- **کلاس تمرین:**
  - دوشنبه (۲۰ اردیبهشت)، ساعت ۱۲:۱۵
  - کلاس ۲۰۲

- **مشاهدهٔ برگه‌های امتحان:**
  - دوشنبه بعد (۲۷ اردیبهشت)، ساعت ۱۲:۰۰ تا ۳:۰۰

فنون طراحی

فنون طراحی

- **هدف:**
  - ❑ آشنایی با تکنیک‌ها برای
    - کیفیت بالای طرح
    - کارامدتر شدن فرایند طراحی

طراحی پودمانی

- **طراحی پودمانی (modular):**
  - سیستم دیجیتال بزرگ: برحسب تعدادی بلوک یا *پودمان*
  - آسان شدن طراحی
    - کنترل پیچیدگی
    - هر پودمان جدا
  - آسان شدن اشکال‌زدایی
  - طراحی موازی پودمان‌ها
    - توسط یک طراح یا یک تیم

طراحی مبتنی بر IP

- **طراحی مبتنی بر IP: مزایا**
  - ❑ به‌صرفه نبودن طراحی همهٔ پودمان‌ها
    - پودمان‌های آماده: بلوک IP
  - ❑ Time to market ←
  - ❑ کاهش هزینهٔ طراحی ←
  - ❑ عدم نیاز به همهٔ مهارت‌ها و تخصص‌های پروژه
    - تمرکز روی مزایای تخصصی شرکت ← مزیت رقابتی

طراحی مبتنی بر IP

- **طراحی مبتنی بر IP، چالش‌ها:**
  - عدم اطمینان از بی‌اشکال بودن IP
  - عدم اطلاع از کیفیت IP
  - عدم آگاهی از نحوهٔ استفادهٔ درست از IP
  - مجتمع‌سازی:
    - اتصال IP به سایر اجزا همیشه سرراست نیست
    - ناهماهنگی Clocking
    - ناهمخوانی ورودی‌ها و خروجی‌های IP با بلوک‌های مرتبط
  - عدم تمایل طراحان به استفاده از طرح‌های دیگران

طراحی مبتنی بر IP

- **طراحی مبتنی بر IP، راه حل‌ها:**
  - ❑ مقاومت در برابر طراحی از ابتدا برای همه چیز
  - ❑ تحویل گرفتن IP با مستندات کافی:
    - قابلیت‌های IP
    - نحوهٔ به‌کارگیری IP
    - نحوهٔ درستی‌سنجی IP

تولید IP

- **توصیه: تبدیل بلوک‌های پروژه به IP برای آینده**
  - تکمیل مستندات بلوک
  - افزودن قابلیت‌های اضافه برای کلی شدن IP جهت کار در شرایط گوناگون
    - با کمک generic، if-generate، for-generate، attributeها
    - لزوم پیش‌بینی شرایط گوناگون
  - اجتناب از افراط در IP کردن پودمان‌ها
    - بلوک‌های کوچک
    - بلوک‌های کم‌کاربرد
      - ← اتلاف هزینه و وقت

تولید IP

- **تولید IP، کدام پودمان‌ها:**
- **معیارها:**
  - ❑ عملکرد آن به گونه‌ای باشد که در کاربردهای گوناگون قابل استفاده باشد.
  - ❑ عملکرد آن را بتوان به آسانی توصیف کرد.
  - ❑ تعداد ورودی‌ها و خروجی‌های آن زیاد نباشد ← دشواری ارتباط با سایر بلوک‌ها
  - ❑ قابلیت توصیف پارامتری برای شرایط مختلف را دارا باشد.

تولید IP

- **مشخصات یک IP خوب:**

کد توصیف آن واضح و قابل فهم باشد.

ورودی‌ها و خروجی‌های آن با ثبات‌هایی ثبت شوند تا در ایجاد ارتباط با بلوک‌های دیگر مشکلات ناهمگامی با کلاک به وجود نیاید.

به تنهایی و به آسانی قابل تست باشد و بردارهای تست و testbench فراهم شده باشد.

