طراحی سیستم‌های تعبیه‌شده
Embedded System Design

فصل دوم ـ قسمت هشتم

مشخص‌سازی
Specifications

کاظم فولادی
دانشکده‌ی مهندسی برق و کامپیوتر
دانشگاه تهران

kazim@fouladi.ir

SystemC
انگیزه‌ها

- بسیاری از استانداردها (مانند GSM و MPEG) به صورت برنامه‌های C منتشر شده‌اند.
  - اگر زبان‌های توصیف سخت‌افزار خاص برای استفاده مورد نظر باشد، استانداردها باید ترجمه شوند.
- کارکرد بسیاری ازسیستم‌ها به وسیله‌ی ترکیبی از مؤلفه‌های سخت‌افزاری و نرم‌افزاری فراهم می‌شود.
  - شبیه‌سازی‌ها به واسطی برای شبیه‌سازهای سخت‌افزاری و نرم‌افزاری نیاز دارند، مگر اینکه یک زبان واحد برای توصیف سخت‌افزار و نرم‌افزار استفاده شود.

تلاش‌هایی برای توصیف نرم‌افزار و سخت‌افزار در یک زبان.

استفاده از لهجه‌های مختلف C برای توصیف سخت‌افزار.

SystemC
ویژگی‌های مورد نیاز

نیازمندی‌ها و راه‌حل‌ها برای مدل‌سازی سخت‌افزار در یک زبان نرم‌افزاری:

- **کتابخانه کلاس C++ شامل توابع لازم**
- **همروندی:** از طریق پردازش‌ها، کنترل شده با لیست‌های حساسیت و فراخوانی‌های فرمان‌های ابتدایی wait *
- **زمان:** اعداد ممیز شناور در SystemC 1.0 و مقادیر صحیح در SystemC 2.0، شامل واحدهایی چون ps، ns، µs *
- **پشتیبانی انواع داده‌ی بیتی:** بردارهای بیتی با طول‌های مختلف؛ منطق چندمقداری (منطق ۲ و ٤ مقداری؛ resolution) *
- **ارتباطات:** مدل کانال plug-and-play،

رفتار قطعی تضمین نمی‌شود.

* Good to know VHDL ☺

Verilog

زبان توصیف سخت‌افزار رقیب VHDL

استانداردها:

- IEEE 1364-1995 (Verilog version 1.0)
- IEEE 1364-2001 (Verilog version 2.0)

ویژگی‌های شبیه به VHDL:

- توصیف طراحی با موجودیت‌های متصل به هم
- بردارهای بیتی و واحدهای زمانی پشتیبانی می‌شوند.

ویژگی‌های متفاوت با VHDL:

- پشتیبانی توکار منطق ٤ مقداری و منطق با ٨ سطح قوت سیگنال که در دو بایت برای هر سیگنال کد می‌شود.
- ویژگی‌های بیشتر برای توصیفات سطح ترانزیستور
- انعطاف کمتر نسبت به VHDL
- متداول‌تر در آمریکا (VHDL بیشتر در اروپا متداول است.)

SystemVerilog

متناظر با نسخه‌های 3.0 و 3.1 از Verilog

شامل:

- عناصر اضافی زبان برای مدل‌سازی رفتار
- انواع داده‌ای زبان C مانند `int`
- امکانات تعریف نوع
- تعریف واسط‌های مؤلفه‌های سخت‌افزاری به عنوان موجودیت‌های جدا
- مکانیزمی برای فراخوانی توابع C/C++ از داخل Verilog
- مکانیزم محدودشده برای فراخوانی توابع Verilog از C.

SystemVerilog
ادامه

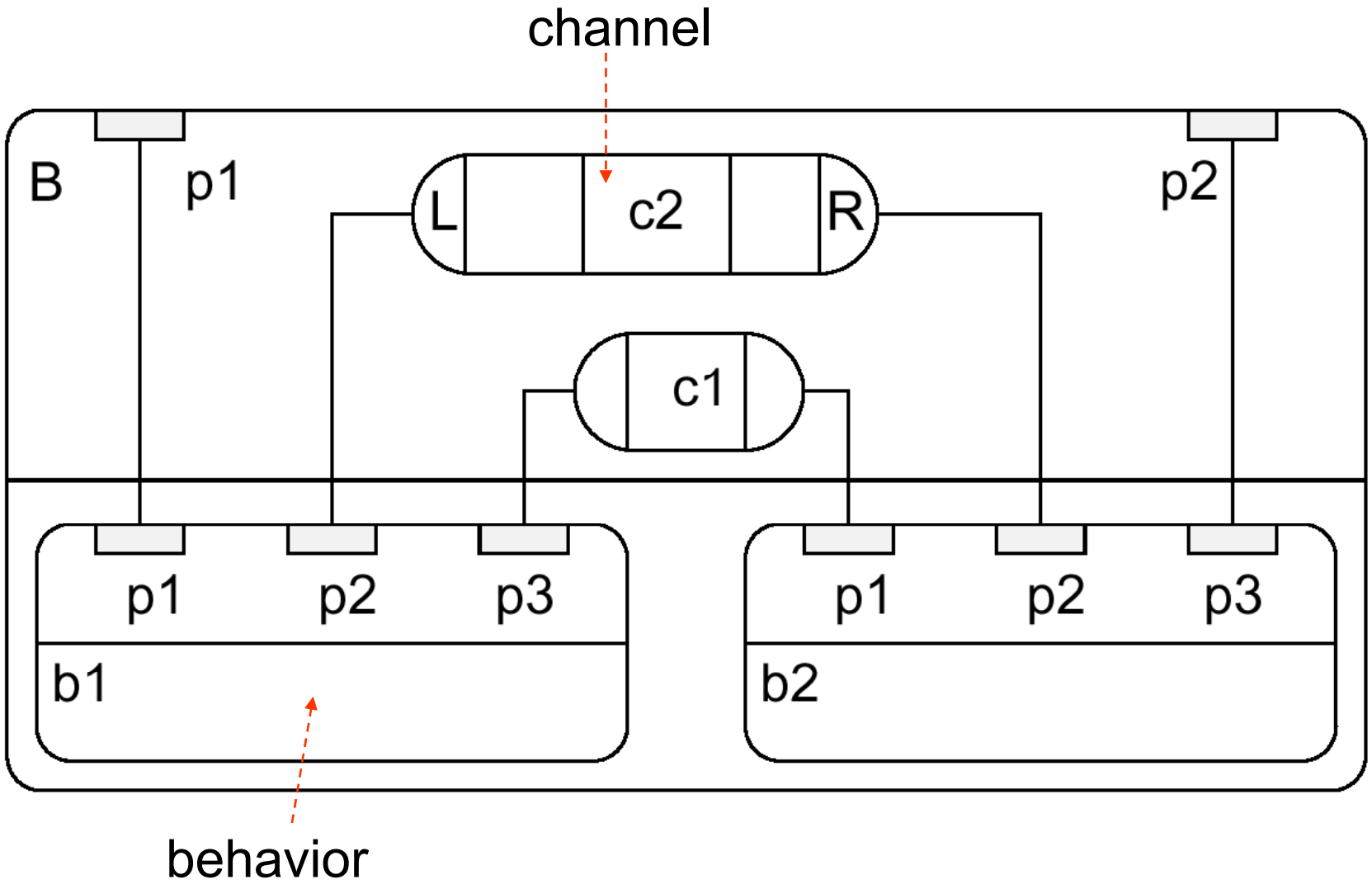
- ویژگی‌های بهبودیافته برای توصیف بستر آزمون (testbench)
- کلاس‌ها می‌توانند در بسترهای آزمون استفاده شوند.
- ایجاد پویای پردازش‌ها
- ارتباطات میان‌پردازشی و همگام‌سازی استاندارد شده شامل سمافورها
- تخصیص و آزادسازی خودکار حافظه
- ویژگی‌های فراهم شده توسط زبان برای وارسی رسمی

