بسمه تعالی

طراحی کامپیوتری سیستمهای دیجیتال

Computer-Aided Digital System Design

مدرس

فاطمه دارائی

f_daraei@semnan.ac.ir

https://fdaraei.profile.semnan.ac.ir

طراحی کامپیوتری
سیستم‌های دیجیتال
تعاریف
اساسی
انواع
سخت‌افزارهای
برنامه‌پذیر
زبان توصیف
سخت افزار
VHDL
آشنایی با ابزارهای
شبیه‌سازی و سنتز

مراجع درسی

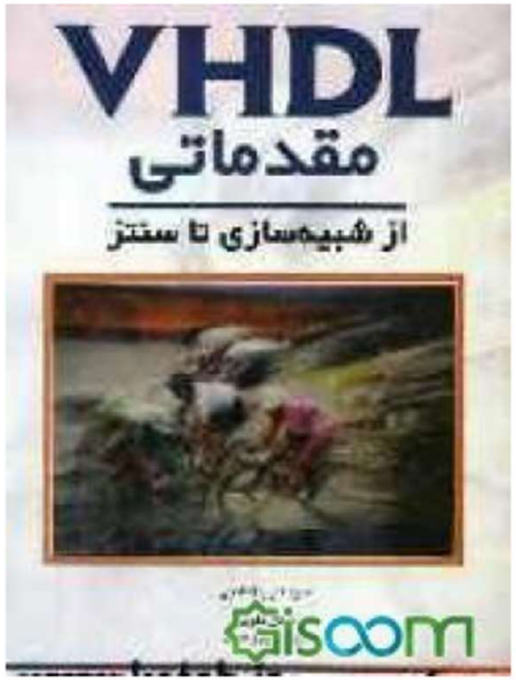
طراحی خودکار مدارهای دیجیتال با

FPGA و زبان توصیف سخت افزار VHDL

نویسنده: دکتر حسن سیدرضی

انتشارات ناقوس

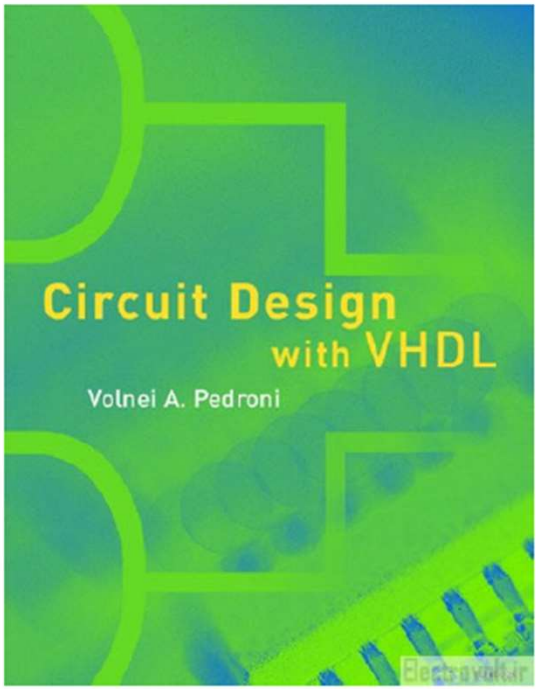
VHDL مقدماتی: از شبیه سازی تا سنتز



نویسنده: سوداکار یالامانچی

مترجم: عبدالکریم زعمری، فرناز نکویی

ناشر: نص



Circuit Design with VHDL

Volnei A. Pedroni

MIT Press

ارزشیابی

میان ترم ۶ نمره

پایان ترم ۱۰ نمره

تکلیف ۲ نمره

پروژه ۳ نمره

مجموع از ۲۱ نمره

دانشگاه سمنان
Semnan University
پردیس فرزانگان

سرفصل دروس

فصل اول

مقدمات و مفاهیم اولیه

فصل دوم

زبان های توصیف سخت افزار

فصل سوم

آموزش زبان توصیف VHDL

فصل چهارم

سنتز سخت افزار

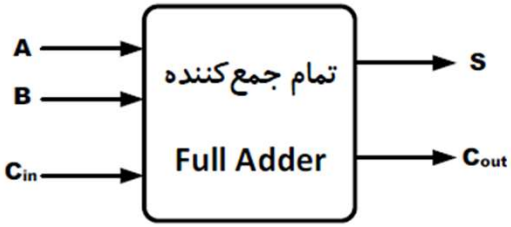
فصل پنجم

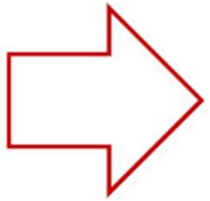
طراحی سیستم های دیجیتال با PLD

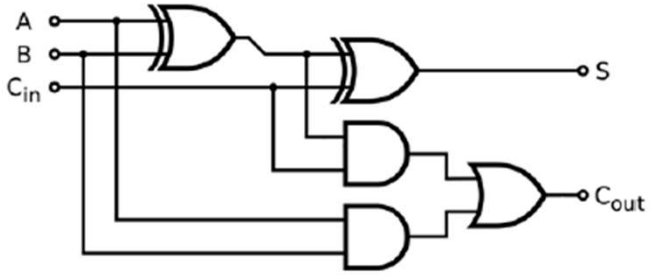
اسلاید ها برگرفته از جزوه دکتر محمد علی شفیعیان
https://shafieian-education.ir/computer-aided-digital-system-design-3/

طراحی سیستم‌های دیجیتال

طراحی سیستم دیجیتال، فرآیند خلق و مدیریت توصیف‌های متعددی از سیستم است که نمایانگر نگرش‌های متمایز با سطوح مختلف تجرید است.

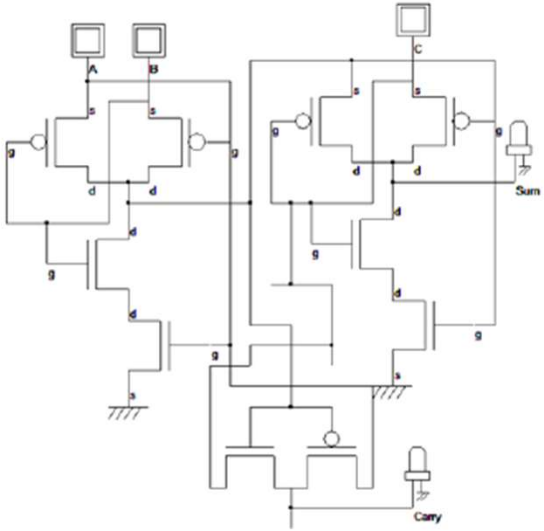


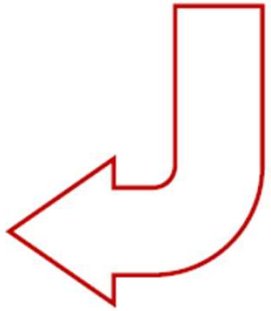


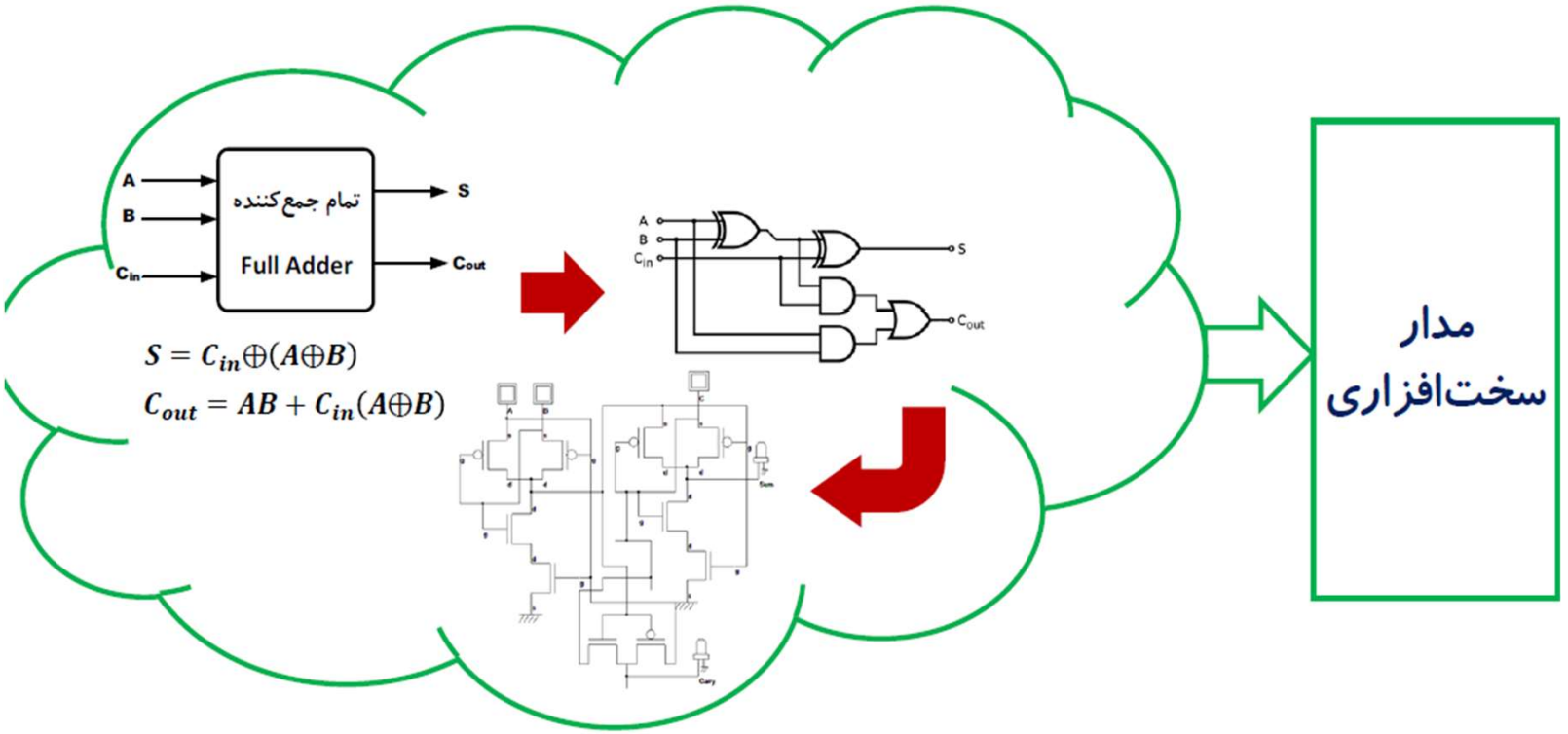


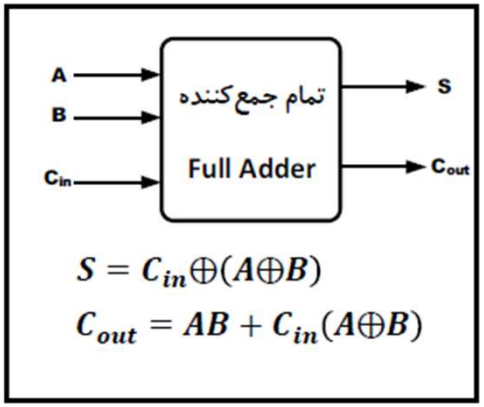
$$S = C_{in} \oplus (A \oplus B)$$

$$C_{out} = AB + C_{in}(A \oplus B)$$






A
B
Cin
تمام جمع‌کننده
Full Adder
S
Cout
$S = C_{in} \oplus (A \oplus B)$
$C_{out} = AB + C_{in}(A \oplus B)$
مدار
سخت‌افزاری

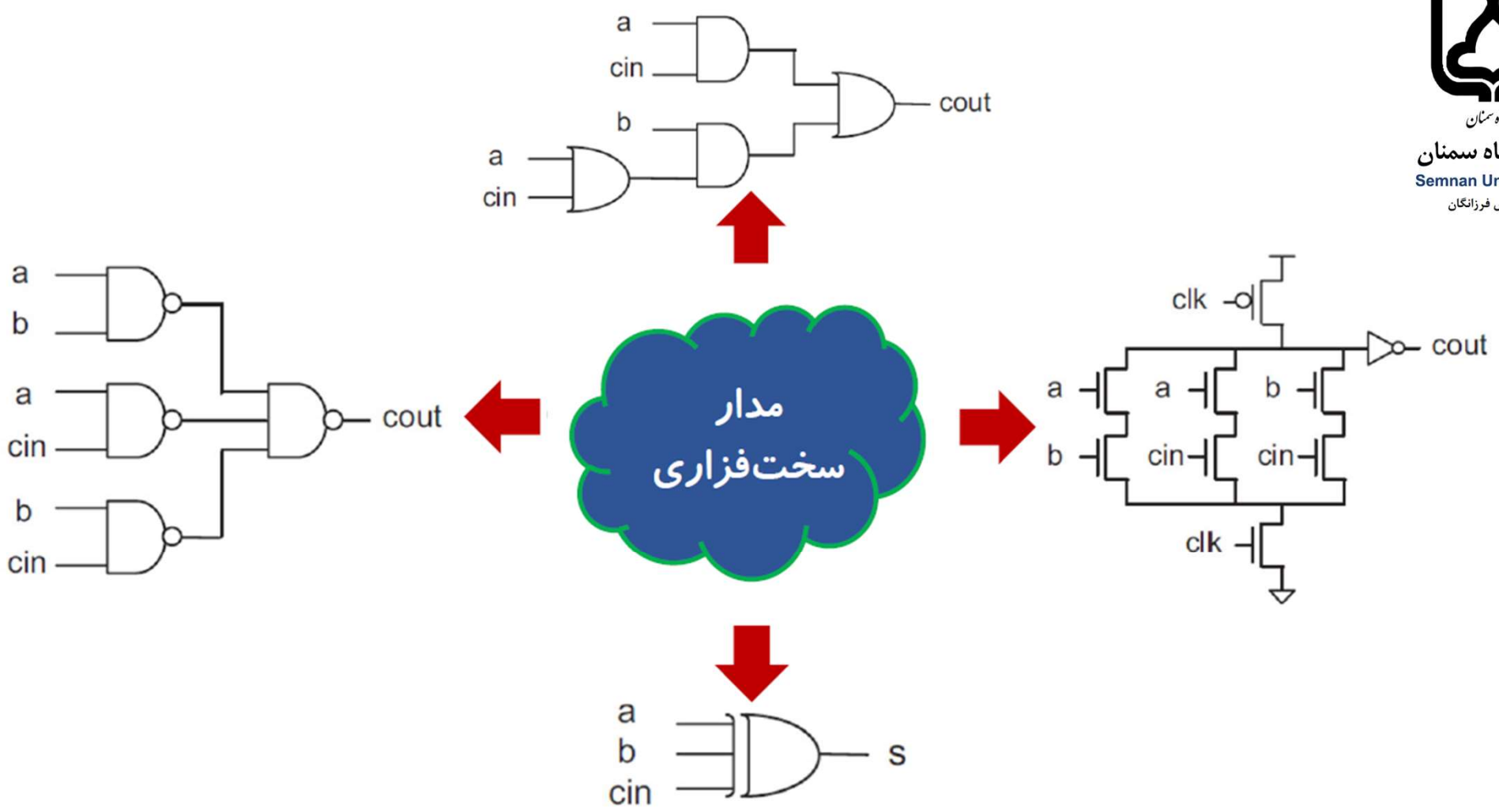


$$S = C_{in} \oplus (A \oplus B)$$

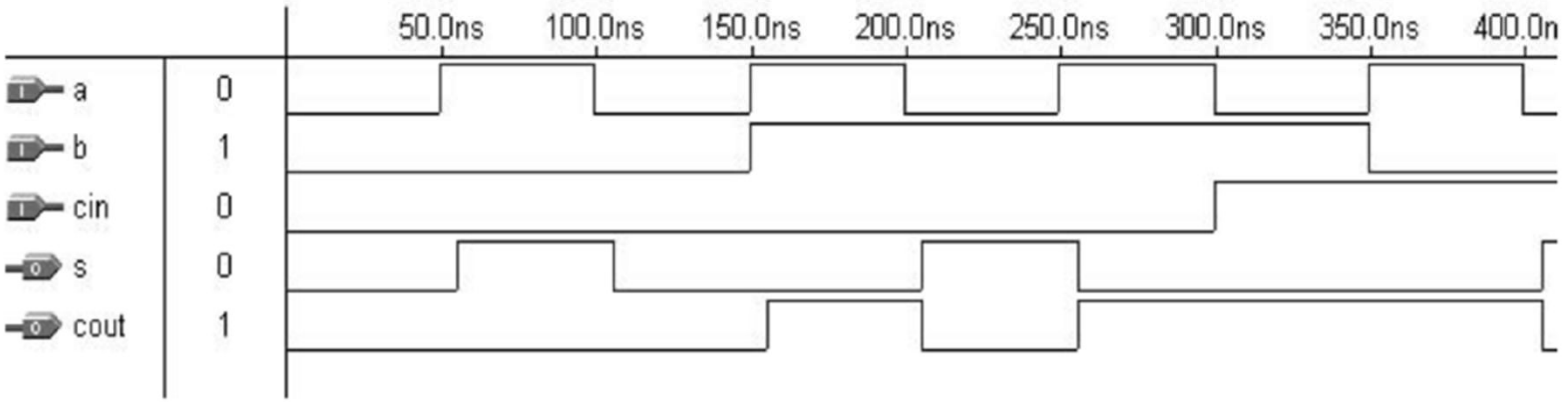
$$C_{out} = AB + C_{in}(A \oplus B)$$

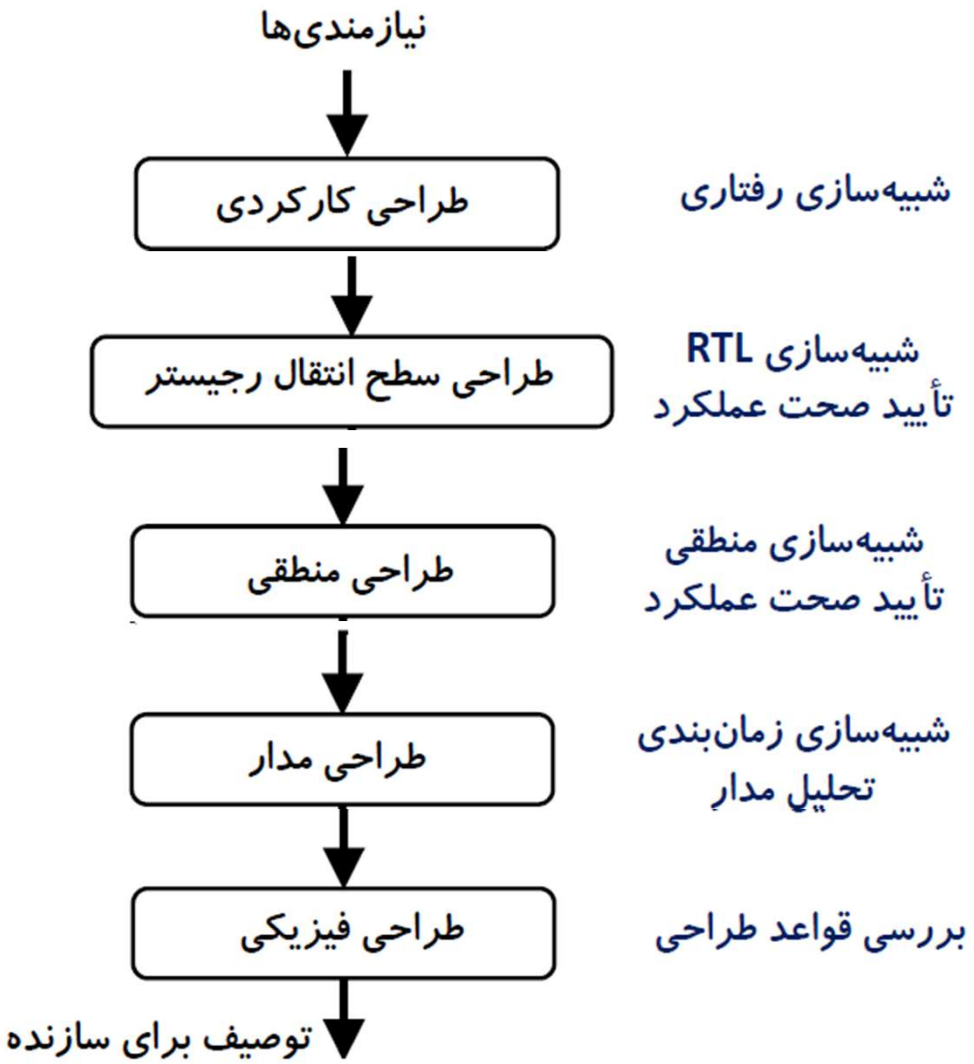
مدار
سخت‌فزاری

```
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;
```


a
cin
b
cout
a
cin
a
b
a
cin
b
cin
cout
مدار
سخت‌فزاری
clk
cout
a
b
a
cin
b
cin
clk
a
b
cin
s

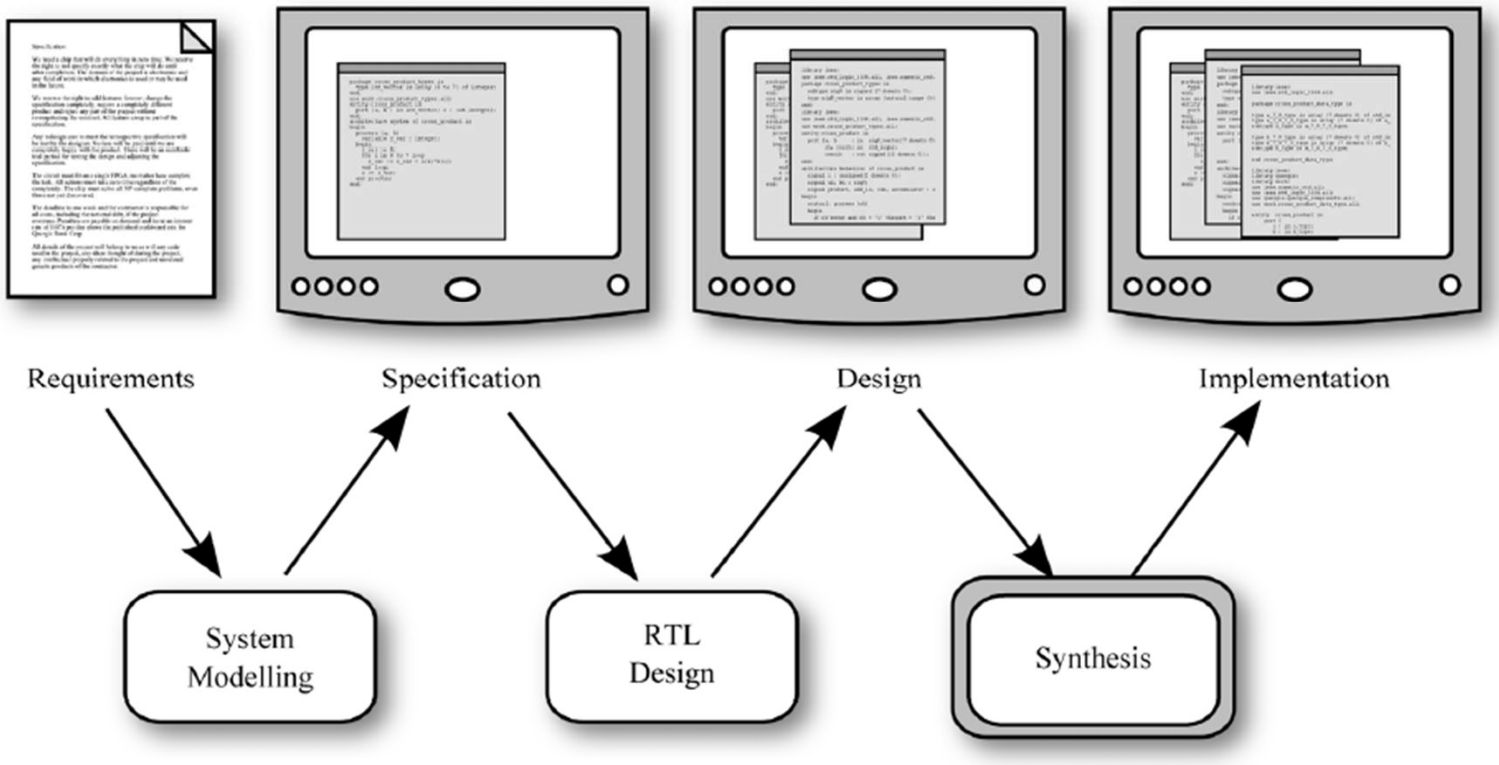
طراحی سیستم‌های دیجیتال
(شبیه‌سازی زمانی)