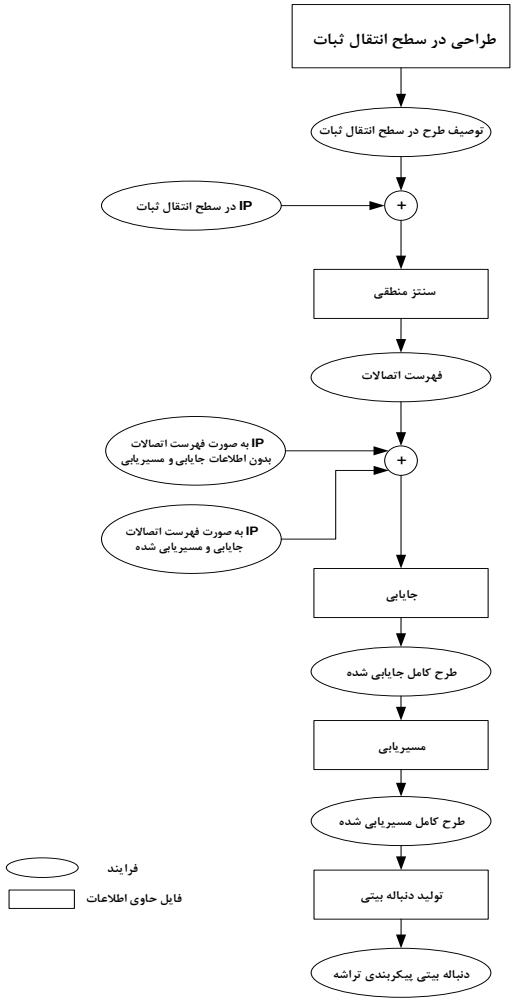
برای بلوک‌های بزرگ، در صورت امکان برای ارتباط با بیرون از بلوک، از استانداردهای واسط (آمبا، ویشبون، ...) استفاده شود.

- IP ← به آسانی با سایر بلوک‌های سازگار با این استانداردها ارتباط برقرار می‌کند ← نیاز به مدار واسط دیگری نیست.
- آسان شدن طراحی تیمی و ارتباط با طراحان سایر بلوک‌ها

طراحی مبتنی بر IP

- **شرایطی که طراحی درون سازمانی توصیه می‌شود:**
  - (غیرضروری یا حتی زیان‌بخش)
    - ❑ کارایی بلوک مهم است ولی بلوک IP فاقد کارایی مورد نیاز است
  - قابلیت اضافی (معمولاً) = کاهش کارایی
    - ❑ خودمان ویژگی‌های بیشتری می‌گذاریم
    - ❑ بلوک در زمینهٔ تخصصی ما (مزیت ما)
  - می‌توانیم قابلیت رقابتی ایجاد کنیم
    - ❑ IP تأثیر زیادی در هزینه و زمان ندارد
    - ❑ هنوز برای این FPGA موجود نیست (یا خیلی گران)
    - ❑ IP قابل اطمینان نیست