☞ Significant hype about the potential of SystemVerilog

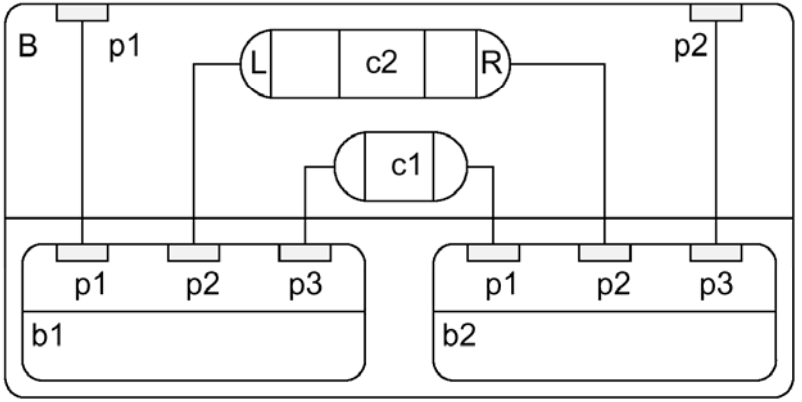
SpecC [Gajski, Dömer et. al. 2000]

- SpecC مبتنی بر تفکیک ارتباطات و محاسبات است.
  برای مؤلفه‌های سیستم "plug-and-play" بودن را فعال می‌کند.
  سیستم‌ها را به صورت شبکه‌های سلسله‌مراتبی از رفتارهای مرتبط با هم از طریق کانال‌ها مدل می‌کند.
- **SpecC** از رفتارها، کانال‌ها و واسط‌ها تشکیل می‌شود.
- **رفتارها** شامل درگاه‌ها، مؤلفه‌های مقداردهی شده به صورت محلی، متغیرها و توابع خصوصی و یک تابع اصلی عمومی هستند.
- **کانال‌ها** ارتباطات را کپسوله می‌کنند. کانال‌ها شامل متغیرها و توابع، هستند و برای تعریف یک پروتکل ارتباطی استفاده می‌شوند.
- **واسط‌ها** رفتارها و کانال‌ها را با یکدیگر مرتبط می‌کنند. آنها پروتکل‌های ارتباطی که در کانال‌ها تعریف می‌شوند را اعلان می‌کنند.

Example



SpecC-Example




```
interface L {void Write(int x);};
interface R {int Read (void);};
channel C implements L,R
{ int Data; bool Valid;
   void Write(int x)  {Data=x; Valid=true;}
   int Read(void) {while (!Valid) waitfor(10); return (Data);}
};
behavior B1 (in int p1, L p2, in int p3)
  {void main(void)    {/*...*/ p2.Write(p1);}  };
behavior B2 (out int p1, R p2, out int p3)
  {void main(void)   {/*...*/ p3=p2.Read();   }  };
behavior B(in int p1, out int p2)
{ int c1; C c2; B1 b1(p1,c2,c1); B2 b2 (c1,c2,p2);
    void main (void)
    {par {b1.main();b2.main();}}
};
```

زبان‌های دیگر

- **Pearl :** طراحی شده در آلمان برای کاربردهای کنترل فرایند.
  (دهه‌ی70). متداول در اروپا.
- **Chill :** طراحی شد برای ایستگاه‌های تبادلات تلفنی، بر اساس پاسکال.
- **IEC 60848 و STEP 7 :**
  زبان کنترل فرایند با استفاده از عناصر گرافیکی.
- **SpecCharts :**
  ترکیب StateCharts و VHDL،
  طراحی شده توسط Gajski و دیگران، جایگزین شده توسط SpecC

زبان‌های دیگر (ادامه)

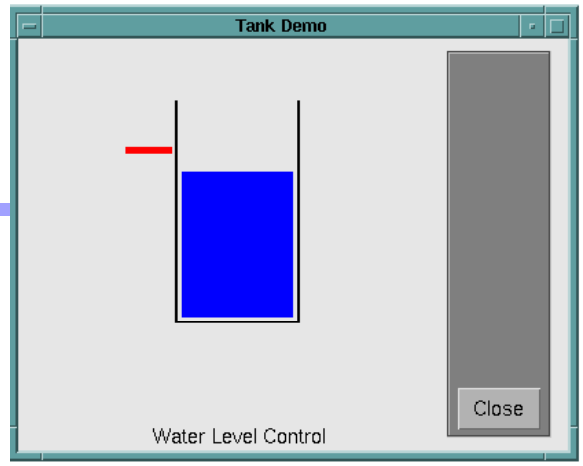
- **Estelle :** طراحی شده برای توصیف پروتکل‌های ارتباطی؛ شبیه SDL؛ متحد سازی هر دوی آنها موفق نبوده است.
- **LOTOS و Z :** زبان‌های مشخص‌سازی جبری
- **Silage :** زبان تابعی برای پردازش سیگنال دیجیتال.
- **Rosetta :** تلاش‌هایی برای زبان‌های جدید طراحی سیستم.
- **Esterel :** زبان‌های واکنشی؛ همگام‌سازی؛
  همه‌ی واکنش‌ها فرض می‌شوند که در زمان صفر انجام می‌شوند.
  ارتباطات بر اساس پخش همگانی («لحظه‌ای»)
  //www.esterel-technologies.com

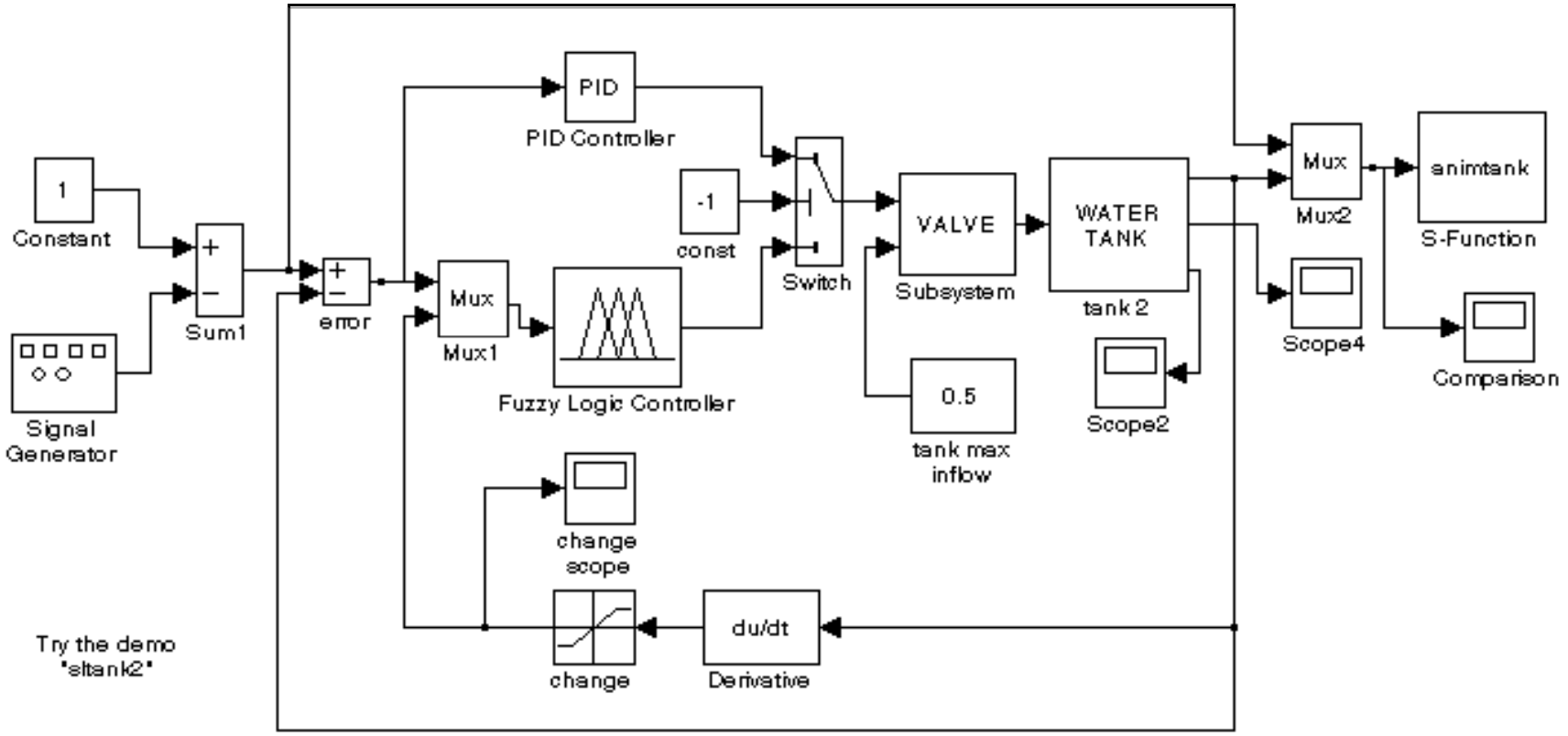
MATLAB/Simulink

- **MATLAB** (Matrix Laboratory) آزمایشگاه ماتریسی
  امکاناتی برای تعریف محاسبات مبتنی بر ماتریس
  توسعه‌ی بسته‌های عددی فرترن، LINPACK و EISPACK با یک GUI
- **Simulink**:
  مشخص‌سازی سیستم‌های کنترل به صورت GUI
  استفاده از MATLAB به صورت داخلی برای حل این معادلات.
- **StateFlow**:
  ابزار مبتنی بر StateCharts مجتمع شده در MATLAB

