VHDL

مدرس: عاطفه سلیمی

فهرست مطالب

- مرور کلی بر طراحی مدارهای مجتمع الکترونیکی
- زبان VHDL (ساختارها و دستورالعمل ها)
- طراحی مدار در سطوح مختلف
  - طراحی در سطح الگوریتم
  - طراحی در سطح رجیستر
  - طراحی در سطح گیت
- مباحث زمان بندی و تخصیص منابع
- بهینه سازی
- نرم افزار شبیه ساز Active-HDL
- نرم افزار Xilinx ISEو پیاده سازی سخت افزاری توسط FPGA

نحوه ارزیابی

- 1. میان ترم 30%
- 2.پایان ترم 50%
- 3. تکالیف 10%
- 4. پروژه 10-20%

مراجع درس

- 1. طراحی سیستم دیجیتالی با استفاده از VHDL، علیرضا فتاح.
- 2. VHDL: Analysis and Modeling of Digital Systems, Navabi
- 3. VHDL: Programming by Example. , Douglas Perry
- 4. RTL Hardware Design Using VHDL: Coding for Efficiency, Portability, and Scalability 1st Edition by Pong P. Chu
- 5. Circuit Design with VHDL, Volnei A. Pedroni
- میانترم 96/1/24

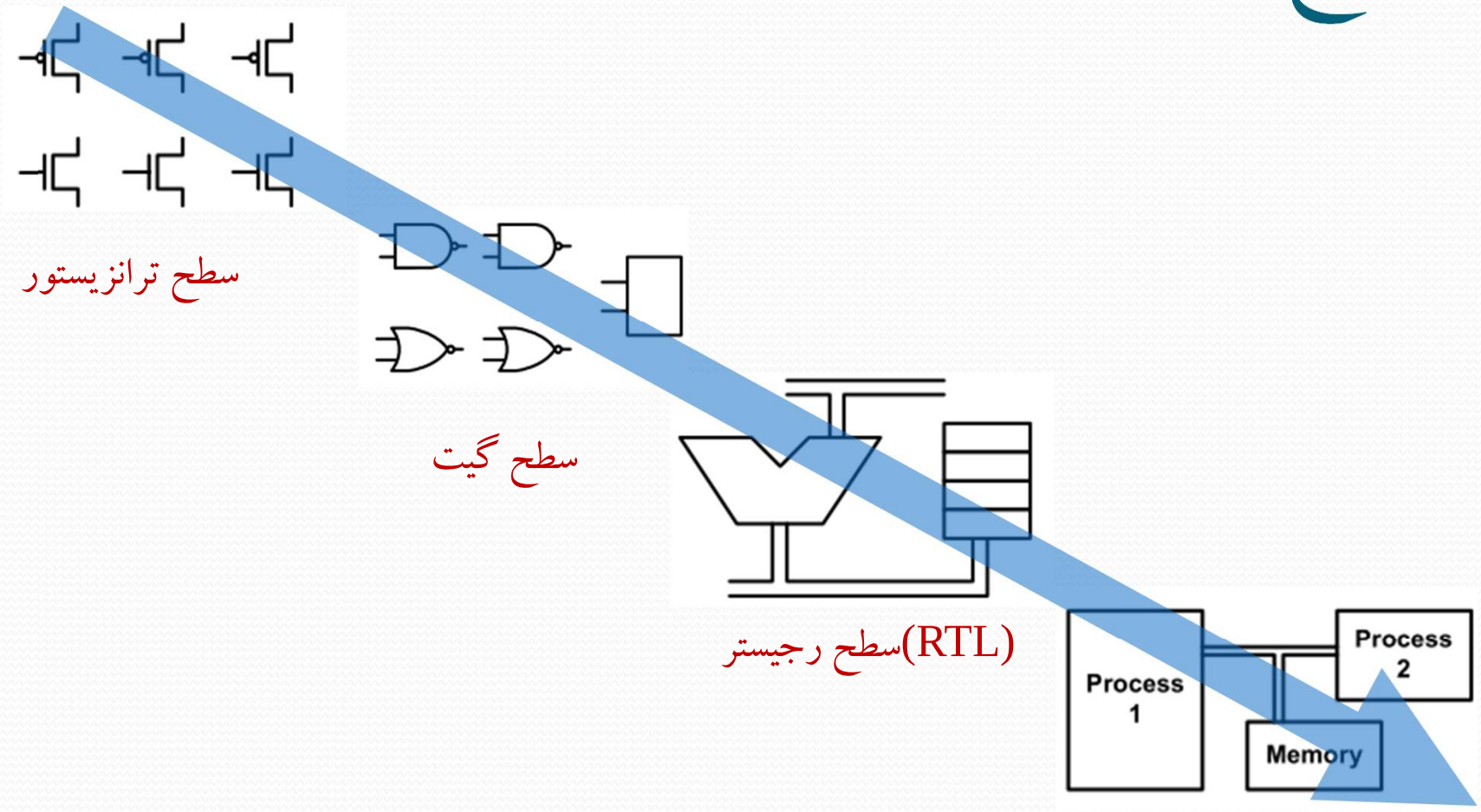
طراحی مدارهای الکترونیکی

- 1. دیجیتال
  - نویز کمتر
  - طراحی ساده تر
  - قابلیت برنامه ریزی مجدد
  - قابلیت پیکر بندی مجدد
- 2.آنالوگ
  - طراحی شامل در نظر گرفتن تعداد زیادی بده بستان است
  - طرح با تغییر شرایط مدار نیازمند به روز شدن است
- 3.دیجیتال - آنالوگ

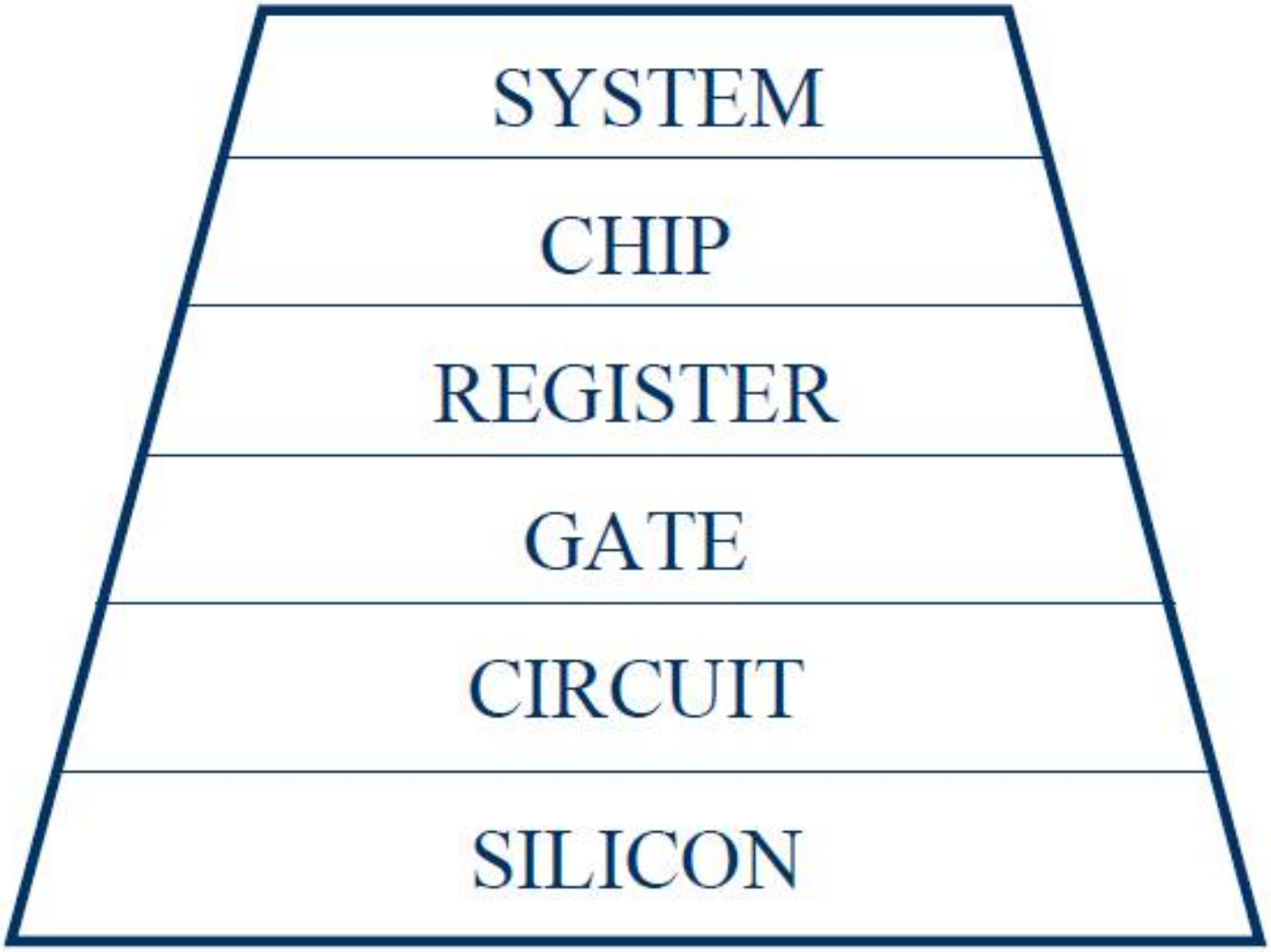
چهار مرحله اساسی در ساخت مدارهای مجتمع

- 1.طراحی (design)
- 2. ساخت (fabrication)
- 3. تست (testing)
- 4. بسته بندی (packaging)

