دانشگاه شهید بهشتی
دانشکده‌ی مهندسی برق و کامپیوتر
گروه طراحی خودکار مدارهای مجتمع پرتراکم

راهنمای کاربری ابزار EduCAD

(Version 9.5)

علی جهانیان

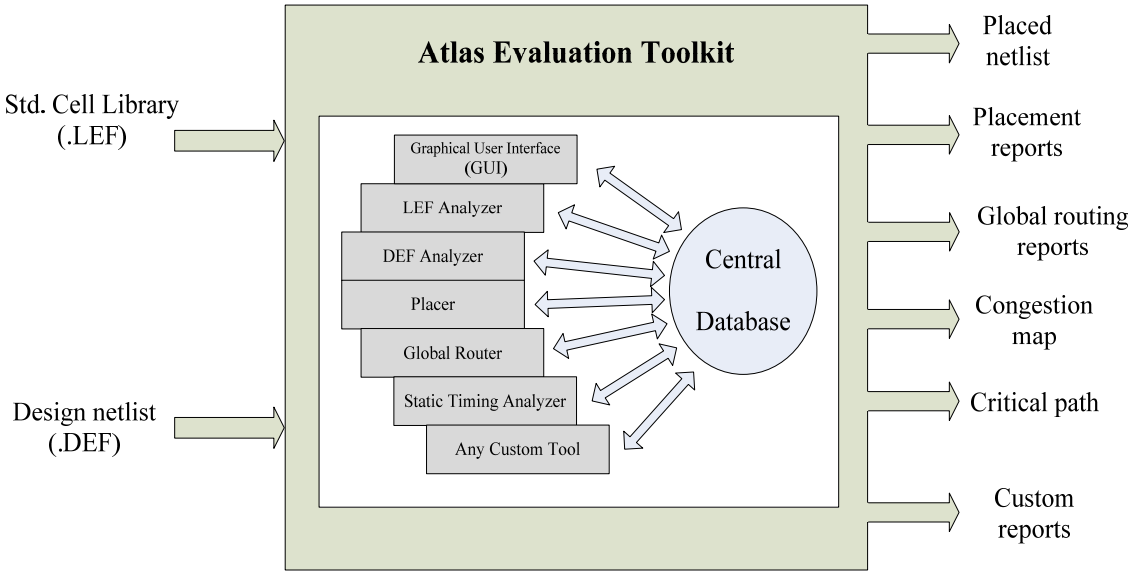
زمستان ۱۳۹۲

محیط طراحی و آزمون EDUCAD

محیط EduCAD که مخفف Educational CAD Tool می‌باشد، توسط گروه طراحی خودکار مدارهای مجتمع دانشگاه شهید بهشتی برای اهداف آموزشی-پژوهشی طراحی و پیاده‌سازی شده است. بانک داده مرکزی این مجموعه، از ابزار Atlas که توسط گروه طراحی خودکار دانشگاه صنعتی امیرکبیر طراحی و پیاده‌سازی شده، اقتباس گردیده است و بقیه بخش‌ها برای استفاده در کاربردهای آموزشی به آن اضافه گردیده‌اند.

۱. مجموعه ابزار EduCAD

محیط EduUCAD یک مجموعه کد به زبان C/C++ است که ساختار کلی و شمای ورودی و خروجی آن بصورت شکل ۱ است:



شکل ۱: ساختار کلی ابزار duCAD

چنانکه در شکل فوق دیده می‌شود، EduCAD شامل ابزارهای لازم جهت تحلیل فایل‌های ورودی و یک ابزار ساده برای چینش خطی سلول‌هاست. مهم‌ترین ویژگی این ابزار این است که بسادگی می‌تواند گسترش یابد و دانشجویان با هزینه‌ی زمانی کمی می‌توانند هسته‌های جدیدی به آن اضافه نمایند یا اجزای داخلی آن را تغییر دهند. در واقع این ابزار یک آزمایشگاه برای آزمودن الگوریتم‌های طراحی خودکار سخت‌افزار می‌باشد. این ابزار بصورتی ساده کدنویسی شده و با اغلب نسخه‌های Linux که مجهز به کتابخانه‌ی گرافیکی Qt باشند، سازگار است. لازم به ذکر است که Qt یک کتابخانه‌ی گرافیکی است که بیشتر نسخه‌های Linux آن را به‌صورت رایگان در بردارند. این ابزار با نسخه‌های 7.0 تا 12.0 سیستم عامل Fedora core آزمایش شده است.