Simulink-example:

From www.mathworks.co.uk/access/helpdesk/help/toolbox/fuzzy/fuzzyt25.shtml





خلاصه

Languages briefly presented

- SystemC
- Verilog, SystemVerilog
- SpecC
- Pearl, Chill, IEC 60848, SpecCharts, Estelle, Lotos, Z, Silage, Rosetta, Esterel, MATLAB/Simulink

سطوح مدل‌سازی سخت‌افزار

۱. سطح سیستم (System level)
۲. سطح الگوریتمی (Algorithmic level)
۳. سطح مجموعه دستورالعمل (Instruction set level)
۴. سطح انتقال ثبات (Register-transfer level (RTL))
۵. مدل‌های سطح گیت (Gate-level models)
۶. مدل‌های سطح سوییچ (Switch-level models)
۷. مدل‌های سطح مدار (Circuit-level models)
۸. مدل‌های سطح دستگاه (Device-level models)
۹. مدل‌های جانمایی (Layout models)
۱۰. مدل‌های فرایند و دستگاه (Process and device models)

سطح سیستم

- اصطلاح سطح سیستم به طور واضح تعریف نشده است.
- این اصطلاح در اینجا برای اشاره به کل سیستم تعبیه شده و سیستمی که پردازش اطلاعات درون آن تعبیه شده است، به کار می‌رود و احتمالاً شامل محیط هم هست.
- چنین مدل‌هایی شامل جنبه‌های مکانیکی و جنبه‌های پردازش اطلاعات است و بنابراین یافتن شبیه‌سازهای مناسب برای آن می‌تواند دشوار باشد. راه‌حل‌هایی شامل VHDL-AMS، SystemC یا MATLAB. MATLAB و VHDL-AMS مدلسازی معادلات دیفرانسیل با مشتقات جزیی را پشتیبانی می‌کند.
- چالش‌هایی برای مدل کردن اجزای پردازش اطلاعات سیستم در چنین روشی که مدل شبیه‌سازی بتواند برای سنتز سیستم‌های تعبیه‌شده نیز استفاده شود.

سطح الگوریتمی

- شبیه‌سازی الگوریتم‌هایی که قصد داریم در سیستم تعبیه‌شده از آنها استفاده کنیم.
- بدون ارجاع به پردازنده یا مجموعه دستورالعمل خاص
- انواع داده‌ای می‌توانند دقت بالاتری نسبت به پیاده‌سازی نهایی داشته باشند.
- اگر انواع داده‌ای به گونه‌ای انتخاب شده باشند که هر بیت دقیقاً با یک بیت در پیاده‌سازی نهایی متناظر باشد،
  این مدل **بیت ـ درست (bit-true)** نام دارد. ترجمه‌ی مدل غیر بیت ـ درست به مدل بیت ـ درست باید با پشتیبانی ابزارها انجام شود.
- می‌تواند متشکل از یک پردازش یا مجموعه‌ای از پردازش‌ها باشد.

Algorithmic level: Example:
-MPEG-4 full motion search -

```
for (z=0; z<20; z++)
 for (x=0; x<36; x++) {x1=4*x;
  for (y=0; y<49; y++) {y1=4*y;
   for (k=0; k<9; k++) {x2=x1+k-4;
    for (l=0; l<9; ) {y2=y1+l-4;
     for (i=0; i<4; i++) {x3=x1+i; x4=x2+i;
      for (j=0; j<4;j++) {y3=y1+j; y4=y2+j;
       if (x3<0 || 35<x3||y3<0||48<y3)
        then_block_1; else else_block_1;
       if (x4<0|| 35<x4||y4<0||48<y4)
        then_block_2; else else_block_2;
}}}}}}
```

سطح دستورالعمل

الگوریتم‌هایی که برای پردازنده‌ای (پردازنده‌هایی) که باید استفاده شود، کامپایل شده‌اند.

شبیه‌سازی در این سطح امکان شمارش تعداد دستورالعمل‌های اجرایی را فراهم می‌کند.

دگرگونی‌ها:

ـ تنها شبیه‌سازی تاثیر دستورالعمل‌ها

ـ مدلسازی سطح تراکنش (Transaction-level modeling):

هر خواندن/نوشتن یک تراکنش است (به جای مجموعه‌ای از انتساب‌های سیگنال)

ـ شبیه‌سازی‌های چرخه ـ درست (Cycle-true simulations):

تعداد دقیق چرخه‌ها

ـ شبیه‌سازی‌های بیت ـ درست (Bit-true simulations):

شبیه‌سازی با استفاده از تعداد درست بیت‌ها به طور دقیق

Instruction level: example

Assembler (MIPS)	Simulated semantics
`and $1,$2,$3`	`Reg[1]:=Reg[2] ∧ Reg[3]`
`or $1,$2,$3`	`Reg[1]:=Reg[2] ∨ Reg[3]`
`andi $1,$2,100`	`Reg[1]:=Reg[2] ∧ 100`
`sll $1,$2,10`	`Reg[1]:=Reg[2] << 10`
`srl $1,$2,10`	`Reg[1]:=Reg[2] >> 10`

سطح انتقال ثبات (RTL)

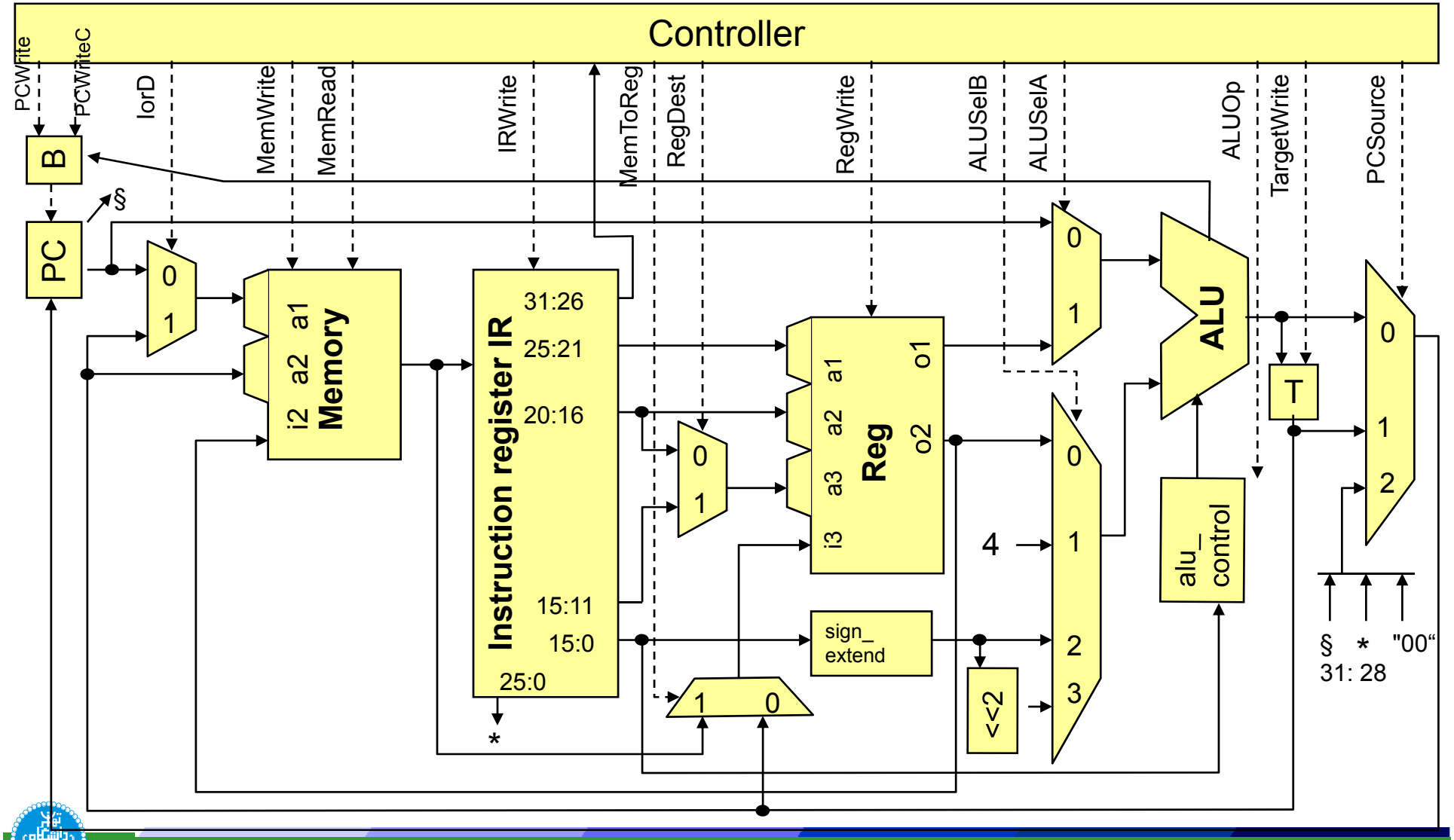
در این سطح، تمام مؤلفه‌های سیستم را در سطح انتقال ثبات مدل می‌کنیم، شامل:

- واحدهای حسابی/منطقی (ALU)،
- ثبات‌ها،
- حافظه‌ها،
- مالتی پلکسرها،
- کدگذارها،

مدل‌های این سطح همیشه **چرخه درست** هستند.

سنتز خودکار از روی چنین مدل‌هایی، چالش بزرگی نیست.

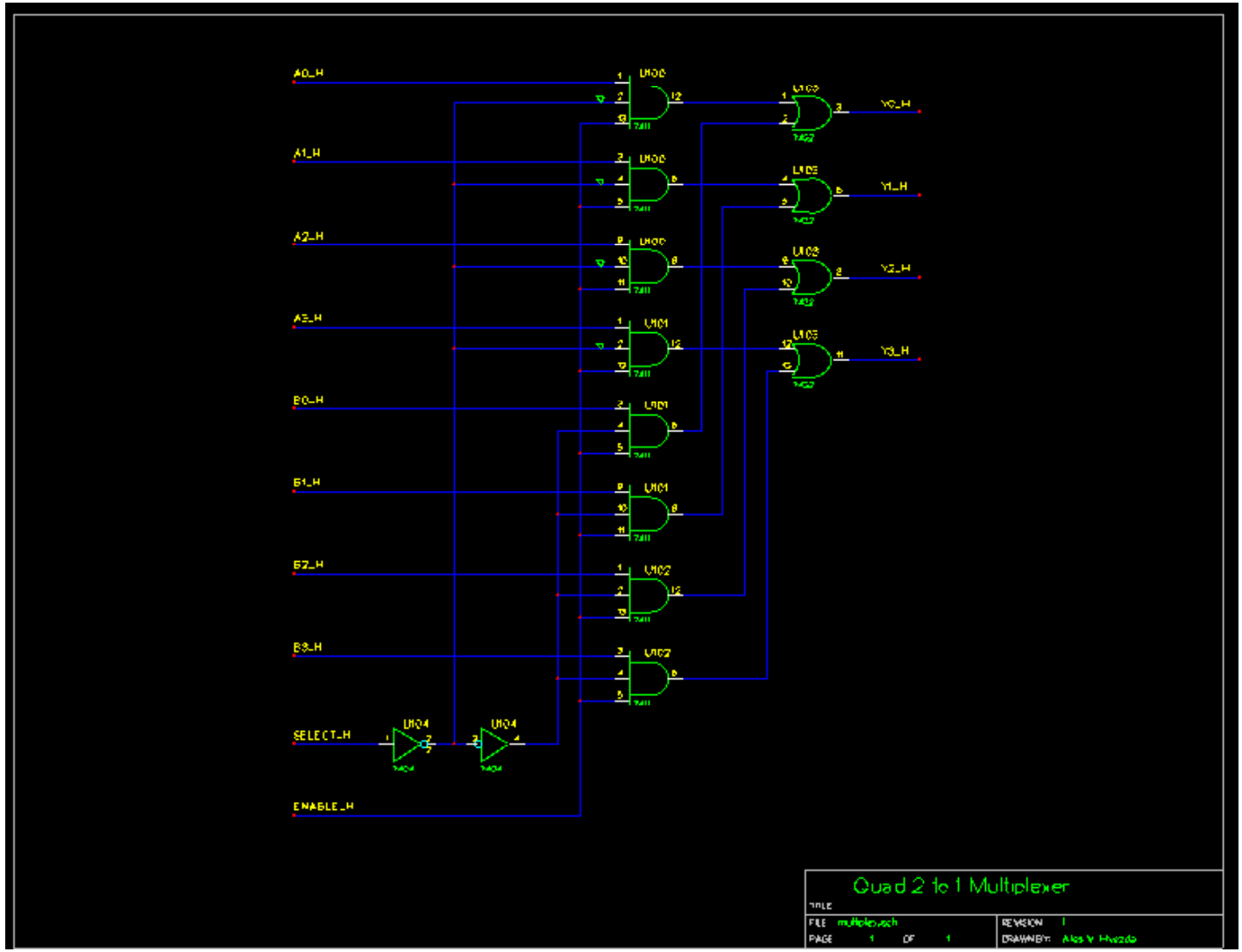
Register transfer level: example (MIPS)



مدل‌های سطح گیت

- مدل‌های حاوی گیت‌ها به عنوان مؤلفه‌های پایه
- اطلاعات دقیقی در مورد احتمالات انتقال سیگنال فراهم می‌کند. بنابراین می‌تواند برای براورد توان مصرفی استفاده شود.
- محاسبات تاخیر می‌تواند دقیق‌تر از RTL باشد.
  معمولاً اطلاعاتی در مورد طول سیم‌ها وجود ندارد (تخمینی)
- اصطلاح مدل سطح گیت گاهی برای اشاره به توابع بولی نیز بکارگرفته می‌شود (نه گیت‌های فیزیکی؛ تنها رفتار گیت‌ها در نظر گرفته می‌شود). چنین مدل‌هایی باید «مدل توابع بولی» نامیده شوند.