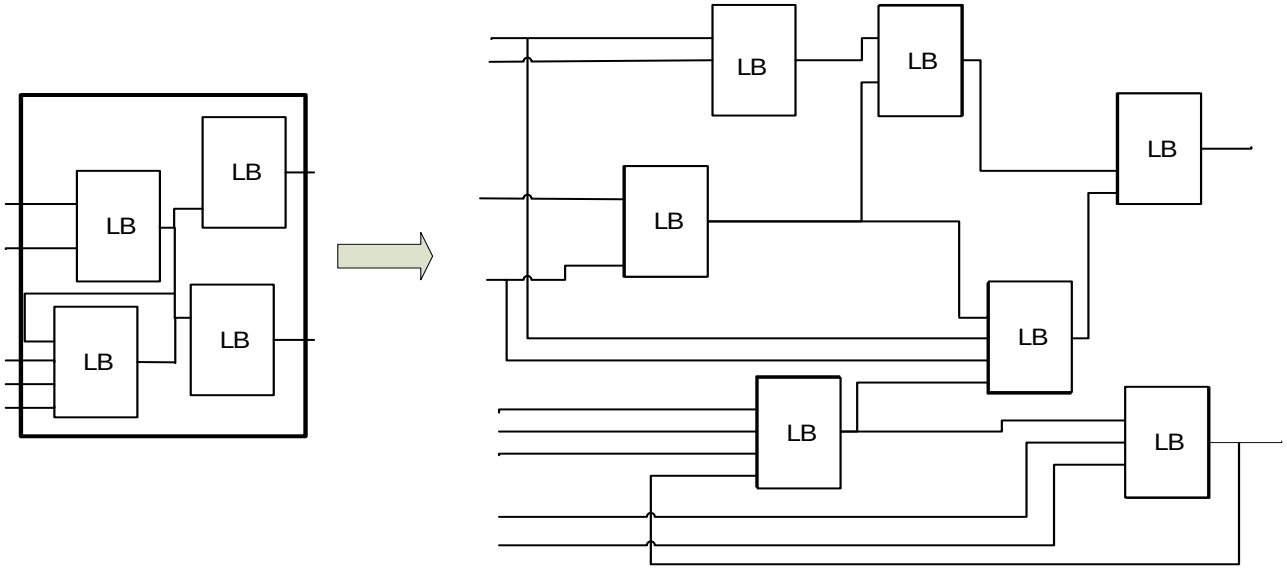
انواع IP



- **دسته‌بندی از دیدگاه سطح تجرید توصیف IP:**
  - مرحله‌ای که از IP استفاده می‌شود
    - RTL:
    - Verilog/VHDL
  - سنتز IP در کنار سایر بلوک‌ها

انواع IP

- **دسته‌بندی از دیدگاه سطح تجرید توصیف IP:**
  - منطقی:
    - به صورت netlist
    - Fit نشده