ابزار EduCAD شامل یک بانک داده‌ی مرکزی بنام AtlasDB است که کلیه‌ی اطلاعات طرح و تکنولوژی در آن ذخیره می‌شود. بانک داده AtlasDB علاوه بر داده‌های ذخیره شده در آن، یک مجموعه توابع دسترسی (API) هم دارد که امکان دسترسی و بازیابی اطلاعات داخل بانک را فراهم می‌کند.

۲. نحوه‌ی نصب و اجرای ابزار EduCAD

چنان‌که گفته شد، کد این ابزار برای سهولت اجرا تا حد زیادی ساده شده است، لذا در اکثر نسخه‌های Linux قابل اجراست. البته باید سیستم عامل مجهز به کتابخانه Qt3 یا بالاتر باشد. برای راه‌اندازی این ابزار کافی است فایل educad.zip را روی یک شاخه‌ی دلخواه کپی کنید و آن را unzip کنید. برای unzip کردن می‌توانید از طریق منو یا از دستور زیر استفاده نمایید.

```
unzip -x educad.zip
```

در صورت unzip شدن صحیح برنامه، زیرشاخه‌های زیر داخل شاخه‌ی اصلی educad ایجاد خواهد شد.

```
bin, db, def, lef, gui, input, lib, main, placer
```

سپس دستور زیر را در شاخه‌ی EduCAD/bin اجرا نمایید.

```
./install
```

با اجرای این دستور پیغام‌های زیر ظاهر می‌گردد.

```
Compiling Placer..
OK
Compiling Global Routing ..
OK
Compiling GUI ..
OK
Compiling STA ..
OK
Compiling main ..
OK
Linking ..
OK
```

در صورتی که کامپایل برنامه با مشکلی مواجه نشود، در شاخه educad\bin فایل اجرایی برنامه با عنوان educad\bin ساخته می‌شود. برای اجرای برنامه دستور زیر را در شاخه educad\bin اجرا نمایید.

```
./educad ../input/xx.lef ../input/yy.def [-place/-load] zz.dat
```

که در آن، ورودی‌های xx.lef و yy.def به ترتیب نام فایل‌های فناوری و طراحی هستند که در شاخه‌ی input قرار داده شده‌اند. به همراه این مجموعه یک فایل فناوری (180nm Crete Library مربوط به شرکت Cadence) و چند فایل طراحی ارائه شده است که با فرمت صنعتی ارائه شده‌اند. این فایل‌ها از ۳۰۰۰ سلول تا حدود ۵۰۰۰۰ سلول دارند. پس از نام فایل‌های فناوری و طرح، مد اجرای ابزار جایابی به صورت زیر مشخص می‌شود:

- **مد اجرای جایابی:** اگر در خط فرمان، سوییچ place– انتخاب گردد، ابزار جایابی اجرا می‌شود و پس از آن نیز ابزار مسیریابی سراسری اجرا می‌شود که مدار جایابی شده را به عنوان ورودی در نظر گرفته و عمل مسیریابی سراسری را انجام می‌دهد. لیست موقعیت سلول‌ها پس از انجام جایابی نیز در فایل zz.dat ذخیره می‌گردد.
- **مد بارکردن جایابی:** اگر در خط فرمان، سوییچ load– انتخاب شود، دیگر ابزار جایابی اجرا نشده و جایابی از پیش انجام شده، از فایل zz.dat که یک جایابی از پیش ذخیره شده است، لود می‌شود. پس از آن نیز ابزار مسیریابی سراسری اجرا می‌شود که مدار خوانده شده را به عنوان ورودی در نظر گرفته و عمل مسیریابی سراسری را انجام می‌دهد. این حالت برای شرایطی در نظر گرفته شده که بخواهیم الگوریتم مسیریابی را روی یک مدار جایابی شده‌ی خاص چندین بار تکرار نماییم. برای این مد اجرا چند فایل از پیش جایابی شده در شاخه‌ی dump قرار داده شده است.

۳. ابزار Placer

این ابزار حاوی دو ابزار جایابی ساده است که به‌عنوان یک محیط برای توسعه‌ی ابزار جایابی طراحی شده است.