Gate-level models: Example

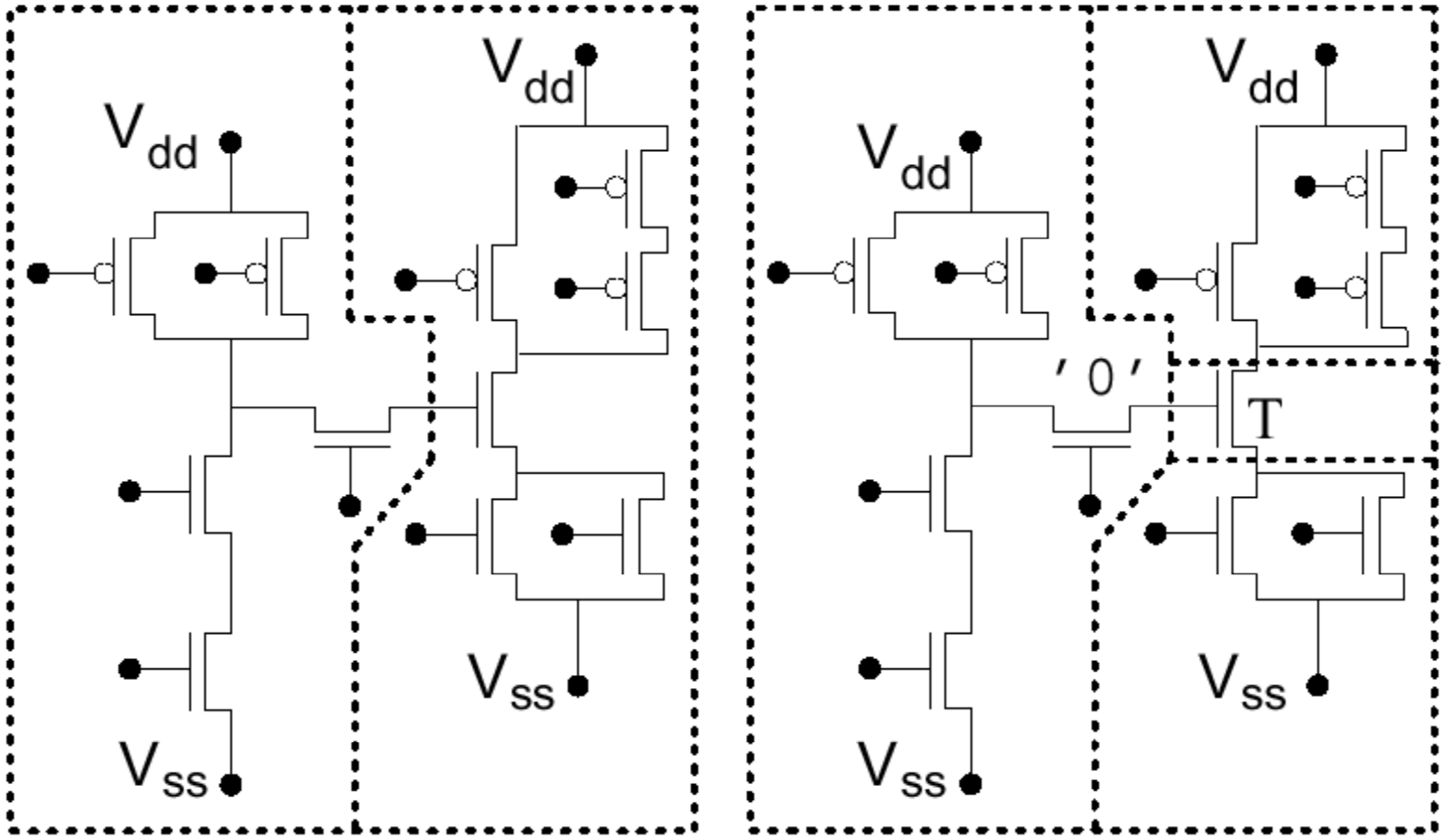


source: http://geda.seul.org/screenshots/screenshot-schem2.png

مدل‌های سطح سوییچ

- مدل‌های سطح سوییچ از سوییچ‌ها (ترانزیستورها) به عنوان مؤلفه‌ی اصلی خود استفاده می‌کنند.
- مدل‌های سطح سوییچ از مدل‌های مقادیر دیجیتال استفاده می‌کنند.
- برخلاف مدل‌های سطح گیت، مدل‌های سطح سوییچ قادر به انعکاس انتقال دوطرفه‌ی اطلاعات هستند.

Switch level model: example



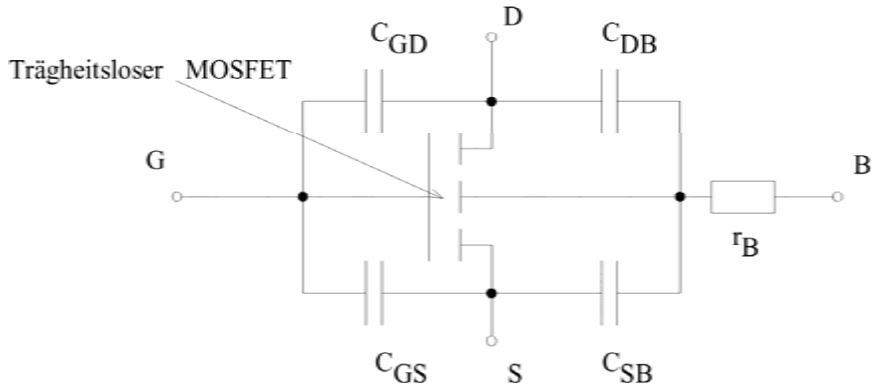
Source: http://vada1.skku.ac.kr/ClassInfo/ic/vlsicad/chap-10.pdf

مدل‌های سطح مدار

- در این سطح نظریه‌ی مدار و مؤلفه‌های آن (منابع جریان و ولتاژ، مقاومت‌ها، خازن‌ها، القاگرها و احتمالاً مدل‌های کلان از نیمه‌هادی‌ها) اساس شبیه‌سازی در این سطح را تشکیل می‌دهند.

Simulations involve partial differential equations.
Linear if and only if the behavior of semiconductors is linearized.

Ideal MOSFET

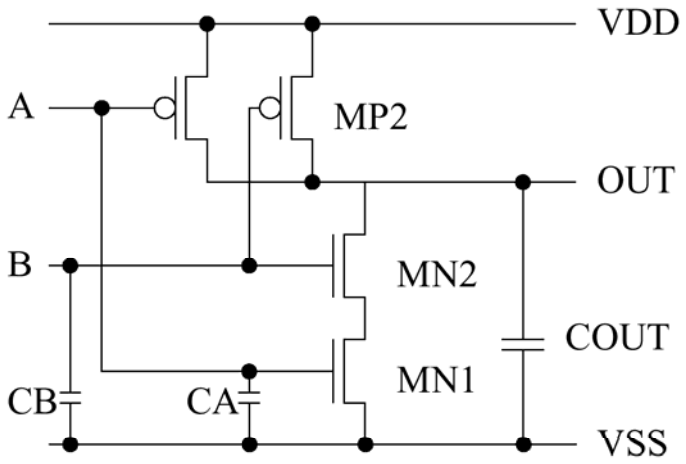


Transistor model

Circuit level models: SPICE

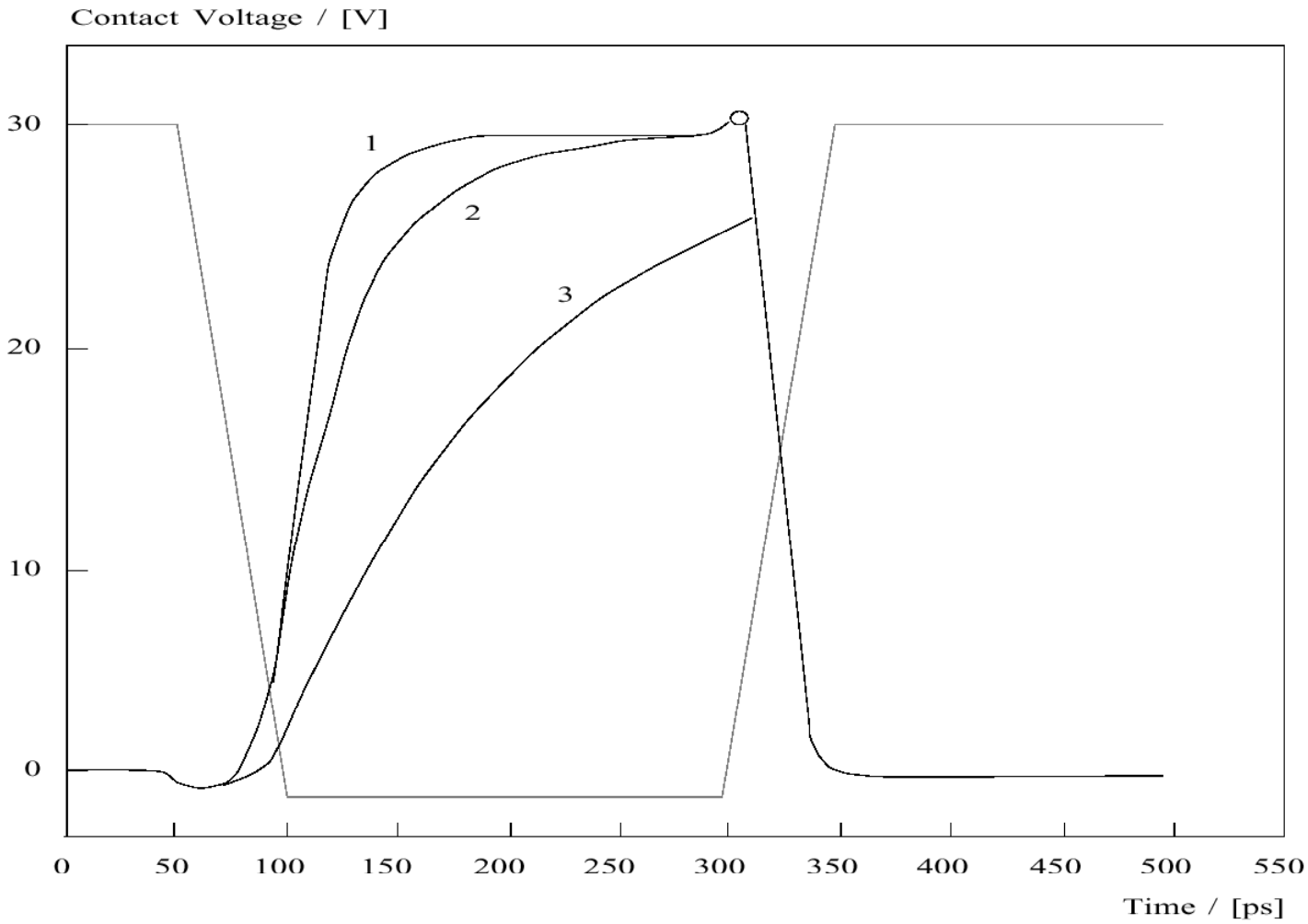
The most frequently used simulator at this level is SPICE [Vladimirescu, 1987] and its variants.