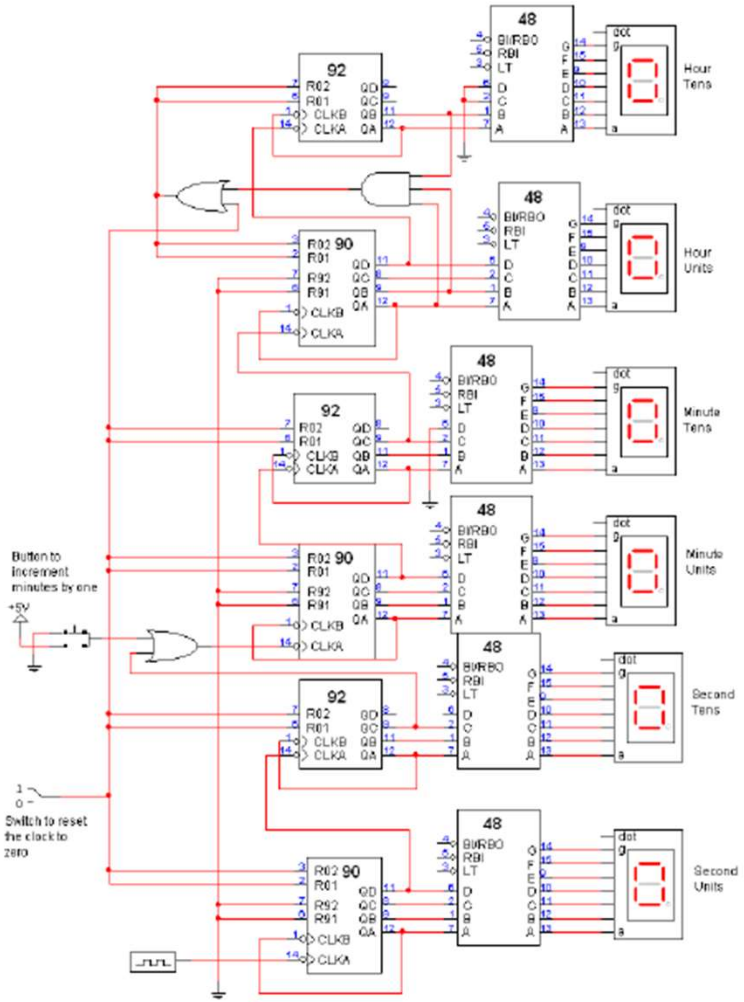

نیازمندی‌ها
طراحی کارکردی
شبیه‌سازی رفتاری
طراحی سطح انتقال رجیستر
شبیه‌سازی RTL
تأیید صحت عملکرد
طراحی منطقی
شبیه‌سازی منطقی
تأیید صحت عملکرد
طراحی مدار
شبیه‌سازی زمان‌بندی
تحلیل مدار
طراحی فیزیکی
بررسی قواعد طراحی
توصیف برای سازنده

واحدهای پیچیده‌تر
تعداد کم (جمع‌کننده، حافظه‌ها)
تجرید (Abstraction) بالاتر
تعداد زیادی واحد ساده اما با امکانات محدود
مثل گیت‌ها و ترانزیستورها
تجرید (Abstraction) کمتر

طراحی سیستم‌های دیجیتال

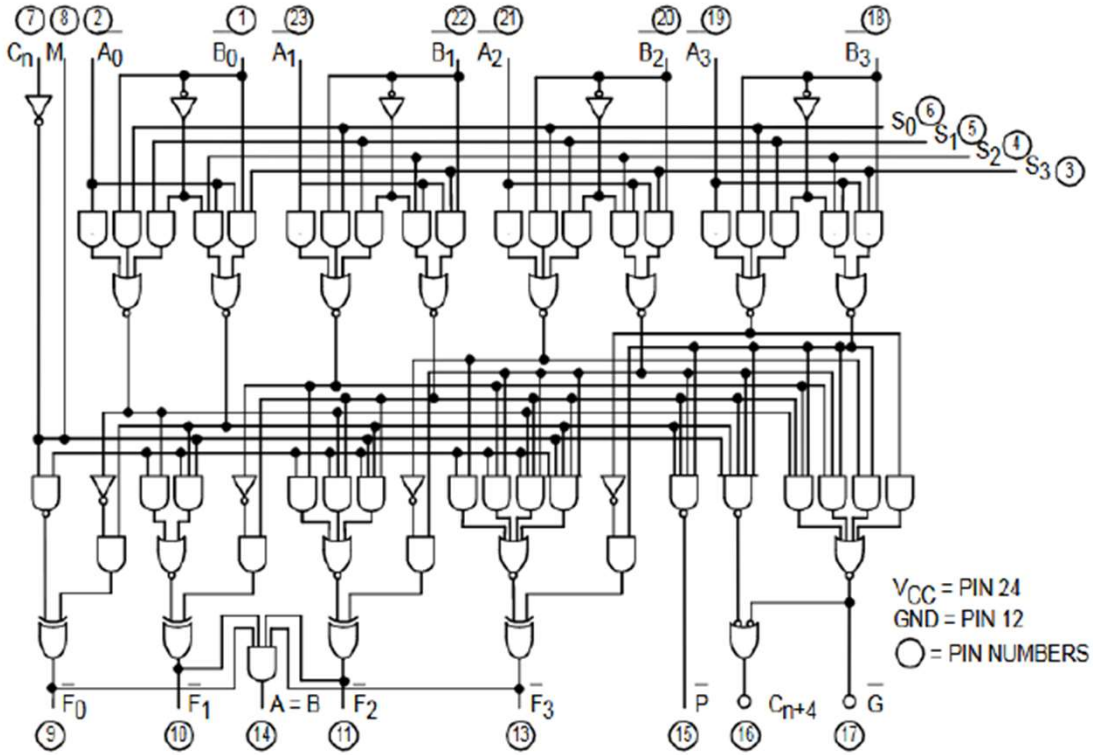


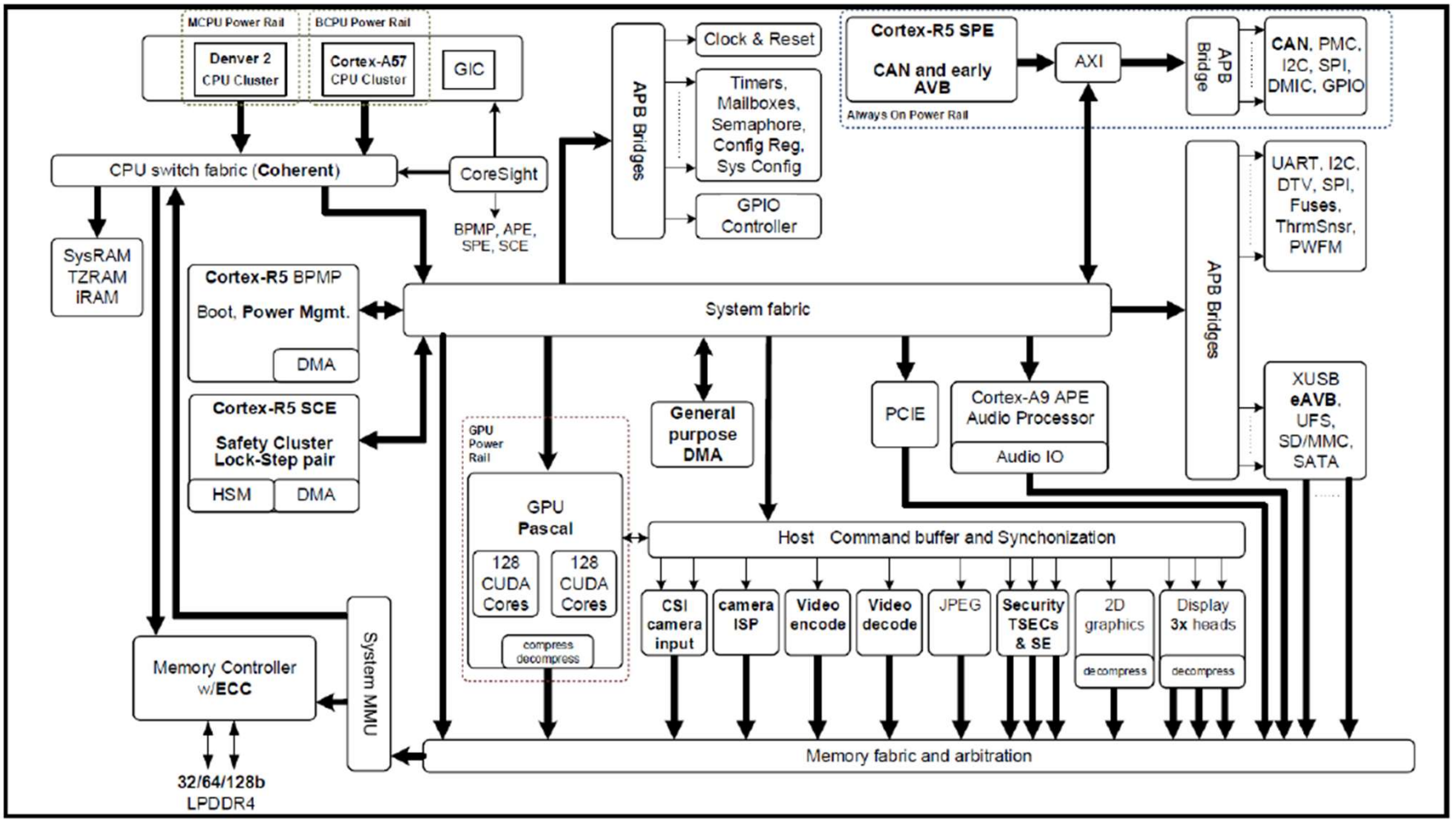
طراحی سیستم‌های دیجیتال

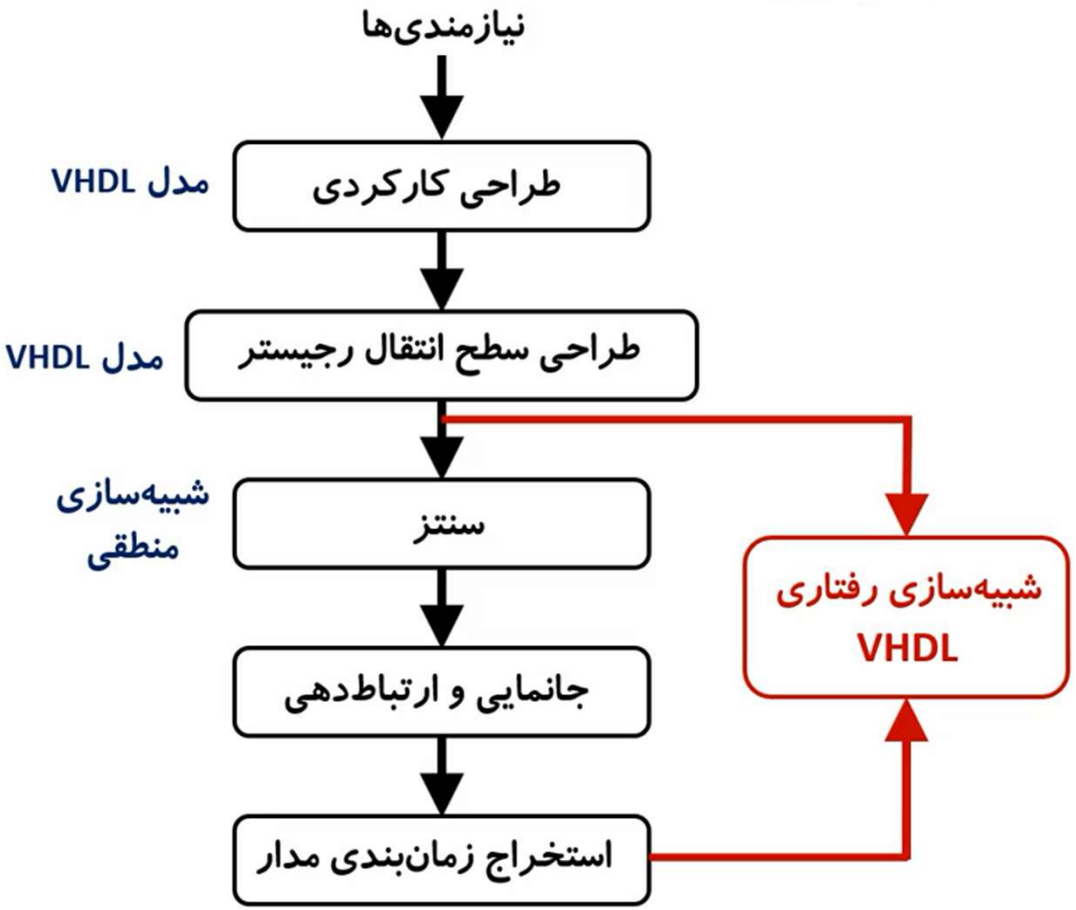


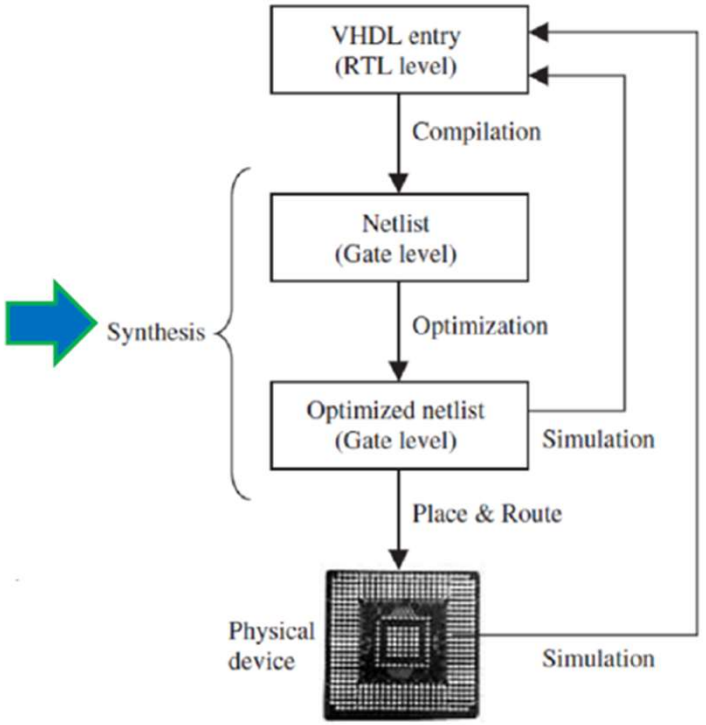
دانشگاه سمنان
Semnan University
پردیس فرزانگان

طراحی سیستم‌های دیجیتال



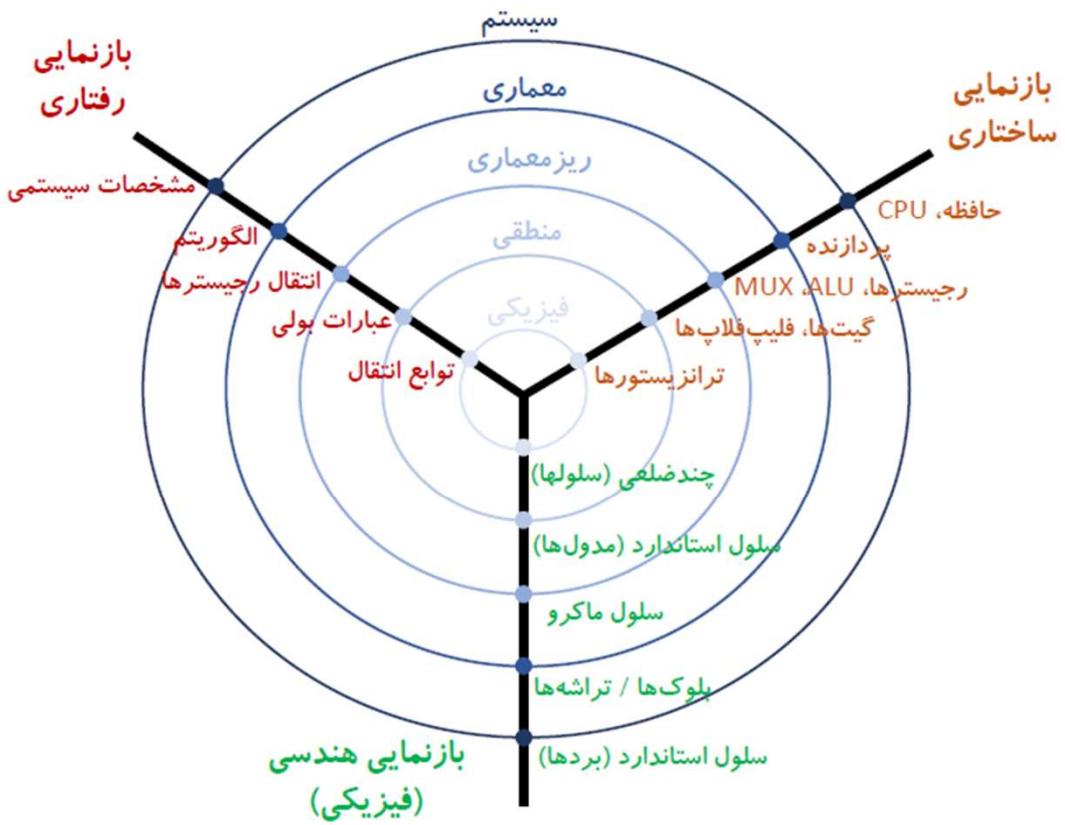

MCPU Power Rail
BCPU Power Rail
Denver 2 CPU Cluster
Cortex-A57 CPU Cluster
GIC
CPU switch fabric (Coherent)
CoreSight
BPMP, APE, SPE, SCE
SysRAM TZRAM iRAM
Cortex-R5 BPMP
Boot, Power Mgmt.
DMA
Cortex-R5 SCE
Safety Cluster Lock-Step pair
HSM
DMA
APB Bridges
Clock & Reset
Timers, Mailboxes, Semaphore, Config Reg, Sys Config
GPIO Controller
Cortex-R5 SPE
CAN and early AVB
Always On Power Rail
AXI
APB Bridge
CAN, PMC, I2C, SPI, DMIC, GPIO
UART, I2C, DTV, SPI, Fuses, ThrmSnsr, PWFM
APB Bridges
System fabric
General purpose DMA
PCIE
Cortex-A9 APE Audio Processor
Audio IO
XUSB eAVB, UFS, SD/MMC, SATA
GPU Power Rail
GPU Pascal
128 CUDA Cores
128 CUDA Cores
compress decompress
Host Command buffer and Synchonization
CSI camera input
camera ISP
Video encode
Video decode
JPEG
Security TSECs & SE
2D graphics
decompress
Display 3x heads
decompress
Memory Controller w/ECC
System MMU
32/64/128b LPDDR4
Memory fabric and arbitration


نیازمندی‌ها
مدل VHDL
طراحی کارکردی
مدل VHDL
طراحی سطح انتقال رجیستر
شبیه‌سازی منطقی
سنتز
جانمایی و ارتباط‌دهی
استخراج زمان‌بندی مدار
شبیه‌سازی رفتاری VHDL


VHDL entry
(RTL level)
Compilation
Netlist
(Gate level)
Optimization
Optimized netlist
(Gate level)
Simulation
Place & Route
Physical
device
Simulation
Synthesis

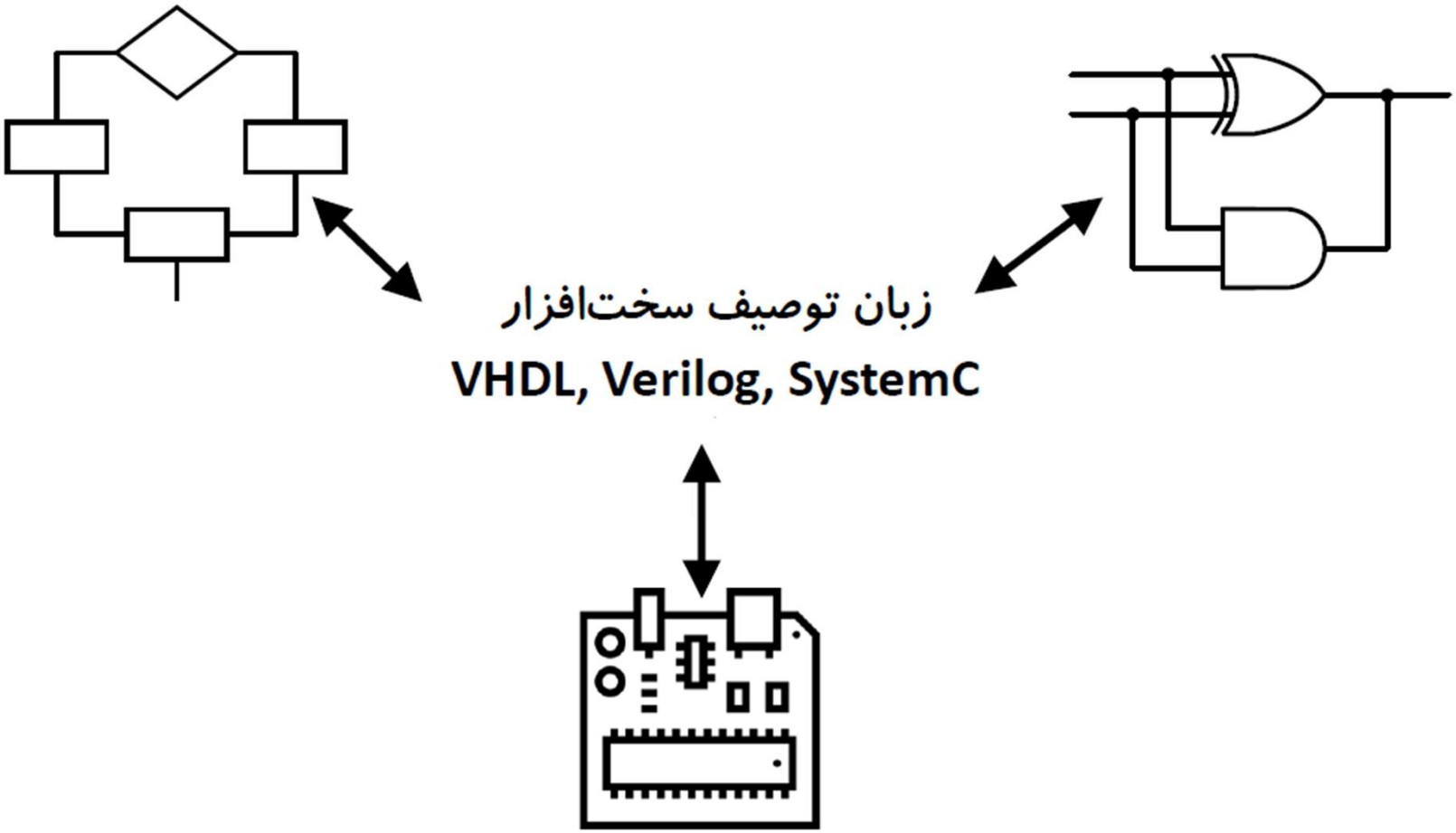
توصیف ساختاری

سطوح تجرید



زبان توصیف سخت افزار

دانشگاه سمنان
Semnan University
پردیس فرزانگان



مدارات مجتمع قابل برنامه ریزی

سیستم های تعبیه شده

سیستم تعبیه شده به این معناست که پروسسور (پردازنده) و سایر امکانات در یک سیستم واحد گنجانده شده اند.

