Edmond Febrinicko Armay
Jurnal Teknosains Kodepena
pp. 8-19

Abstrak
Saat ini desain rangkaian elektronika semakin kompleks, sehingga kebutuhan perangkat lunak *Electronic Design Automation* (EDA) juga semakin besar. Perangkat lunak yang ada hanya mampu mendesain rangkaian elektronika dalam skala kecil, dan belum mengakomodasi peralatan sintesis dan analisis yang baru. Selain itu, perangkat lunak yang tersedia juga belum memanfaatkan geometri dan konektivitas *layout* komponen elektronika. Untuk memenuhi kebutuhan-kebutuhan tersebut di atas, telah dibuat suatu perangkat lunak yang menggunakan pendekatan integrasi dan basis data. Pendekatan integrasi digunakan karena memiliki keunggulan dalam hal kemudahan, kumpulan *Input/Output* (I/O) handlers dapat digunakan, fleksibel, dan fungsional, sehingga interaksi pengguna menjadi lebih seragam dan kuat. Pendekatan basis data, khususnya basis data berorientasi objek, memiliki keunggulan dalam hal banyaknya objek yang tak terbatas dan dapat menambah atribut pada objek apapun. Menggunakan kedua pendekatan di atas, perangkat lunak memperlihatkan peralatan-peralatannya dapat berinteraksi satu sama lain, kontrol perubahan sangat mudah dikelola, dan pengguna dapat membangun sistem kendala, sehingga memungkinkan manipulasi basis data.

Kata kunci: *Electronic Design Automation*, basis data berorientasi objek

1. PENDAHULUAN

Ketika rangkaian elektronika menjadi semakin rumit, maka permintaan sistem *Electronic Design Automation* (EDA) juga semakin meningkat. Tidak hanya kebutuhan untuk meningkatkan kapasitas desain yang lebih besar, sistem elektronika harus turut berkembang mengakomodasi banyaknya peralatan sintesis dan analisis yang baru. Peralatan sintesis semakin berkembang karena teknik analisis yang baru dan metode pemecahan masalah yang lebih efektif. Kebutuhan untuk memotong biaya pabrikasi dan menghasilkan komponen elektronika yang lebih cepat, menuntut peralatan sintesis baru yang dapat menghapus banyaknya iterasi desain. Selain itu, kerapatan dan kelajuan *Integrated Circuits* (IC) modern sangat sensitif terhadap fitur layout sehingga peralatan sintesis yang efektif harus mampu memanfaatkan geometri komponen elektronika dan konektivitas antar komponen. Oleh karena itu, beberapa peralatan sintesis tradisional belum cukup memenuhi kebutuhan di atas dan harus ditulis ulang.

Perangkat lunak Magic VLSI (*Very-large-scale integration*) menawarkan penggunaan geometri "*corner-stitched*", yaitu seluruh *layout* digambarkan sebagai tumpukan bidang, dan masing-masing bidang terdiri dari "*tile*" (persegi panjang). Penggambaran *corner-stitched* sangat cocok untuk pencarian di dalam bidang tunggal, sehingga corner-stitched unggul dalam waktu eksekusi [1]. *Corner-stitched* sangat tidak sesuai dengan basis data yang sangat besar: pengelolaan empat *pointer* untuk masing-masing *tile*, dan penyimpanan *tile*, membuatnya lebih banyak memakan memori daripada representasi berbasis objek.

Perangkat lunak LASI (*Layout System for Individuals*) menawarkan penggambaran layout dari sel hirarkis yang disimpan sebagai *file* tunggal TLC (*Transportable Layout Cell*), yang mana dapat diubah menjadi *file* GDS (*Graphic Database System*) atau *file* CIF (*Cal-tech Intermediate Format*). Walaupun dapat dijalankan secara *portable* [2], LASI tidak dapat menjalankan simulasi SPICE.

Perangkat lunak Microwind menawarkan desain dan simulasi rangkaian mikroelektronika pada level *layout*. Walaupun dilengkapi dengan fasilitas edit gambar, simulator analog SPICE, pemandangan atraktif seperti karakteristik MOS (*metal-oxide-semiconductor*), dan tampang lintang dua dimensi [3], fasilitas pemandangan *layout* secara tiga dimensi tidak tersedia pada versi gratis (*lite version*).

Memperhatikan kelemahan-kelemahan di atas, dan disamping menangani banyak fasilitas, sistem EDA harus mengakomodasi lingkungan desain yang berbeda-beda. Sistem EDA tidak hanya menangani teknologi IC yang beragam (MOS, bipolar, galium arsenida, dan lainnya) tapi sistem EDA juga harus beradaptasi terhadap perubahan aturan desain. Sistem EDA harus mendukung spesifikasi desain level yang lebih tinggi, seperti skematik, dan bahasa deskripsi perangkat keras, seperti VHDL. Masalah lain sistem EDA adalah antarmuka pengguna. Idealnya, antarmuka harus menjangkau seluruh fasilitas sistem sehingga sekumpulan perintah (*command*) perlu dipelajari. Antarmuka juga harus fleksibel untuk memenuhi kebutuhan kelompok dan individu. Antarmuka utama seharusnya memiliki bahasa pemrograman tertentu seperti Lisp atau Prolog.

Untuk memperkaya interaksi pengguna, sistem EDA dapat menyediakan kendala yang ditentukan oleh pengguna. Contoh dari kendala yang berguna adalah aturan geometrik dan persyaratan konektivitas. *Solver* kendala yang ideal dapat menerapkan aturan tersebut pada saat desain.