Example:




```
.SUBCKT NAND2 VDD VSS A B OUT
MN1 I1 A VSS VSS NFET W=8U L=4U AD=64P AS=64P
MN2 OUT B I1 VSS NFET W=8U L=4U AD=64P AS=64P
MP1 OUT A VDD VDD PFET W=16U L=4U AD=128P AS=128P
MP2 OUT B VDD VDD PFET W=16U L=4U AD=128P AS=128P
CA A VSS 50fF
CB B VSS 50fF
COUT OUT VSS 100fF
.ENDS
```

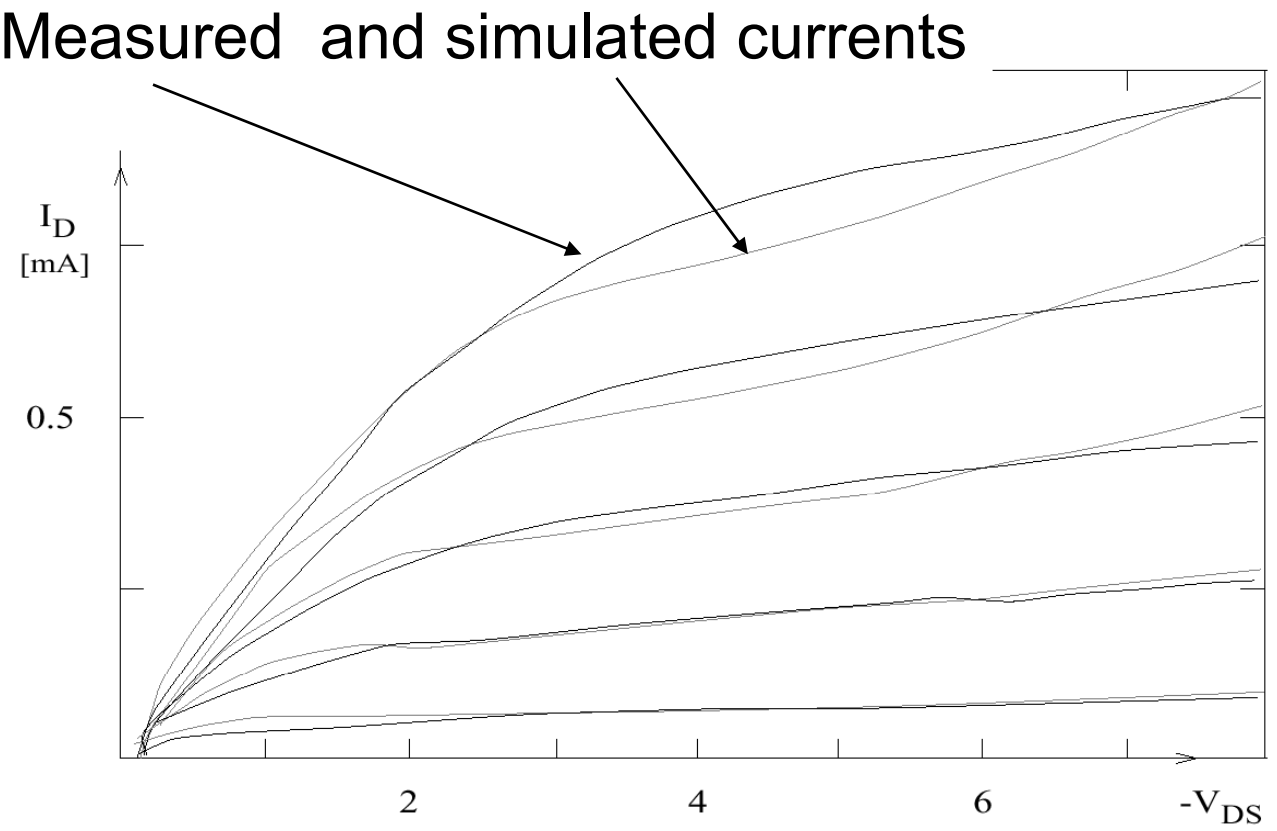
Circuit level models: sample simulation results



سطح دستگاه

Simulation of a single device (such as a transistor).

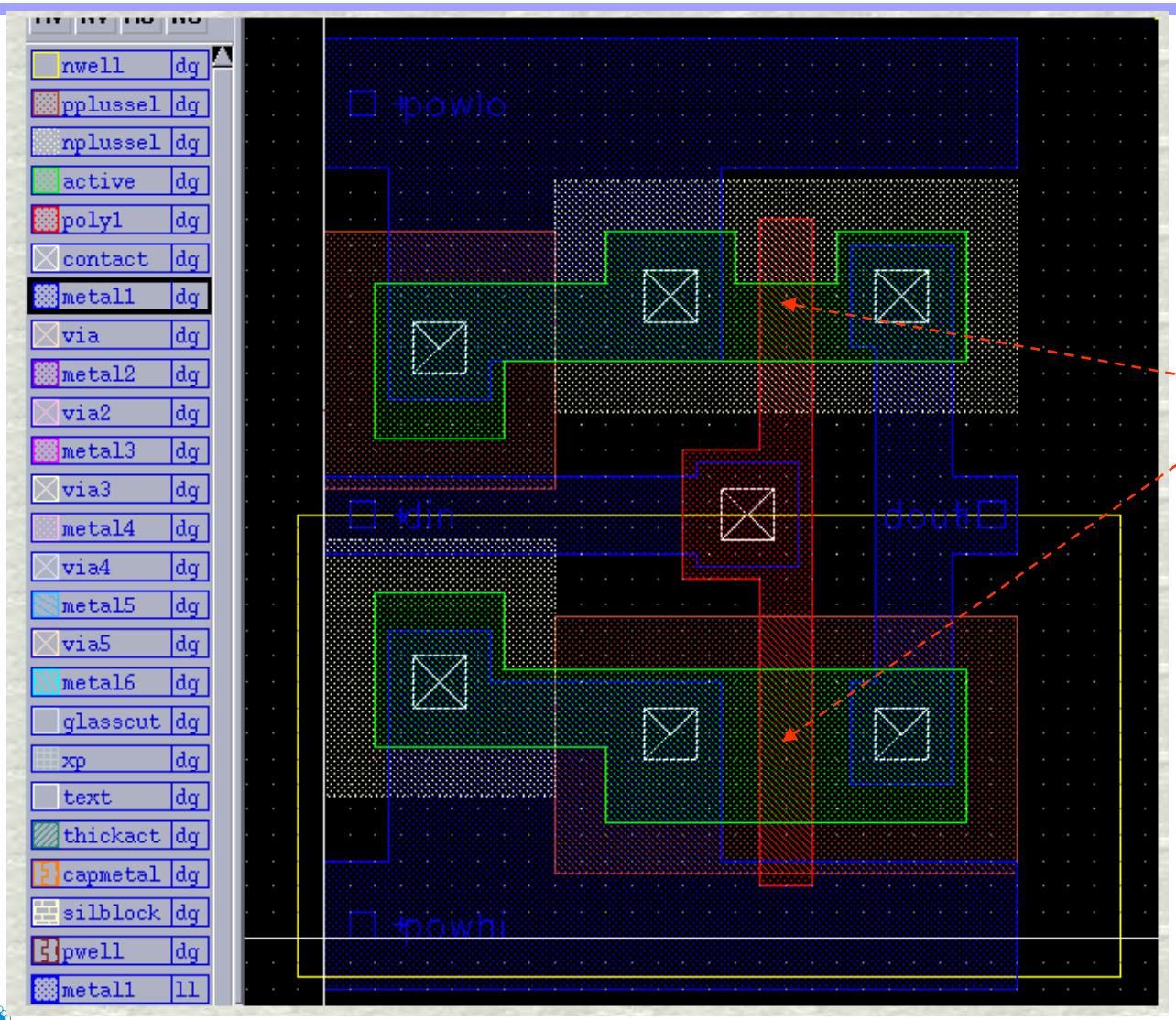
Example (SPICE-simulation [IMEC]):