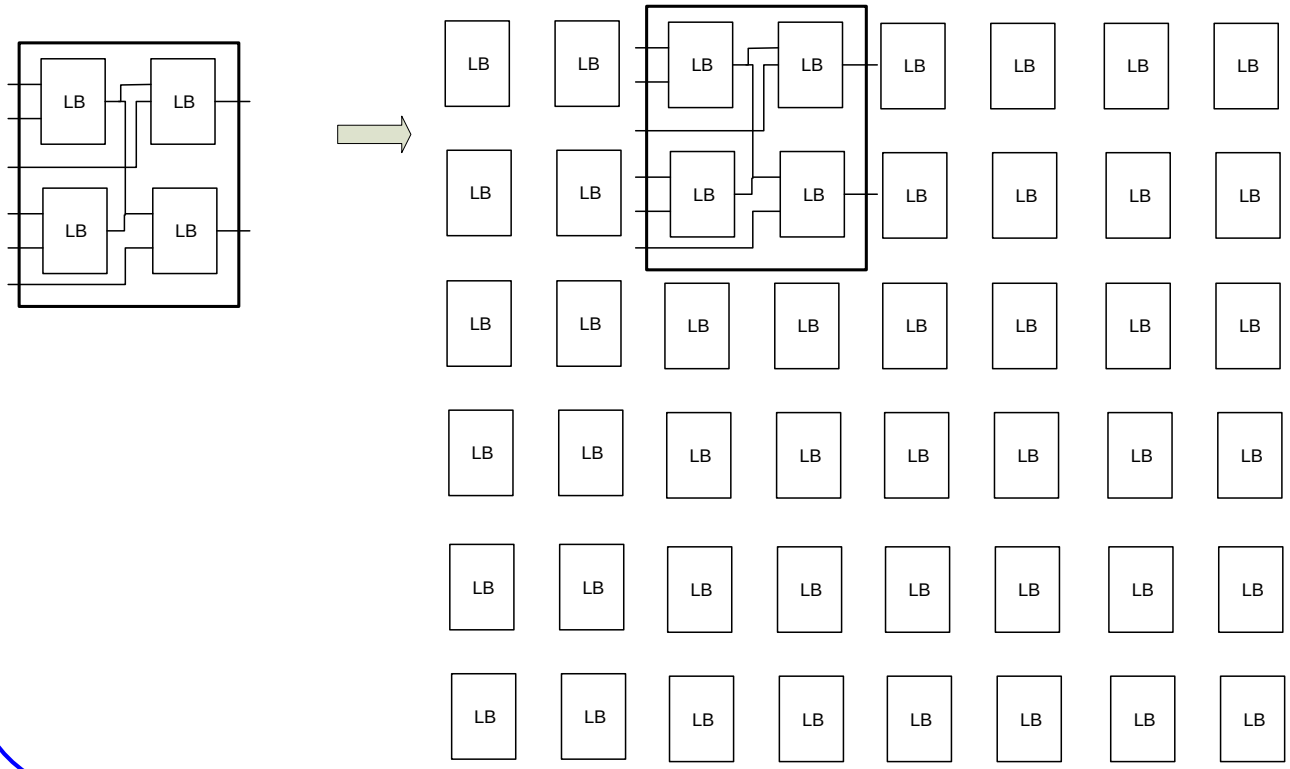


انواع IP

- دسته‌بندی از دیدگاه سطح تجرید توصیف IP:

❑ چینش:

- به صورت جایابی و مسیریابی شده
- ابزار جایابی و مسیریابی، جعبهٔ سیاه می‌بینند



انواع IP

- **IP نرم:**
  - سطح RTL
- **IP ثابت (firm):**
  - سطح منطقی و سطح چینش
    - سطح چینش: RPM (Relationally Placed Macro)
- **IP سخت:**
  - هسته‌های سخت در تراشه

مقایسهٔ IPها

- IP نرم:
  - ❑ قابل فهم‌تر
  - ❑ قابل اصلاح
    - قابل توسعه (افزودن امکانات)
    - قابل بهبود کارایی
  - ❑ آسان‌تر بودن تجمیع با سایر بخش‌ها
    - در RPM: نیاز به رزرو جای مشخص در طرح اصلی
  - ❑ قابل سرقت و کپی‌برداری
    - ← عدم تمایل طراحان IP به ارائه
  - ❑ وابستگی بسیار کم به ابزار و تراشه
  - ❑ اما پرطرفدارتر

مقایسهٔ IPها

- **IP ثابت:**
  - ❑ اگر ابزار و تراشهٔ مشخص است
    - ← احتمالاً محصول بهتر (از لحاظ مساحت، سرعت، توان)
      - علت: تلاش زیاد برای بهینه‌سازی
- **IPهای ساده‌تر: در ابزارها**
- **IPهای پیچیده: پولی**

انواع IP

- **دسته‌بندی از دیدگاه عملکرد:**
  - پردازنده
  - کنترل‌کننده‌های حافظه
  - واسط‌های استاندارد
    - USB
    - PCI اکسپرس
    - I2C
    - UART
    - اترنت
  - مدارهای محاسباتی
    - CORDIC
    - توابع مثلثاتی و هایپربولیک، جذر، و لگاریتم

انواع IP

- **دسته‌بندی از دیدگاه عملکرد (ادامه):**
  - هسته‌های امنیتی (رمزنگاری و رمزگشایی)
    - DES، AES، RSA، SHA، MD5
  - مدارهای آشکارساز خطا:
    - کد CRC