Fitur penting lainnya dari sistem EDA adalah bebas platform. Mengingat perkembangan *workstation*, tidak masuk akal meng-*install* sistem EDA pada satu sistem operasi tertentu. Sebaliknya, sistem EDA harus menyediakan kemudahan portabilitas sehingga sistem dapat berjalan secara mudah.

2. METODOLOGI

Langkah-langkah dalam penelitian ini adalah mendata kebutuhan sistem dalam bentuk analisis kebutuhan, kemudian membangun rancangan fungsional sistem dan rancangan objek sistem. Langkah selanjutnya adalah mengimplementasikan rancangan dalam bentuk *mimic router* dan simulasi SPICE.

2.1. Rumusan Masalah dan Tujuan

Rumusan masalah dari penelitian ini adalah bagaimana memanfaatkan geometri dan konektivitas *layout* dalam desain komponen elektronika semikonduktor. Mengakomodasi lingkungan desain yang berbeda terhadap teknologi IC dan perubahan aturan dalam desain komponen elektronika semikonduktor. Menciptakan antarmuka pengguna yang menjangkau seluruh fasilitas desain komponen elektronika semikonduktor. Menyediakan kemudahan portabilitas bagi *engineer* dalam mendesain komponen elektronika semikonduktor.

2.2. Integrasi

Usaha awal pada integrasi peralatan EDA adalah menggabungkan dua paket yang berbeda. Jika editor *layout* mampu mengantarmuka ke simulator, maka itu bisa disebut terintegrasi. Akhirnya, sistem ini berkembang menjadi urutan program kompleks untuk konversi *file*, sehingga setiap fasilitas dapat dipadukan. Teknik ini menentukan bentuk yang paling dasar dari integrasi – pertukaran *file* data. Hal ini untuk menambah integrasi seperti itu pada sistem EDA yang ada, tetapi umpan-balik *error* yang efektif, atau koreksi, sulit untuk diatur. Dalam Ulysses, "bahasa *scripts*" kompleks mengatur interaksi peralatan EDA [4][19].

Bentuk yang lebih maju dari integrasi *file* data tercapai ketika tipe *file* tunggal dimandatkan, misalnya sistem OCT [5], atau menggunakan *Electronic Design Interchange Format* (EDIF) [6]. Semua peralatan bekerja dari titik ini, dan interaksi pengguna sangat sederhana. Masing-masing peralatan harus ditulis ulang untuk menangani tipe *file* yang baru, dan format itu harus dipilih untuk menangani semua kebutuhan. Tetapi kelebihannya adalah kumpulan *Input/Output* (I/O) *handlers* dapat digunakan oleh masing-masing peralatan, sehingga *task* pemrograman menjadi lebih mudah.

Integrasi paling ketat, digunakan oleh banyak sistem [7][8][9][18], terjadi ketika seluruh data dalam memori. Peralatan kemudian menetap dalam program tunggal, atau berjalan dalam lingkungan *multitasking* dihubungkan dengan komunikasi antar-proses. Ini membutuhkan pengkodean ulang peralatan dan pemilihan basis data, tetapi peralatan menulis yang paling sederhana, karena fasilitas tersedia untuk manipulasi basis data, I/O, antarmuka pengguna, dan lain-lain. Faktanya, hanya kernel komputasi masing-masing peralatan yang perlu ditulis. Selain menyederhanakan peralatan, penempatan seluruh peralatan dalam memori memungkinkan interaksi satu sama lain menjadi lebih efektif. Masing-masing

peralatan dapat menyediakan fasilitas yang relevan, dan kebijakan penggunaan dapat diimplementasikan. Bentuk integrasi ini menyediakan fleksibilitas terbesar [10][17], dan hirarki fungsionalitas peralatan dapat diimplementasikan [11][16].

2.3. Basis Data

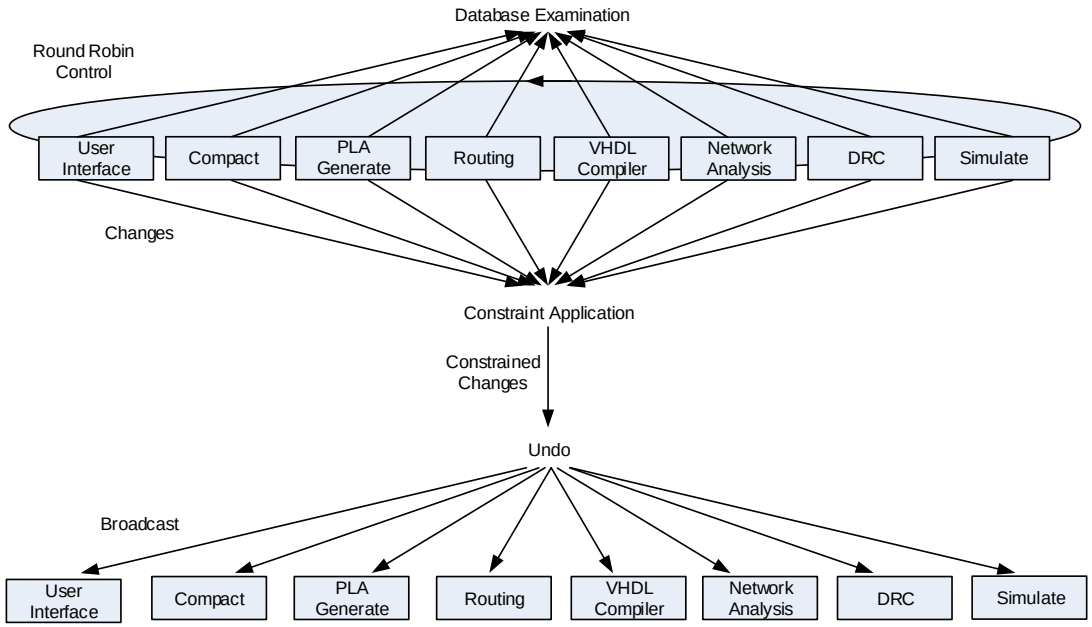
Mengingat bahwa integrasi yang efektif membutuhkan basis data, maka perlu menyediakannya dengan yang paling kuat. Tidak ada pembatasan pada ukuran desain dan harus mudah menambah informasi tambahan sesuai kebutuhan. Ini berarti bahwa basis data berorientasi objek adalah yang terbaik yang memungkinkan banyaknya objek tak terbatas dan dapat menambah atribut pada objek apapun. Model objek/atribut ini membentuk dasar basis data. Model ini memiliki struktur bertingkat tiga lapis yang terdiri dari *design entities*, *circuitry*, dan *design environments* [12].