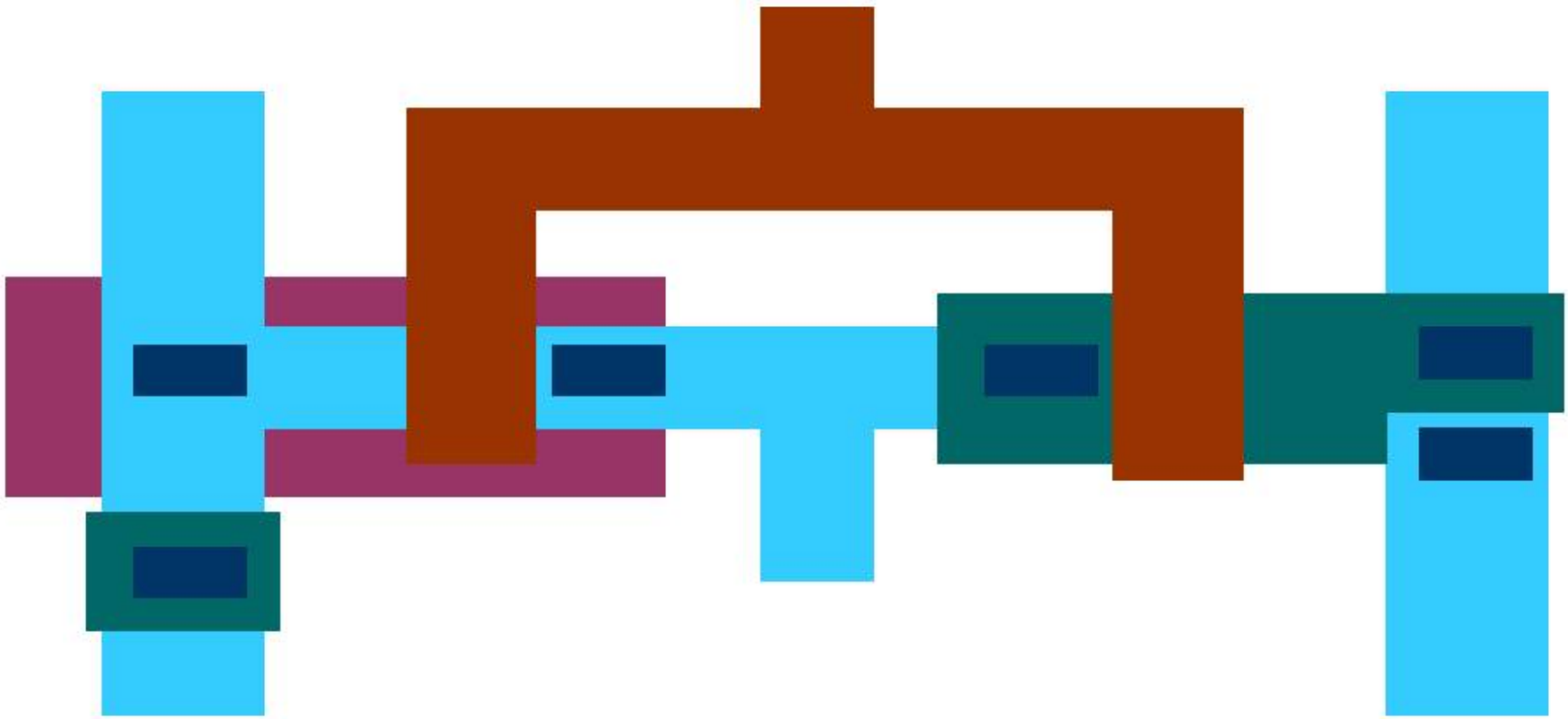
سطوح توصیف مدار



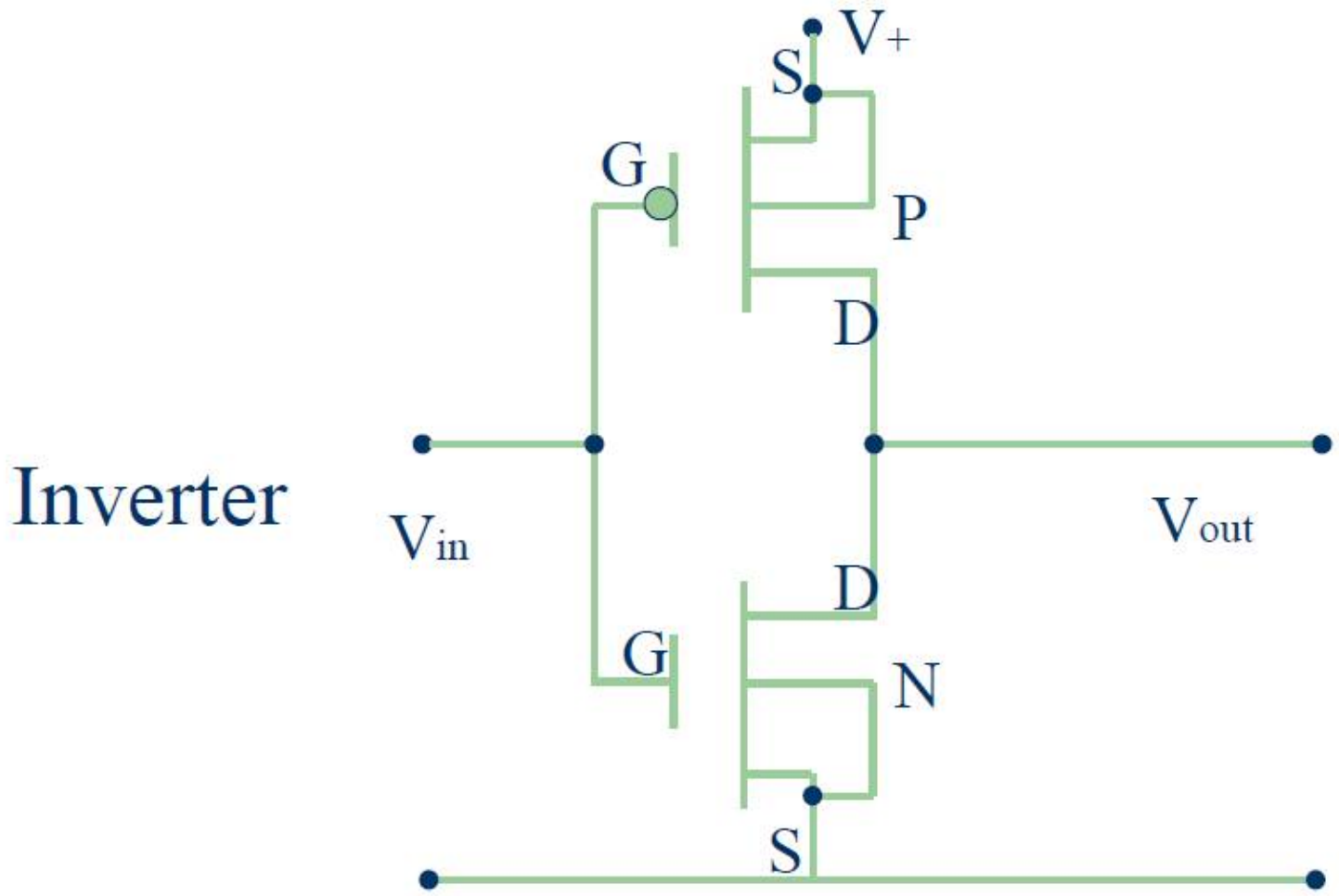
سطوح توصیف مدار



Silicon Level

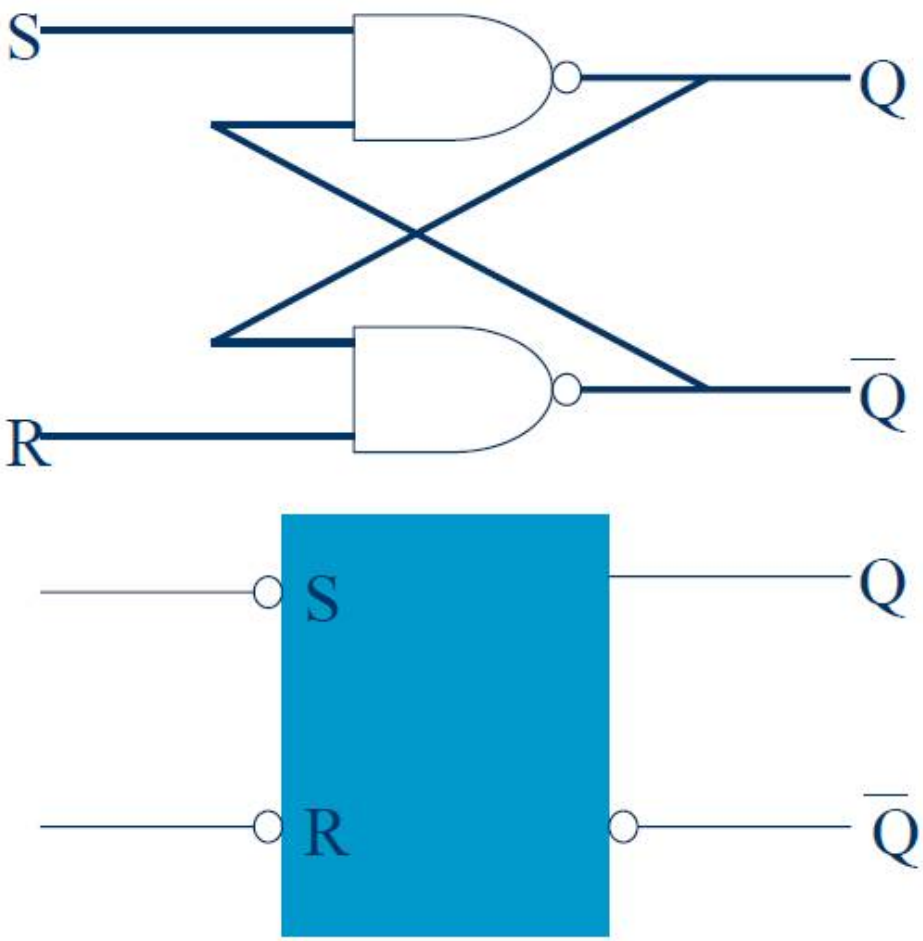


Circuit Level

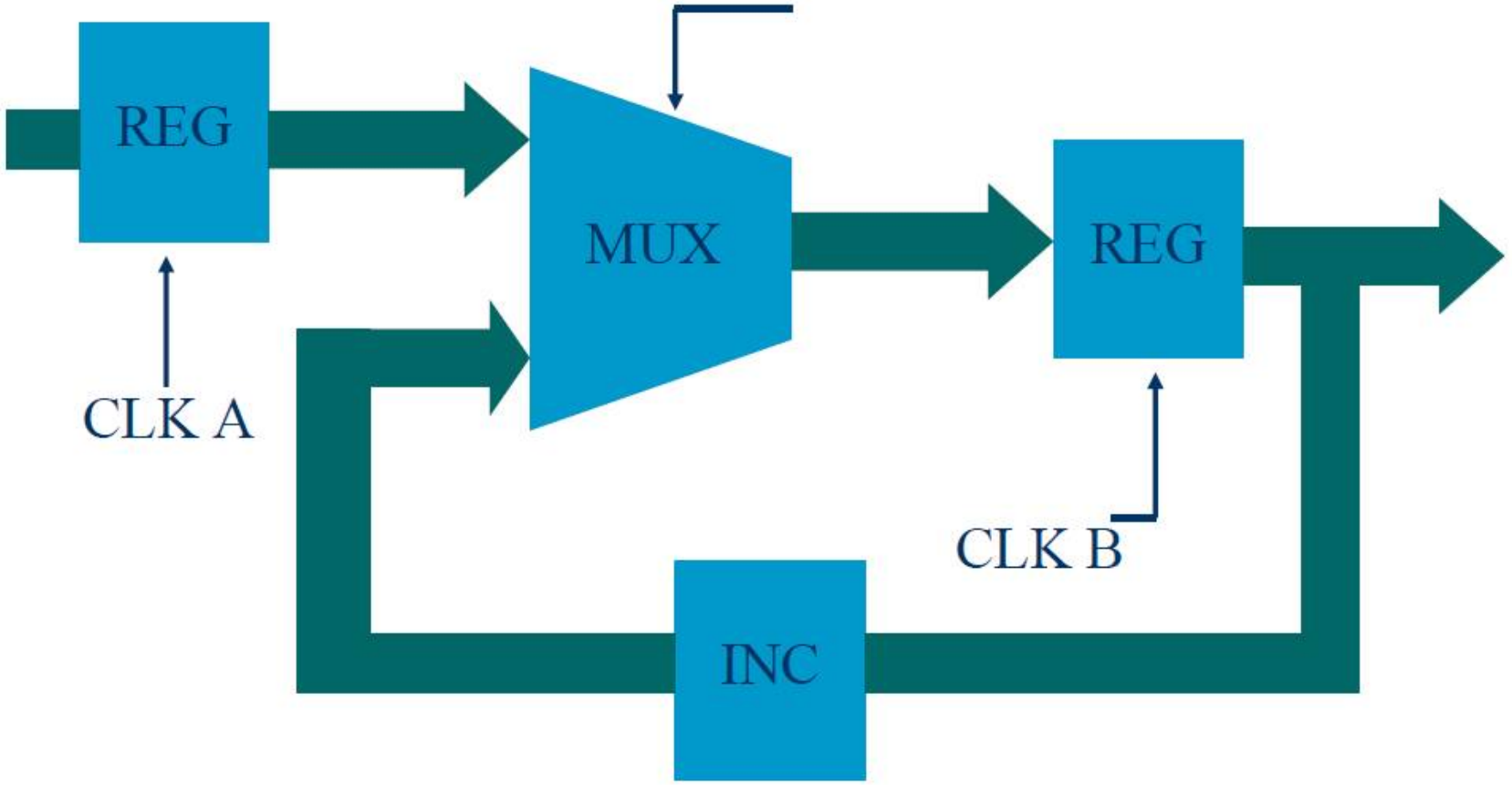


Gate Level

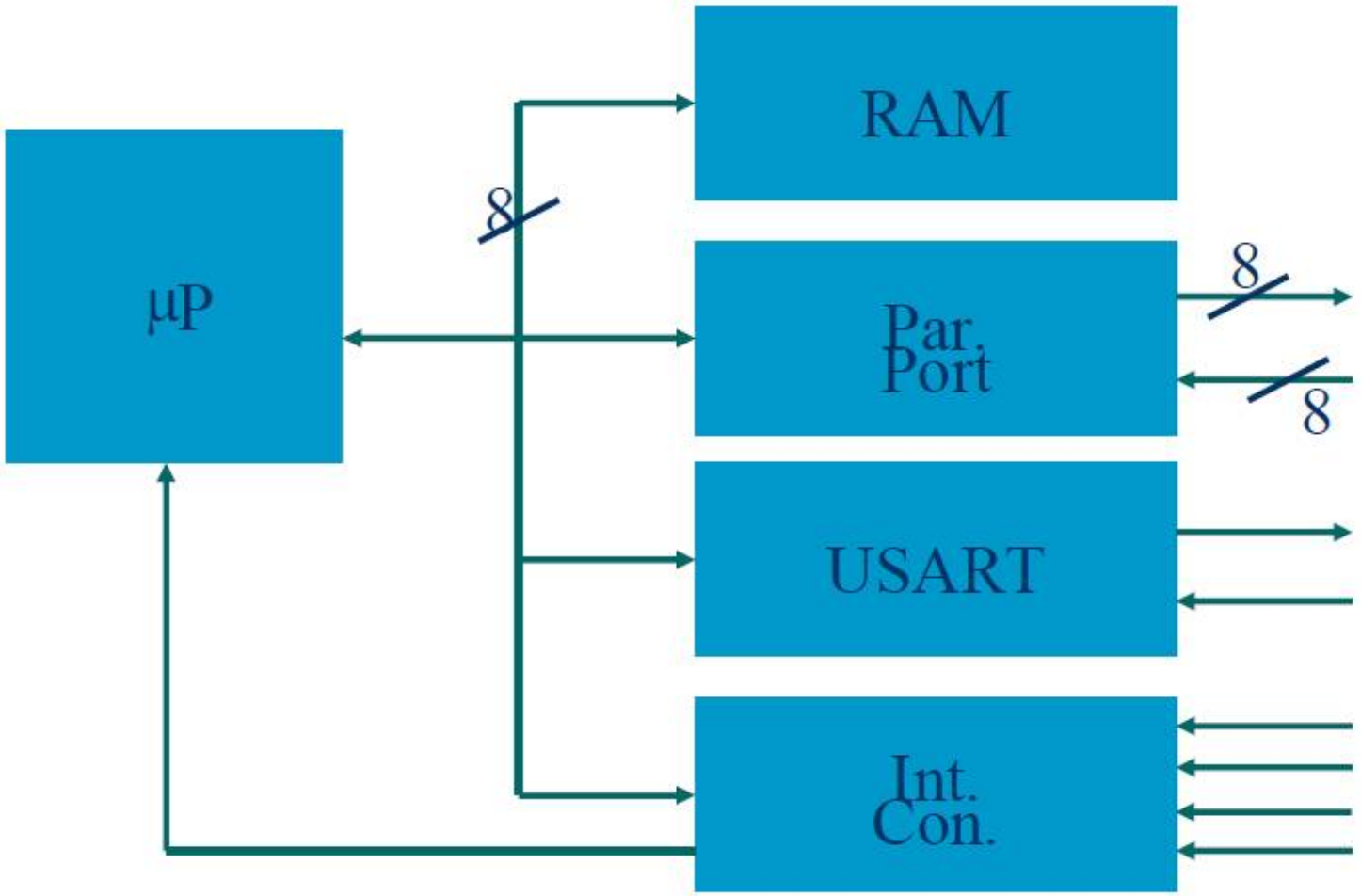
SR Flip Flop



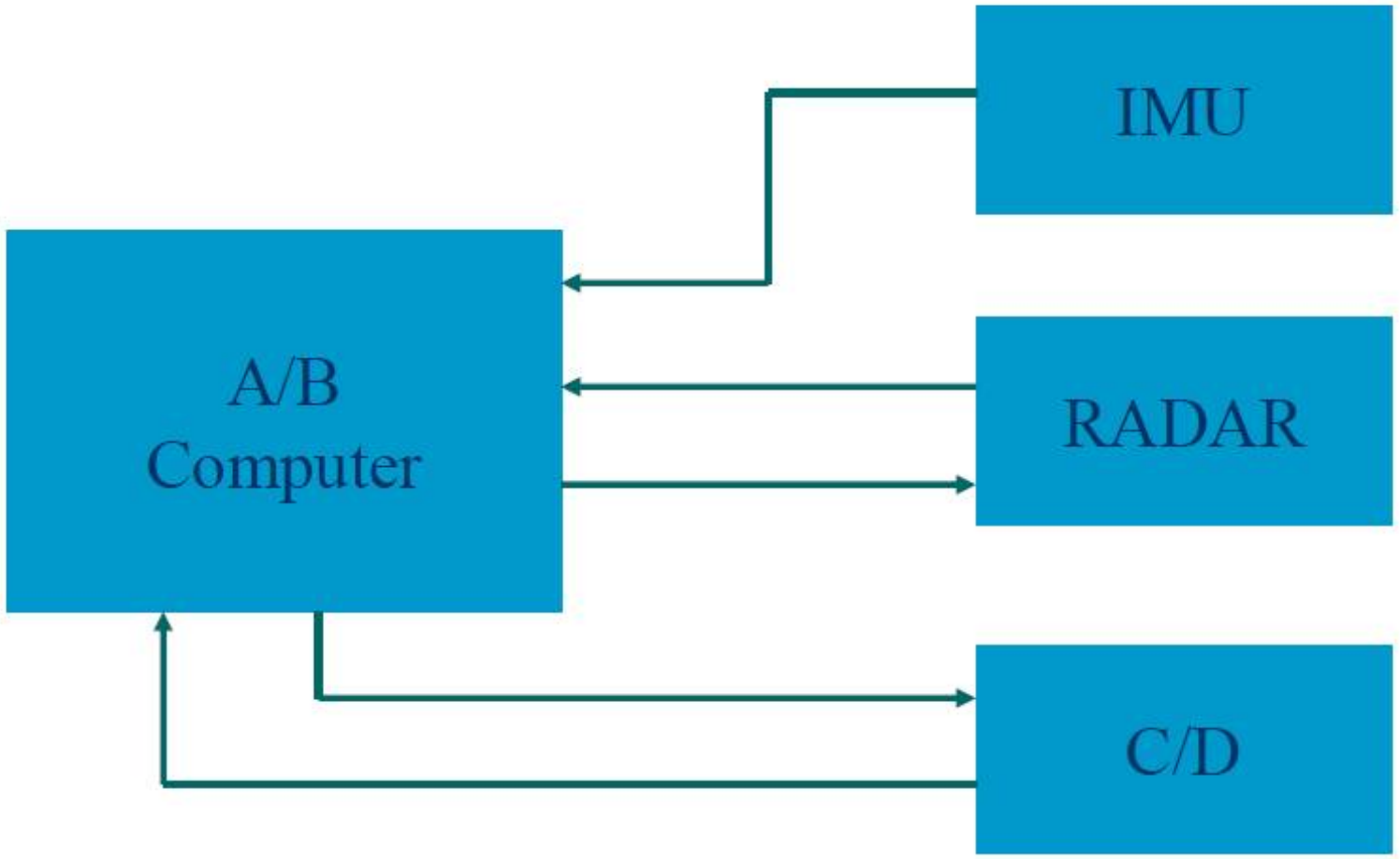
Register Level



Chip Level



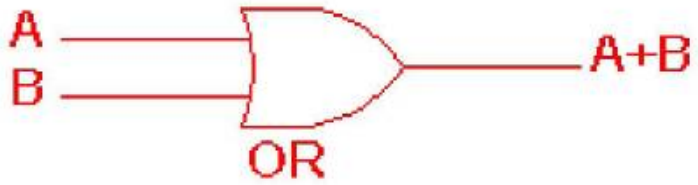
System Level



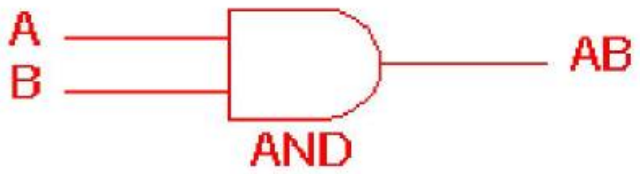
یادآوری جدول درستی و جدول کارنو

- جبر بولی یا جبر دو مقداره عملیات روی سیگنال های دو مقداره که فقط مقدار صفر یا یک دارند را گویند.
- برای عملگرهای مختلف در جبر بولی می توان جدول درستی رسم کرد که عبارت است از جدولی که تمام حالتهای ممکن متغیرها به همراه مقدار حاصل از اعمال عملگر مورد نظر روی آن را نشان دهد.
- سه عملگر اصلی در جبر دو مقداره عبارتند از
  - OR
  - AND
  - NOT