روش های پیاده سازی سخت افزار در سیستم های تعبیه شده

- استفاده از تراشه *ASIC*
- استفاده از قطعات گسسته
- استفاده از کامپیوترهای صنعتی
- استفاده از تراشه های برنامه پذیر از قبیل *CPLD* و *FPGA*
- استفاده از پردازنده ها

روش های پیاده سازی سخت افزار در سیستم های تعبیه شده

استفاده از تراشه ASIC

- ASIC مخفف *A*pplication *S*pecific *I*ntegrated *C*ircuit است.
- در این روش کلیه نیازهای سیستم در قالب یک مدار مجتمع، توسط تکنولوژی VLSI و به صورت بهینه طراحی می شود.

عیب

- هزینه اولیه طراحی مهندسی بالا
- طی نمودن پروسه زمان بر و پرهزینه

طراحی تراشه در زمان ارتقای سیستم

فقط در سیستم هایی با تولید بالا مانند گوشی های تلفن همراه یا کاربردهای نظامی که کارآیی و قابلیت اطمینان اهمیت دارد، استفاده می شود.

روش های پیاده سازی سخت افزار در سیستم های تعبیه شده

استفاده از قطعات گسسته

- طراحی سیستم توسط المان هایی مانند ترانزیستورها و تراشه هایی مانند سری *74XX* انجام می شود.

- به دلیل تعداد المان های زیاد مورد استفاده، ابعاد و صرف توان سیستم بسیار افزایش می یابد

روش های پیاده سازی سخت افزار در سیستم های تعبیه شده

استفاده از کامپیوترهای صنعتی

- در برخی از کاربردها مانند دستگاه های خودپرداز *ATM* می توان از کامپیوترهای صنعتی به عنوان پردازنده استفاده کرد.

- سخت افزار سیستم به صورت آماده بوده و نیازی به طراحی ندارد.
- سیستم عامل های موجود از قبیل *Linux* و *Windows CE* کلیه سرویس های اصلی مورد نیاز سیستم را تأمین می کند.
- می توان از بسته های نرم افزاری متعددی برای نوشتن برنامه استفاده نمود.

روش های پیاده سازی سخت افزار در سیستم های تعبیه شده

استفاده از کامپیوترهای صنعتی

- ابعاد بزرگ و مصرف توان بالای سیستم باعث می شود که در بسیاری از سیستم های کم مصرف و قابل حمل نتوان از این روش استفاده کرد.
- سیستم عامل های معمول مانند ویندوز قابلیت اطمینان بالایی ندارند و نمی توان در کاربردهای حساس از آن ها استفاده کرد.
- به صورت بلادرنگ (*Real Time*) عمل نمی کنند که همین امر، کاربرد آنها را محدود می کند.

روش های پیاده سازی سخت افزار در سیستم های تعبیه شده

استفاده از تراشه های برنامه پذیر از قبیل *CPLD* و *FPGA*

- این ادوات در مقایسه با تراشه های *ASIC* هزینه طراحی اولیه کمتری دارند.
- کارد اصلی تراشه های *FPGA* در سیستم های پردازش اطلاعات موازی و پرسرعت است.
- از تراشه های *CPLD* بیشتر به عنوان کنترل کننده حافظه استفاده می شود.

عیب

- مصرف توان تراشه های *FPGA* بیش از میکروکنترلرها است

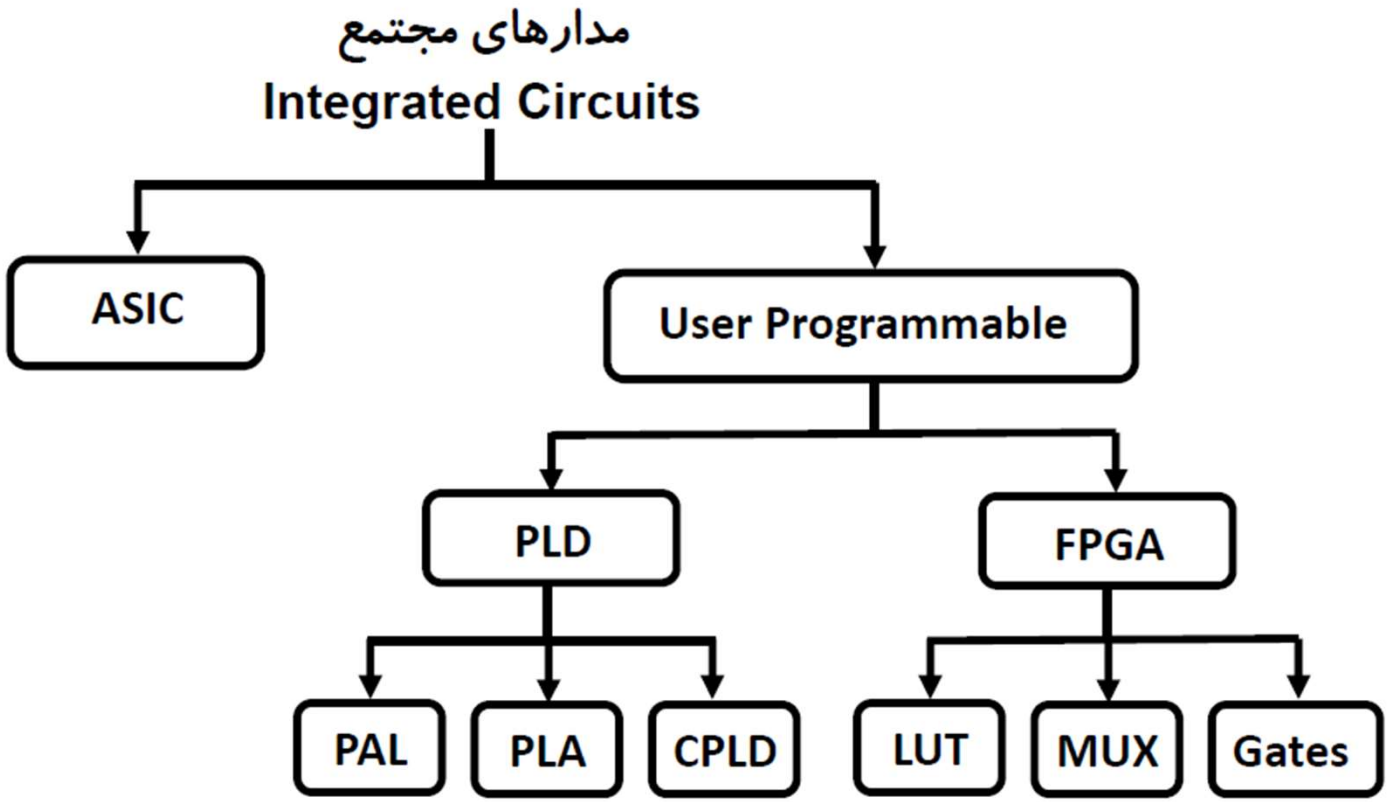
کمتر به عنوان کنترل کننده اصلی سیستم در ادوات تعبیه شده استفاده می شوند.

روش های پیاده سازی سخت افزار در سیستم های تعبیه شده

استفاده از پردازنده ها

با توجه به نوع کاربرد
- میکروپروسسورهای عمومی (همه منظوره)
- پردازنده‌های *DSP*
- میکروکنترلرها

روش های پیاده سازی سخت افزار در سیستم های تعبیه شده



دانشگاه سمنان
Semnan University
پردیس فرزانگان

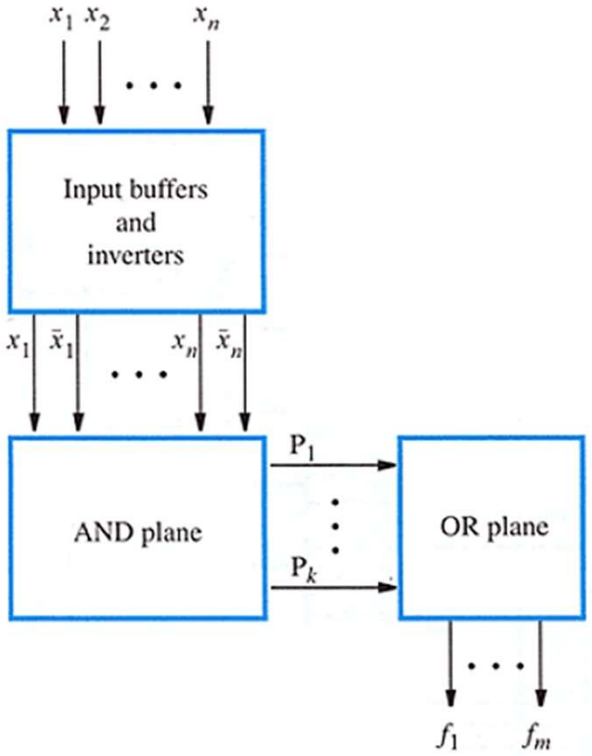
انواع مدارهای منطقی برنامه‌پذیر

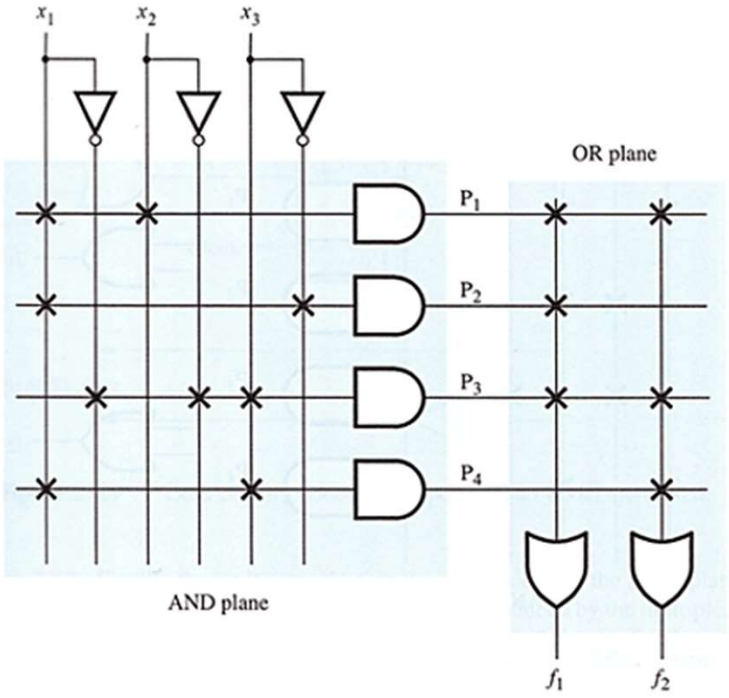
- در گذشته، طراحی سیستم‌های دیجیتال با مجموعه‌ای از گیت‌ها و ICهای استاندارد انجام می‌شد.
- امروزه طراحی این سیستم‌ها با استفاده از مدارهای منطقی برنامه‌پذیر تغییرات زیادی نموده است.
- این مدارها با ظرفیتی در حدود ۱۰۰ تا چندهزار گیت در یک مدار مجتمع VLSI قرار دارند.

Field Programmable Device (FPD)
Programmable Logic Device (PLD) ➡ مدارهای منطقی برنامه‌پذیر

Very Large Scale Integrated Circuits (VLSI)

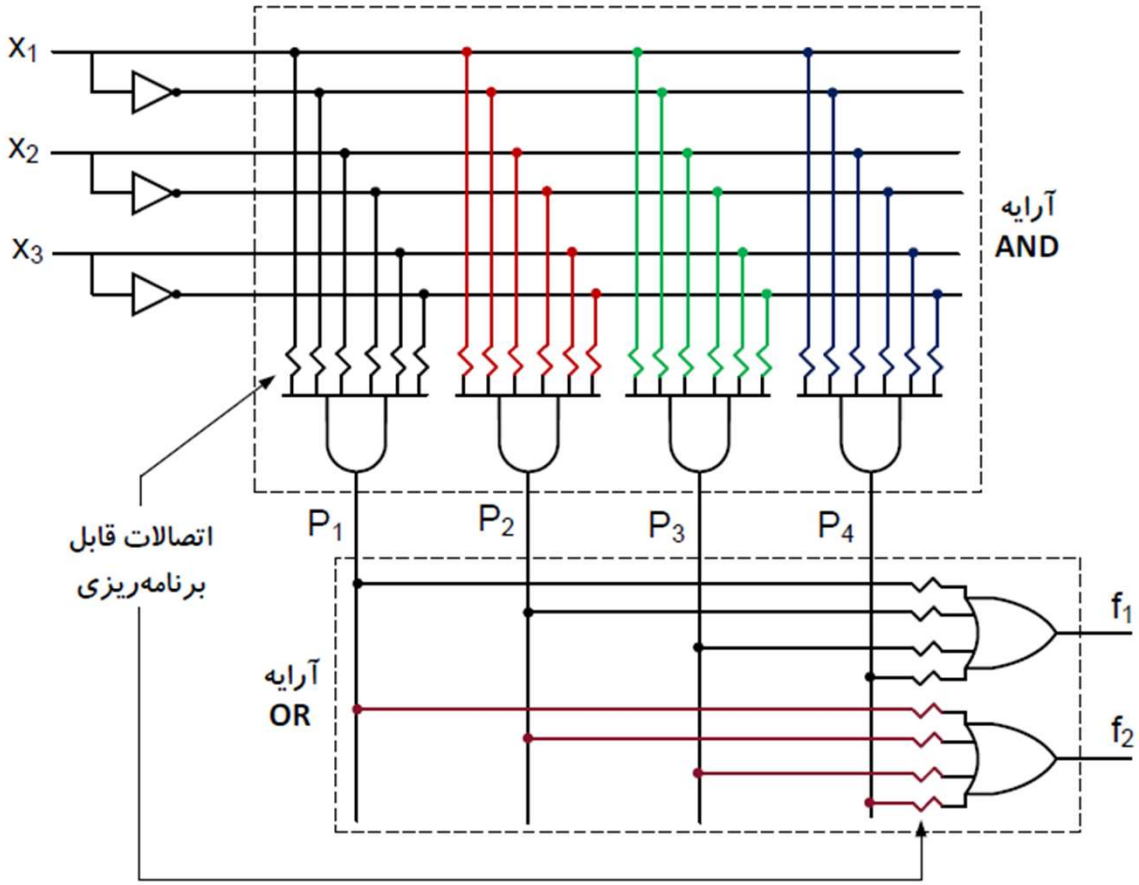
مدارهای منطقی برنامه‌پذیر PLA





PLA = Programmable Logic Array

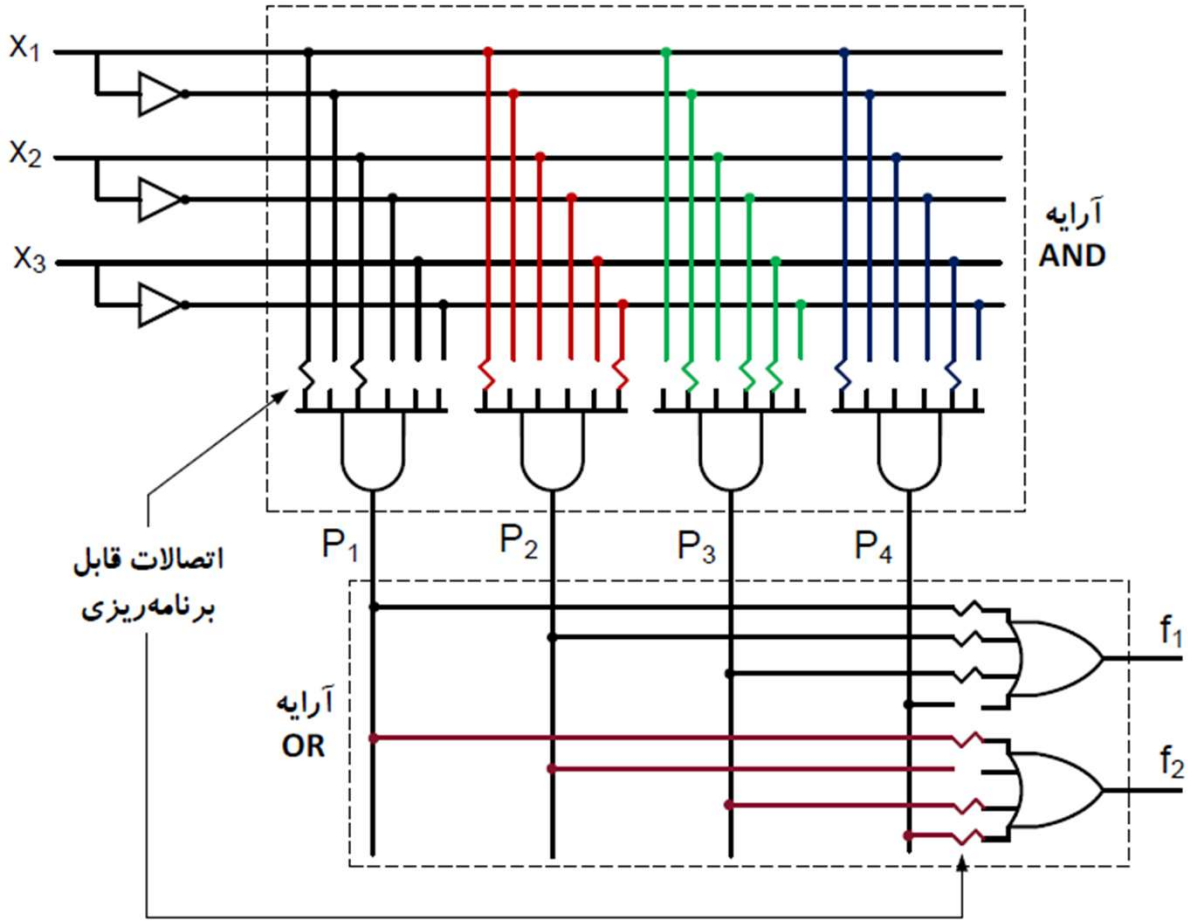
مدارهای منطقی
برنامه‌پذیر PLA



$$f_1 = x_1x_2 + x_1\overline{x_3} + \overline{x_1}\,\overline{x_2}x_3$$

$$f_2 = x_1x_2 + x_1x_3 + \overline{x_1}\,\overline{x_2}x_3$$

مدارهای منطقی برنامه‌پذیر PLA



$$f_1 = x_1x_2 + x_1\overline{x_3} + \overline{x_1}\,\overline{x_2}x_3$$

$$f_2 = x_1x_2 + x_1x_3 + \overline{x_1}\,\overline{x_2}x_3$$

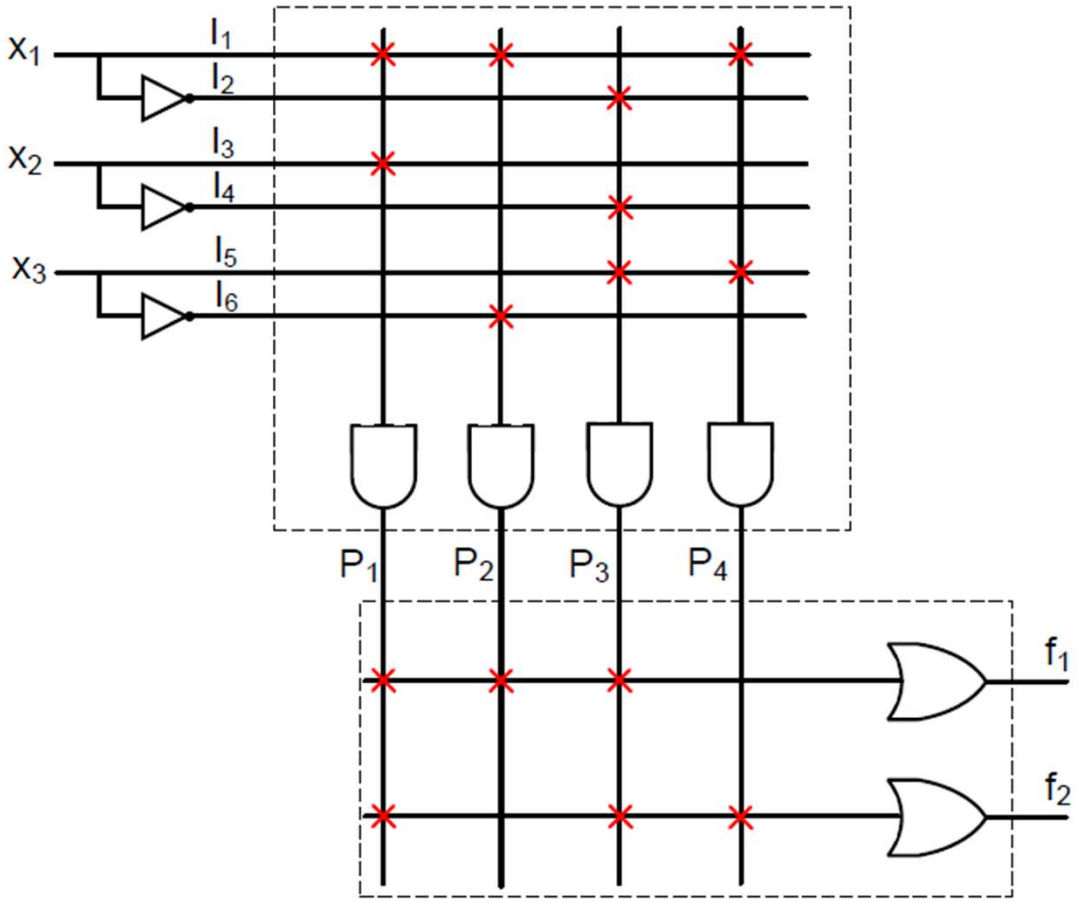
مدارهای منطقی برنامه‌پذیر PLA

- PLA شامل یک طبقه آرایه‌ای از گیت AND و یک طبقه آرایه از گیت OR می‌باشد.
- هر یک از ورودی‌ها یا معکوس آن‌ها به گیت AND متصل می‌شوند، در نتیجه خروجی گیت AND حاصلضرب ورودی یا معکوس آنها است.
- خروجی هر گیت OR، مجموع منطقی خروجی‌های AND می‌باشد.
- مدار PLA برای پیاده‌سازی توابع منطقی که به‌صورت مجموع حاصلضرب (SOP) است بسیار مناسب می‌باشد.