Level tertinggi basis data adalah *design entities*. *Design entities* adalah koleksi *circuitry* yang berkumpul membentuk keseluruhan artefak. Biasanya, *design entity* disebut "sel," dan sel tunggal diedit sekaligus. Akan tetapi, basis data mengikuti konsep OCT dan memanggil masing-masing *design entity* suatu *facet* . Penggunaan *facet* didalam *facet* memberikan kompleksitas untuk rangkaian yang besar. Dibalik hirarki dasar, sangat berguna mengidentifikasi masing-masing *facet* sebagai bagian tampilan yang berbeda, sehingga beberapa gaya desain dapat dilakukan. Tipe tampilan mengandung *layout* dan skematik, isi dan tubuh dokumentasi, simulasi, dan lain-lain. Berhubung desainer sering membuat spesifikasi rangkaian abstrak (skematik) kemudian melakukan *layout* aktual, ketersediaan tipe tampilan yang berbeda membolehkan sistem EDA menelusuri perkembangan desain yang lebih baik [12].

Fitur lain *facet* yang diperlukan adalah nomor versi. Dengan mengenal sejarah aktivitas desain, pekerjaan dapat dipartisi antara beberapa desainer, dan evolusi desain dapat ditelusuri. Masing-masing *facet* memiliki kreasi dan modifikasi informasi tanggal [12]. Aspek terakhir *facet* adalah cara digabungkan. Pustaka harus ada untuk mengelompokkan *facet* . Dalam basis data, masing-masing pustaka adalah *disk file* tunggal yang mengandung koleksi *facet* , hirarki lengkap, atau kombinasi *facet* yang lain [12].

Level pertengahan organisasi basis data adalah *circuitry*. Sistem skematik menggambarkan *circuitry* sebagai jaringan abstrak komponen dan kabel, sedangkan sistem *layout* IC sering menggambarkan *circuitry* sebagai geometri di atas lapisan. Agar kuat dan fleksibel, basis data tidak hanya menangani geometri dan konektivitas, tetapi juga aturan. Oleh karena itu, *circuitry* terdiri dari komponen dan kabel, dengan notasi aturan penempatan, dan ukuran. Masing-masing *facet* mengandung beberapa informasi *circuitry* – jaringan, pengklasifikasi geometris (*R-Trees*) [13][14], dan atribut aturan seperti *directionality* dan *delay*.

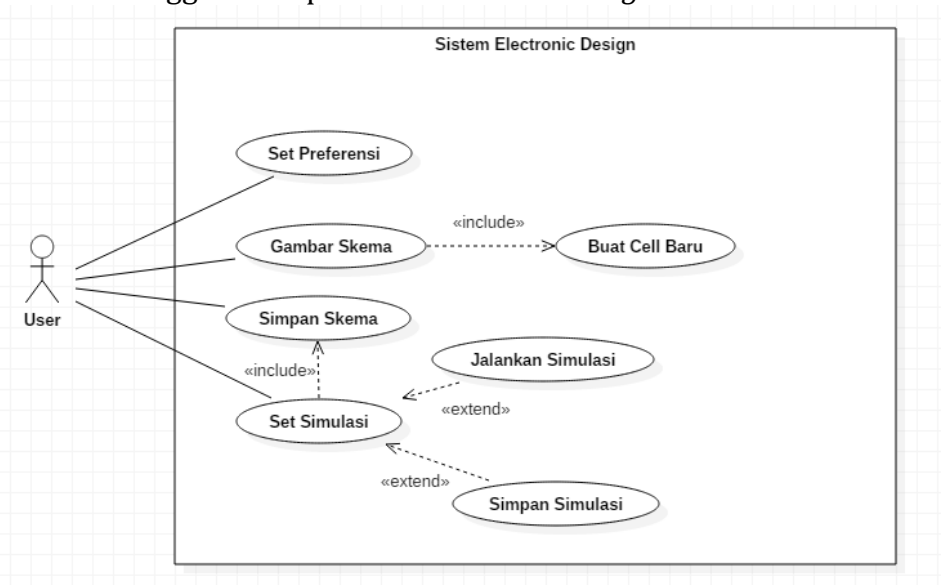
Level terendah basis data adalah *design environment* dimana *circuitry* didefinisikan. Daripada menyimpan seluruh informasi pada level *circuitry*, efisiensi menyatakan bahwa atribut umum dijelaskan secara terpisah. Oleh karena itu, bentuk, warna, aturan desain, aturan konektivitas, dan informasi lain dikumpulkan menjadi *design environments*. Selain menghemat ruang, penempatan informasi dalam *design environment* memungkinkan standar yang dapat diganti dalam *circuitry*. Sebagai contoh, aturan simulasi transistor dapat ditemukan dalam *design environment*, atau diganti jika transistor memiliki aturan tertentu. Berhubung keseragaman basis data, dimungkinkan menempatkan aturan pada *facet* , sehingga mengesampingkan informasi pada seluruh *circuitry* didalamnya [12][15].