۳–۱. ساختمان داده ابزار جایابی

ساختمان داده این مجموعه ابزار در قالب کلاس simplePlacer در فایل placer.h تعریف شده است. فرم کلی این کلاس به‌صورت زیر است.

```
class SimplePlacer {
      int simPlacer();
      int readDB(atlasDB *);
      void updateDB(atlasDB *);
      double THPWL();

private:
      list <simpleInstance *> *mList;
      list <simpleNet *> *nList;
      float rowHeight;
      unsigned maxlayoutW;
      unsigned maxlayoutH;
      int rowNumber;
};
```

که متغیرهای آن عبارتند از:

متغیر rowHeight: ارتفاع یک سطر از سلول‌های استاندارد می‌باشد.

متغیر maxlayoutW: عرض کل چینش را نشان می‌دهد.

متغیر maxlayoutH: ارتفاع کل چینش را نشان می‌دهد.

متغیر rowNumber: تعداد سطرهای سلول استاندارد را نشان می‌دهد.

متغیر mList: یک لیست پیوندی که هر عنصر آن یک سلول از نوع داده‌ای simpleInstance می‌باشد.

متغیر nList: یک لیست پیوندی که هر عنصر آن یک سیم از نوع داده‌ای simpleNet می‌باشد.

لازم به ذکر است که سه مقدار maxlayoutW، maxlayoutH و rowNumber در طی انجام جایابی ممکن است تغییر کنند.

و توابع اصلی این کلاس عبارتند از:

تابع readDB(): این تابع، اطلاعات لازم برای جایابی را از بانک AtlasDB خوانده و در ساختار داخلی جایاب می‌ریزد.

تابع simPlacer(): این تابع، عمل جایابی خطی را انجام می‌دهد.

تابع updateDB(): این تابع، پس از انجام جایابی نتایج جایابی –یعنی محل سلول‌ها- را در بانک AtlasDB به‌روز می‌نماید.

تابع THPWL(): این تابع مجموع طول سیم‌های طرح را محاسبه می‌کند. اگر در مراحل میانی جایابی صدا زده شود، موقعیت سلول‌های جایابی شده را نیز در محاسبه‌ی طول سیم در نظر می‌گیرد.

مهم‌ترین اجزای داده‌ای در این کد عبارتند از:

ساختار simplePoint: این ساختار داده، یک زوج نقطه را نگهداری می‌کند.

ساختار simpleRect: این ساختار داده، یک مستطیل را نگهداری می‌کند.

ساختار simpleNet: این ساختار داده، اطلاعات مربوط به یک سیم را نگهداری می‌کند. فیلدهای این ساختار عبارتند از:

- فیلد netName: یک رشته‌ی کاراکتری حاوی نام سیم است.
- فیلد NetID: شماره‌ی سیم است.
- فیلد instance: لیست سلول‌هایی توسط این سیم به هم متصل شده‌اند.
- فیلد tag: یک بیت که برای پیاده‌سازی الگوریتم‌ها مورد استفاده قرار می‌گیرد

ساختار simpleInstance: این ساختار داده، اطلاعات مربوط به یک سلول را نگهداری می‌کند.

- فیلد InstName: یک رشته‌ی کاراکتری حاوی برچسب سلول است.
- فیلد CellName: یک رشته‌ی کاراکتری حاوی نام نوع سلول است. مثلاً AND21.

- فیلد InstID: شماره‌ی سلول است.
- فیلد TL: یک زوج عدد که حاوی محل سلول است که می‌تواند موقعیت گوشه یا مرکز آن باشد.
- فیلدهای width و height: عرض و ارتفاع سلول. البته ارتفاع همه‌ی سلول‌ها یکسان هستند.
- فیلد rowNo: شماره‌ی سطر حاوی سلول را نشان می‌دهد.
- فیلد net: سیمی را نشان می‌دهد که خروجی سلول به آن متصل است. این فیلد در واقع حاوی سلول‌هایی است که خروجی سلول به آنها متصل است.

۳-۲. هسته‌های جایابی

ابزار educad دارای دو هسته جایابی ساده است که دانشجویان می‌توانند این دو هسته را تغییر دهند و تأثیر بهبود آنها را در گزارش‌های جایابی یا واسط گرافیکی مشاهده نمایند.

جایاب خطی: هسته اول یک جایاب خطی است که سلول‌های طرح را به ترتیب مشاهده در فایل ورودی جایابی می‌کند. عملکرد آن به این صورت است که سلول‌ها به‌صورت خطی و به‌طور سطر به سطر چیده می‌شوند و هر گاه سلول چیده شده از مستطیل محیطی چینش بیرون بزند، کنترل به سطر بعد منتقل می‌شود و سلول بعدی از ابتدای سطر بعدی جایابی می‌شود.