آیا هر تابع منطقی را می‌توان با PLA پیاده‌سازی کرد؟

چون هر تابع منطقی را می‌توان برحسب مجموع حاصل‌ضرب (SOP) نوشت،
لذا هر تابع منطقی قابل پیاده‌سازی بر PLA است

مدارهای منطقی برنامه‌پذیر PLA



$$f_1 = x_1x_2 + x_1\overline{x_3} + \overline{x_1}\,\overline{x_2}x_3$$

$$f_2 = x_1x_2 + x_1x_3 + \overline{x_1}\,\overline{x_2}x_3$$

مدارهای منطقی برنامه‌پذیر PLA

- در آرایه PLA هر خط عمودی داخل آرایه AND، خروجی یک مدار AND است که در محل تقاطع هر سطر و ستون عمودی یک سوییچ قابل برنامه‌ریزی گذاشته شده است.

- سوییچ قابل برنامه‌ریزی یک دیود با فیوز به‌طور سری یا یک ترانزیستور MOS با گیت شناور باشد.

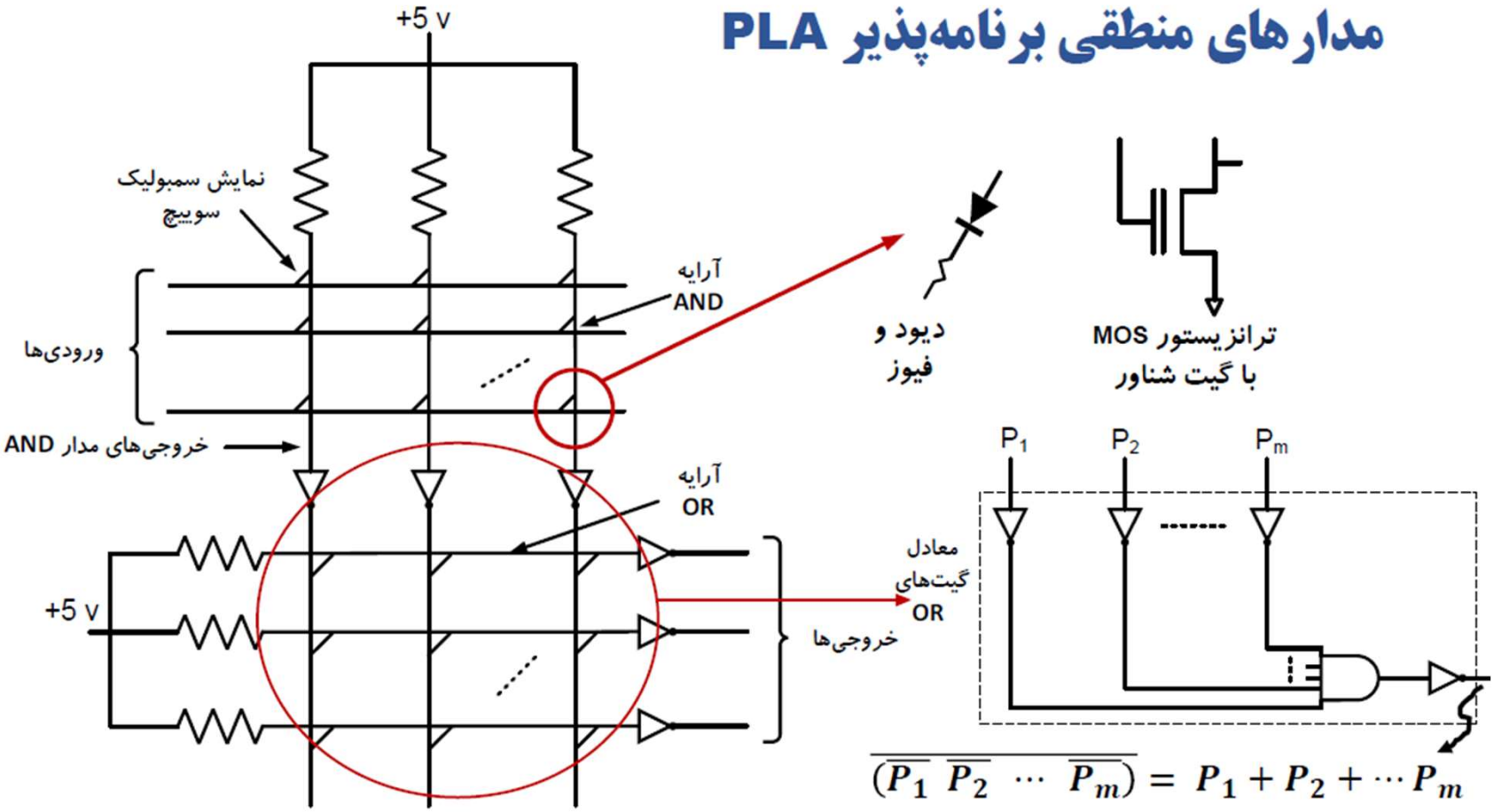
- این سوییچ‌های قابل برنامه‌ریزی با ابزارهای برنامه‌ریزی CAD برنامه‌ریزی می‌شوند.

Computer Aided Design (CAD)

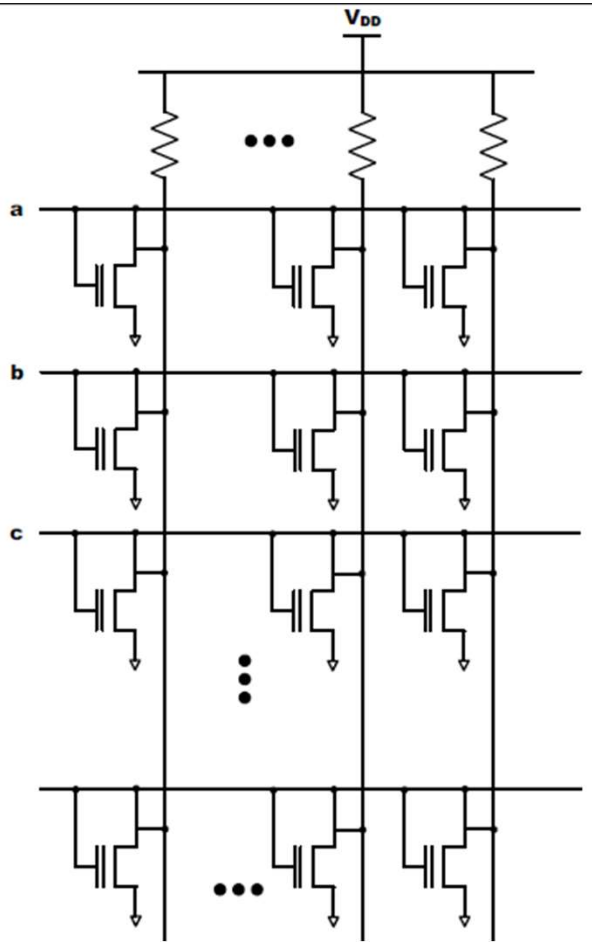
Programmable Switch = سوییچ قابل برنامه‌ریزی

Floating Gate = گیت شناور

مدارهای منطقی برنامه‌پذیر PLA



مدارهای منطقی برنامه‌پذیر PLA



مدارهای منطقی برنامه‌پذیر PLA

- چون PLA دو طبقه گیت قابل برنامه‌ریزی داردگران تمام می‌شود و تأخیر مدار نسبتاً زیاد است، در نتیجه سرعت آن کم است.

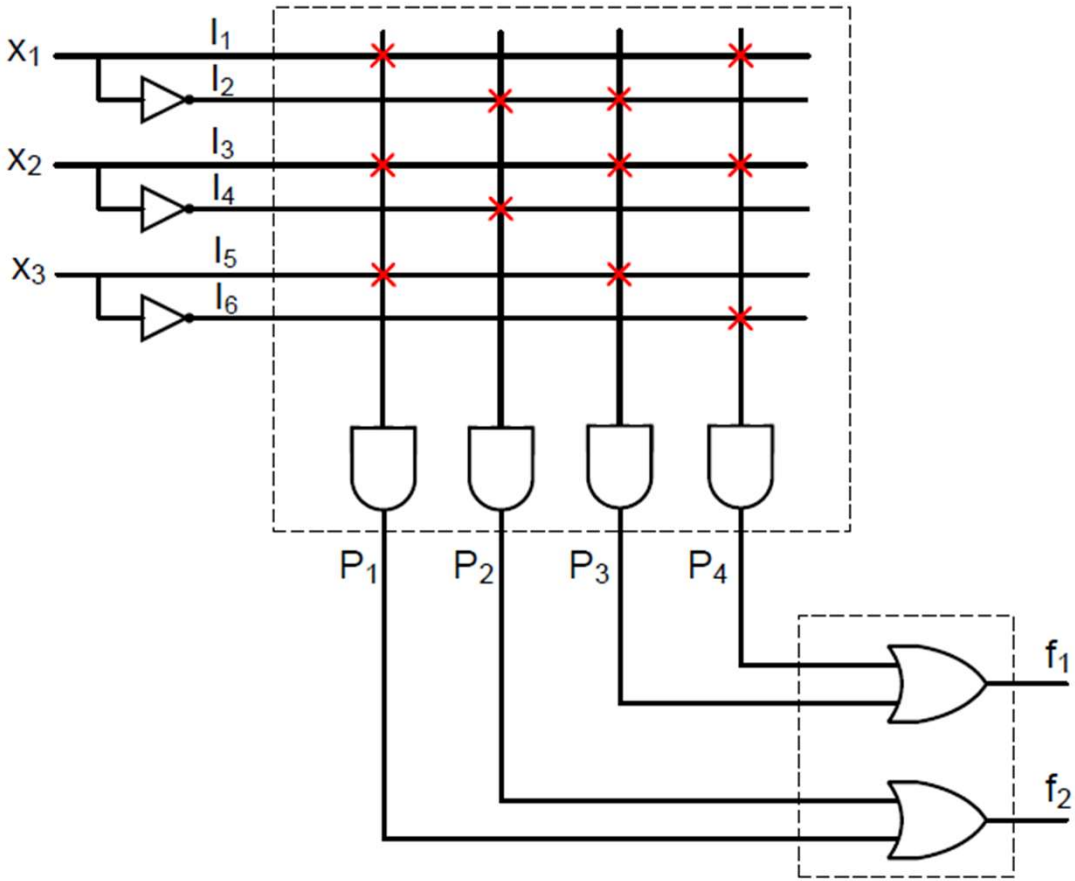
- برای حل این مسأله از مدارهای قابل برنامه‌ریزی PAL استفاده می‌شود.

مدارهای منطقی برنامه‌پذیر PAL

- در PLA هر دو آرایه AND و OR قابل برنامه‌ریزی هستند.

- در PAL فقط آرایه AND قابل برنامه‌ریزی است و خروجی گیت‌های AND به‌طور دائم به ورودی‌های OR متصل می‌باشند.

مدارهای منطقی برنامه‌پذیر PAL



$$f_1 = x_1 x_2 \overline{x_3} + \overline{x_1} x_2 x_3$$

$$f_2 = \overline{x_1}\,\overline{x_2} + x_1 x_2 x_3$$

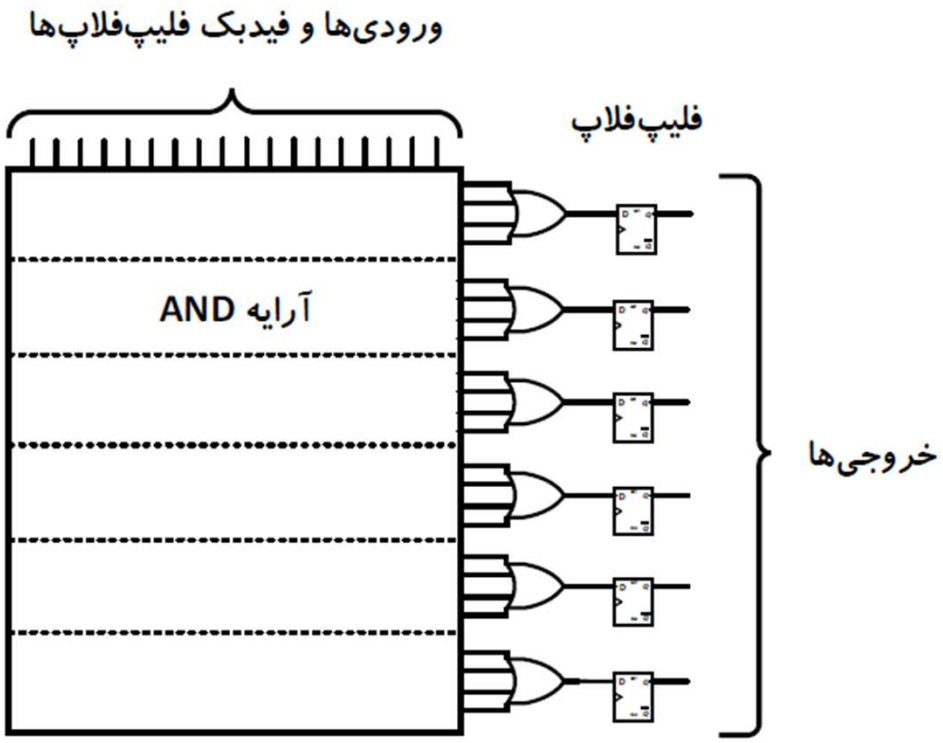
مدارهای منطقی برنامه‌پذیر PAL

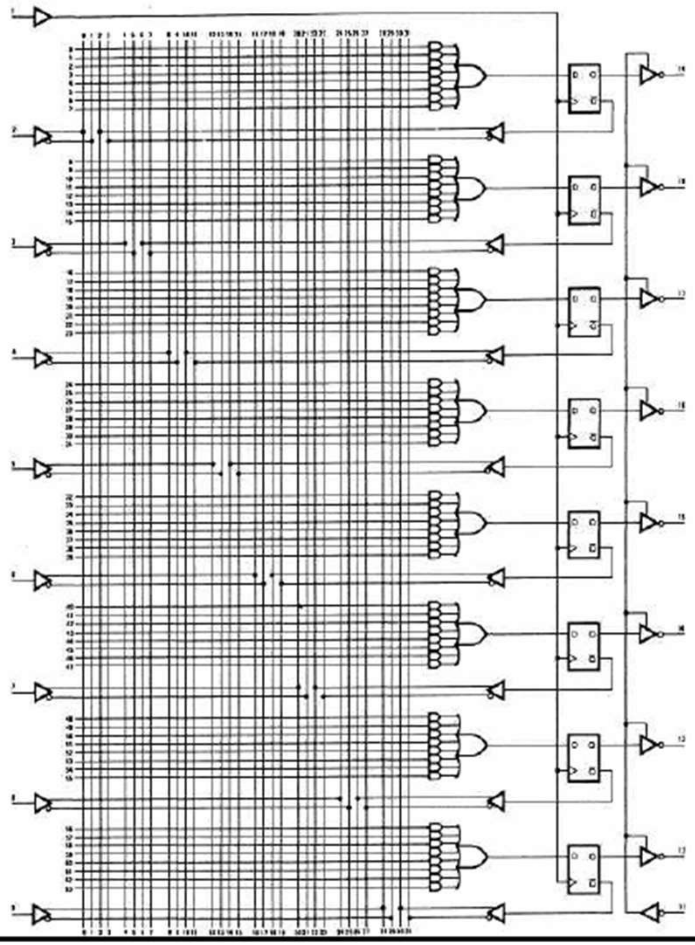
- در PAL چون فقط گیت‌های AND قابل برنامه‌ریزی هستند، لذا ساخت PAL کم‌هزینه‌تر از PLA می‌باشد و سرعت آن نیز بیشتر است.

- برای جبران آرایه غیرقابل برنامه‌ریزی OR، سازندگان انواع مختلفی از PAL را به بازار عرضه کرده‌اند که از لحاظ تعداد گیت‌های OR و نیز تعداد ورودی‌های گیت‌های OR با یکدیگر متفاوت هستند.

- در PALها معمولاً خروجی گیت‌های OR به فلیپ‌فلاپ متصل هستند، پس می‌توان مدارهای ترتیبی با پالس ساعت را در آن‌ها پیاده‌سازی نمود.

مدارهای منطقی برنامه‌پذیر PAL





مدارهای منطقی برنامه‌پذیر PAL

PAL16R6

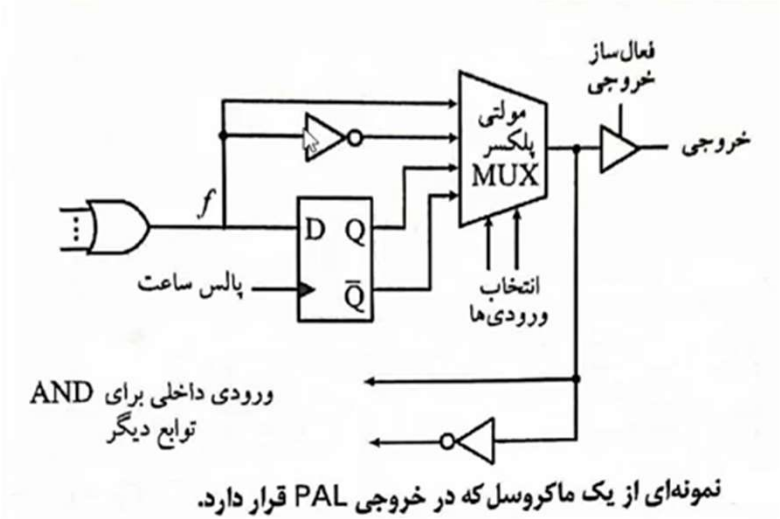
PAL16R8

مدارهای منطقی برنامه‌پذیر PAL

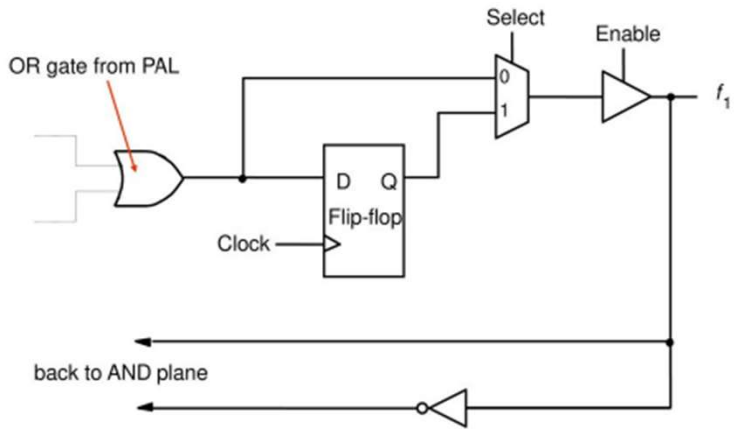
دانشگاه سمنان
Semnan University
پردیس فرزانگان

- در مدارهای PAL نسل بعدی به جای فلیپ‌فلاپ از مجموعه‌ای از فلیپ‌فلاپ، مولتی‌پلکسر و ... به نام ماکروسل (Macrocell) استفاده شد.

- با برنامه‌ریزی ورودی مولتی‌پلکسر می‌توان خروجی PAL را مستقیماً برابر تابع f یا معکوس آن استفاده نمود و یا خروجی تابع را از فلیپ‌فلاپ گرفت.



نمونه‌ای از یک ماکروسل که در خروجی PAL قرار دارد.



مدارهای منطقی برنامه‌پذیر PAL

- امروزه PALهای مختلف با انواع ماکروسل‌ها در بازار وجود دارندکه نمونه‌ای از آن‌ها PAL یا PLD به شماره PAL22V10 است

- مدارهای منطقی برنامه‌پذیر PAL و PLA به مدارهای برنامه‌پذیر ساده (SPLD) معروف هستند و ظرفیت آنها حدود ۲۰۰ گیت می‌باشد.

- SPLDها برای محدوده وسعیی از کاربردها مناسب هستند و امروزه تقریباً در هر سیستم دیجتالی وجود دارند.

Programmable Logic Device (PLD)

Simple Programmable Logic Device (SPLD)

مدارهای منطقی برنامه‌پذیر CPLD

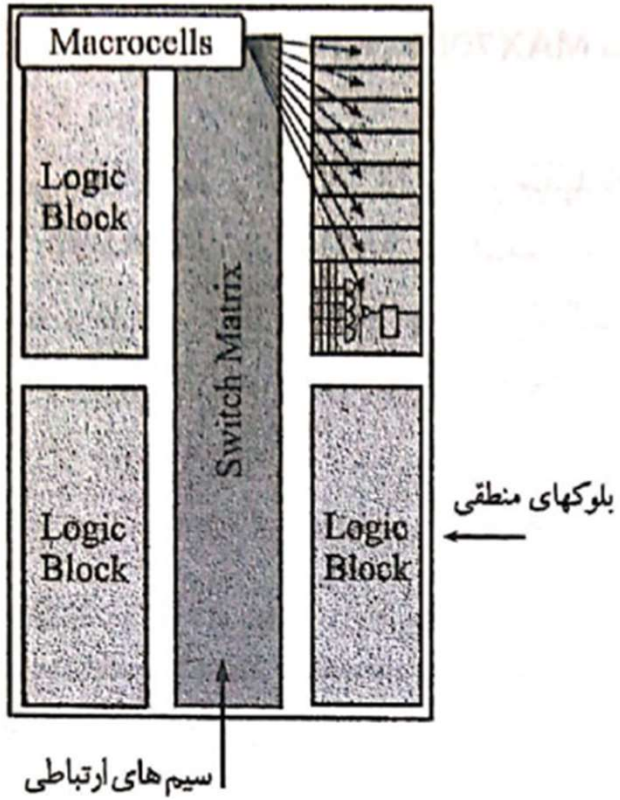
- PALها، ICهای قابل برنامه‌ریزی خوبی هستند اما برای طراحی سیستم‌های پیچیده دیجیتال ممکن است چندین PAL نیاز باشد.