Gambar 1. Interaksi peralatan yang diusulkan dengan basis data

2.4. Rancangan Sistem

Rancangan sistem dibagi dalam dua bagian utama yaitu rancangan objek sistem (basis data objek), dan rancangan fungsional; menggambarkan interaksi *user* dengan sistem menggunakan pemodelan *use case diagram*.



Gambar 2. Use Case Diagram

2.4.1. Rancangan Objek Sistem

Perangkat lunak EDA yang dibuat mengandung beberapa fasilitas. *Tool dispatcher* adalah fasilitas terpadu yang canggih. *Dispatcher* adalah sistem operasi berbentuk *kernel* yang mendistribusikan kontrol ke peralatan dan menjaganya terhadap perubahan yang dibuat oleh pihak lain. *Dispatcher* bekerja dengan baik untuk peralatan *incremental* dan *batch*, peralatan sintesis dan analisis, dan memungkinkan antarmuka pengguna yang bertindak sebagai peralatan, sehingga

menyatukan seluruh fungsi non basis data dalam sistem EDA. Struktur *dispatcher* menangani distribusi multiprosesor dari aktivitas peralatan.

Jantung *tool dispatcher* adalah *manajer round-robin* yang memberikan masing-masing peralatan suatu giliran. Hanya peralatan yang hidup mendapat giliran. Oleh karena itu, peralatan *batch* seperti *generator Programmable Logic Array* (PLA) dimatikan dan tidak menerima giliran. Pengguna juga dapat mematikan peralatan *incremental* seperti pemeriksa aturan desain. Ketika peralatan dihidupkan, maka diinformasikan ke seluruh *facet* yang telah berubah ketika mati, sehingga dimungkinkan untuk menyusul.

Saat mendapat giliran, peralatan dapat melakukan tugas apapun secepat mungkin. Pemeriksa aturan desain menggunakan gilirannya untuk memeriksa seluruh perubahan yang dibuat. Antarmuka pengguna menunggu perintah tunggal dan mengeksekusinya. Ini adalah waktu satu-satunya yang perubahan dapat dibuat ke basis data.

Perubahan basis data adalah penciptaan / penghapusan objek atau penciptaan/penghapusan/modifikasi atribut. Sebagai tambahan, kelompok perubahan pada objek diberi tanda kurung dengan tanda awal/akhir sehingga beberapa perubahan pada objek dapat dikoordinasikan dengan baik. Seluruh perubahan dibuat melalui antarmuka prosedural untuk mencatat aktivitas dan mencegah peralatan dari kerusakan. Tabel 1 memperlihatkan perubahan yang diusulkan.

Tabel 1. Perubahan Basis Data yang Diusulkan

Operasi	Langkah
Ciptakan transistor	Ciptakan objek transistor Akhiri perubahan objek transistor
Hapus kabel	Mulai perubahan objek kabel Hapus objek kabel
Ubah ukuran kontak	Mulai perubahan objek kontak Modifikasi atribut ukuran pada objek kontak Akhiri perubahan objek kontak

Ketika giliran berakhir, basis data akan menyiarkan seluruh perubahan yang dibuat oleh alat. Aktivitas ini dibuat ke seluruh peralatan yang hidup. Banyak peralatan tidak tertarik pada perubahan basis data, sehingga informasi dapat diabaikan. Untuk beberapa alat, hal ini sangat berguna. Pemeriksa aturan desain mengantri seluruh perubahan sehingga dapat memeriksa dan bereaksi selama gilirannya. Antarmuka pengguna memperbaharui tampilan ketika menerima seluruh perubahan. Hal ini dimungkinkan untuk menjaga tampilan dari sumber perubahan.

Perangkat lunak yang dibuat memiliki beberapa kelebihan. Pertama, masing-masing peralatan dapat berinteraksi satu sama lain. Kedua, jika melewatkan seluruh perubahan melalui kanal tunggal, maka kontrol perubahan sangat mudah dikelola. Ketiga, dimungkinkan untuk membangun sistem kendala, sehingga memungkinkan manipulasi basis data. Gambar 1 memperlihatkan seluruh proses yang diusulkan.

Dalam perangkat lunak ini, sistem kendala yang pertama menerima perubahan yang dibuat oleh alat. Masing-masing tipe perubahan (penciptaan / penghapusan objek dan penciptaan/penghapusan/modifikasi atribut) melewati sistem kendala, kemudian dapat menghasilkan perubahan lain. Pada akhir giliran,

solver kendala memiliki kesempatan akhir bereaksi pada perubahan, atau menundanya. Sistem kendala baru mudah di-install karena itu adalah sekumpulan rutin untuk masing-masing tipe perubahan.

Masuknya antarmuka pengguna dalam daftar peralatan yang aktif berarti bahwa pengguna selalu berada pada satu titik tertentu dalam urutan pengiriman. Setelah memasukkan perintah, peralatan beroperasi dalam urutan yang tetap. Oleh karena itu, peralatan sintesis mengikuti antarmuka pengguna dan mendahului peralatan analisis. Hal ini memungkinkan seluruh aksi generatif, seperti *compaction* dan *routing*, terjadi sebelum analisis akhir, seperti pemeriksaan aturan desain dan kelistrikan.