جایاب دست‌ساز: هسته بعدی یک ابزار جایابی با الگوریتم سردسازی شبیه‌سازی شده است. این هسته با جابجایی‌های مکرر سلول‌ها، تابع هدف را بهبود می‌دهد. تابع هدف مورد استفاده در ابزارمجموع طول سیم است. این ابزار شامل دو بخش جایابی سراسری و جایابی جزئی است. در مرحله جایابی سراسری، محل کلی سلول‌ها بدون حذف روی هم افتادگی‌ها تعیین می‌شود و در مرحله جایابی جزئی، محل دقیق سلولها تعیین شده و روی هم افتادگی‌ها کاملاً رفع می‌گردد.

۴. ابزار مسیریاب سراسری

این ابزار یک مسیریابی سراسری ساده است که به‌عنوان یک محیط برای توسعه‌ی ابزارهای مسیریابی سراسری طراحی شده است. این هسته، یک ابزار مسیریابی برپایه‌ی درخت اشتاینر است که به ساده‌ترین شکل ممکن یک مسیریابی سراسری ساده را ایجاد می‌نماید. الگوریتم مورد استفاده در آن به‌صورت زیر است:

۴-۱. ساختمان داده ابزار مسیریاب سراسری

این ابزار در قالب کلاس simpleGlobalRouter در فایل gr.h پیاده‌سازی شده است:

```
class simpleGlobalRouter
{
	...
	public:
		vector < simpleGrNode *> nodes;
		vector < simpleGrEdge *> edges;
		vector <simpleGrNet *> nets;
		unsigned cellH, cellW;
		unsigned rowNumber, columnNumber;


	public:
		simpleGlobalRouter ( atlasDB *_db );
		~simpleGlobalRouter() {};
		void createGrMesh ( unsigned, unsigned, long int, long int);
		void simpleGrRoute ( );
		void reportCapacityViolations( );
		void PrimMST( );

	...
};
```

که متغیرهای آن عبارتند از:

متغیرهای rowNumber و columnNumber: به ترتیب تعداد سطرها و تعداد ستون‌ها در مش مسیریابی سراسری را نشان می‌دهند.

متغیرهای cellW و cellH: به ترتیب عرض و ارتفاع هر سلول مش فوق را نشان می‌دهد.

متغیر nets: یک لیست پیوندی است که هر عنصر آن یک سیم مدار را نشان می‌دهد.

متغیر edges: یک لیست پیوندی که هر عنصر آن یک یال از گراف مسیریابی را مدل می‌نماید.

متغیر nodes: یک لیست پیوندی که هر عنصر آن یک گره از گراف مسیریابی را مدل می‌نماید.

گفتنی است که دو لیست nodes و edges گراف مسیریابی را تشکیل داده‌اند.

ساختارهای اصلی مورد استفاده در این ابزار، عبارتند از:

ساختار simpleGrEdge: یک یال مش مسیریابی است و شامل دو گره دو طرف یال است. فیلدهای آن عبارتند از:

- فیلدهای node1 و node2: دو سلول متصل شده با یال هستند.
- فیلد capacity: کل ظرفیت یال را نشان می‌دهد که با توجه به تعداد و pitch لایه‌های فلز و نیز ابعاد هر سلول شبکه توسط دو تابع computeHorizontalEdgesCapacity و computeVerticalEdgesCapacity تعیین می‌شود.
- فیلد used_tracks: تعداد شیارهای مورد استفاده در طی عمل مسیریابی را نشان می‌دهد. مسلما اگر بیش از ظرفیت یک یال، سیم به آن اختصاص یابد، شرط ظرفیت روی آن یال نقض می‌گردد.

ساختار simpleGrNode: این ساختار هم یک گره از مش مسیریابی را مدل می‌کند. هر گره چهار اشاره‌گر به چهار گره اطراف خود دارد.

ساختار simpleGrNet: این ساختار اطلاعات سیم‌های طرح را در خود حفظ می‌کند. مهم‌ترین فیلدهای آن عبارتند از:

- لیست terminalBins: این لیست حاوی لیست سلول‌های مش است که هر کدام از ترمینال‌های سیم در آن قرار گرفته است.
- لیست routingTree: این لیست هم شامل یال‌هایی از مش است که پس از مسیریابی، سیم از آنها رد شده است.