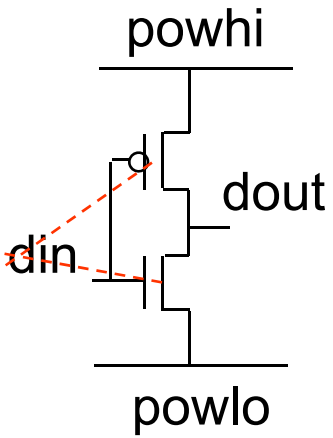


مدل‌های جانمایی (Layout models)

- جانمایی واقعی مدار را منعکس می‌کند.
- حاوی اطلاعات **هندسی** است.
- نمی‌تواند به طور مستقیم شبیه‌سازی شود:
  رفتار مدار می‌تواند با همبسته کردن مدل جانمایی با یک توصیف رفتاری در سطحی بالاتر یا با استخراج مدار از روی جانمایی استنتاج شود.
- طول سیم‌ها و ظرفیت‌های خازنی معمولاً از روی جانمایی استخراج می‌شود که با توصیفاتی در سطوح بالاتر
  پشت‌نویسی (back-annotated) می‌شود
  (دقیق‌تر برای تخمین تاخیر و توان).

Layout models: Example

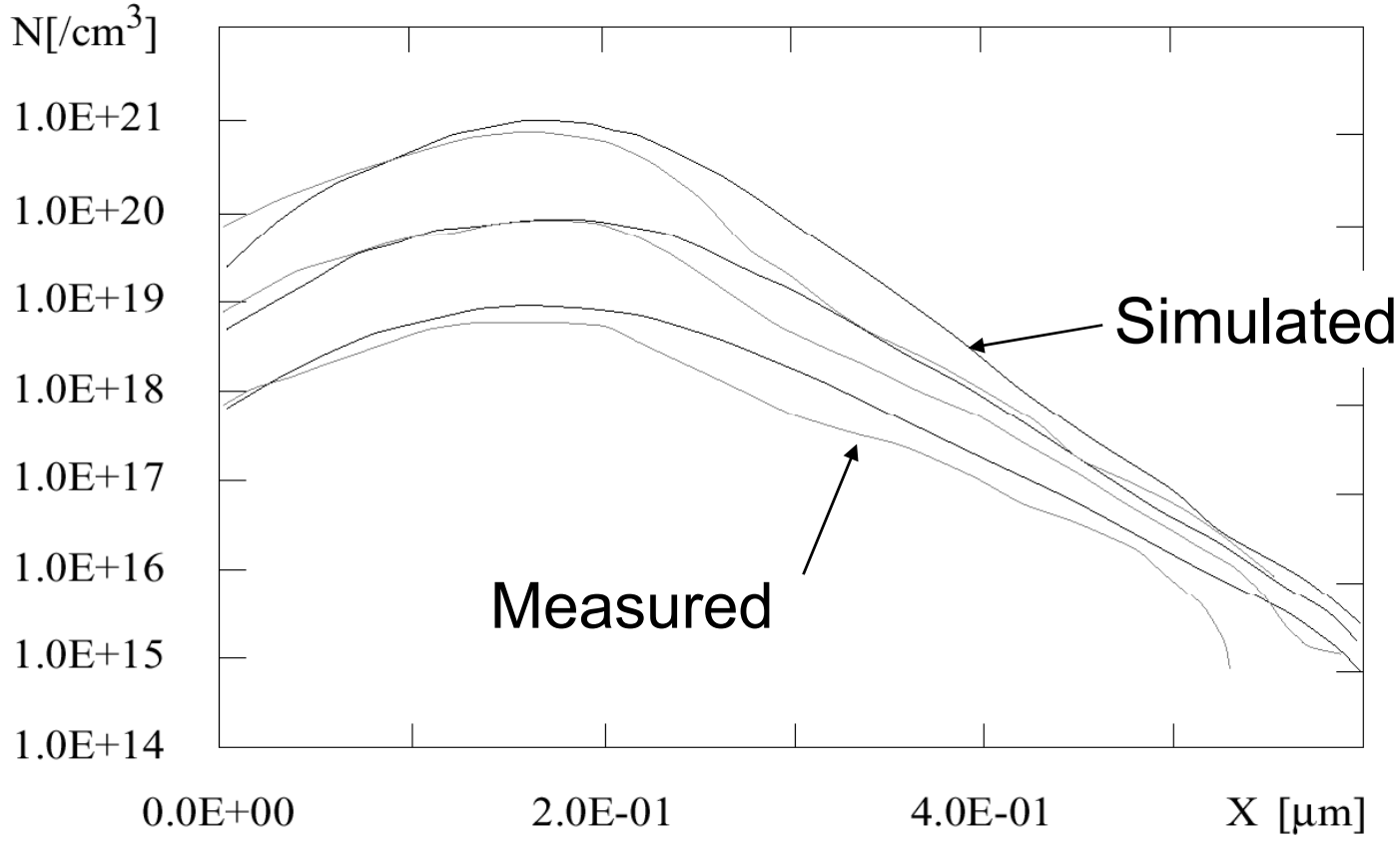




© Mosis (http://www.mosis.org/Technical/Designsupport/polyflowC.html); Tool: Cadence