Perangkat lunak ini bekerja sangat baik untuk peralatan *incremental*. Peralatan *batch* seperti generator cocok dengan skema ini saat dimatikan. Ketika dihidupkan, generator menyelesaikan fungsinya dan mati sendiri. Oleh karena itu, perangkat lunak ini menyediakan kerangka serba guna untuk seluruh peralatan EDA.

2.4.2. Rancangan Fungsionalitas Sistem

Perangkat lunak ini memiliki beberapa fungsi yang bisa digunakan oleh *user*. Pemodelan fungsionalitas sistem menggunakan *use case diagram*. *Use case diagram* menampilkan interaksi *actor*, dalam hal ini adalah *user* dengan fungsi-fungsi didalam sistem. *Use case* utama memiliki beberapa ketergantungan dengan *use case* lainnya (*include*), dan *use case* utama memiliki beberapa fungsi tambahan pada *use case* lainnya (*extend*).

3. HASIL DAN PEMBAHASAN

Perangkat lunak berbasis desktop ini ditulis menggunakan instrumen bahasa pemrograman C++ dalam Microsoft Visual Studio 2015 sebagai *Integrated Development Environment*. Sumber data diambil dari ON Semiconductor, sedangkan datanya masih dalam bentuk data mentah yang berisi tentang aturan lambda, dan karakteristik listrik dari komponen-komponen elektronika semikonduktor. Data-data mentah tersebut kemudian dikelompokkan menjadi tiga kelompok besar, yaitu *design entities*, *circuitry*, dan *design environment*. Setelah dikelompokkan, semua data di-*parsing* ke lingkungan pemrograman C++. Disana kode-kode program ditulis untuk memanggil semua data yang telah dikelompokkan. Selain itu, juga dibuat *Graphical User Interface* yang memudahkan pengguna dalam mendesain komponen elektronika semikonduktor.

3.1. Mimic Router

electriC++ memiliki router yang disebut "*mimic-router*" yang memperkuat aksi wiring pengguna dengan cara menduplikasinya pada tempat lain yang sejenis dalam rangkaian. Hal ini sangat berguna dalam desain berbasis *array*, ketika pengguna hanya menghubungkan elemen tunggal dalam *array* dan router melakukan pekerjaan sisanya. Peralatan ini diaktifkan dengan cara menghidupkannya, kemudian menerima *broadcast* dari seluruh perubahan yang dibuat ke basis data.

Mimic-router mencari perubahan basis data yang menambah kabel-kabel. Perubahan ini harus berasal dari antarmuka pengguna. Perubahan *wiring* tidak harus menjadi bagian dari operasi yang lebih kompleks, seperti duplikasi dari koleksi rangkaian.

Selama hidupnya peralatan antarmuka pengguna, *command* pengguna

dapat dieksekusi. Jika pengguna membuat penambahan kabel, *mimic-router* melihat dan mengantrekan penambahan sampai gilirannya.

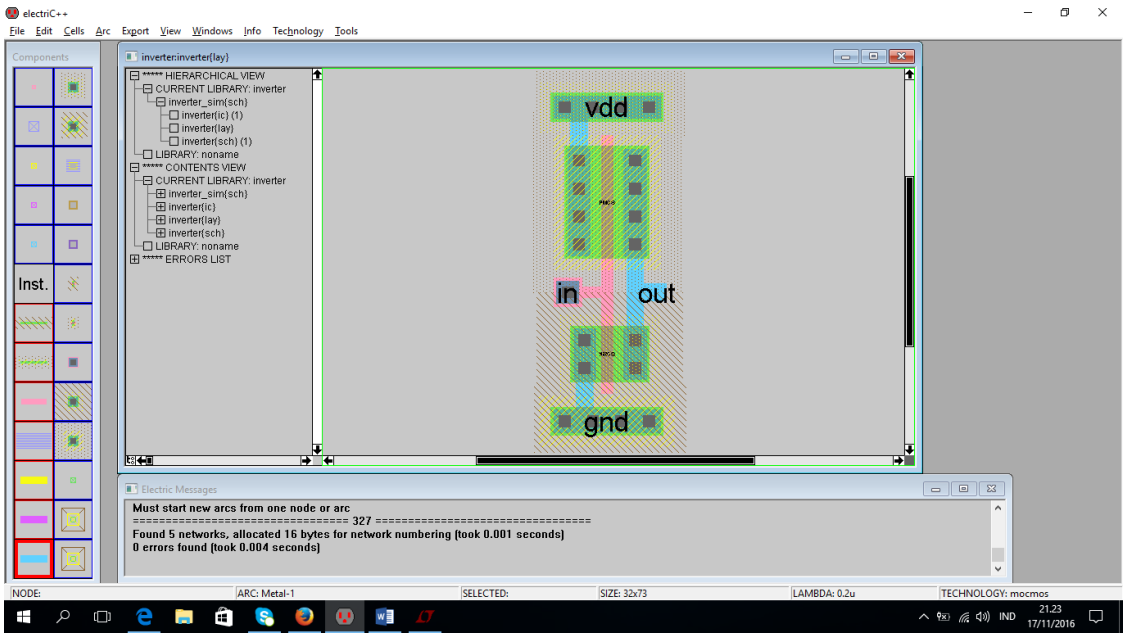
Selama hidupnya *mimic-router*, peralatan melihat pada masing-masing penambahan kabel dan mencari situasi yang mirip di tempat lain di *facet* . Situasi yang mirip didefinisikan yaitu memiliki *endpoint* dari jenis yang sama, terpisah oleh jarak yang sama, dan tidak terhubung. *Mimic-router* akan menambah kabel-kabel.