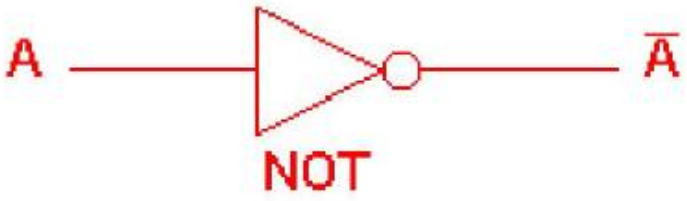
جدول درستی مربوط به سه عملگر اصلی



2 Input OR gate		
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1



2 Input AND gate		
A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

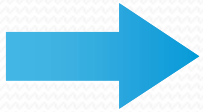


NOT gate	
A	$\bar{A}$
0	1
1	0

جدول کارنو

- جدول کارنو روشی گرافیکی برای ساده سازی یک عبارت جبر دو مقداره است.
- ساده سازی عبارت جبر دو مقداره به منظور طراحی ساده تر صورت می پذیرد.
- با توجه به شکل زیر برای یک جدول درستی دو متغیره جدول کارنویی معادل شکل زیر وجود دارد.
- برای 2 متغیر 4 حالت مختلف مینترم وجود دارد

x	y	minterm
0	0	x'y'
0	1	x'y
1	0	xy'
1	1	xy



X \ Y	0	1
0	x'y'	x'y
1	xy'	xy

ادامه:

- مینترم های چپ و راست به ترتیب شامل y’ و y هستند.
- مینترم های بالا و پایین به ترتیب شامل x’ و x هستند.

X \ Y	0	1
0	x’y’	x’y
1	xy’	xy

	Y’	Y
X’	x’y’	x’y
X	xy’	xy

- عبارت زیر برای یک جدول دو متغیره شامل جمع دو مینترم را در نظر بگیرید

$$x’y’ + x’y$$

- هر دوی این عبارات در سطر بالای جدول کارنو ظاهر می شوند که فقط شامل x’ است. بنابراین با ساده سازی با جدول کارنو مقدار عبارت بالا برابر x’ است.
- اگر از جبر دو مقدار برای ساده سازی عبارت بالا استفاده شود داریم

$x’y’ + x’y = x’(y’ + y)$ [خاصیت توزیع پذیری]

$= x’ \bullet 1$ [$y + y’ = 1$]

$= x’$ [$x \bullet 1 = x$]

مثالهای دیگر از ساده سازی با جدول دو متغیره

- مثال دیگر عبارت x’y + xy است. هر دو مینترم در سمت راست جدول کارنو هستند که شامل y است بنابراین مقدار عبارت برابر با y است.

		Y
	x’y’	x’y
X	xy’	xy

- مثال دیگر عبارت x’y’ + x’y + xy است.
  - مینترم های x’y’ + x’y در سطر بالای جدول کارنو هستند که شامل x’ است
  - مینترم های x’y + xy در سمت راست جدول کارنو هستند که شامل y است.
  - بنابراین حاصل ساده سازی عبارت بالا به صورت x’ + y خواهد بود

		Y
	x’y’	x’y
X	xy’	xy

جدول کارنوی 3 متغیره

- نحوه قرار گیری مینترم ها در جدول 3 متغیره به صورت زیر است.

X	YZ = 00	YZ = 01	YZ = 11	YZ = 10
0	x'y'z'	x'y'z	x'yz	x'yz'
1	xy'z'	xy'z	xyz	xyz'

X	YZ = 00	YZ = 01	YZ = 11	YZ = 10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

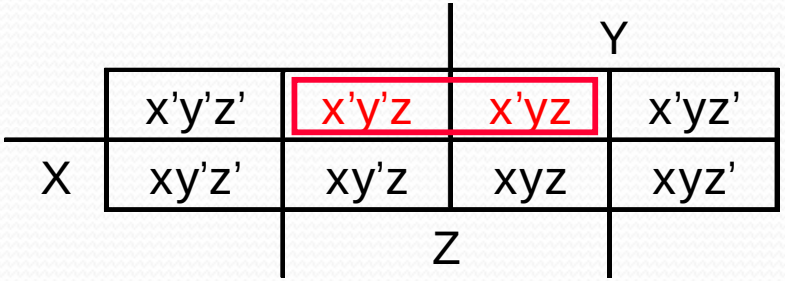
- راه دیگر برای برچسب گذاری جدول کارنو به صورت زیر است

X \ Y,Z	Y,Z = 00	Y,Z = 01	Y,Z = 11	Y,Z = 10
X = 0	x'y'z'	x'y'z	x'yz	x'yz'
X = 1	xy'z'	xy'z	xyz	xyz'

X \ Y,Z	Y,Z = 00	Y,Z = 01	Y,Z = 11	Y,Z = 10
X = 0	m_0	m_1	m_3	m_2
X = 1	m_4	m_5	m_7	m_6

ادامه

- با این نحوه قرار گیری مینترم ها هر ترکیبی از مربع های 2 4 یا 8 تایی از مینترم ها قابل ساده سازی است.



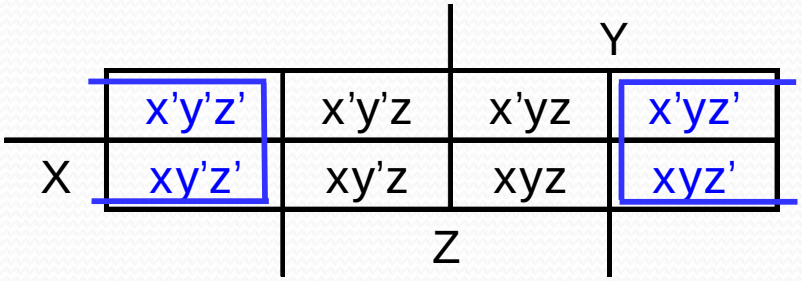
$x'y'z + x'yz$

$= x'z(y' + y)$

$= x'z \bullet 1$

$= x'z$

- در شکل زیر 4 مینترم از ستون سمت راست و چپ جدول قابل ساده سازی هستند.