- مدارهای منطقی CPLD از دو یا چندین بلوک منطقی و تعدادی ماکروسل تشکیل شده‌اند که با سیستم‌های ارتباطی و سوییچ‌های قابل برنامه‌ریزی با هم ارتباط داده می‌شوند.

Switch Matrix = سوییچ‌های قابل برنامه‌ریزی

Complex Programmable Logic Device (CPLD)

مدارهای منطقی برنامه‌پذیر CPLD

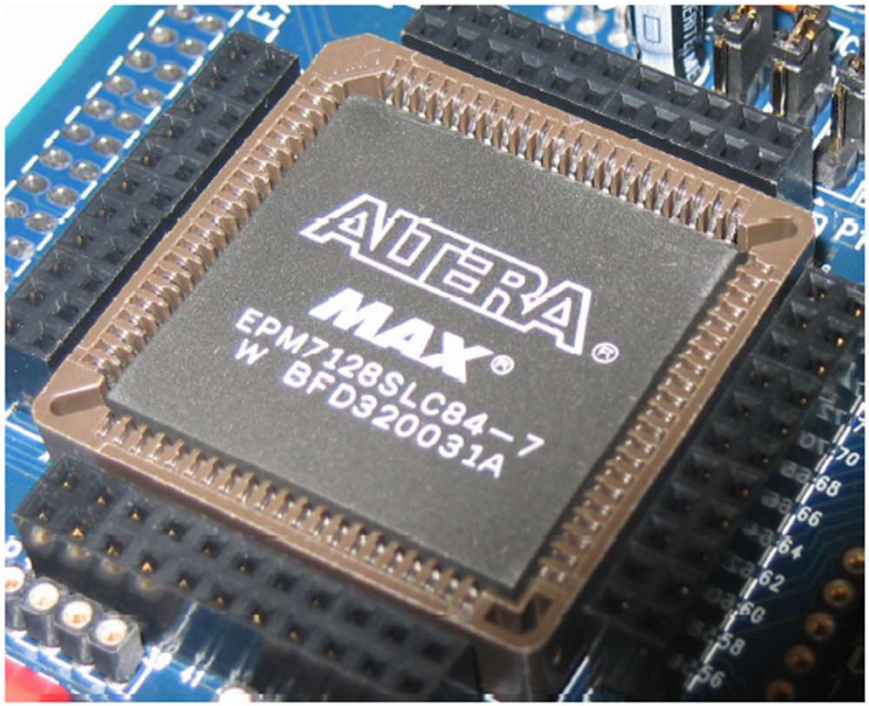


مدارهای منطقی برنامه‌پذیر CPLD

- CPLDهای تجاری با اندازه‌های مختلف از ۲ تا ۱۰۰ بلوک منطقی یا SPLD و با ظرفیتی در حدود ۱۰۰۰ تا ۱۵۰۰۰ گیت ساخته شده‌اند.

- CPLD خانواده MAX9000 از شرکت Altera دارای ظرفیت در حدود ۱۲۰۰۰ گیت می‌باشد.

مدارهای منطقی برنامه‌پذیر CPLD



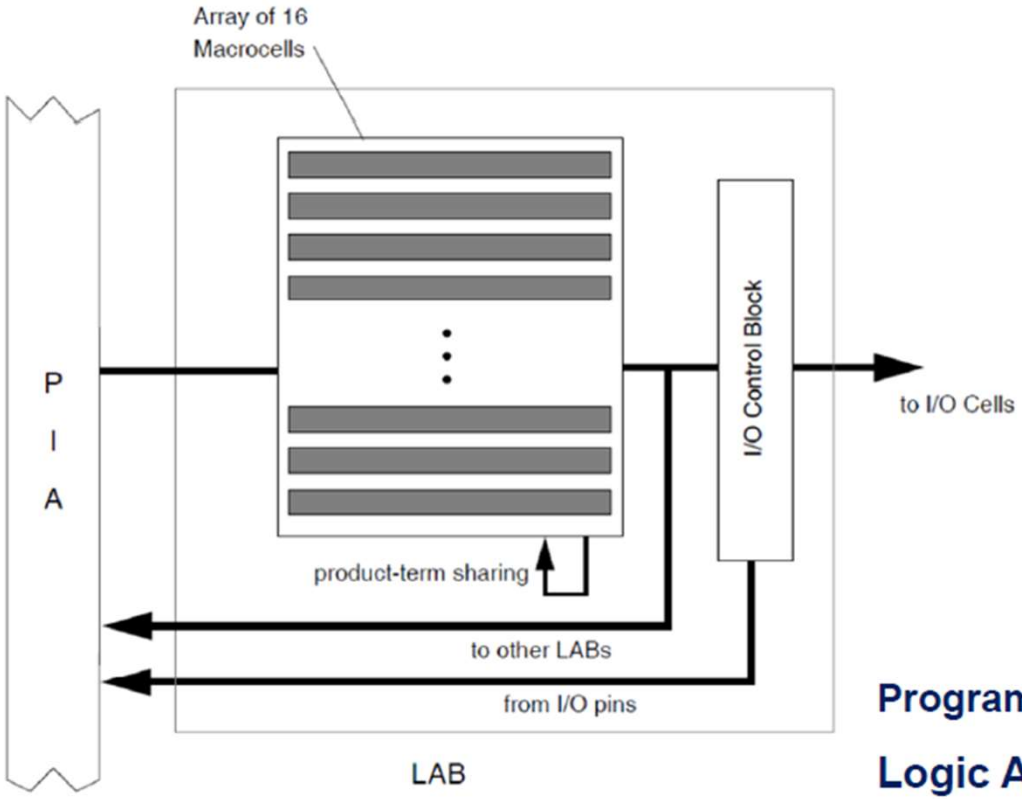
سری‌های CPLD شرکت Altera

- MAX5000
- **MAX7000**
- MAX9000

مدارهای منطقی برنامه‌پذیر CPLD

دانشگاه سمنان
Semnan University
پردیس فرزانگان

CPLD سری MAX7000

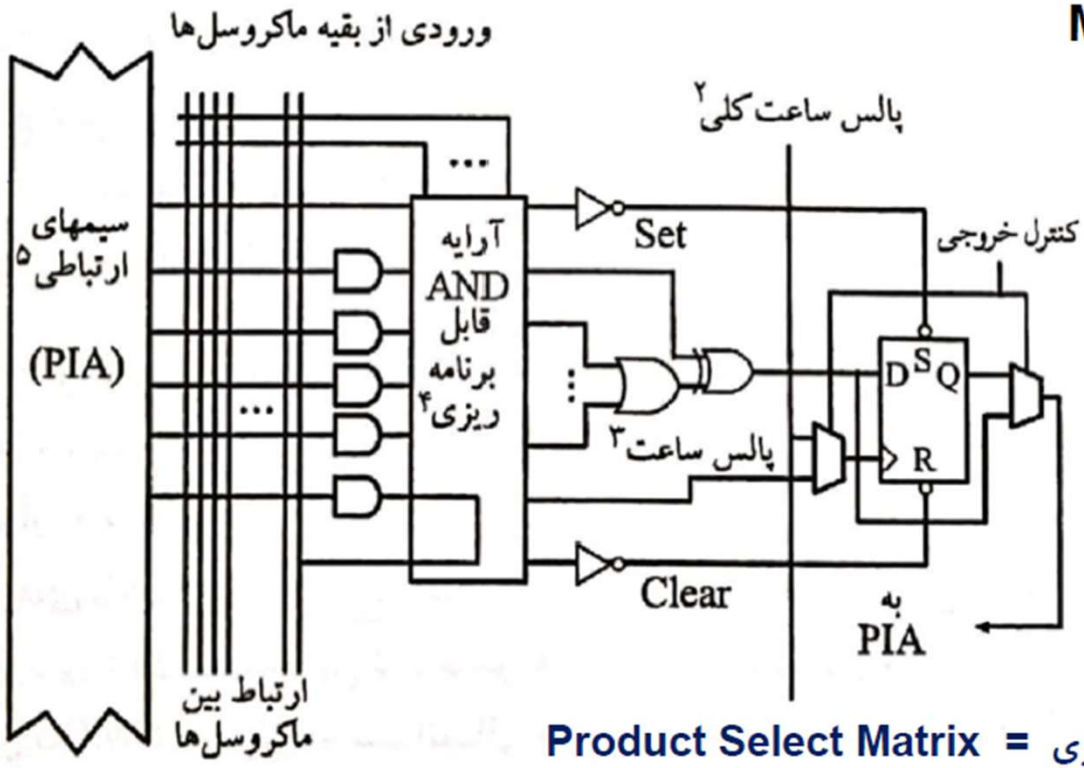


Programmable Interconnect Array (PIA)

Logic Array Block (LAB)

مدارهای منطقی برنامه‌پذیر CPLD

CPLD سری MAX7000



Programmable Interconnect Array (PIA)

Array Clock = پالس ساعت

Product Select Matrix = آرایه AND قابل برنامه ریزی

مدارهای منطقی برنامه‌پذیر CPLD

CPLDهای شرکت Xilinx

XC7000

XC9500

مشابه سری MAX7000 شرکت Altera

مزایای مدارهای منطقی برنامه‌پذیر CPLD

- CPLD تأخیر کمی در حد چند نانوثانیه دارند، لذا بسیار سریع و حدود فرکانس ۱۰۰ مگاهرتز کار می‌کنند.

- CPLDها با ابزارهای برنامه‌ریزی CAD که توسط سازندگان تهیه شده است قابل برنامه‌ریزی هستند.

- CPLD به علت سرعت زیاد و ظرفیت بالا (حدود چند هزار گیت در یک IC) برای طراحی نمونه‌سازی سیستم دیجیتال، کنترل‌کننده‌های گرافیکی، شبکه‌های کامپیوتری LAN، قسمت‌های مختلف پروسسور و ... به‌کار برده می‌شوند.

مزایای مدارهای منطقی برنامه‌پذیر CPLD

به‌طور کلی CPLD برای طراحی سیستم‌های دیجیتالی که به تعداد زیاد گیت‌های AND و OR نیاز دارند به‌کار برده می‌شود.

یکی از محاسن CPLD این است که قابل برنامه‌ریزی مجدد می‌باشد.

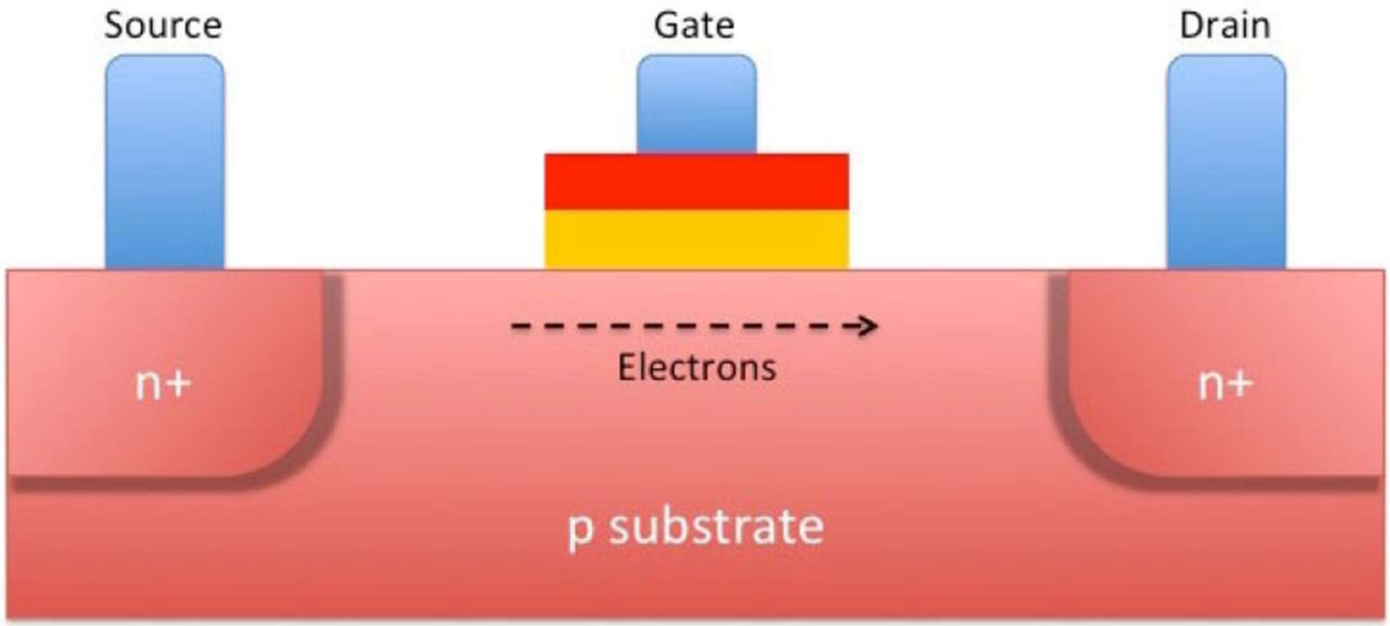
سوییچ‌های قابل برنامه‌ریزی CPLD و PPGA

۱ فیوز ← نخستین نوع سوییچ قابل برنامه‌ریزی با برنامه‌ریزی ارتباط آن قطع می‌شد ← کاربرد در PLAها

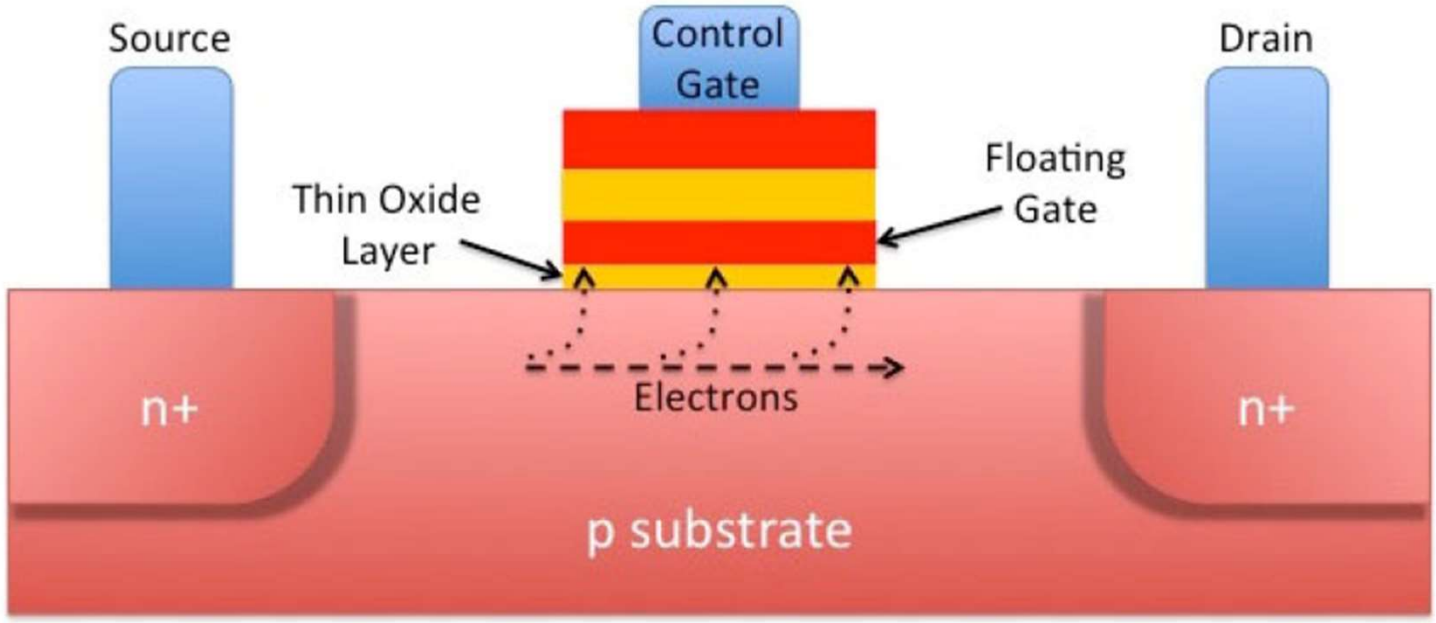
۲ ترانزیستور با گیت شناور
↵ کاربرد در مدارهای مجتمع CMOS با ظرفیت زیاد مانند CPLD مشابه حافظه‌های EPROM و EEPROM

۳ سوییچ‌های با کنترل SRAM
۴ آنتی‌فیوز
← کاربرد در FPGAها

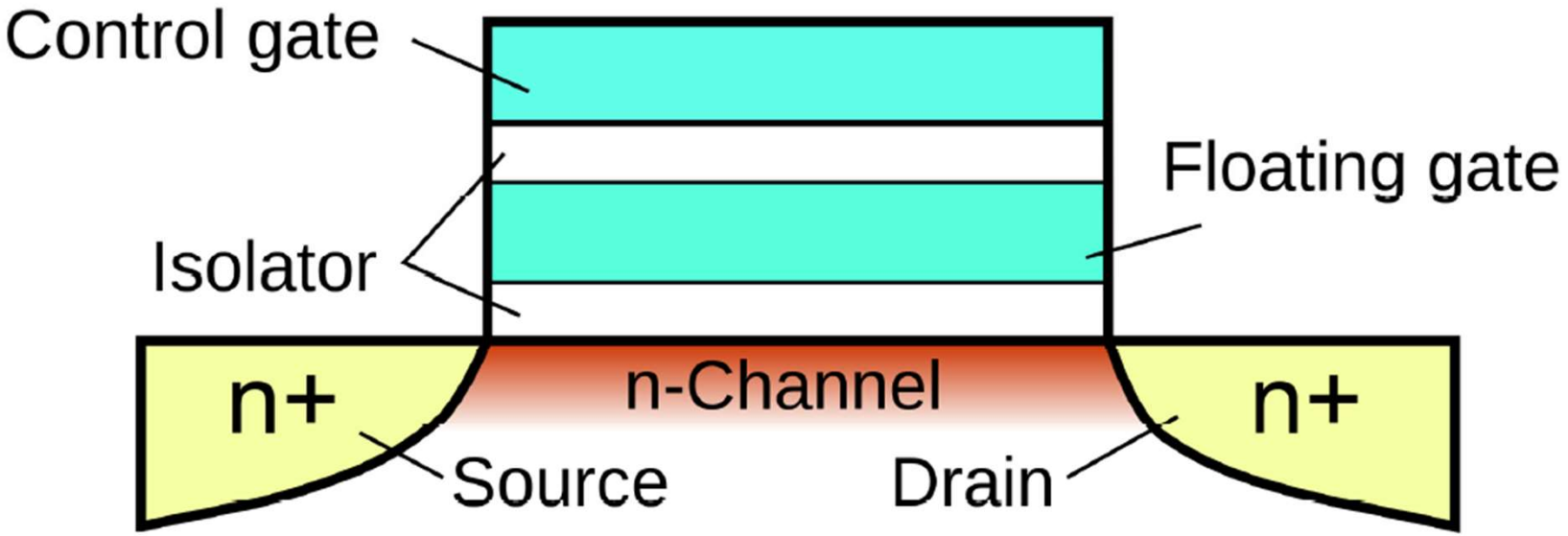
ترانزیستور سوییچ قابل برنامه‌ریزی با گیت شناور



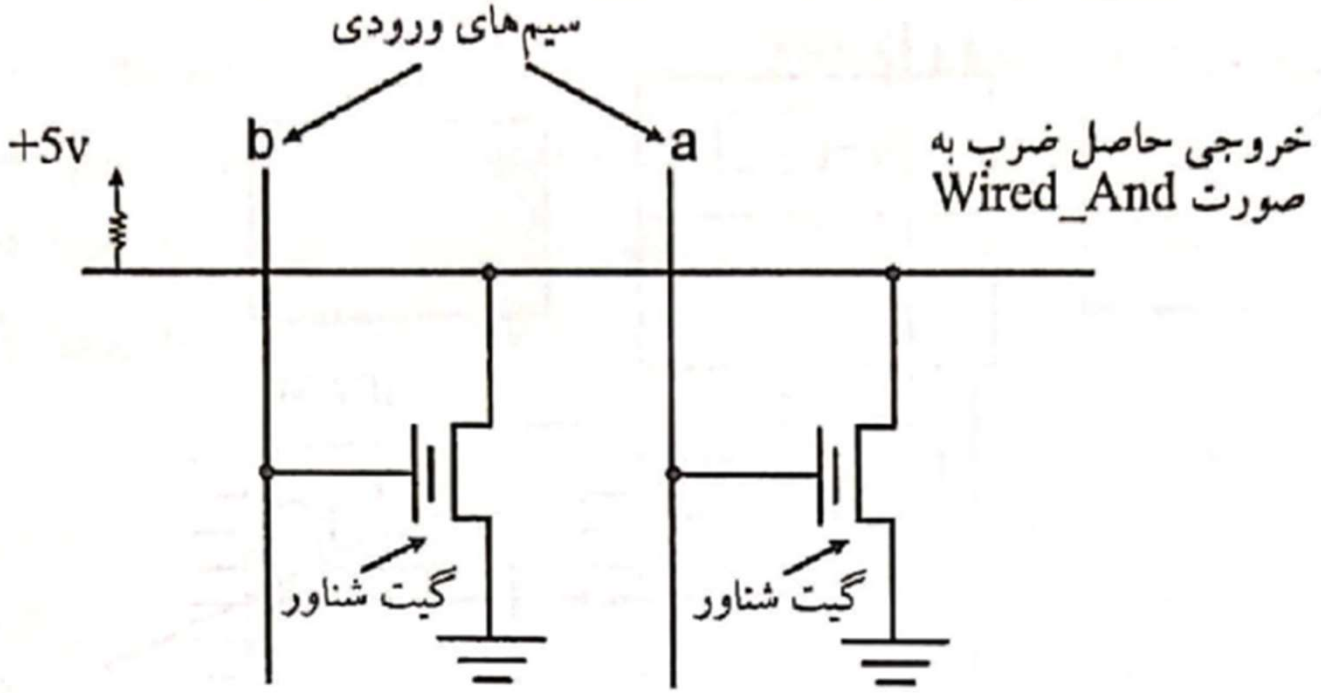
ترانزیستور سوییچ قابل برنامه‌ریزی با گیت شناور