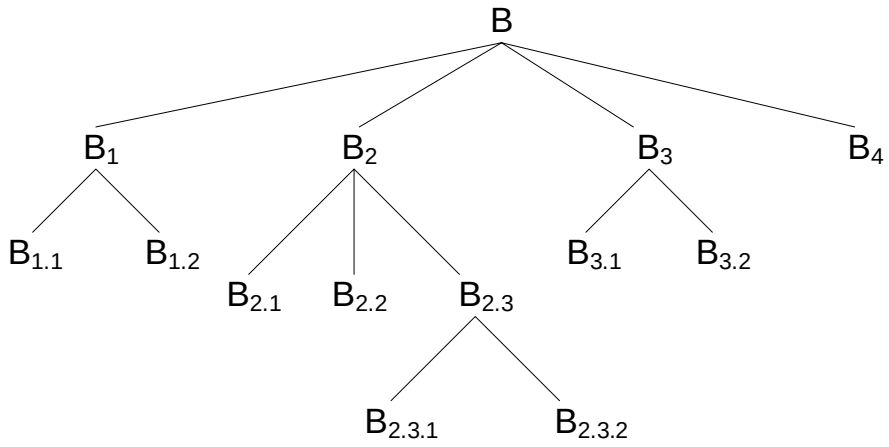
منابع IP

- از کجا تهیه کنیم؟
  - ❑ شرکت‌های عرضه‌کنندهٔ FPGA و کتابخانهٔ ابزارهای طراحی
  - ❑ گروه‌های عرضه‌کنندهٔ هسته‌های متن باز
    - سایت OpenCores
  - ❑ شرکت‌های فروشندهٔ هسته‌های IP
  - ❑ دانشگاه‌ها و مراکز پژوهشی

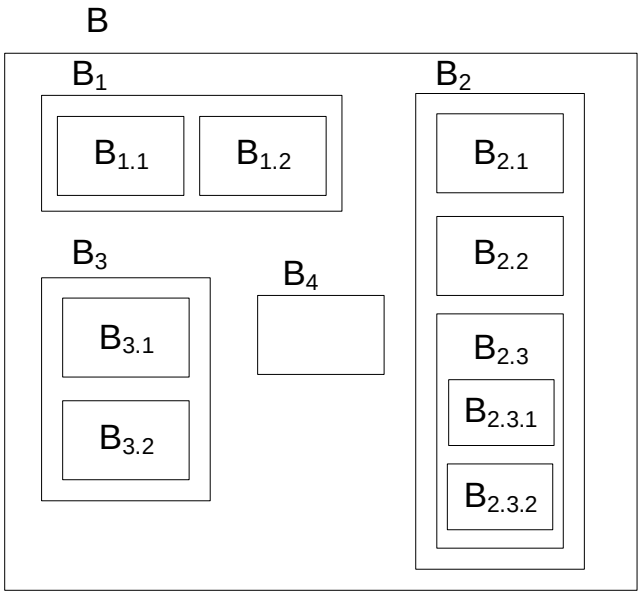
- از وجود مستندات کافی مطمئن شوید:
  - اطلاعات دقیق عملکرد هسته
  - نحوهٔ استفاده و سفارشی کردن آن
  - فرایند درستی‌سنجی
  - مجموعهٔ بردارهای آزمون و برنامهٔ آزمون

طراحی سلسله‌مراتبی

- **مزایا در سیستم‌های بزرگ:**
  - آسان‌تر شدن فرایند طراحی، توصیف، و اشکال‌زدایی



)ب(



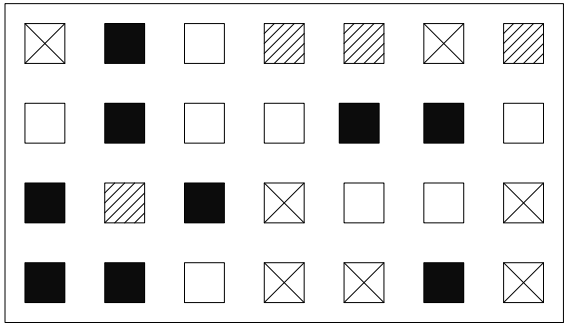
)الف(

طراحی سلسله‌مراتبی

- **مزایا در سیستم‌های بزرگ:**
  - بهینه‌سازی هر بلوک: جداگانه
    - علت: اعمال محدودیت‌های طراحی (مانند بهینه‌سازی برای سرعت) به بلوک‌های مختلف
      - در طراحی مسطح، اعمال محدودیت‌ها فقط به طور سراسری به کل طرح
  - آسان‌تر شدن درج IP

طراحی سلسله‌مراتبی

- **ملاحظات طراحی سلسله‌مراتبی:**
  - اجتناب از افراط در تعدد سطوح
  - اجتناب از بزرگ شدن هر بلوک
    - بلوک بزرگ ← تعداد سطرهای بیشتر ← احتمال اشکال، بیشتر
  - ابزارها: مسطح‌سازی طرح
    - علت: بهینه‌سازی مؤثرتر
    - پس از جایابی: متفرق
      - حتی داخل یک LB