$x'y'z' + xy'z' + x'yz' + xyz'$

$= z'(x'y' + xy' + x'y + xy)$

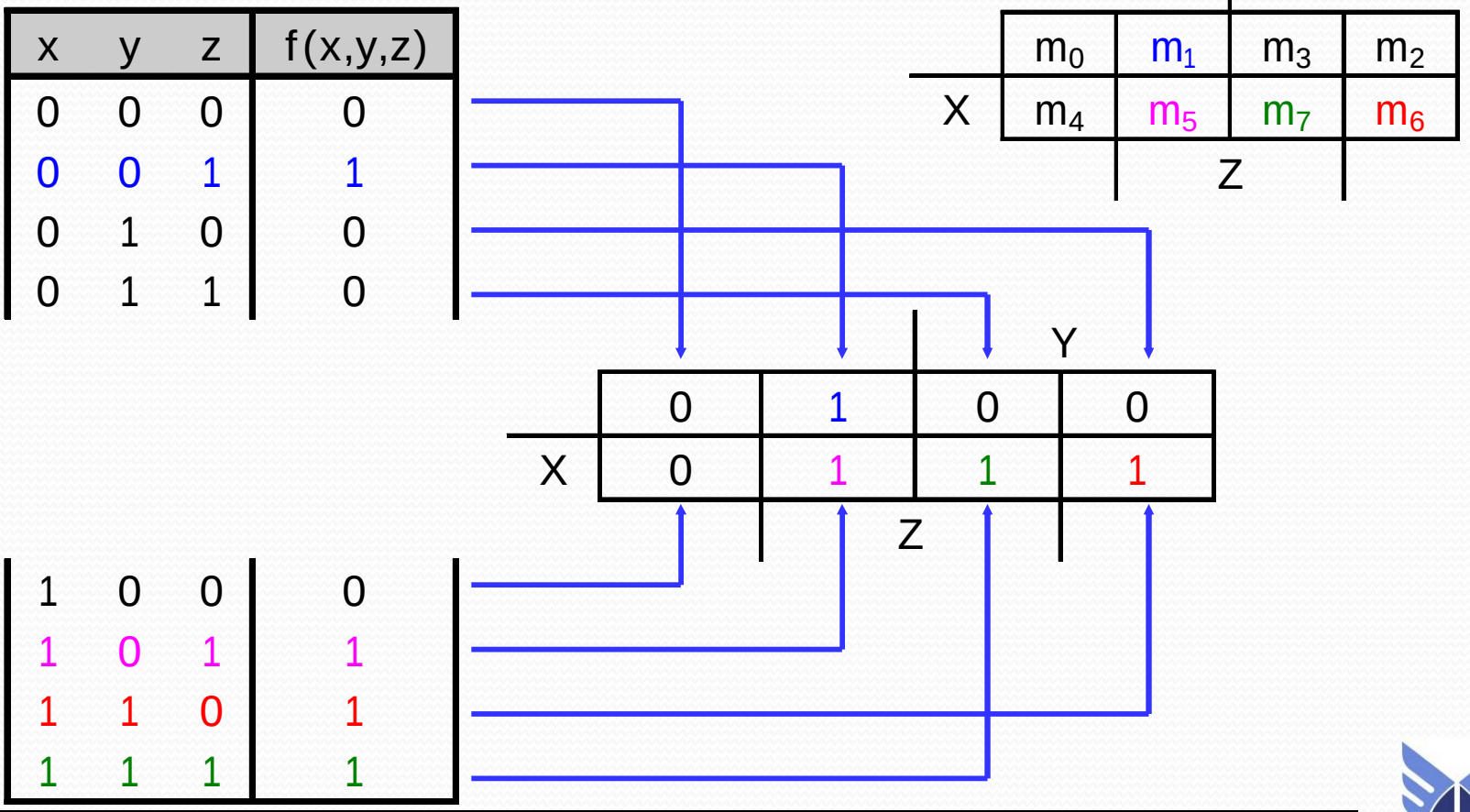
$= z'(y'(x' + x) + y(x' + x))$

$= z'(y'+y)$

$= z'$

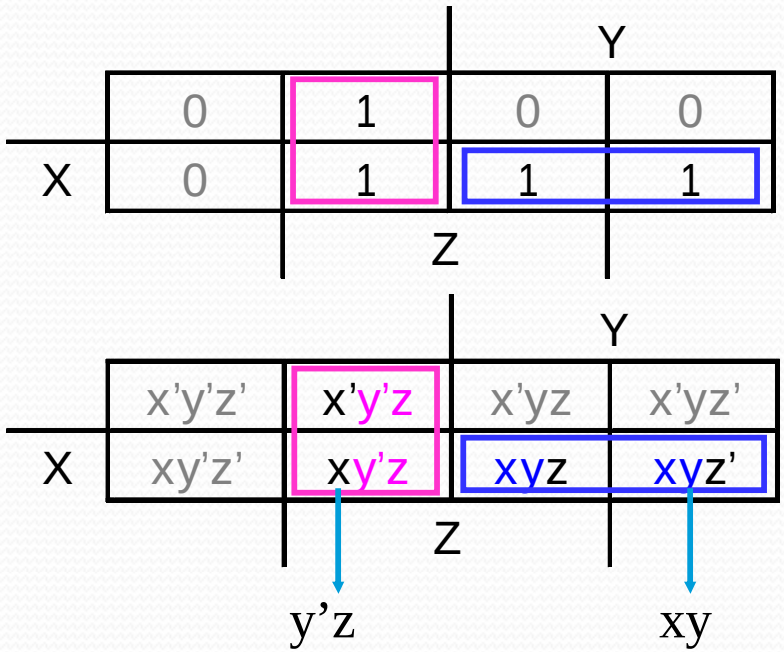
پر کردن جدول کارنو با استفاده از جدول درستی

- می توان به طور مستقیم جدول کارنو را با استفاده از جدول درستی پر کرد.
- در اینصورت خروجی هر سطر i از جدول درستی به مربع mi از جدول کارنو منتقل می شود.



نوشتن عبارات مینترم ها از روی جدول کارنو

- سخت ترین قسمت در ساده سازی با جدول کارنو گروه بندی 1 های جدول است.
- گروههای 1، 2 ، 4 و 8 تایی از 1 های جدول را دسته بندی کرده و دور آنها مستطیل کشیده و مینترم معادل آنها را بنویسید.



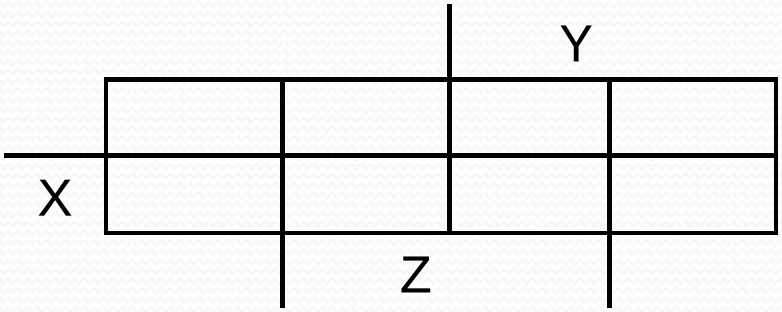
$F(x,y,z)= y'z + xy$

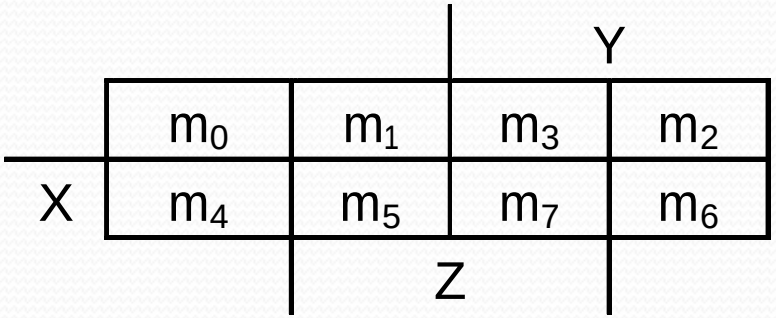
به منظور داشتن ساده ترین پاسخ

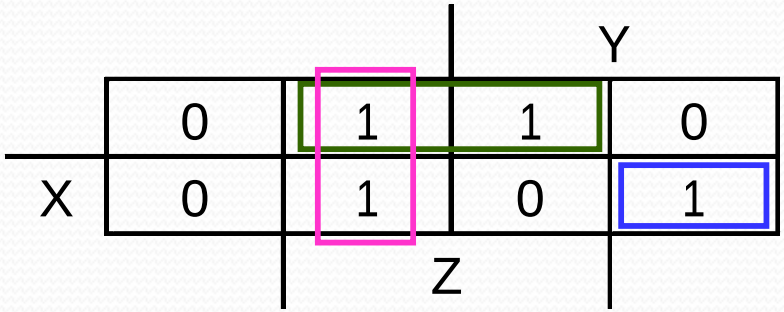
- باید مستطیل هایی که دور یک ها کشیده می شوند تعدادشان مینمم شود تا تعداد مینترم های نهایی حداقل شوند.
- مستطیل ها باید تا حد امکان ماکزیمم 1 های مجاور را در برگیرند مثلا 4 تا 1 مجاور بهتر است یک مستطیل 4 تایی تشکیل دهند تا دو تا دو تایی تا تعداد مینترم های حاصل مینمم شوند.
- مستطیل ها می توانند همپوشانی داشته باشند تا بزرگتر شوند.

تمرین: ساده سازی با جدول کارنو

- ساده سازی جمع مینترم های m1+m3+m5+m6



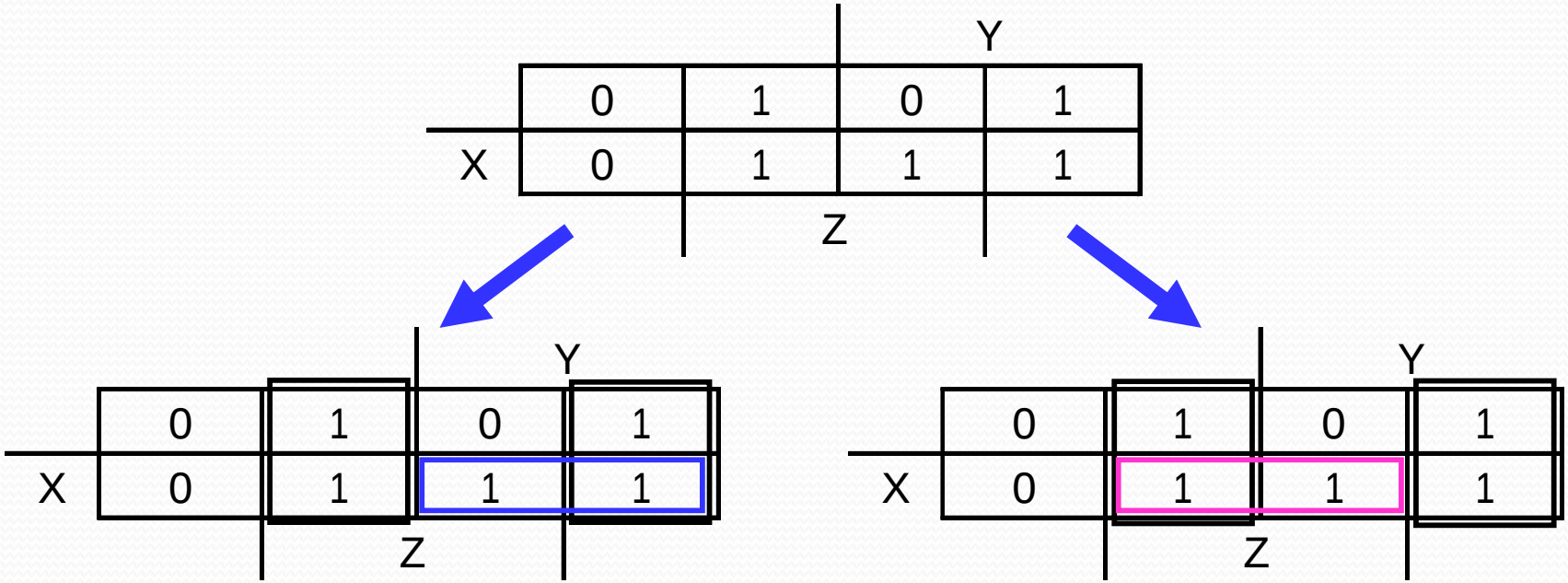




- عبارت ساده شده نهایی x’z + y’z + xyz’

جوابهای ممکن با جدول کارنو

- ساده سازی با جدول کارنو لزوما جواب یکتایی نمی دهد.