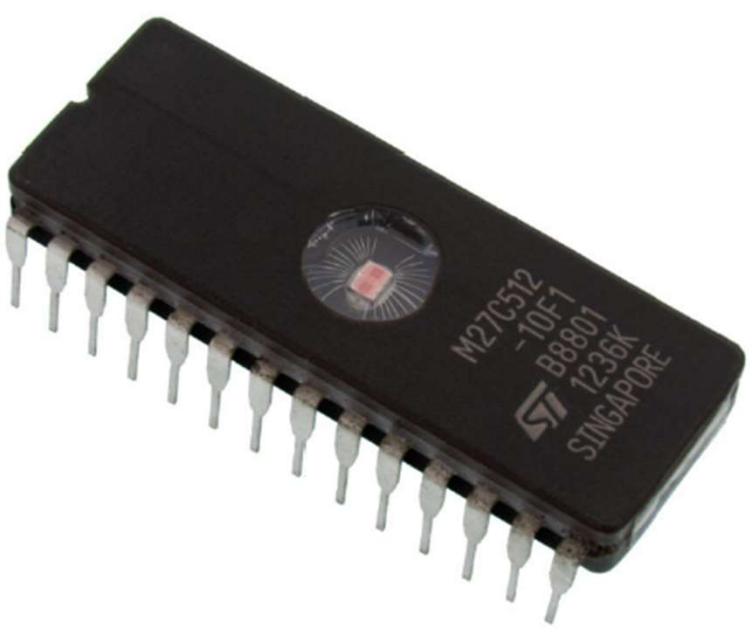
ترانزیستور سوییچ قابل برنامه‌ریزی با گیت شناور

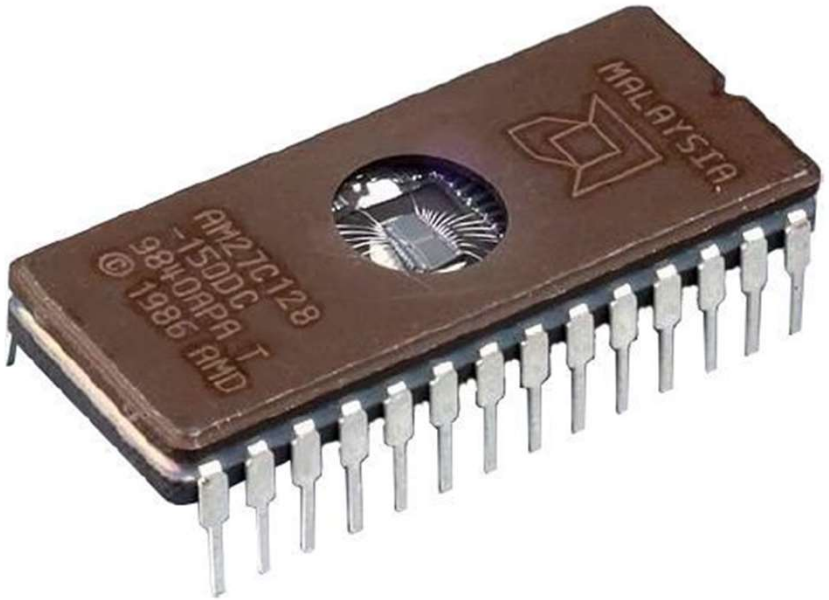


ترانزیستور سوییچ قابل برنامه‌ریزی با گیت شناور

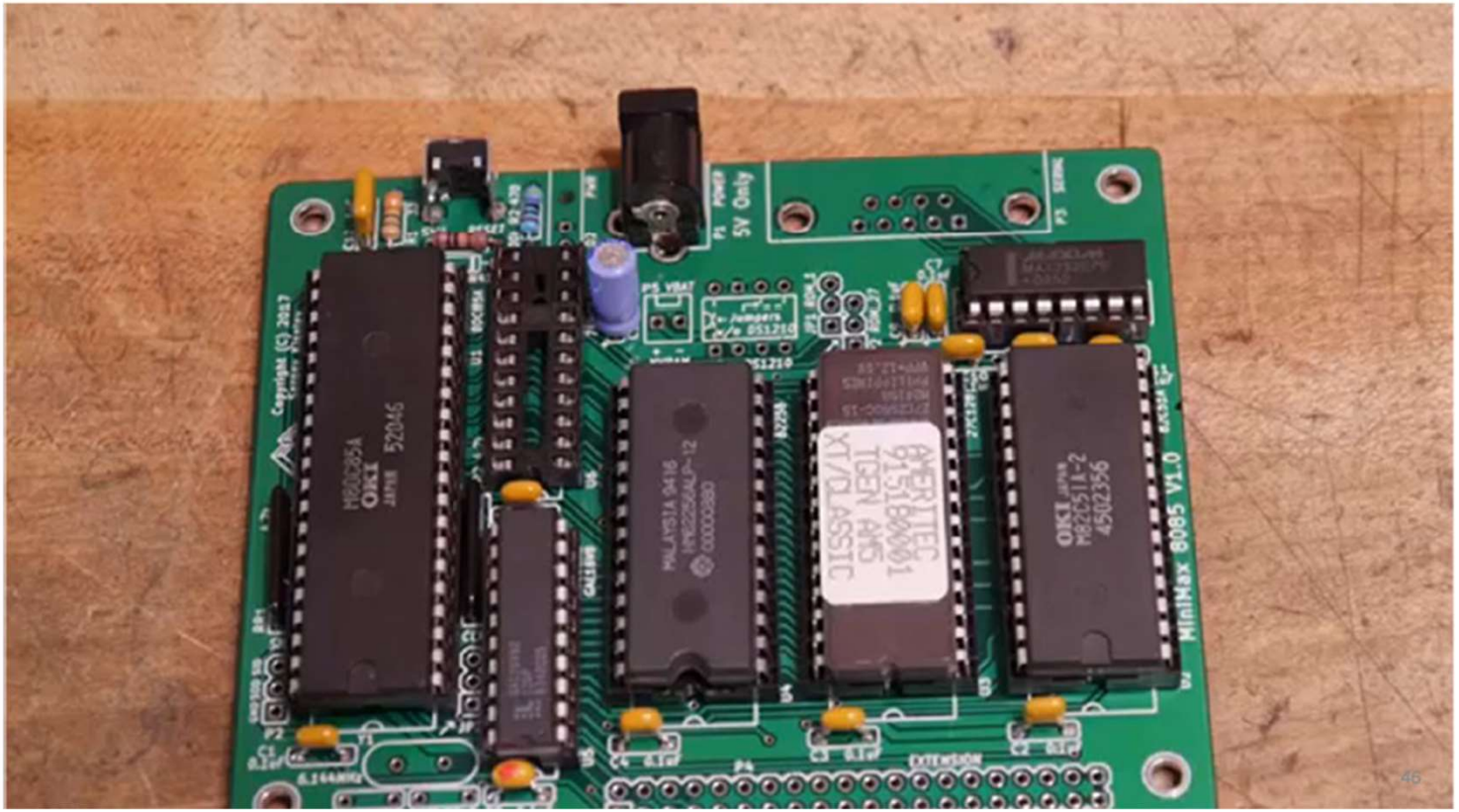


ترانزیستور سوییچ قابل برنامه‌ریزی با گیت شناور (EPROM)

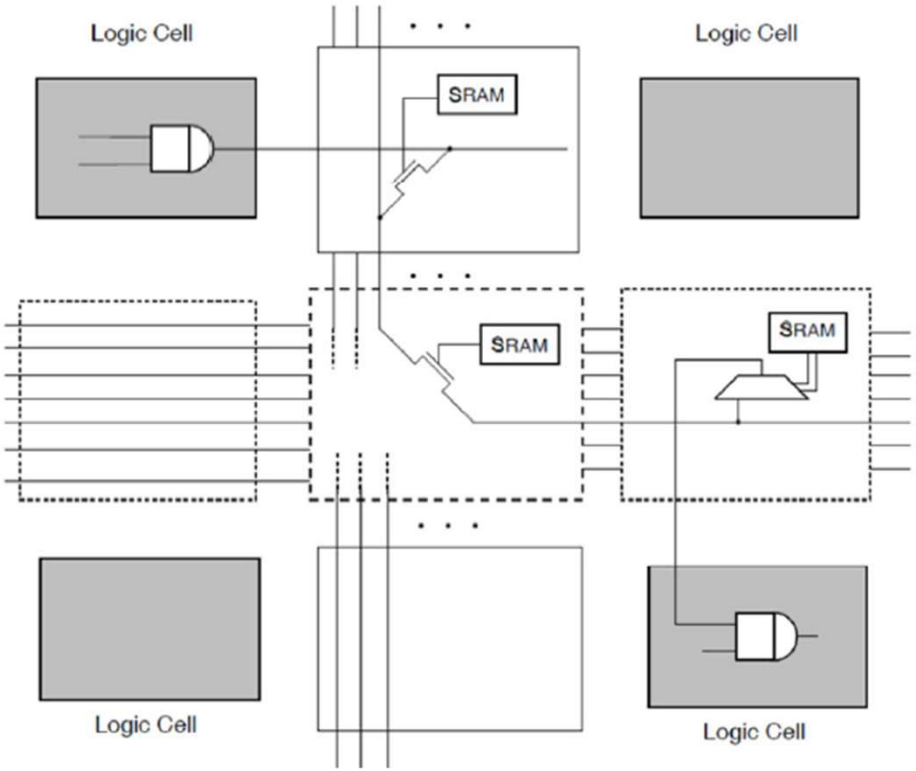




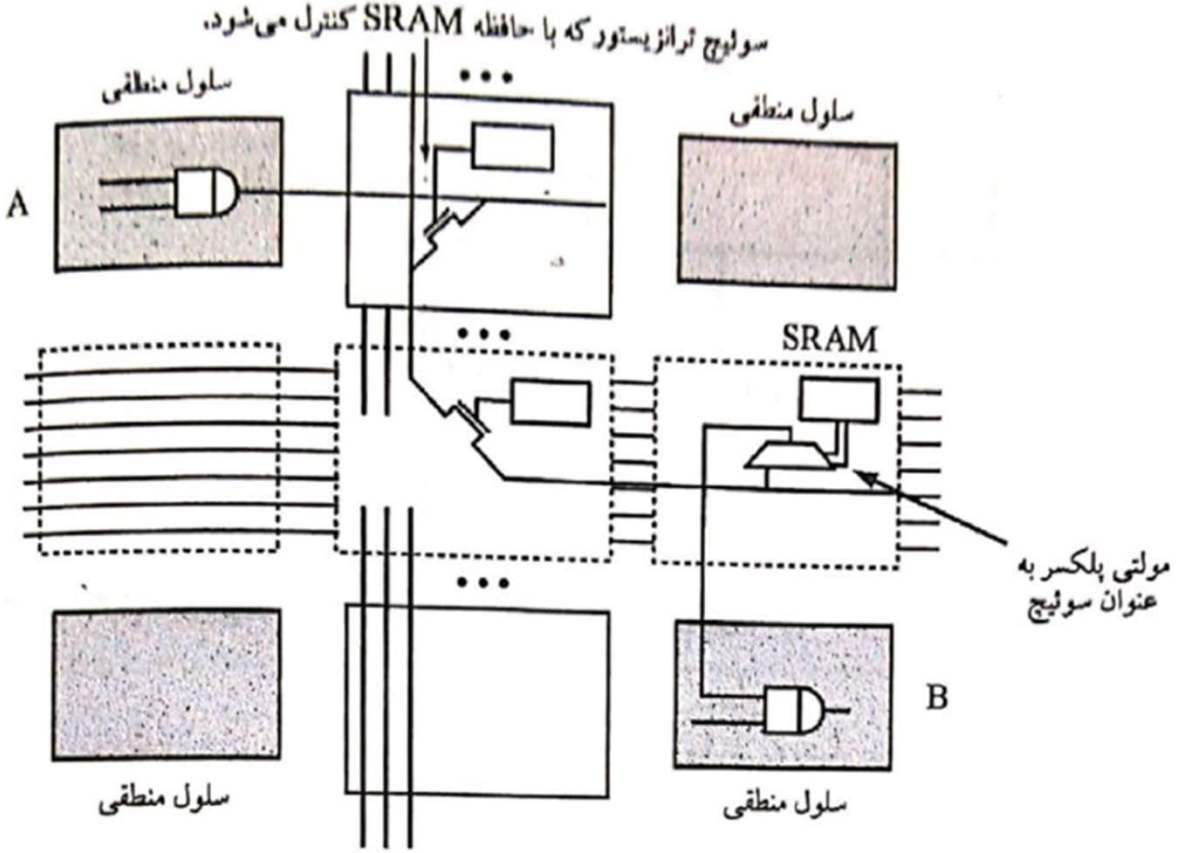
ترانزیستور سوییچ قابل برنامه‌ریزی با گیت شناور (EPROM)



سوییچ قابل برنامه‌ریزی با حافظه SRAM



سوییچ قابل برنامه‌ریزی با حافظه SRAM



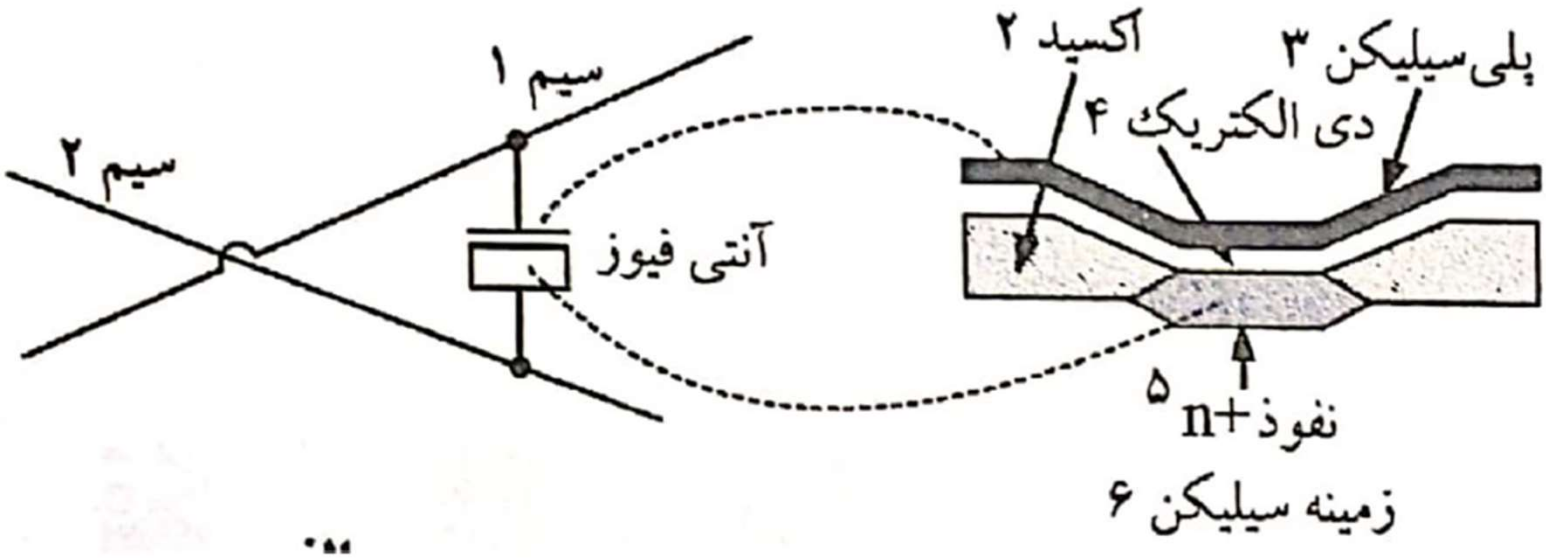
سوییچ قابل برنامه‌ریزی با حافظه SRAM

- در برخی از FPGAها، سوییچ‌های قابل برنامه‌ریزی ترانزیستوری به کار برده می‌شود که با حافظه SRAM برنامه‌ریزی یا کنترل می‌شوند.

- اگر خروجی حافظه SRAM برابر با ۱ شود، سوییچ ترازیستوری بسته شده و دو سیم را به هم متصل می‌کند.

- در برخی از FPGAها از مولتی‌پلکسر به‌عنوان سوییچ استفاده می‌شود که با قرار دادن مقدار حافظه SRAM در قسمت کنترل مولتی‌پلکسر، هر یک از ورودی‌های مورد نظر مولتی‌پلکسر به خروجی آن متصل می‌شود.

سوییچ قابل برنامه‌ریزی آنتی‌فیوز



سوییچ قابل برنامه‌ریزی آنتی‌فیوز

- سوییچ آنتی‌فیوز فقط یک بار برنامه‌ریزی می‌شود لذا FPGA برای همیشه برنامه‌ریزی می‌گردد و قابل برنامه‌ریزی مجدد نمی‌باشد.

- سوییچ SRAM با تغییر مقدار SRAM مجدداً برنامه‌ریزی می‌شود، لذا FPGAهای با سوییچ SRAM را می‌توان مجدداً برنامه‌ریزی نمود.

- اشکال سوییچ SRAM این است که با قطع برق اطلاعات آن از بین می‌رود لذا باید باتری یا وسایل دیگر برای حالت قطع برق در مدار FPGA قرار داده شود.

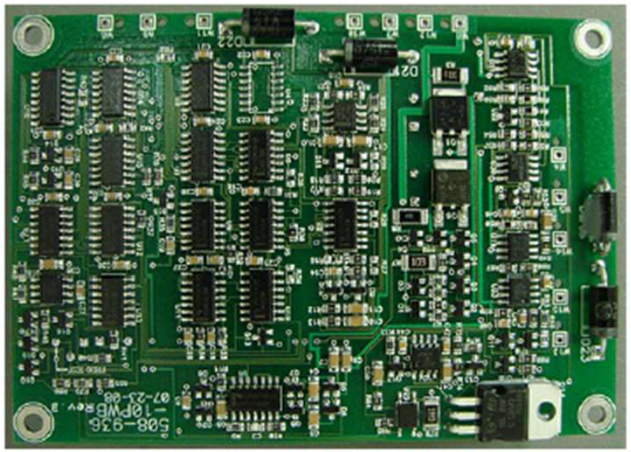
مقایسه سوییچ‌های قابل برنامه‌ریزی

نام	قابلیت برنامه‌ریزی مجدد	فرار بودن	فن‌آوری ساخت
فیوز	خیر	خیر	Bipolar
EPROM	بله خارج از مدار	خیر	UVCMOS
EEPROM	بله درون مدار	خیر	EECMOS
SRAM	بله درون مدار	بله	CMOS
آنتی فیوز	خیر	خیر	CMOS+

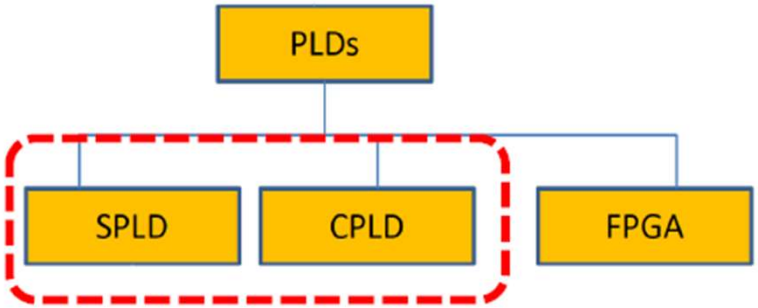
چرا FPGA؟

چند نوع پیاده سازی سخت افزاری داریم؟

با قطعات گسسته و آماده



با سخت افزارهای برنامه پذیر



چرا FPGA؟

چند نوع پیاده سازی سخت افزاری داریم؟

با سخت افزارهای برنامه پذیر

FPGA:
- سریع ✓
- کم توان ✓
- قابل تغییر ✓
- مناسب برای طراحی های پیچیده ✓
- موجود در اندازه های کوچک ✓

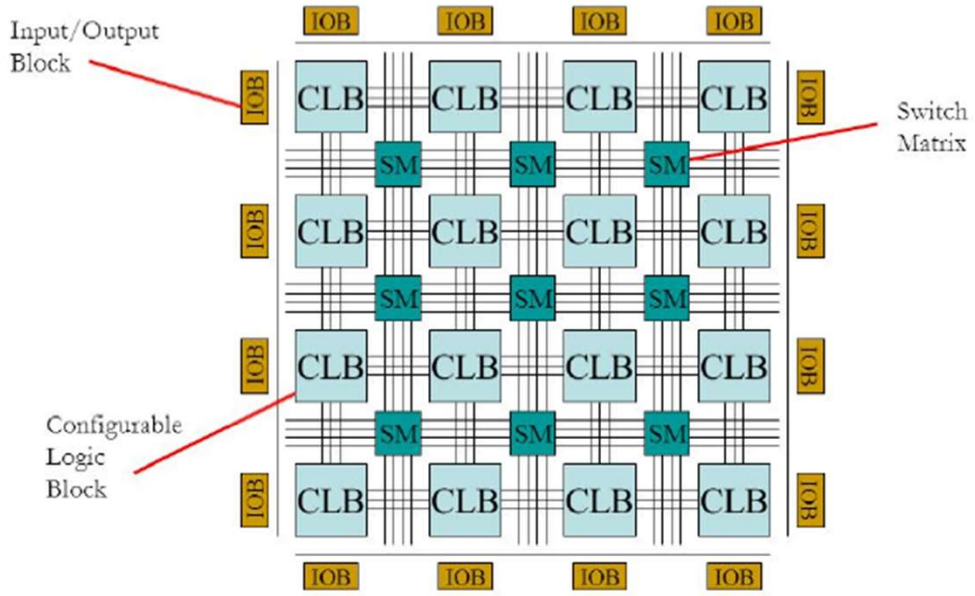
برای طراحی های ساده

با قطعات گسسته و آماده

- تعداد قطعات زیاد ✓
- غیر قابل تغییر ✓
- حساس به نویز ✓
- قطعات محدود ✓
- طراحی زمان بر ✓

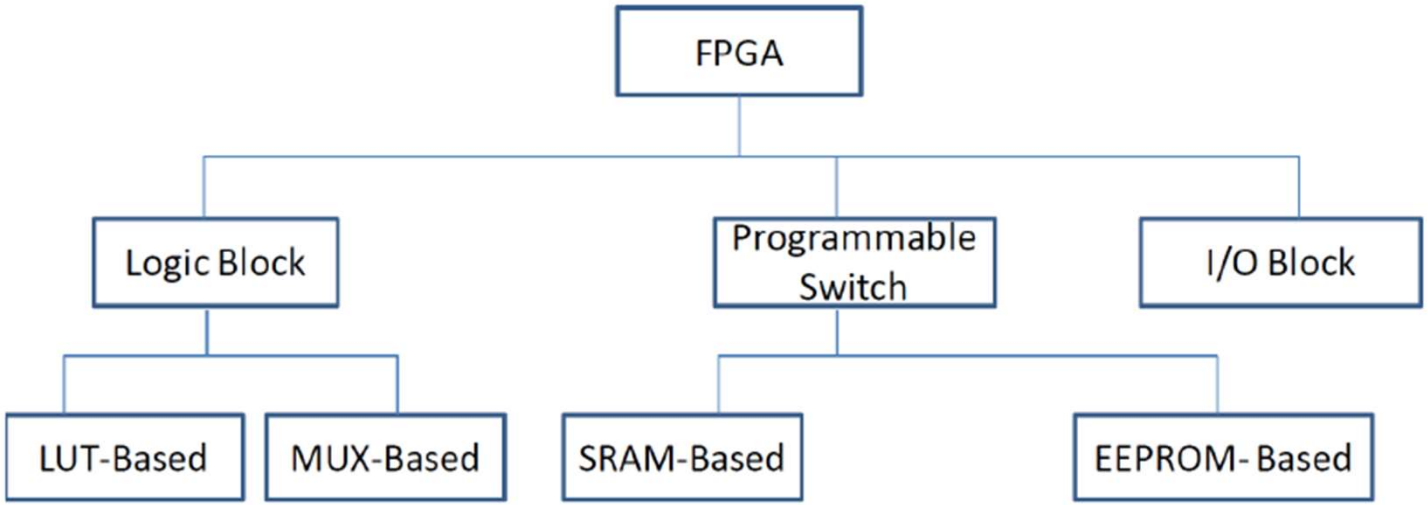
به شکل کلی چند نوع FPGA داریم؟

- یک FPGA از چه چیزی ساخته شده؟



به شکل کلی چند نوع FPGA داریم؟

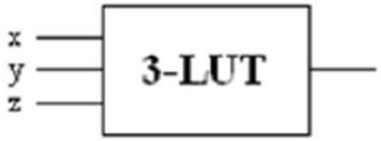
- یک FPGA از چه چیزی ساخته شده؟



به شکل کلی چند نوع FPGA داریم؟

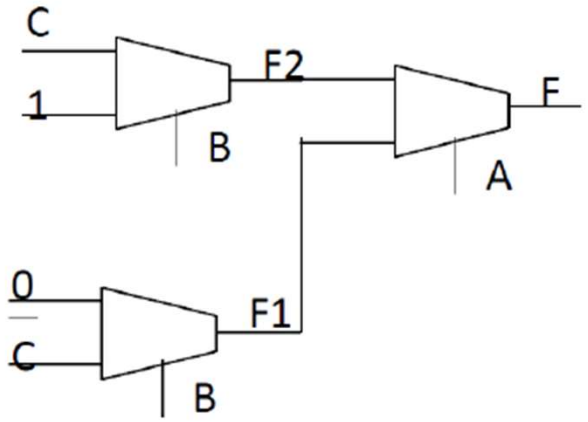
- Logic Block ها چند نوع هستند؟

LUT



x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

MUX



به شکل کلی چند نوع FPGA داریم؟

➢ Logic Block ها چند نوع هستند؟

MUX

✓ تعداد ترانزیستور وابسته به طراحی است.