Ketika *mimic-router* selesai bekerja, seluruh perubahan di-*broadcast* ke seluruh peralatan, menyebabkan antarmuka pengguna menampilkan penambahan kabel. Penambahan kabel yang sederhana menghasilkan penampakan pada tampilan.

3.2. Simulasi SPICE

Simulasi SPICE adalah peralatan yang berbeda. Simulator berjalan pada proses yang terpisah dan tidak berinteraksi dengan pengguna. Seluruh stimulus *input* dan permintaan plot *output* harus dikodekan terlebih dahulu, bersama dengan informasi struktural.

Untuk menjalankan SPICE, pengguna menyajikan deskripsi grafis kebutuhan I/O menggunakan fasilitas Export pada komponen-komponen sumber tegangan, *ground*, *input* dan *output* dari teknologi skematis, misalnya *inverter layout*, seperti terlihat pada Gambar 3.



Gambar 3. Persiapan SPICE

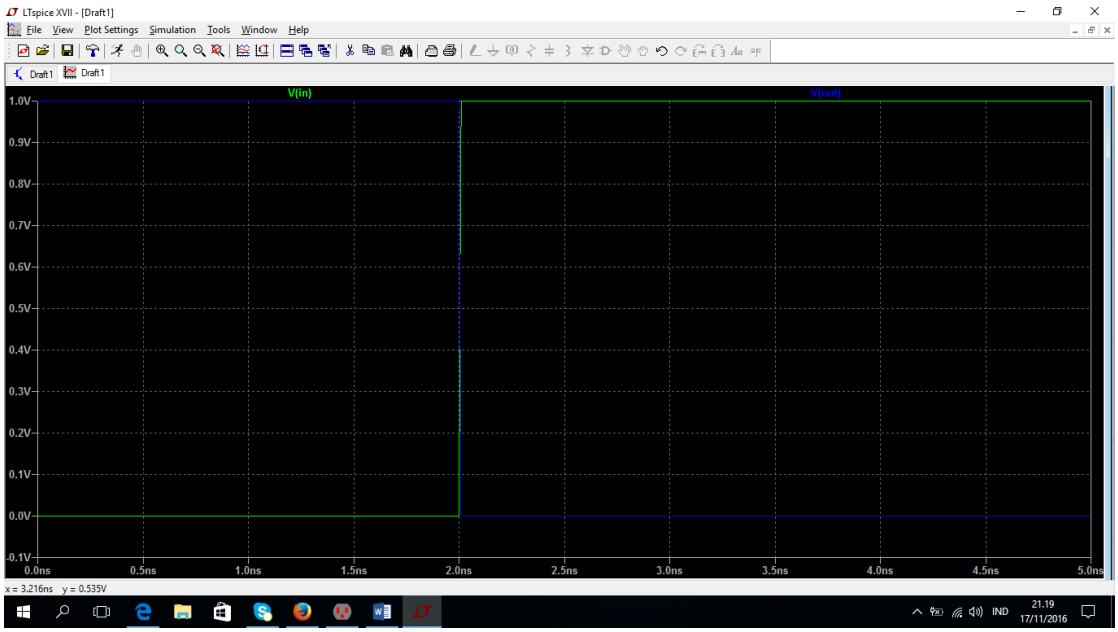
Kemudian pengguna dapat menghidupkan peralatan simulasi khusus untuk SPICE. Pada penelitian ini, simulator yang digunakan adalah perangkat lunak LTspice XVII. Pada *inverter layout* di atas, kode program SPICE yang digunakan adalah sebagai berikut:

```
.include C5.txt
.tran 0 5n 0 10p
PULSE(0 1 2n 10p)
vdd vdd 0 DC 1
```

Ketika peralatan simulasi SPICE bekerja, electriC++ menulis *facet* terbaru

dalam format yang sesuai untuk simulator SPICE. Disini electriC++ menciptakan *file* terbaru dengan format *.spi. Kemudian, SPICE meminta simulator sebagai sub-proses dan menunggunya sampai selesai. Selanjutnya, SPICE membaca daftar *output* simulator dan menciptakan grafik pada monitor komputer, seperti terlihat pada gambar 4.

Akhirnya, peralatan simulasi SPICE berhenti dengan sendirinya. Jadi, peralatan simulasi SPICE bekerja sebagai aktivitas tunggal selama gilirannya.