y’z + yz’ + xy

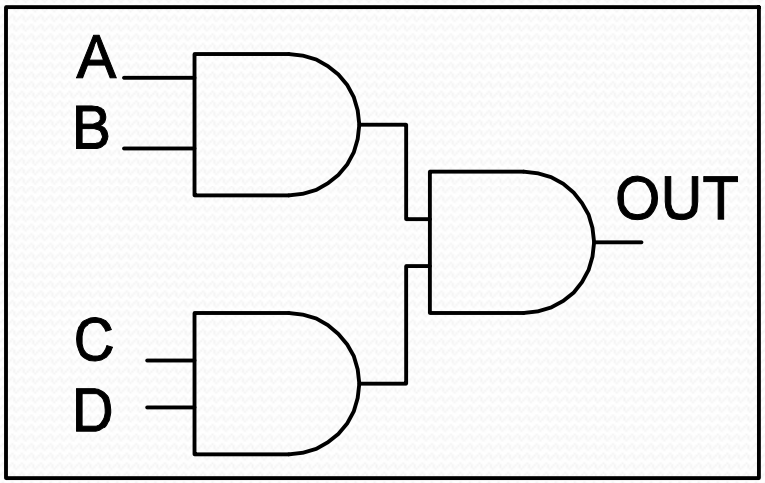
y’z + yz’ + xz

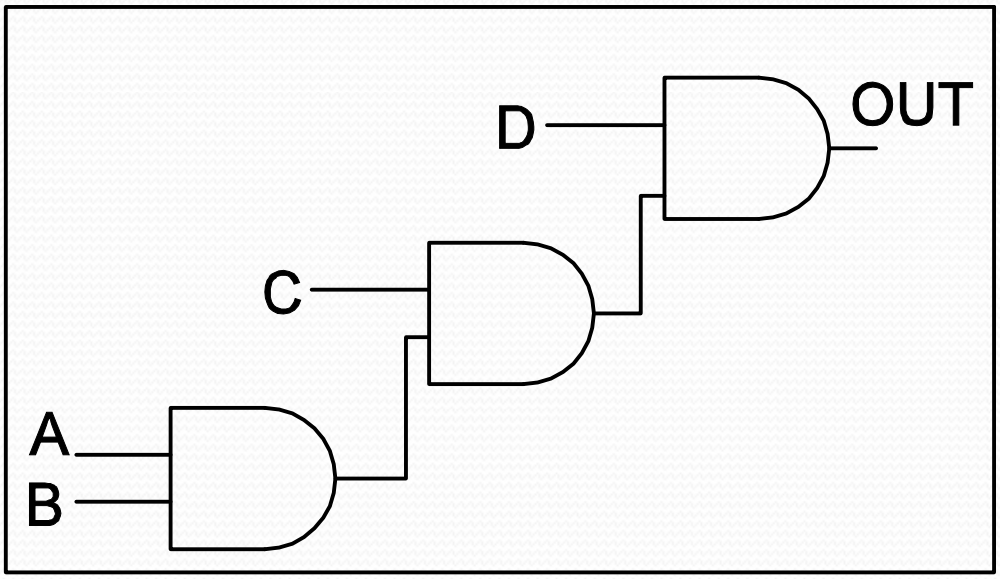
معیارهای ارزیابی طرح

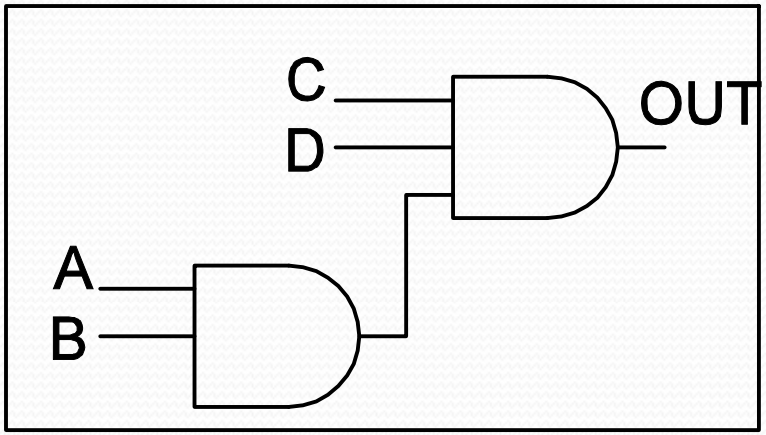
- 1. توان مصرفی
- 2. سطح اشغالی
- 3. تاخیر و سرعت
- 4. قابلیت تست
- 5. قیمت

مثال :پیاده سازی F=A.B.C.D

- طرح بهینه کدام است؟

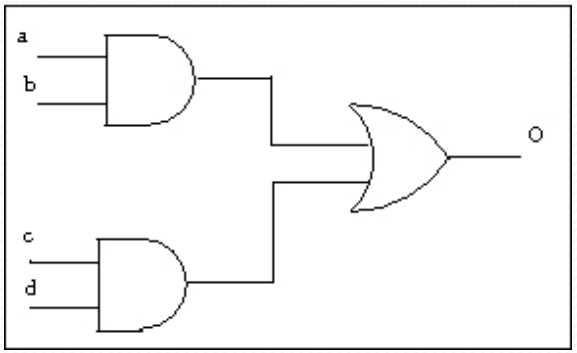


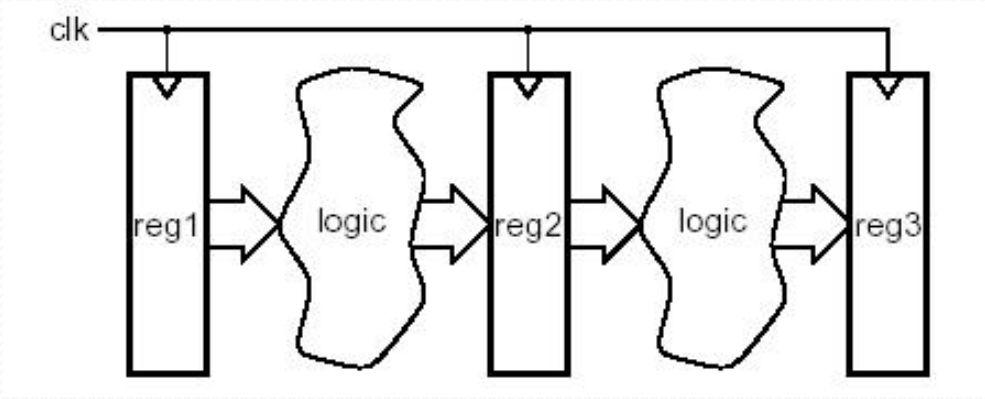




مدارهای ترتیبی و ترکیبی

- مدار ترکیبی فاقد حافظه برای ذخیره سازی اطلاعات (مربوط به ورودی های) قبلی است.
- مدار ترتیبی دارای حافظه است.





روند طراحی

- 1. با استفاده از روش های مرسوم شماتیک مداری ، جدول کارنو و ...

  عدم کارامدی در شرایطی که مدار پیچیده است و ترانزیستورهای مدار بسیار زیاد است.

- 2. استفاده از ابزارهای خودکار کردن طراحی

(Electronic Design Automation) EDA Tools

به وجود آمدن زبان های توصیف سخت افزاری (HDL)

مثال:پیاده سازی ضرب کننده 16*16

- دو ورودی 16 بیتی و یک خروجی 32 بیتی در مجموع 64 ورودی و خروجی
- نیاز به 16*16 گیت AND برای انجام ضرب بیت به بیت
- نیاز به تعداد زیادی مدار Full Adder برای جمع حاصلضرب های بیتی

- در چنین طرحی، طراحی بدون استفاده از ابزار CAD بسیار پیچیده و زمانبر خواهد بود و خطای زیادی خواهد داشت.

کد VHDL

- Entity MULT is

  port (A,B: in std _ logic(15 down to 0);

  Y:out std_logic (31 down to 0);

end MULT;

architecture Behav of MULT is

begin

Y <= A* B;

End Behav ;

مزایای استفاده از زبان HDL

- سرعت بالای طراحی
- خطای کم
- قابل دنبال کردن و خطایابی برنامه نوشته شده با شبیه سازی
- قابلیت استفاده از سخت افزار طراحی شده به صورت یک بسته در کنار سایر سخت افزارها.

- با استفاده از مثال ذکر شده توانایی HDL در طراحی سخت افزار تا حد زیادی قابل درک خواهد بود.

انواع HDL

- دو زبان استاندارد توصیف سخت افزاری پشتیبانی شده توسط IEEE وجود دارد:
- VHDL:

V : Very High Speed Integrated Circuit

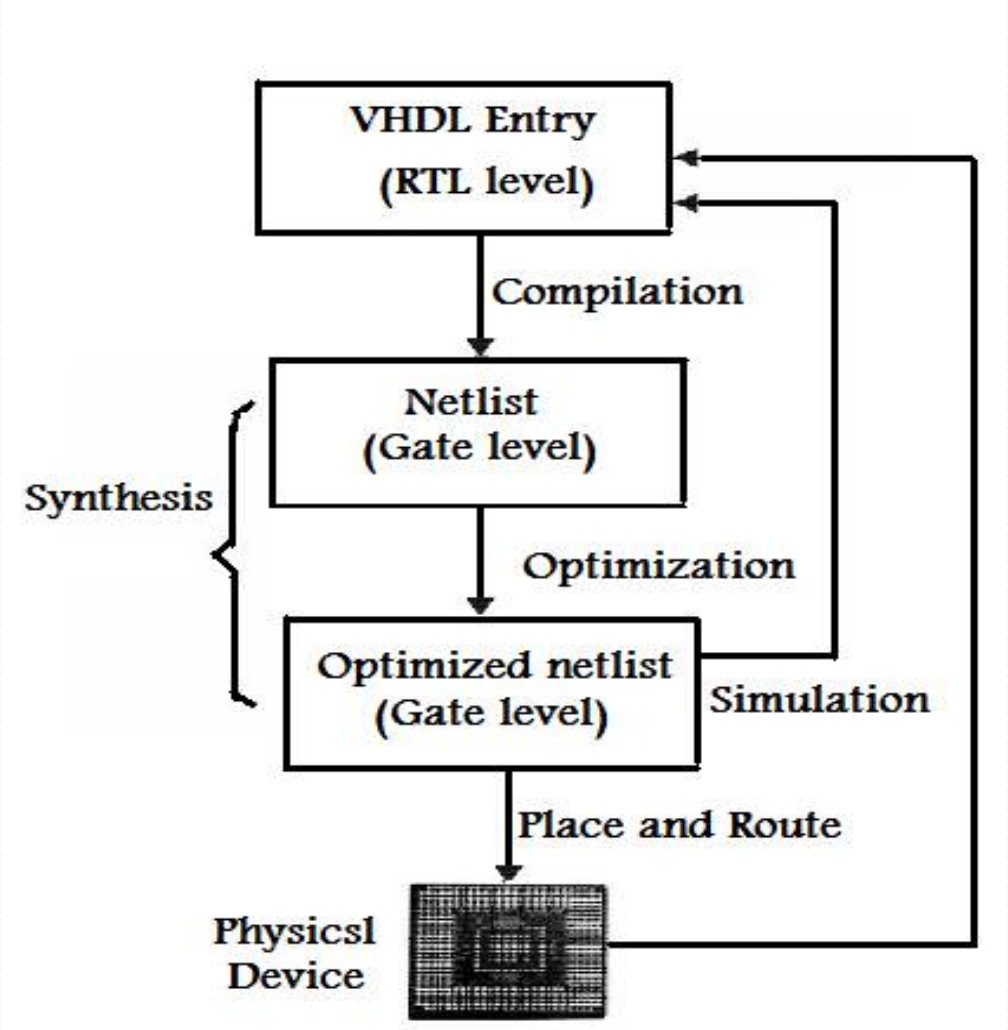
H : Hardware

D : Description

L: Language

- Verilog

مراحل طراحی با HDL



عبارات استفاده شده در طراحی با HDL

- Synthesis : تبدیل کد نوشته شده به مدار

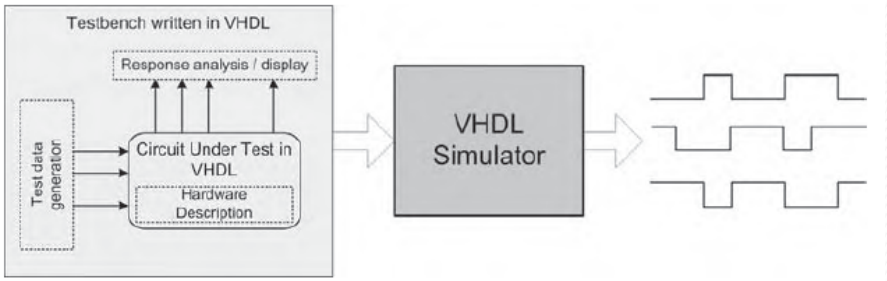
- Simulation:

  اعمال یک سری ورودی به طرح و تست عملکرد آن

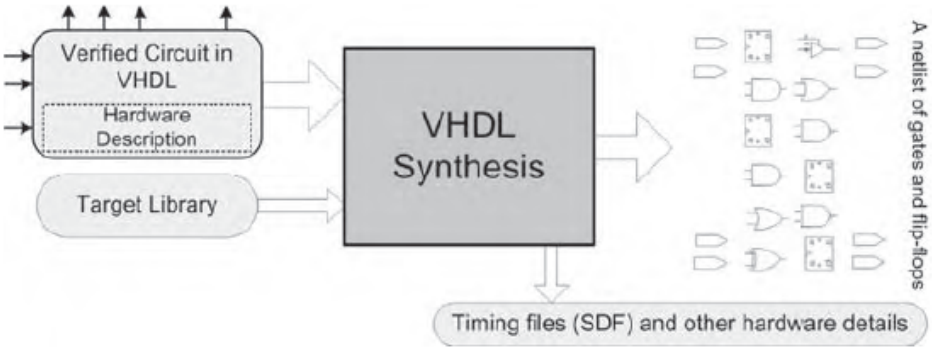
- Testbench:

  فراهم کردن محیطی برای اعمال ورودی و تست عملکرد سیستم

مراحل مختلف طراحی با جزئیات بیشتر



- 1.شبیه سازی طرح قبل از سنتز



- 2. سنتز طرح

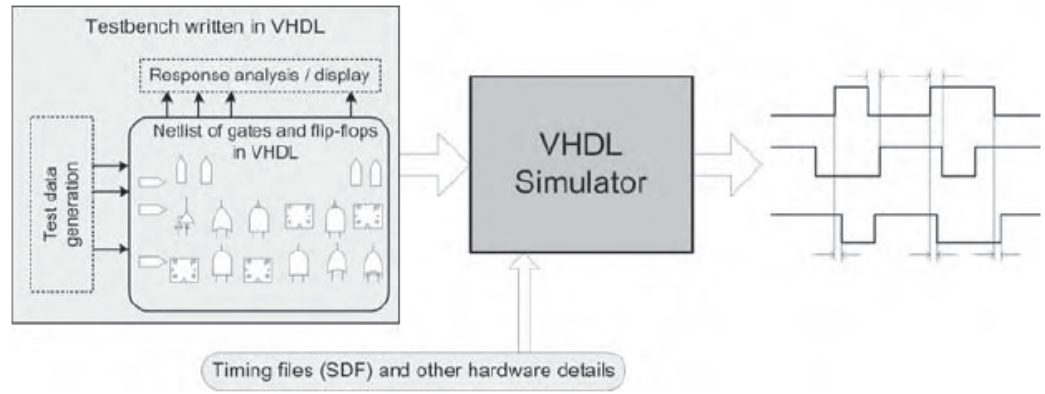
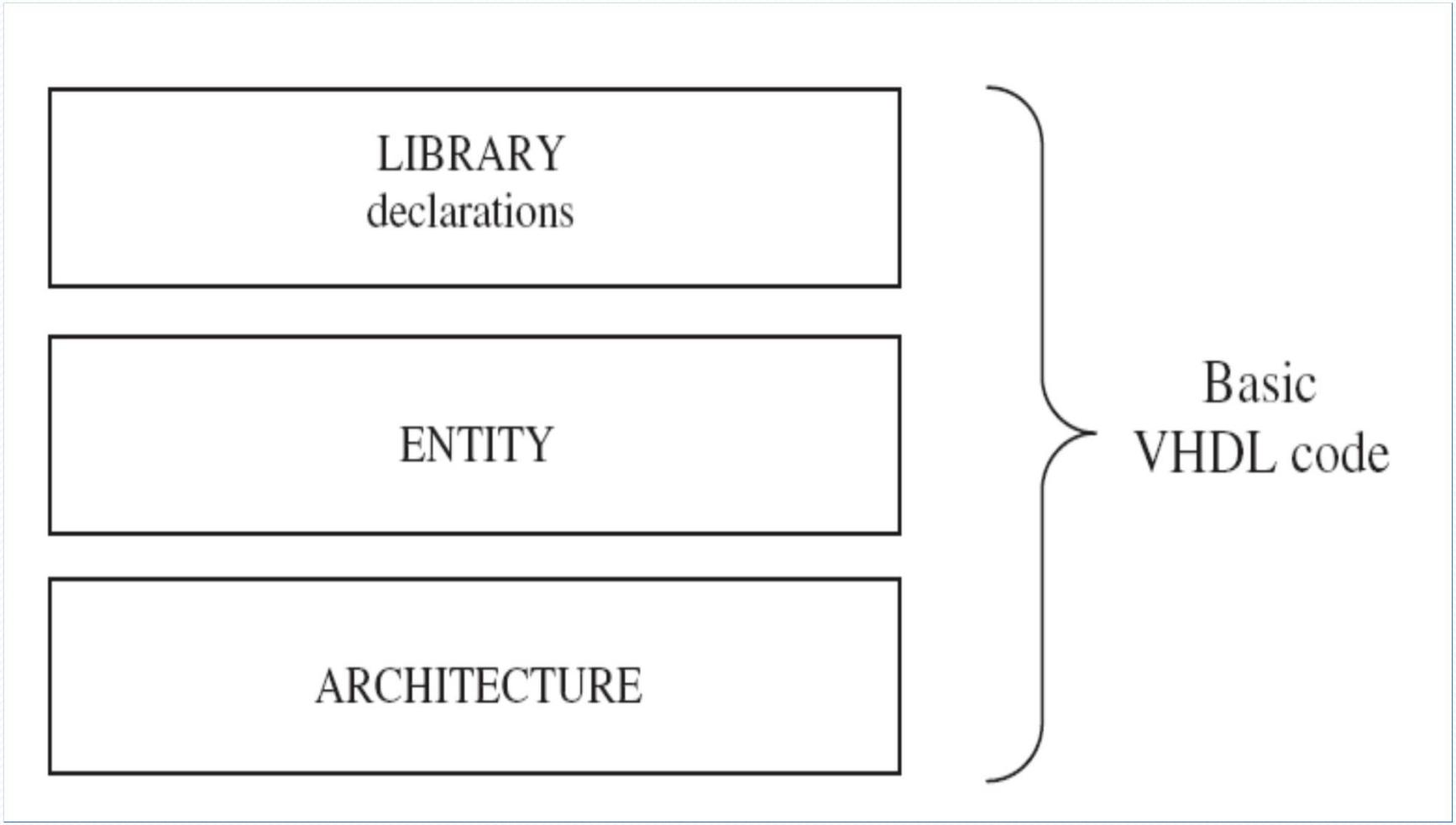


Figure 2.3 Post-synthesis Simulation in VHDL

- 3. شبیه سازی بعد از سنتز

برنامه نویسی به زبان VHDL

قسمت های اصلی کد VHDL



هر کد VHDL سه قسمت اصلی دارد:

- LIBRARY declarations:

  شامل کتابخانه هایی که در طرح استفاده می شوند.

- ENTITY:

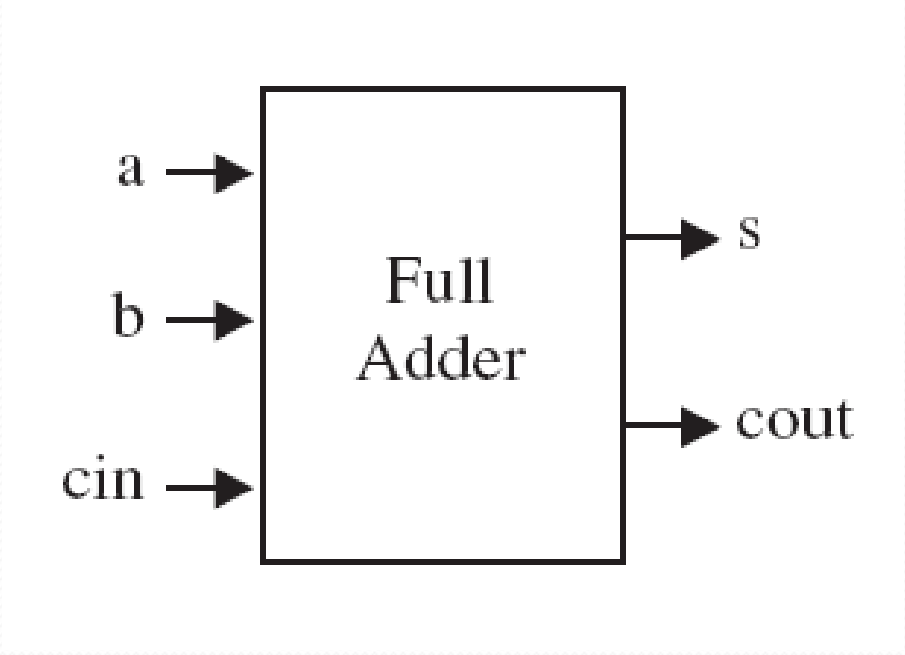
  بیان اتصالات خارجی طرح (مشخص کننده پین های ورودی-خروجی طرح).

- ARCHITECTURE:

  شامل یک کد VHDL که توصیفی از نحوه رفتار مدار می کند.