➢ تاخیر برای توابع پیچیده بیشتر می شود.

➢ روند طراحی پیچیده تر است.

LUT

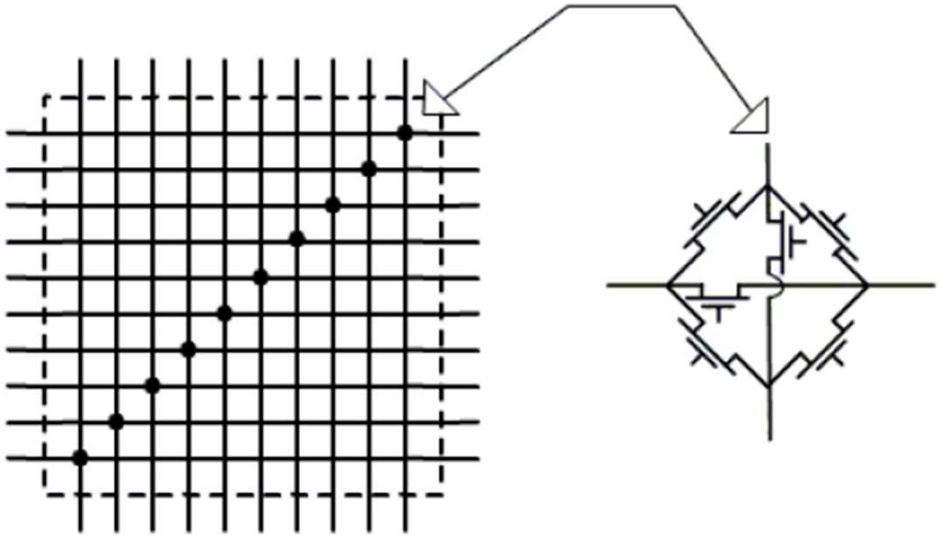
✓ فرآیند طراحی ساده تر است.

✓ پیچیدگی و تاخیر طراحی به پیچیدگی تابع وابسته نیست.

➢ برای ساخت، ترانزیستور زیاد نیاز دارد.

به شکل کلی چند نوع FPGA داریم؟

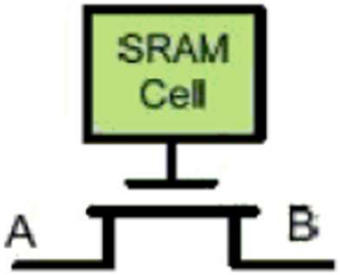
- Switch ها چند نوع هستند؟



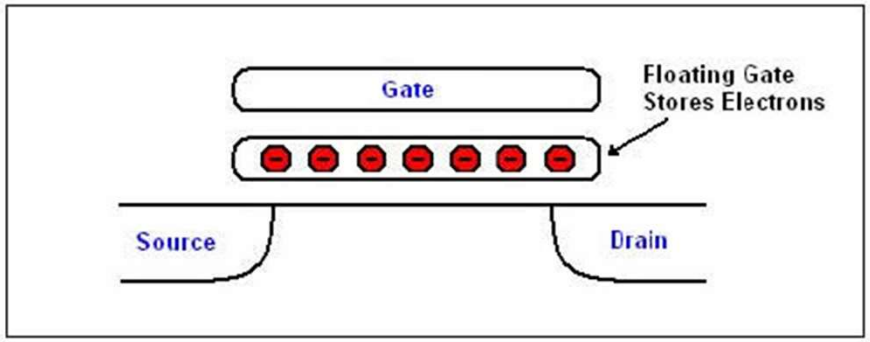
به شکل کلی چند نوع FPGA داریم؟

- Switch ها چند نوع هستند؟

SRAM



EEPROM



به شکل کلی چند نوع FPGA داریم؟

➢ Switch ها چند نوع هستند؟

EEPROM

✓ قابل برنامه ریزی مجدد.
✓ ترانزیستور کمتر می خواهد.
✓ حافظه آن غیر فرار است.

➢ گران تر است.

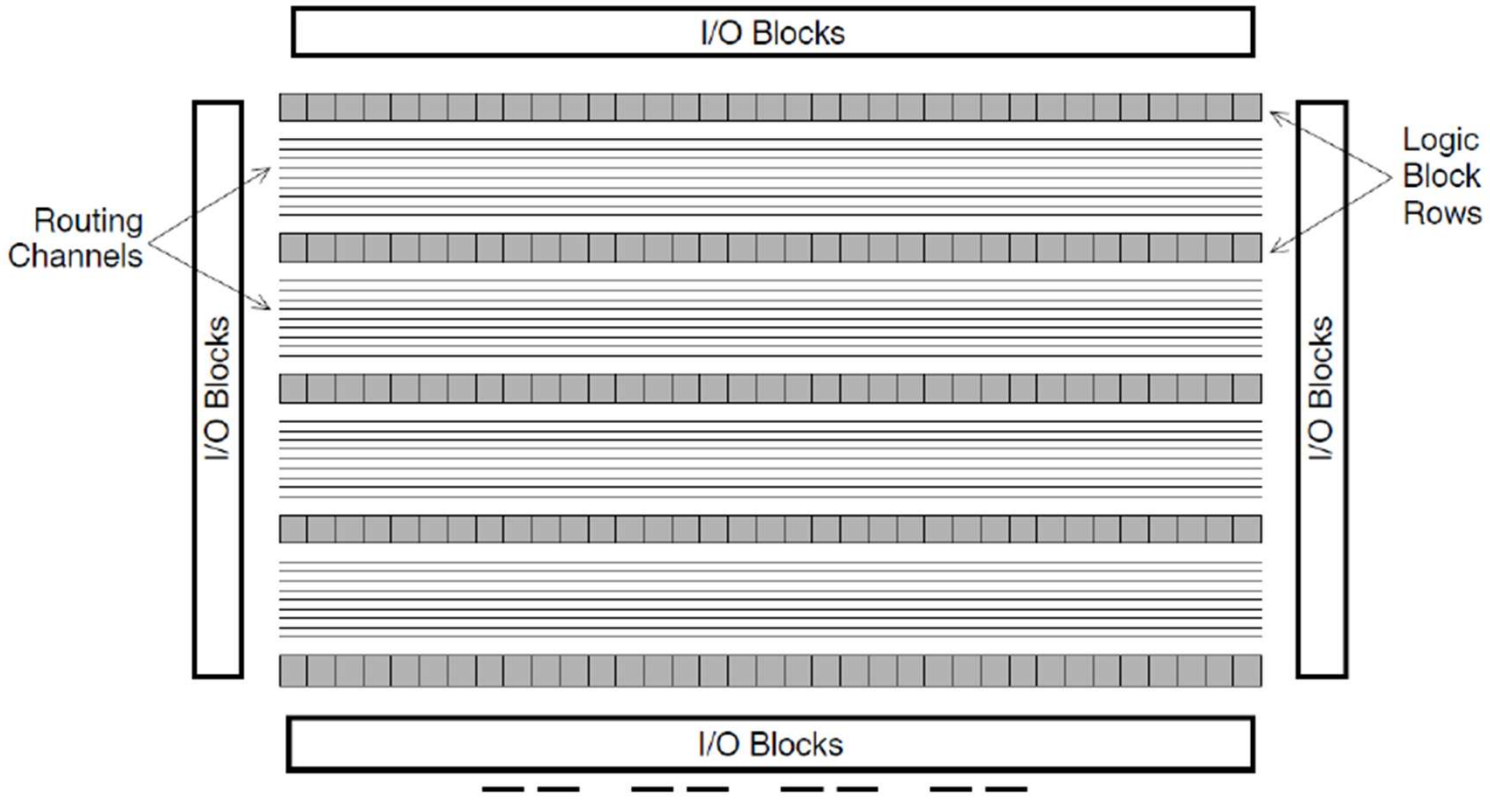
SRAM

✓ قابل برنامه ریزی مجدد.

➢ ترانزیستور زیادی نیاز دارد.

➢ حافظه آن فرار است.

ساختار FPGA با سلول‌های منطقی به‌صورت ردیفی



دو اختلاف اساسی بین CPLD و FPGA

امکانات

FPGAها با ظرفیت در حدود چند هزار گیت، دارای امکانات بیشتری نسبت به CPLDها می‌باشند.

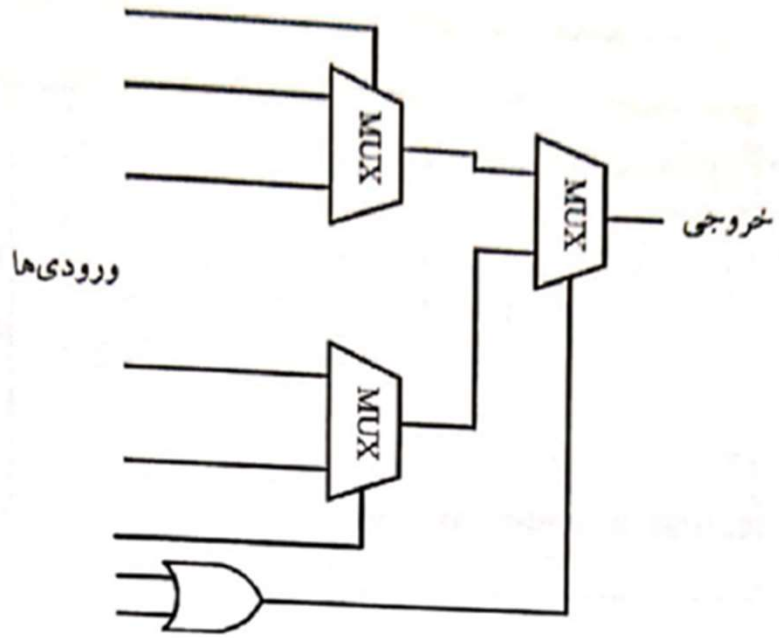
می‌توان با استفاده از FPGAها مدارهای پیچیده‌تر و بزرگتری را طراحی نمود

سرعت

در FPGA، سوییچ‌های قابل برنامه‌ریزی زیادی برای اتصال بلوکهای FPGA به‌کار می‌رود که دارای تأخیر می‌باشند.

FPGAها تأخیر بیشتری نسبت به CPLDها و PALها دارند.

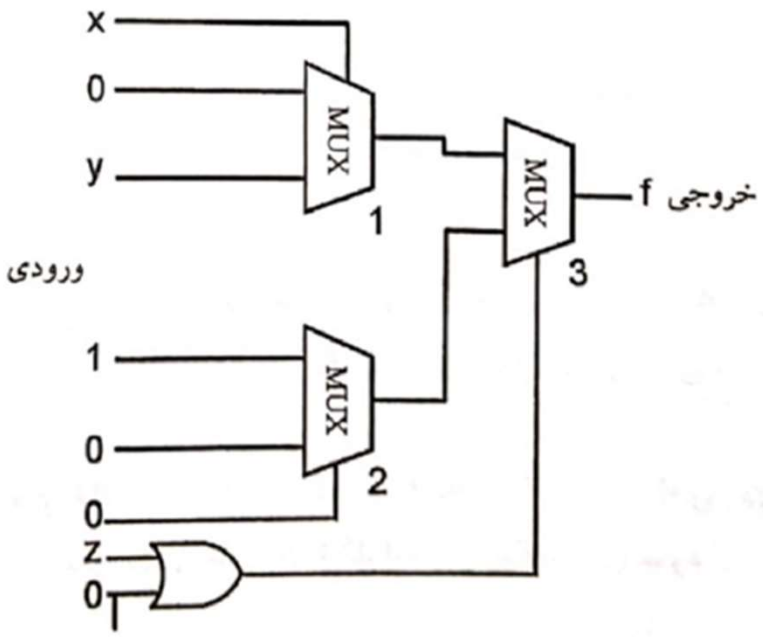
سلول منطقی FPGA مبتنی بر MUX
خانواده Act-1 از شرکت Actel



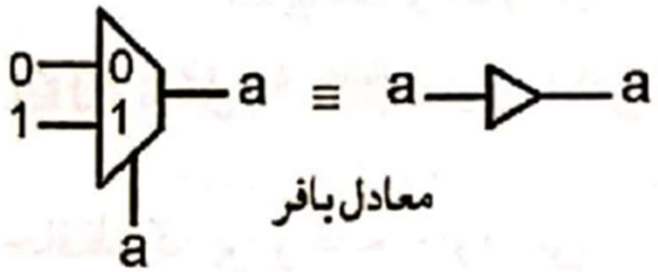
سلول منطقی FPGA مبتنی بر MUX
خانواده Act-1 از شرکت Actel

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	0	1	1
1	1	0	1
1	1	1	1

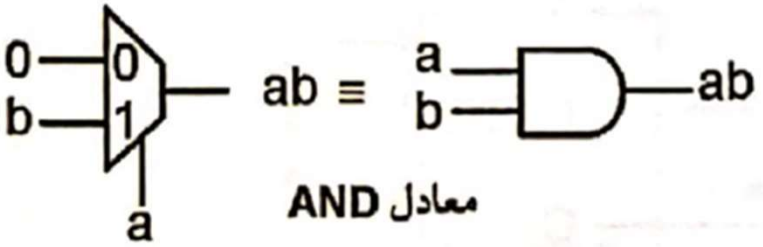
$$f = xy + z$$



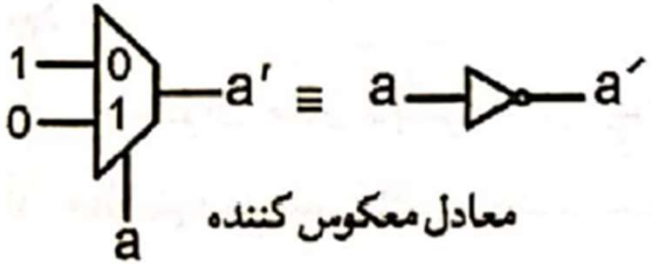
سلول منطقی FPGA مبتنی بر MUX
خانواده Act-1 از شرکت Actel