توابع اصلی این کلاس عبارتند از:

تابع readNetlist(): این تابع، اطلاعات لازم برای جایابی را از بانک AtlasDB خوانده و در ساختار داخلی مسیریاب می‌ریزد.

تابع simpleRoute(): این تابع، عملکرد اصلی مسیریابی را پیاده‌سازی نموده است.

تابع createGrMesh(): این تابع، مش مسیریابی را تشکیل می‌دهد.

تابع reportCapacityViolation(): این تابع، پس از انجام مسیریابی، تعداد یال‌هایی که ظرفیت آنها نقض شده است، را گزارش می‌کند.

تابع PrimMST(): این تابع، درخت پوشای مینی‌مال را برای ترمینال‌های یک سیم خاص محاسبه می‌کند. این تابع می‌تواند برای ایجاد درخت اشتاینر هر سیم استفاده شود. در ساختمان داده مسیریاب سراسری، هر سیم (simpleGrNet) لیستی از ترمینال‌ها دارد. تابع PrimMST، اشاره‌گر یک سیم در لیست nets را دریافت کرده و پس از اجرای این تابع روی آن سیم، متغیرهای parent و Lambda برای ترمینال‌های آن سیم را تعیین می‌کند. در صورتی که تولید درخت پوشا بدون مشکل انجام شود، مقدار TRUE بر می‌گرداند. پس از تشکیل درخت پوشا، با دنبال کردن parent هر ترمینال می‌توان درخت پوشا را تشکیل داد.

۴-۲. هسته ابزار مسیریابی سراسری

در ابتدا یک شبکه روی طرح کشیده می‌شود که در قالب مش مسیریابی سراسری مدل می‌شود. سپس هر سیم چندترمینالی به تعدادی سیم‌های دوترمینالی تقسیم می‌شود و برای هر سیم دو ترمینالی، از مبدأ آن تا هر کدام از مقصدها، یک مسیر به شکل L ایجاد می‌کند. در نهایت مسیر ایجاد شده روی شبکه‌ی ایجاد شده نگاشته می‌شود.

۵. ابزار تحلیل زمانی ایستا

این ابزار یک مسیریابی تحلیل زمانی ایستاست که مسیر بحرانی طرح را به صورتی ساده تشخیص می‌دهد. لازم به ذکر است که این ابزار یک ابزار ساده است و کاربرد صنعتی ندارد. مهم‌ترین محدودیت آن این است که پارامترهای آن متناسب با فناوری CRETE 180nm قرار داده شده است، در حالی که برای کاربردهای صنعتی باید فایل LIB. تحلیل شود. این ابزار لیست مسیرهای نزدیک بحرانی را تشکیل می‌دهد.

۶. افزودن ابزار طراحی خودکار جدید به ابزار

برای افزودن هر ابزار جدید به این مجموعه،کافی است یک شیء از ابزار جدید در فایل educad.cpp ایجاد نمایید و هر کدام از توابع آن ابزار که بخواهند با بانک AtlasDB کار کنند،اشاره‌گر بانک داده را نیز داشته باشند. بعنوان مثال کلاس ابزار SimplePlacer بصورت زیر تعریف شده است:

```
class SimplePlacer {
public:
	SimplePlacer();
	~SimplePlacer();
	int readDB(atlasDB *);
	void updateDB(atlasDB *);
	...
private:
	...
};
```

و در فایل EduCAD.cpp نیز یک شیء از این کلاس ایجاد شده است.

```
…
#include "placer.h"
#include "atlasDB.h"

atlasDB *db;
SimplePlacer *sPL;
SimpleGlobalRouter *sGR;


int main()
{
	...
	db = new atlasDB;
	sPL = new SimplePlacer();
	sPL->readDB(db);
	sPL->simPlacer();
	sPL->updateDB(db);
	...
	sGR -> createGrMesh ( 10, 10, LayoutWidth, LayoutHeight );
	sGR -> simpleGrRoute ();
	sGR -> reportCapacityViolations();
	...
}
```

دقت داشته باشید که ساختار این سیستم به این صورت است که هر ابزار جدیدی که به این مجموعه اضافه می‌شود، در ابتدای اجرای خود اطلاعات را از بانک AtlasDB خوانده و سپس الگوریتم‌های خود را روی

ساختار داده‌ی داخلی خود انجام می‌دهد. پس از تکمیل عملیات، نتایج را در این بانک بروز می‌سازد و ابزارها، مستقیماً الگوریتم‌های خود را روی AtlasDB اجرا نمی‌کنند.