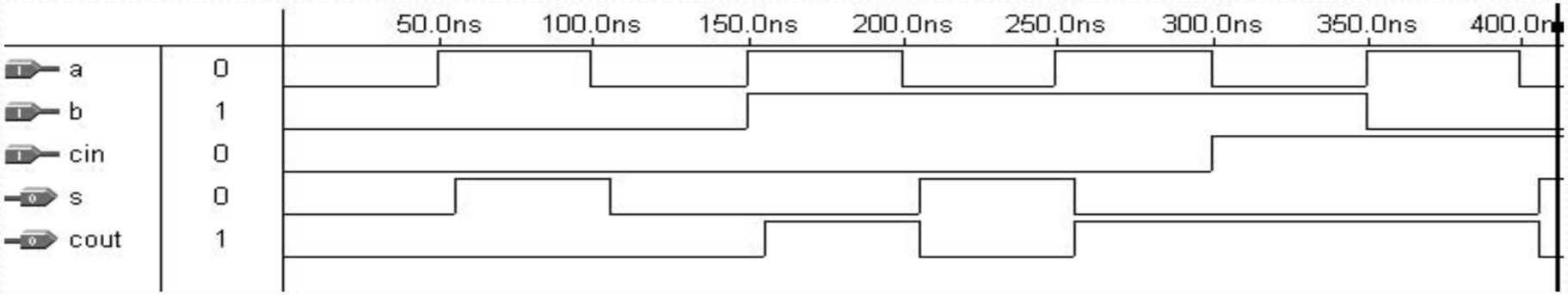
مثال از کد VHDL

شماتیک Full-adder



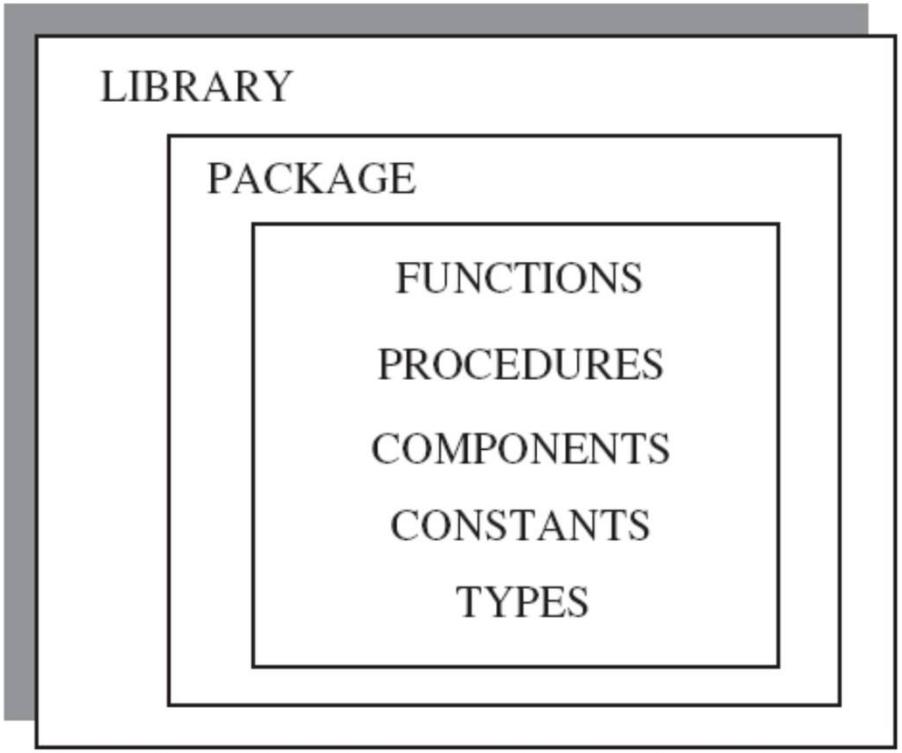
کد VHDL مربوط به مدار Full-adder

```
ENTITY full_adder IS
PORT (a, b, cin: IN BIT;
      s, cout: OUT BIT);
END full_adder;
-------------------------------------
ARCHITECTURE dataflow OF full_adder IS
BEGIN
   s <= a XOR b XOR cin;
   cout <= (a AND b) OR (a AND cin) OR
           (b AND cin);
END dataflow;
```



نتایج شبیه سازی کد VHDL مربوط به طرح

LIBRARY



- کتابخانه مجموعه ای از کدهایی است که معمولا در طرح استفاده می شوند.
- کدهای قرار داده شده در کتابخانه را می توان در طرح های مختلف استفاده کرد بدون آنکه نیاز به باز نویسی وجود داشته باشد.

- نحوه تعریف کتابخانه:

- LIBRARY library_name ;
- USE library_name.package_name.package_parts ;

ادامه

- معمولا حداقل سه بسته (package) از 3 کتابخانه متفاوت در یک طرح نیاز است.
- ieee.std_logic_1164(from the ieee library),
- standard (from the std library) , and
- work (work library)

نحوه تعریف بسته های مختلف

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

LIBRARY std;
USE std.standard.all;

LIBRARY work;
USE work.all;
```

- کتابخانه های std و work بصورت پیش فرض تعریف شده اند.
- فقط کتابخانه ieee باید تعریف شود.

ادامه

- بسته std_logic_1164 از کتابخانه ieee سیستم منطقی چند سطحی را تعریف می کند. این استاندارد یک نوع داده با 9 مقدار را مشخص می کند.

- کتابخانه workمکانی است که طرح مورد نظر و کل فایل هایی که توسط کامپایلر و سیمولاتور ساخته می شود در آنجا ذخیره می شود.

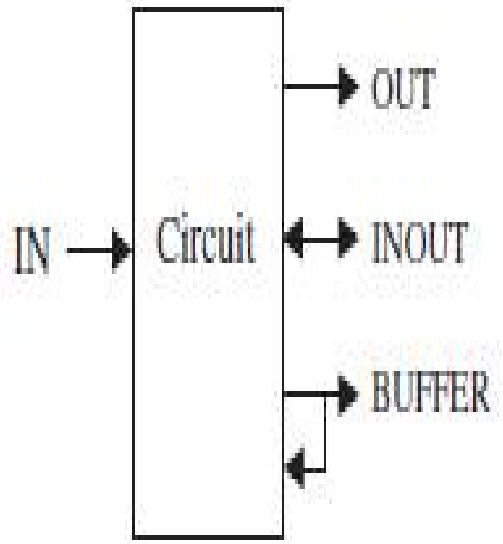
بسته های مختلف کتابخانه ieee

std_logic_1164	شامل متغیرهای STD_LOGIC(8 سطحی) و STD_ULOGIC(9 سطحی)
std_logic_arith	تعریف نوع داده SIGNEDو UNSIGNED و عملگرهای ریاضی و منطقی مربوط توابع تبدیل نوع داده مثل conv_integer(p)، conv_unsigned(p,b)، conv_signed(p,b) و conv_std_logic_vector(p,b)
std_logic_signed	شامل توابع برای کار با نوع داده علامتدار
std_logic_signed	شامل توابع برای کار با نوع داده بدون علامت

ENTITY

- ENTITY برای تعریف پین های (PORTS) ورودی- خروجی طرح مورد استفاده قرار می گیرد.
- نحوه تعریف ENTITY به صورت زیر است:

- ENTITY entity_name IS
  PORT (
  port_ name : signal_mode signal_type ;
  port_ name : signal_mode signal_type ;
- ...);
- END entity_name ;

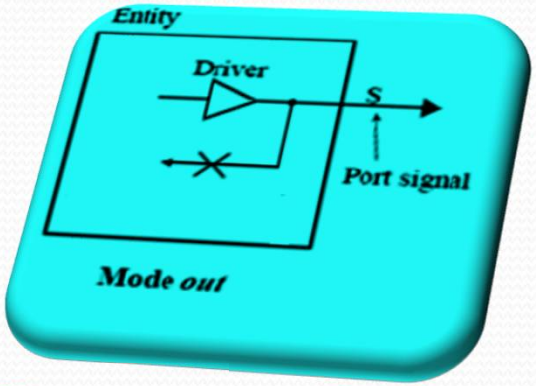


ادامه

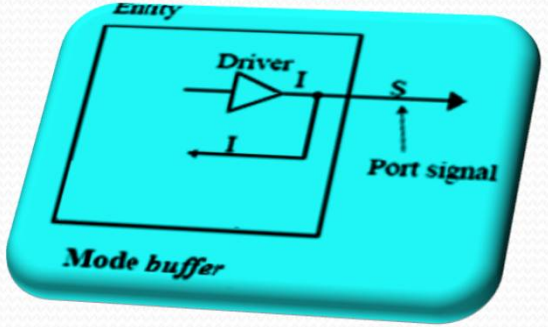
- مد سیگنال می تواند IN،OUT ، INOUT و یا BUFFER باشد.
- IN و OUT پین های یک جهته به ترتیب ورودی و خروجی هستند، در حالی که INOUT دو جهته ورودی-خروجی است.
- BUFFER این سیگنال در داخل طرح، هم خوانده و هم مقدار دهی می شود. بر خلاف پورت inout چیزی از بیرون طرح، داخل آن نمی توان ریخت.
- Entity هر نامی به غیر از کلمات رزرو شده VHDL را می تواند داشته باشد.

مد سیگنال

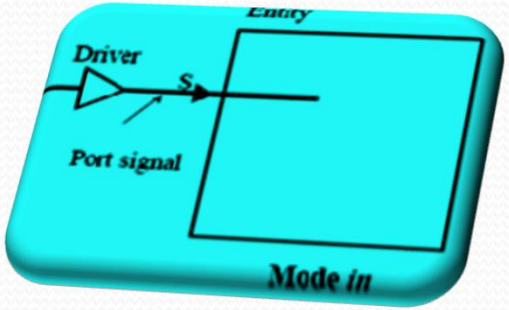
***out*: signal cannot be read inside the entity**



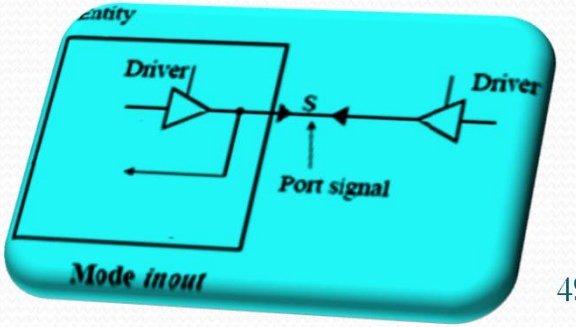
***buffer*: signal can read inside the entity**



***in*: input signal**



***inout*: signal can be read inside the entity**



تبدیل بین فرمت های مختلف