Gambar 4. Grafik SPICE

Pada simulasi *inverter layout* pada gambar 3, dipanggil *file* khusus yaitu C5.txt. *File* ini adalah kode program SPICE untuk proses C5 yang diciptakan oleh ON Semiconductor. Proses C5 adalah proses CMOS *non-silicided* yang memiliki 3 lapisan *metal*, 2 lapisan *polysilicon*, dan sebuah lapisan resistansi tinggi. Proses ini untuk aplikasi 5 volt. Kode program SPICE untuk proses C5 adalah sebagai berikut:

```
* BSIM3 models for AMI Semiconductor's C5 process
*
* Don't forget the .options scale=300nm if using drawn lengths
* and the MOSIS SUBM design rules
*
* 2<Ldrawn<500   10<Wdrawn<10000 Vdd=5V
* Note minimum L is 0.6 um while minimum W is 3 um
* Change to level=49 when using HSPICE or SmartSpice

.MODEL NMOS NMOS (                                LEVEL   = 8
+VERSION = 3.1            TNOM    = 27            TOX     = 1.39E-8
+XJ      = 1.5E-7         NCH     = 1.7E17        VTH0    = 0.6696061
+K1      = 0.8351612      K2      = -0.0839158    K3      = 23.1023856
+K3B     = -7.6841108     W0      = 1E-8          NLX     = 1E-9
+DVT0W   = 0              DVT1W   = 0             DVT2W   = 0
+DVT0    = 2.9047241      DVT1    = 0.4302695     DVT2    = -0.134857
+U0      = 458.439679     UA      = 1E-13         UB      = 1.485499E-18
+UC      = 1.629939E-11   VSAT    = 1.643993E5    A0      = 0.6103537
+AGS     = 0.1194608      B0      = 2.674756E-6   B1      = 5E-6
+KETA    = -2.640681E-3   A1      = 8.219585E-5   A2      = 0.3564792
+RDSW    = 1.387108E3     PRWG    = 0.0299916     PRWB    = 0.0363981
```

```
+WR      = 1              WINT    = 2.472348E-7    LINT    = 3.597605E-8
+XL      = 0              XW      = 0              DWG     = -1.287163E-8
+DWB     = 5.306586E-8    VOFF    = 0              NFACTOR = 0.8365585
+CIT     = 0              CDSC    = 2.4E-4         CDSCD   = 0
+CDSCB   = 0              ETA0    = 0.0246738      ETAB    = -1.406123E-3
+DSUB    = 0.2543458      PCLM    = 2.5945188      PDIBLC1 = -0.4282336
+PDIBLC2 = 2.311743E-3    PDIBLCB = -0.0272914     DROUT   = 0.7283566
+PSCBE1  = 5.598623E8     PSCBE2  = 5.461645E-5    PVAG    = 0
+DELTA   = 0.01           RSH     = 81.8           MOBMOD  = 1
+PRT     = 8.621          UTE     = -1             KT1     = -0.2501
+KT1L    = -2.58E-9       KT2     = 0              UA1     = 5.4E-10
+UB1     = -4.8E-19       UC1     = -7.5E-11       AT      = 1E5
+WL      = 0              WLN     = 1              WW      = 0
+WWN     = 1              WWL     = 0              LL      = 0
+LLN     = 1              LW      = 0              LWN     = 1
+LWL     = 0              CAPMOD  = 2              XPART   = 0.5
+CGDO    = 2E-10          CGSO    = 2E-10          CGBO    = 1E-9
+CJ      = 4.197772E-4    PB      = 0.99           MJ      = 0.4515044
+CJSW    = 3.242724E-10   PBSW    = 0.1            MJSW    = 0.1153991
+CJSWG   = 1.64E-10       PBSWG   = 0.1            MJSWG   = 0.1153991
+CF      = 0              PVTH0   = 0.0585501      PRDSW   = 133.285505
+PK2     = -0.0299638     WKETA   = -0.0248758     LKETA   = 1.173187E-3
+AF      = 1              KF      = 0)
*
.MODEL PMOS PMOS (                                LEVEL   = 8
+VERSION = 3.1            TNOM    = 27             TOX     = 1.39E-8
+XJ      = 1.5E-7         NCH     = 1.7E17         VTH0    = -0.9214347
+K1      = 0.5553722      K2      = 8.763328E-3    K3      = 6.3063558
+K3B     = -0.6487362     W0      = 1.280703E-8    NLX     = 2.593997E-8
+DVT0W   = 0              DVT1W   = 0              DVT2W   = 0
+DVT0    = 2.5131165      DVT1    = 0.5480536      DVT2    = -0.1186489
+U0      = 212.0166131    UA      = 2.807115E-9    UB      = 1E-21
+UC      = -5.82128E-11   VSAT    = 1.713601E5     A0      = 0.8430019
+AGS     = 0.1328608      B0      = 7.117912E-7    B1      = 5E-6
+KETA    = -3.674859E-3   A1      = 4.77502E-5     A2      = 0.3
+RDSW    = 2.837206E3     PRWG    = -0.0363908     PRWB    = -1.016722E-5
+WR      = 1              WINT    = 2.838038E-7    LINT    = 5.528807E-8
+XL      = 0              XW      = 0              DWG     = -1.606385E-8
+DWB     = 2.266386E-8    VOFF    = -0.0558512     NFACTOR = 0.9342488
+CIT     = 0              CDSC    = 2.4E-4         CDSCD   = 0
+CDSCB   = 0              ETA0    = 0.3251882      ETAB    = -0.0580325
+DSUB    = 1              PCLM    = 2.2409567      PDIBLC1 = 0.0411445
+PDIBLC2 = 3.355575E-3    PDIBLCB = -0.0551797     DROUT   = 0.2036901
+PSCBE1  = 6.44809E9      PSCBE2  = 6.300848E-10   PVAG    = 0
+DELTA   = 0.01           RSH     = 101.6          MOBMOD  = 1
+PRT     = 59.494         UTE     = -1             KT1     = -0.2942
+KT1L    = 1.68E-9        KT2     = 0              UA1     = 4.5E-9
+UB1     = -6.3E-18       UC1     = -1E-10         AT      = 1E3
+WL      = 0              WLN     = 1              WW      = 0
+WWN     = 1              WWL     = 0              LL      = 0
+LLN     = 1              LW      = 0              LWN     = 1
+LWL     = 0              CAPMOD  = 2              XPART   = 0.5
+CGDO    = 2.9E-10        CGSO    = 2.9E-10        CGBO    = 1E-9
+CJ      = 7.235528E-4    PB      = 0.9527355      MJ      = 0.4955293
+CJSW    = 2.692786E-10   PBSW    = 0.99           MJSW    = 0.2958392
+CJSWG   = 6.4E-11        PBSWG   = 0.99           MJSWG   = 0.2958392
+CF      = 0              PVTH0   = 5.98016E-3     PRDSW   = 14.8598424
+PK2     = 3.73981E-3     WKETA   = 5.292165E-3    LKETA   = -4.205905E-3
+AF      = 1              KF      = 0)
```

Jika eksekusi simulator memakan waktu, peralatan simulasi SPICE dapat diminta menyiapkan *input*, tetapi tidak menjalankan simulator. Pengguna dapat menjalankan simulasi secara manual, dan peralatan simulasi SPICE dapat diminta membaca daftar *output* pada waktu berikutnya.