مدل‌های فرایند

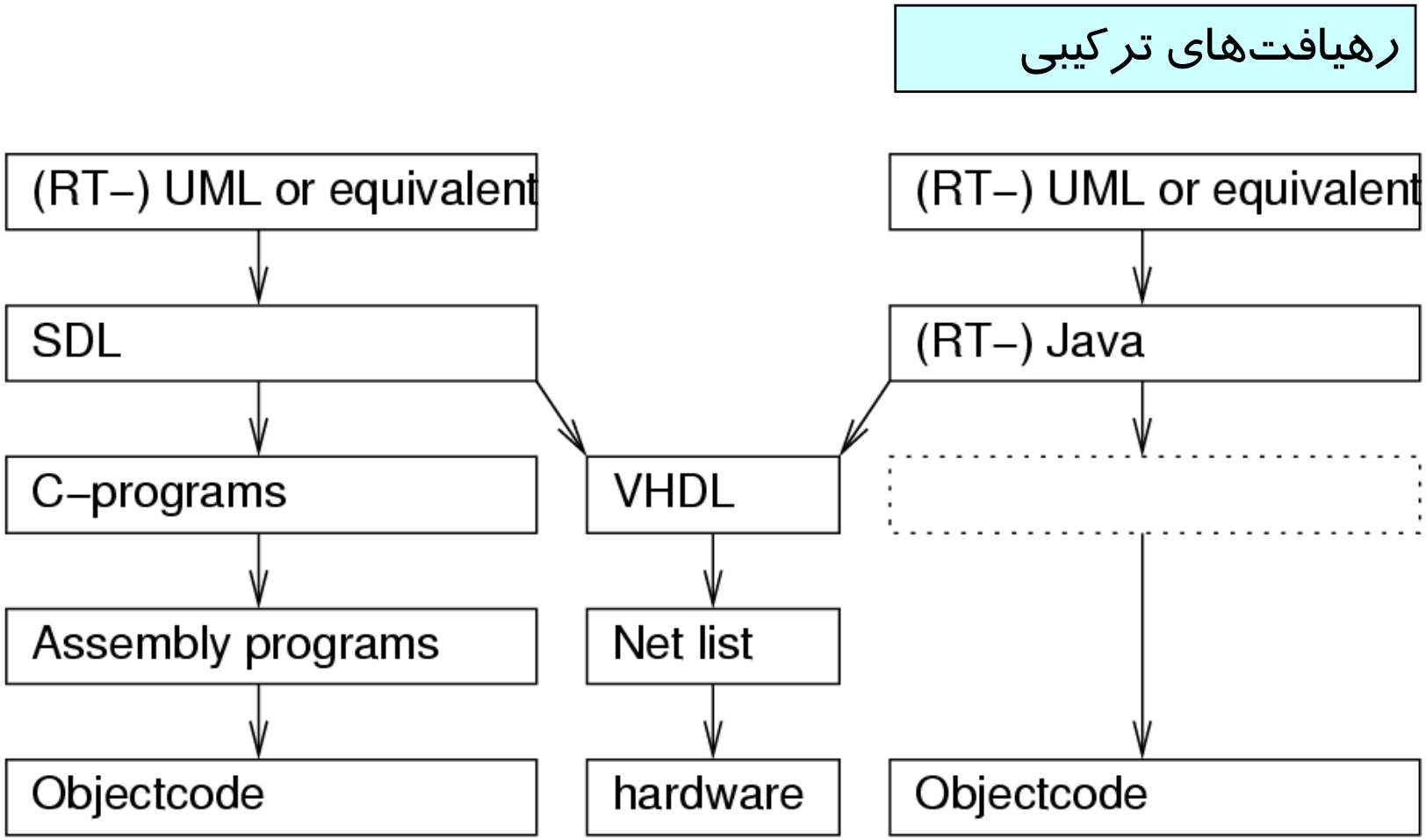
Model of fabrication process; Example [IMEC]:
Doping as a function of the distance from the surface



مقایسه‌ی زبان‌ها

Language	Behavioral Hierarchy	Structural Hierarchy	Programming Language Elements	Exceptions Supported	Dynamic Process Creation
StateCharts	+	-	-	+	-
VHDL	+	+	+	-	-
SpecCharts	+	-	+	+	-
SDL	+-	+-	+-	-	+
Petri nets	-	-	-	-	+
Java	+	-	+	+	+
SpecC	+	+	+	+	+
SystemC	+	+	+	- (2.0)	- (2.0)
ADA	+	-	+	+	+

چگونه در عمل بر مسایل زبان‌ها فایق آییم؟



نیازمندی‌های قابلیت اتکا

شکست‌های مجاز از مرتبه‌ی 1 در 10^9 ساعت است.

~ 1000 مرتبه کمتر از نرخ‌های شکست در تراشه‌های واقعی

- برای سیستم‌های ایمنی ـ حیاتی، کل سیستم باید از تک تک اجزای آن قابل اتکاتر باشد.
- مکانیزم‌های تحمل نقص باید استفاده شوند.

نرخ شکست قابل قبول پایین: ⇐ سیستم‌ها ۱۰۰٪ آزمون‌پذیر نیستند.

- ایمنی باید با ترکیبی از آزمون و استنتاج نشان داده شود.

انتزاع باید استفاده شود تا سیستم با استفاده از مجموعه‌ای سلسله‌مراتبی از مدل‌های رفتاری قابل تشریح گردد.

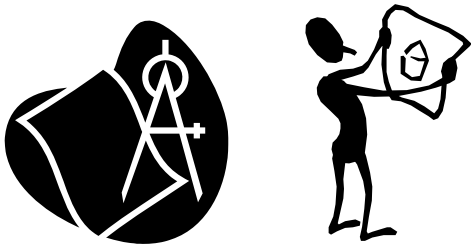
اشتباهات انسانی و نقص‌های طراحی باید به حساب آورده شود.

Kopetz‘s 12 design principles (1-3)

۱. ملاحظات ایمنی که باید به عنوان بخش مهم مشخص‌سازی به کارگرفته شود، کل فرایند طراحی را هدایت می‌کند.

۲. مشخص‌سازی‌های دقیق فرضیات طراحی باید به درستی در ابتدا انجام شود. (شامل شکست‌های قابل انتظار و احتمال آنها)

۳. نواحی آلوده به نقص (FCR) باید در نظر گرفته شوند. نقص‌های یک FCR نباید بر سایر FCRها تاثیر بگذارد.

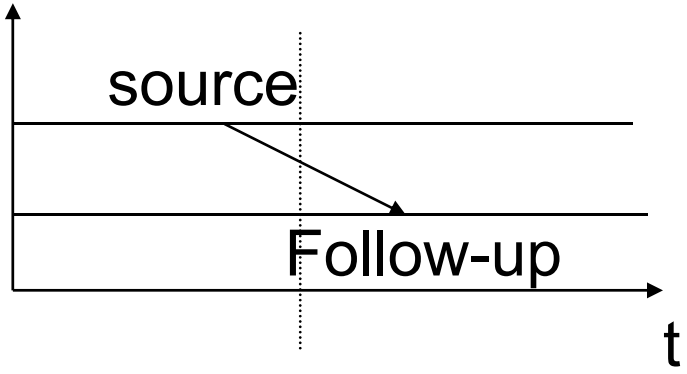


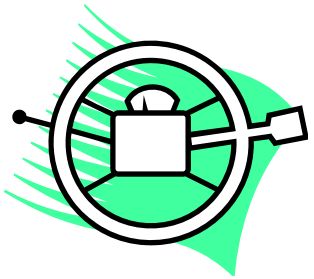
Kopetz‘s 12 design principles (4-6)

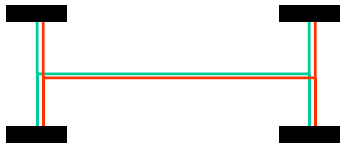
۴. تصور سازگاری از زمان و حالت باید بنا شود. در عیر این صورت تمایز بین خطاهای اصلی و پیرو غیرممکن خواهد بود.

۵. واسط‌های خوش‌تعریف باید مؤلفه‌های داخلی را پنهان نمایند.

۶. باید مطمئن شد که مؤلفه‌ها به طور مستقل از کار می‌افتند.



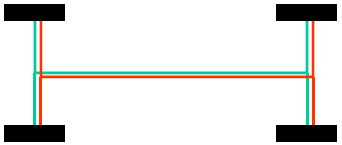




2 independent brake hose systems

Kopetz‘s 12 design principles (7-9)

۷. مؤلفه‌ها باید خودشان را درست در نظر بگیرند؛



one of the systems sufficient for braking

۸. مکانیزم‌های تحمل‌پذیری نقص باید به گونه‌ای طراحی شوند که در تشریح رفتار سیستم دشواری اضافی ایجاد نکنند.
مکانیزم‌های تحمل‌پذیری نقص باید از کارکرد عادی مجزا شود.

۹. سیستم باید برای عیب‌یابی طراحی شود. برای مثال باید ممکن باشد خطاهای موجود (اما پنهان) را تعیین کرد.

Kopetz's 12 design principles (10-12)

۱۰. واسط انسان ـ ماشین باید شهودی و بخشاینده باشد. ایمنی باید با وجود اشتباهات ایجاد شده به وسیله‌ی انسان حفظ شود.

۱۱. هر ناهنجاری باید تبت شود. این ناهنجاری‌ها می‌توانند در سطح واسط عادی غیر قابل مشاهده باشند. ثبت کردن ناهنجاری می‌تواند شامل اثرات داخلی باشد چرا که در غیر این صورت می‌تواند با مکانیزم‌های تحمل نقص پوشیده شوند.

۱۲. یک راهبرد هرگز دست‌برنداشتنی را ایجاد کنید. باید سرویس‌دهی بدون وقفه را فراهم کنید. تولید پنجره‌های pop-up یا آف‌لاین شدن قابل قبول نیست.