■ بلوک های منطقی بلوک A

▨ بلوک های منطقی بلوک B

☒ بلوک های منطقی بلوک C

طراحی سلسله‌مراتبی

- **ملاحظات طراحی سلسله‌مراتبی:**
  - ❑ گاهی ترجیح می‌دهیم سطوح سلسله‌مراتب طراحی به هم نخورد
    - آسان‌تر شدن اشکال‌زدایی
      - ادغام بلوک‌ها ← از بین رفتن نام سیگنال‌ها ← دشواری دنبال کردن آنها بعد از سنتز
      - هر بلوک می‌تواند به طور جدا سنتز، شبیه‌سازی، اشکال‌زدایی و بهینه‌سازی
    - تسریع سنتز، جایابی، و مسیریابی
      - علت: بهینه‌سازی‌های ابزارها، درون بلوک‌ها و مستقل از یکدیگر

طراحی سلسله‌مراتبی

- **ملاحظات طراحی سلسله‌مراتبی:**
  - ورودی‌ها و خروجی‌های بلوک‌ها را در فلیپ فلاپ‌ها ثبت کنید
    - علت: هنگام بهینه‌سازی و timing closure، مسیر بحرانی در درون بلوک ← بهینه‌سازی آسان‌تر
  - برای بلوک‌های سلسله‌مراتبی از درگاه‌های سه حالته و دو طرفه استفاده نکنید (مگر آنکه به درگاه‌های تراشه وصل شده باشند.
    - علت: در FPGA گذرگاه‌های سه حالته وجود ندارد ← تبدیل به مالتی‌پلکسر ← تفاوت نتیجهٔ شبیه‌سازی قبل و بعد از سنتز

بلوک‌های اولیه

- **استفاده از بلوک‌های اولیه (primitive)، مزایا:**
  - ❑ کارایی بهتر در مقایسه با استنتاج ابزار سنتز
    - مثال: DSP blocks برای جمع، تفریق، ضرب، MAC
      - سرعت و توان بهتر
      - وجود دارند و استفاده شوند ← صرفه‌جویی در منابع ← تراشهٔ ارزان‌تر
    - نحوهٔ استفاده:
      - تعیین محدودیت
      - Instantiation

بلوک‌های اولیه

- **استفاده از بلوک‌های اولیه (primitive)، مزایا:**
  - ❑ طراحی در سطح LB:
    - ابزار به طور اتوماتیک ولی طراح باتجربه ← مساحت و توان بهتر
  - ❑ در عمل:
    - ابتدا در سطح RTL به ابزار بسپارید
    - اگر نیازها برآورده نشد، استفاده از primitive
  - ❑ توصیه:
    - افزودن بلوک‌های طراحی شده در سطح primitive به کتابخانه
    - محدودیت black_box روی box_type بلوک ← ابزار داخل آن را سنتز نمی‌کند

بلوک‌های اولیه

- **استفاده از بلوک‌های اولیه (primitive)، مزایا:**
  - گاهی الزامی:
    - مثال: Gigabit Transceiver
    - مثال: بلوک مدیریت کلاک
  - نحوهٔ استفاده:
    - تعیین مشخصات به ابزار
    - سپس Instantiation

بلوک‌های اولیه

- **استفاده از بلوک‌های اولیه، معایب:**
  - وابستگی طرح به ابزار و تراشه
  - شبیه‌سازی کندتر
    - علت: استفاده از مدل دقیق بلوک