Salah satu aspek menarik dari grafik SPICE adalah grafik itu sebenarnya *facet* baru dengan gambaran "*output* simulasi". Grafik itu terbentuk dari komponen grafis khusus dan menjadi bagian permanen basis data. Pengguna dapat mencetak, memperbesar area tertentu, bahkan memodifikasi data.

4. KESIMPULAN

Sistem CAD menjadi semakin kompleks untuk memenuhi permintaan desainer rangkaian. Untuk menjawab tantangan pada masa depan, sistem tersebut harus dapat diperluas dalam banyak cara. Selain mengakomodasi peralatan sintesis dan analisis yang baru, kemampuan integrasi harus mampu menangani lingkungan desain yang berkembang, batasan sistem, bahkan platform perangkat keras yang baru. electriC++ menyediakan semua fleksibilitas yang dimaksud. Sistem ini juga memiliki skema integrasi peralatan yang memungkinkan banyak peralatan berinteraksi secara efisien.

5. DAFTAR PUSTAKA

[1] J. Solomon dan M. Horowitz, "Using Texture Mapping with Mipmapping to Render a VLSI Layout," Proceedings of the 38th Design Automation Conference, pp. 500-505, 2001.

[2] B. Rivera, R. J. Baker, J. Melngailis, "Design and Layout of Schottky Diodes in a Standard CMOS Process," 2001 International Semiconductor Device Research Symposium, pp. 79-82, 2001.

[3] E. Sicard dan S. B. Dhia, "An Illustration of 90 nm CMOS Layout on PC," Proceedings of the Fifth IEEE International Caracas Conference on Devices, Circuits and Systems, pp. 315-318, 2004.

[4] M. Bushnell dan S. W. Director, "Automated Design Tool Execution in the Ulysses Design Environment," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 8, no. 3, pp. 279- 287, March 1989.

[5] D. S. Harrison, P. Moore, R. Spickelmier, A. R. Newton, "Data Management and Graphics Editing in the Berkeley Design Environment," Proceedings of the International Conference on Computer-Aided Design, pp. 24-27, 1986.

[6] W. Li dan H. Switzer, "A Unified Data Exchange Environment Based on EDIF," Proceedings of the 26th ACM/IEEE Design Automation Conference, pp. 803-806, June 1989.

[7] G. Chen dan T. Parng, "A Database Management System for a VLSI Design System," Proceedings of the 25th ACM/IEEE Design Automation Conference, pp. 257-262, June 1988.

[8] J. Soukup, "Organized C: A Unified Method of Handling Data in CAD Algorithms and Databases," Proceedings of the 27th ACM/IEEE Design Automation Conference, pp. 425-430, January 1991.

[9] M. Donlin, "CAD Framework Vendors Wrestle with Standards," Computer Design, vol. 29, issue 19, pp. 83-90, October 1990.

[10] W. D. Smith, D. Duff, M. Dragomirecky, J. Caldwell, M. Hartman, J. Jasica, M. A. d'Abreu, "FACE Core Environment: The Model and its Application in

CAE/CAD Tool Development," Proceedings of the 26th ACM/IEEE Design Automation Conference, pp. 466-471, June 1989.

[11] J. Daniell, dan S. W. Director, "An Object-Oriented Approach to CAD Tool Control within a Design Framework," Proceedings of the 26th ACM/IEEE Design Automation Conference, pp. 197-202, June 1989.

[12] S. M. Rubin, "An Integrated Aid for Top-Down Electrical Design," Proceedings of the International Conference on Computer-Aided Design (ICCAD-83), 1983.

[13] A. Guttman, "R-Trees: A Dynamic Index Structure for Spatial Searching," ACM SIGMOD, vol. 14, issue 2, pp. 47-57, June 1984.

[14] P. Nagarkar, A. Bhattacharya, O. Jafari, "Exploring State-of-the-Art Nearest Neighbor (NN) Search Techniques," CODS COMAD 2021: 8th ACM IKDD CODS and 26th COMAD, pp. 443-446, January 2021.

[15] J. Huang, M. Zhu, P. Gupta, S. Yang, S. M. Rubin, G. Garret, J. He, "A CAD Tool for Design and Analysis of CNFET Circuits," 2010 IEEE International Conference of Electron Devices and Solid-State Circuits (EDSSC), pp. 1-4, 2010.

[16] Y. Rezgui, S. Boddy, M. Wetherill, G. Cooper, "Past, present and future of information and knowledge sharing in the construction industry: Towards semantic service-based e-construction?" Computer-Aided Design, vol. 43, issue 5, pp. 502-515, May 2011.

[17] M. Bales, "Design Databases" in Electronic Design Automation for IC Implementation, Circuit Design, and Process Technology (2nd ed.), Chapter 15, CRC Press, 2016.

[18] Z. A. Alzamil, "Application of redundant computation in program debugging," Journal of Systems and Software, vol. 81, issue 11, pp. 2024-2033, November 2008.

[19] V. R. L. Shen, "A PN-based approach to the high-level synthesis of digital systems," Integration, vol. 39, issue 3, pp. 182-204, June 2006.