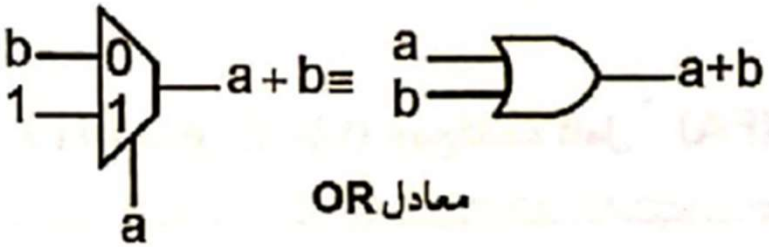
معادل بافر



معادل AND



معادل معکوس کننده

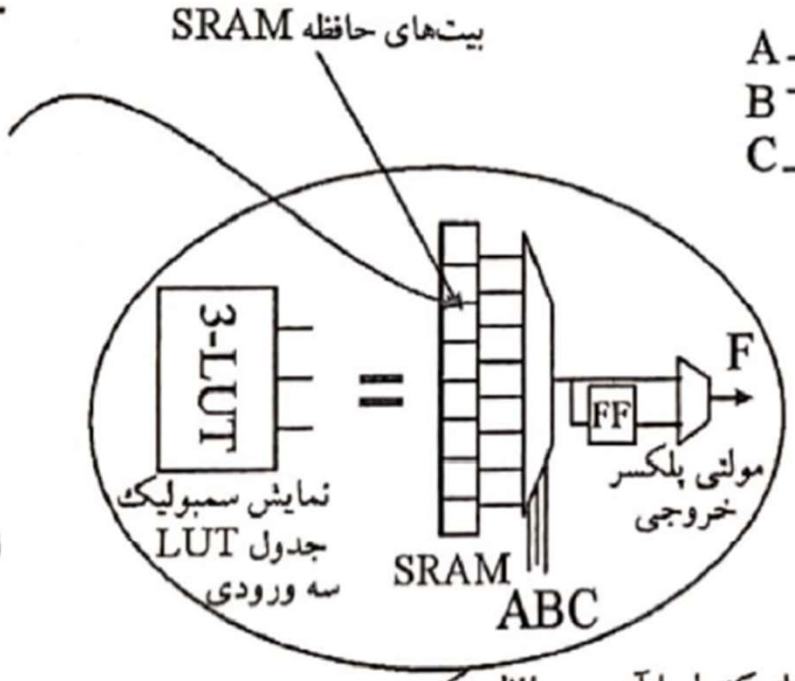


معادل OR

سلول منطقی FPGA مبتنی بر LUT

ABC	F
000	0
001	1
010	0
011	1
100	0
101	1
110	1
111	1

الف: جدول درستی تابع F
F=AB+C

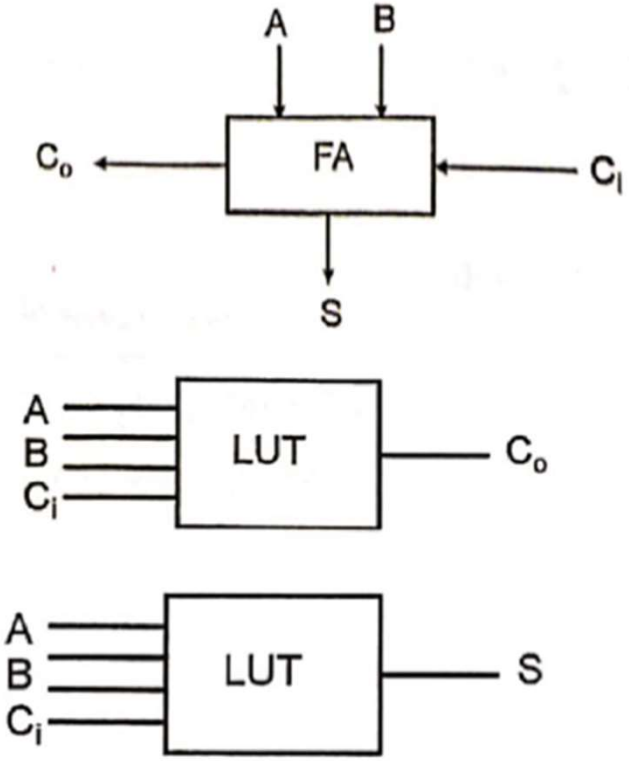


ورودی‌های کنترل با آدرس حافظه یک بیتی

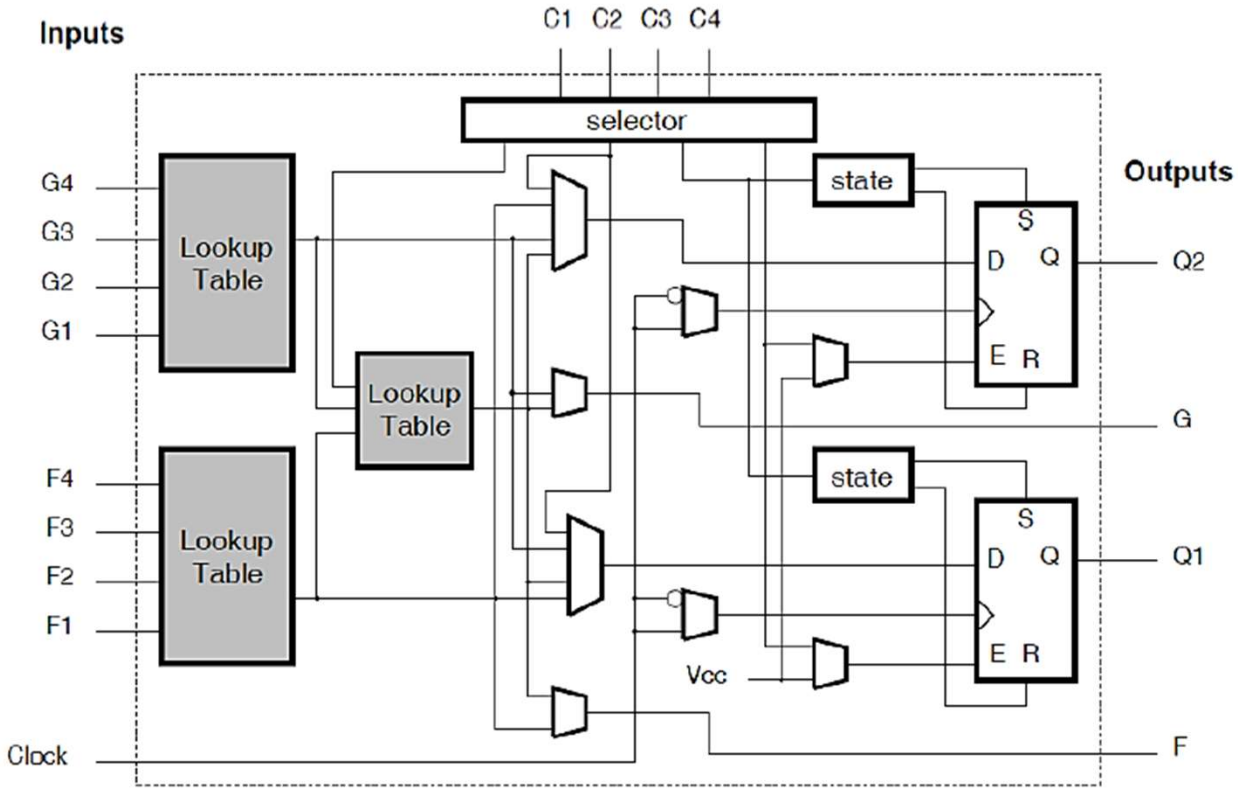
ج: مدار تابع

سلول منطقی FPGA مبتنی بر LUT

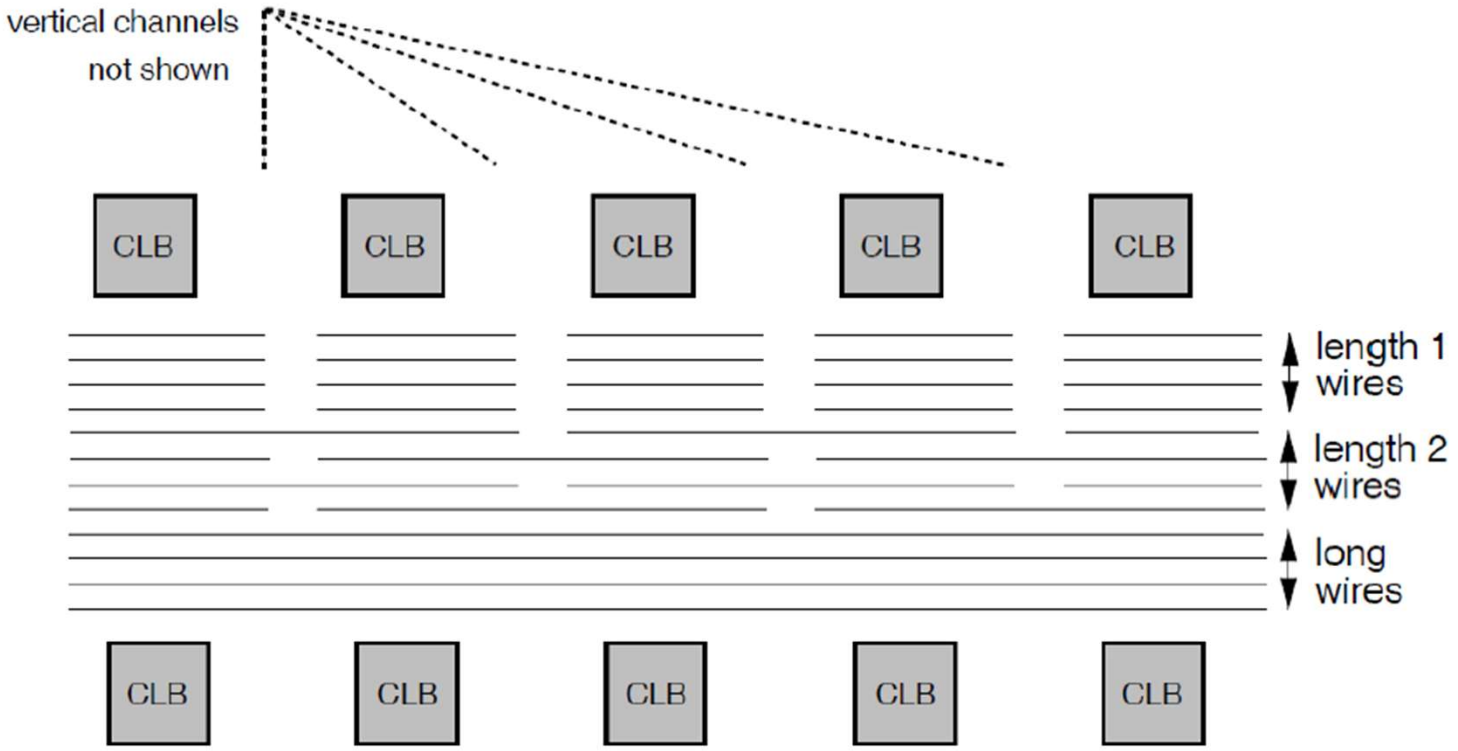
A	B	C_I	S	C_O
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



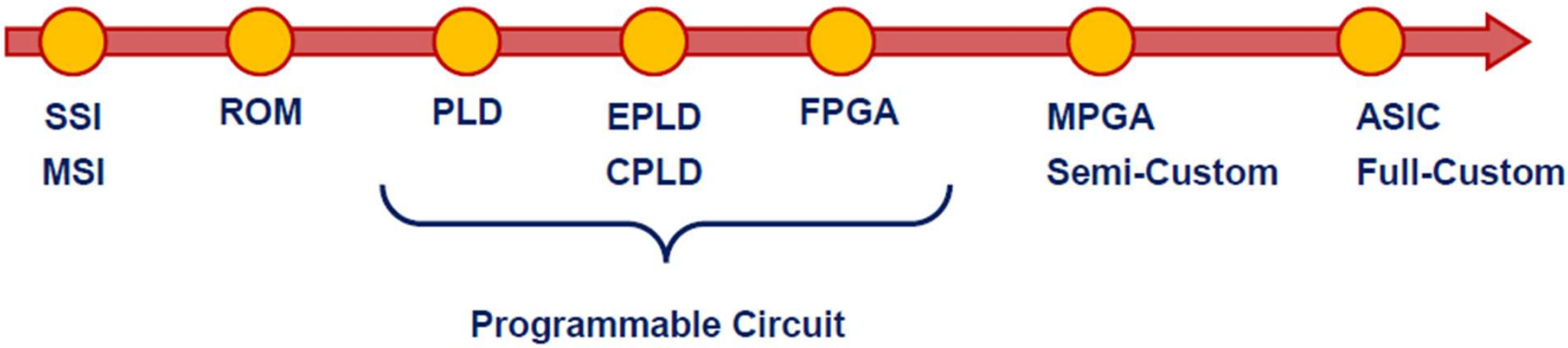
سلول منطقی CLB، FPGAهای سری XC4000 شرکت Xilinx



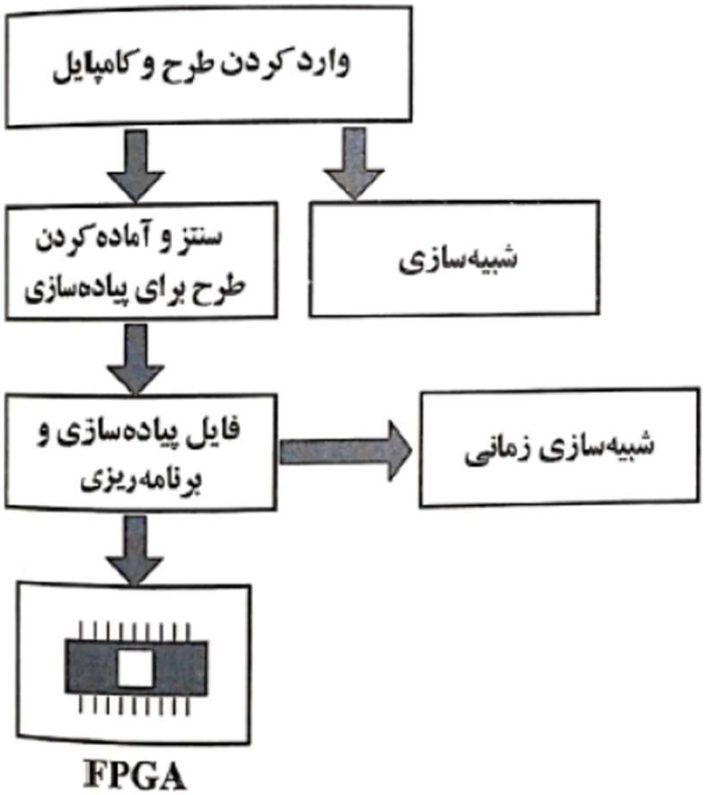
آرایه بلوک‌های منطقی CLB، FPGAهای سری XC4000



مقایسه تراشه‌ها از دیدگاه تجمع‌پذیری



روش طراحی سیستم‌های دیجیتال با FPGA



وارد کردن طرح اولیه و کامپایل

۱- ترسیم شماتیک با ادیتور گرافیک در محیط ابزار برنامه‌ریزی PPGA

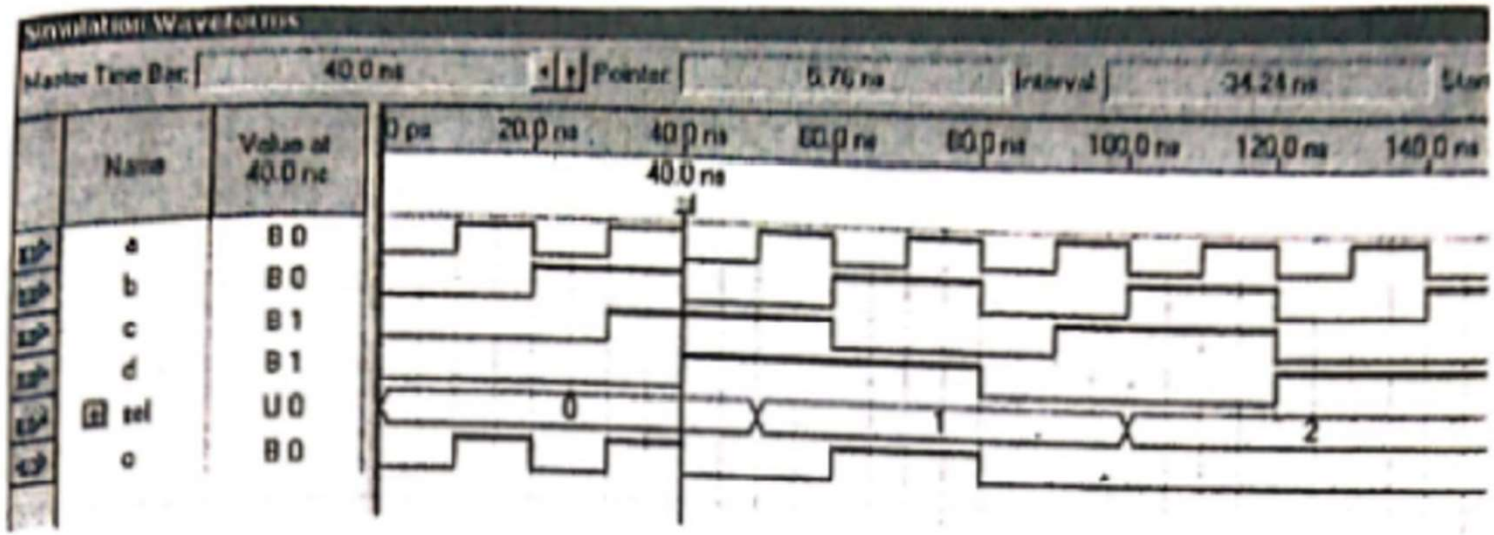
۲- توصیف طرح با زبان توصیف سخت‌افزار VHDL با استفاده از ویرایشگر متن HDL

وارد کردن طرح اولیه و کامپایل

```
library ieee;
use ieee.std_logic_1164.all;
entity pro  is
     port(a,b,c,d: in std_logic;
                        sel: in  std_logic_vector(1 downto 0);
                        o: out std_logic);
end;
architecture  MULCA     of pro  is
begin
process(a,b,c,d,sel)
begin
      case sel is
                        when "00" => o <=a;
                        when "01" => o <=b;
                        when  others => null;
       end case;
end process;
end:
```

دانشگاه سمنان
Semnan University
پردیس فرزانگان

شبیه‌سازی


Simulation Waveforms
Master Time Bar: 40.0 ns
Pointer: 5.76 ns
Interval: 34.24 ns
Name
Value at 40.0 ns
a
b
c
d
sel
o
B 0
B 0
B 1
B 1
U 0
B 0
20.0 ns
40.0 ns
60.0 ns
80.0 ns
100.0 ns
120.0 ns
140.0 ns
0
1
2

وارد کردن طرح اولیه، کامپایل و شبیه‌سازی

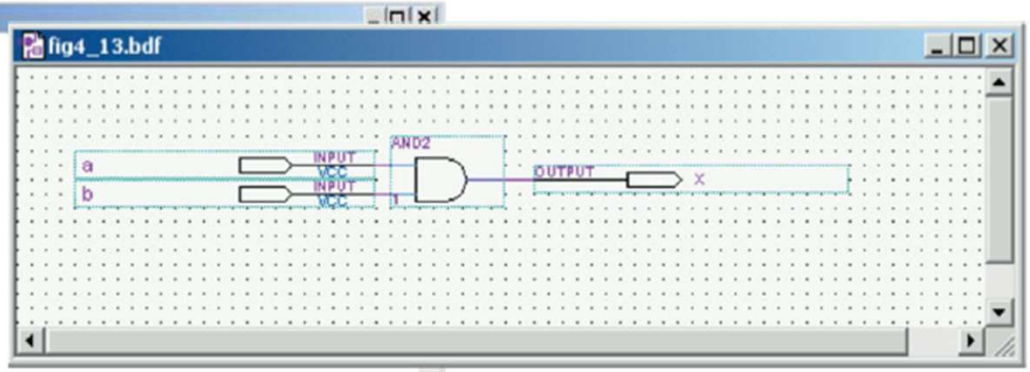
fig4_13.vhd

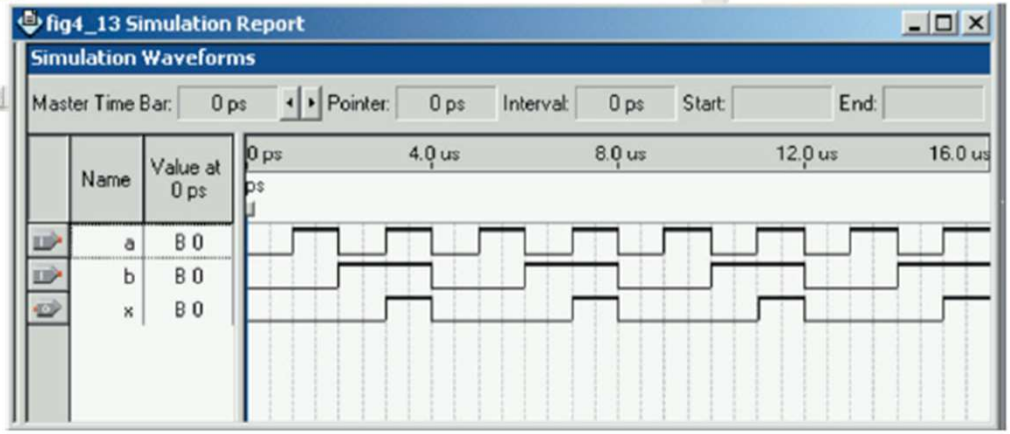
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY fig4_13 IS
    PORT(
        a, b: IN std_logic;
        x:    OUT std_logic);
END fig4_13;

ARCHITECTURE arc OF fig4_13 IS
    BEGIN
        x<=a AND b;
END arc;
```

Line 17 Col 1 INS





سنتز و آماده کردن طرح برای پیاده‌سازی

- سنتز یعنی تبدیل برنامه VHDL به معادل عناصر منطقی مانند جمع‌کننده، مولتی‌پلکسر، ثبات، دیکدر، فلیپ‌فلاپ و .... و اتصالات مربوطه که به آن netlist گفته می‌شود.

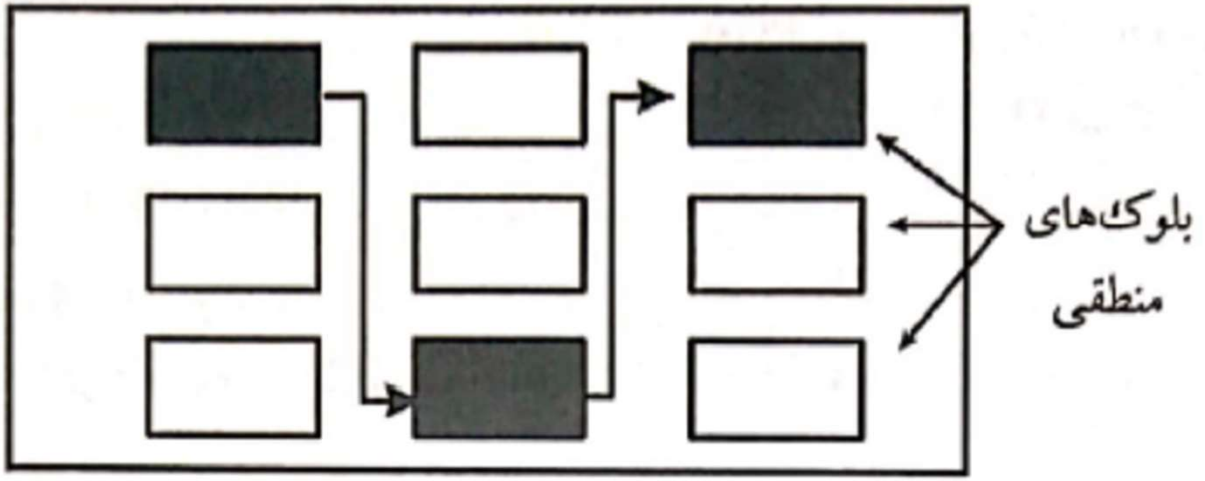
- چون طرح اولیه منطقی معمولاً به‌طور بهینه نیست، در ابزارهای برنامه‌ریزی FPGA الگوریتم‌هایی وجود دارد که طرح اولیه را بهینه می‌کند.

فایل پیاده‌سازی

- در این مرحله نوع FPGA را مشخص می‌کنیم و با استفاده از فایل سنتز که شامل بلوک‌های منطقی و ارتباطات آنها می‌باشدفایلی با یک سری بیت‌های باینری تولید می‌شود که با استفاده از آن تعدادی سوییچ‌ها و بلوک‌های منطقی FPGA می‌توانند برنامه‌ریزی شوند.

- در این مرحله محل بلوک‌های منطقی مورد نیاز در FPGA مشخص و ارتباطات آنها در سیلیکن FPGA نیز مشخص می‌شود.

انتخاب بلوک‌های منطقی (Placement) و ارتباط (Routing) آنها در FPGA

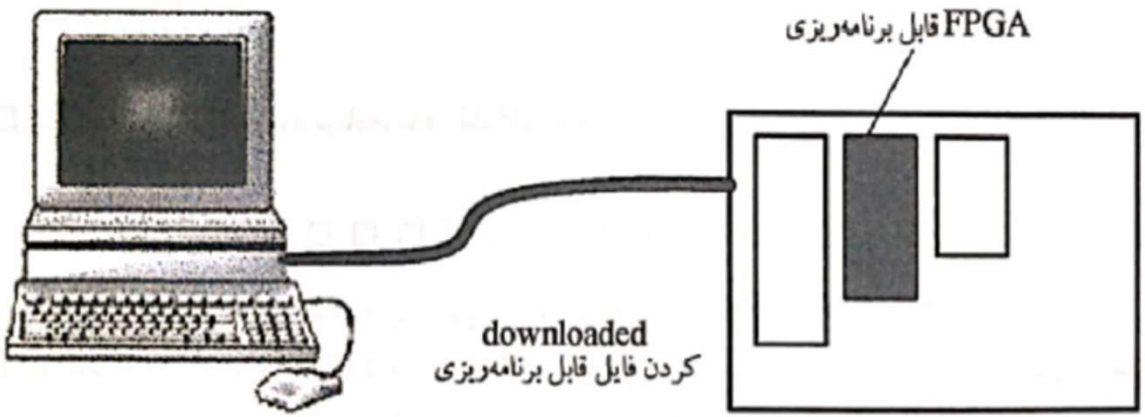


شبیه‌سازی زمانی (اختیاری)

- در این مرحله مقدار تأخیر گیت‌های بلوک‌های منطقی مشخص می‌شود در نتیجه زمان‌بندی و تأخیر واقعی مدار بررسی می‌گردد.

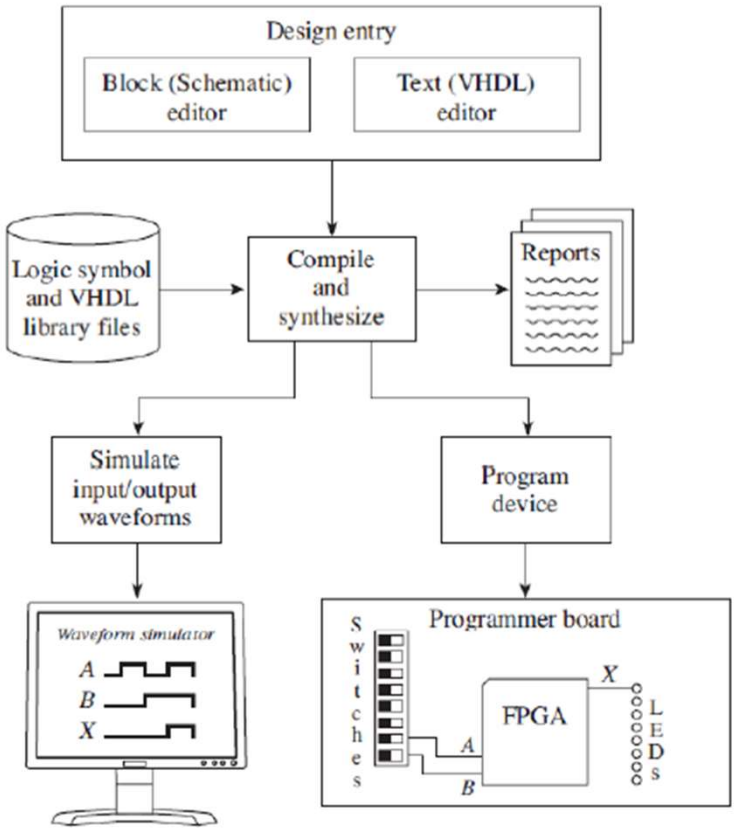
- برخی ابزارهای برنامه‌ریزی FPGA این مرحله را مرحله Verification می‌نامند.

برنامه‌ریزی FPGA



کامپیوتر با ابزار برنامه‌ریزی FPGA
مانند QUARTUS II یا
MAX+PLUS II یا ...

روش طراحی سیستم‌های دیجیتال با FPGA



دانشگاه سمنان
Semnan University
پردیس فرزانگان