پیاده‌سازی مدارهای محاسباتی و منطقی با حافظه

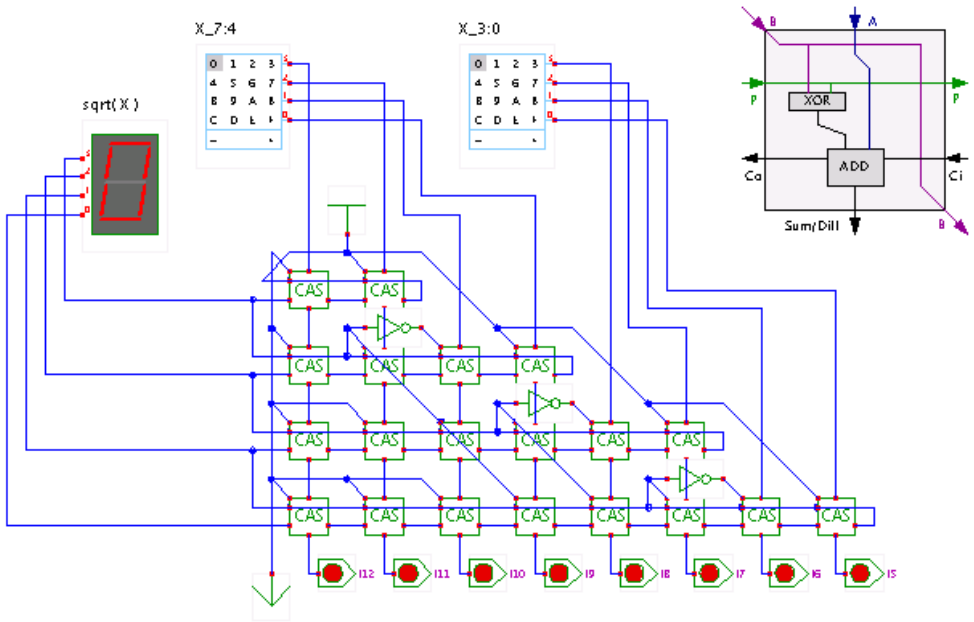
پیاده‌سازی با حافظه

- راه‌های طراحی مدارهای محاسباتی و منطقی:

1. توصیف RTL و سنتز

– مثال: عمل جذر:

– پیاده‌سازی الگوریتم جذر با LBها و DSPها ← پیچیده و مصرف شدن منابع زیاد



پیاده‌سازی با حافظه

- **راه‌های طراحی مدارهای محاسباتی و منطقی:**

۲. طراحی با حافظه

1. ذخیره‌سازی مقادیر در جدول

پیاده‌سازی با حافظه

- ذخیرهٔ مقادیر در جدول جستجو

آدرس	داده
۰	۰
۱	۱
۲	۱/۴
۳	۱/۷
۴	۲
۵	۲/۲
⋮	⋮
N	√n

پیاده‌سازی با حافظه

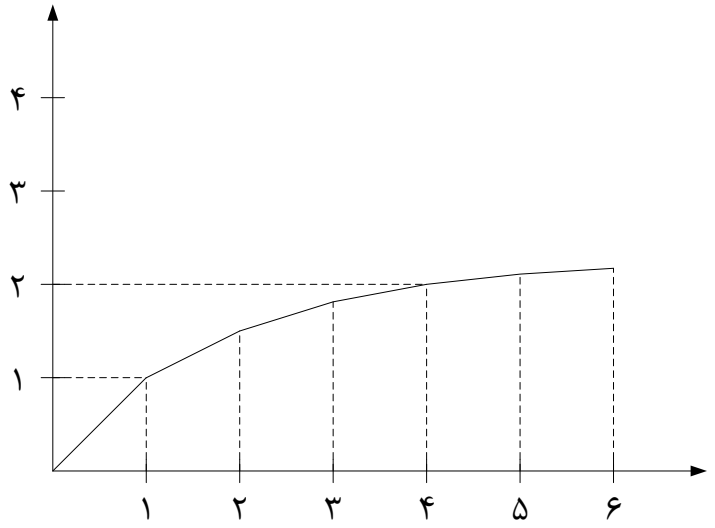
- **دقت محاسبه:**
  - ❑ تعداد اعداد ذخیره شده
    - تقسیم بازه به ۱۰ قسمت یا ۱۰۰۰۰۰ قسمت؟
  - ❑ دقت اعداد ذخیره شده
    - تا چند رقم اعشار
    - حذف ارقام سمت راست در اعداد بزرگ

آدرس	داده
۰	۰
۱	۱
۲	۱/۴
۳	۱/۷
۴	۲
۵	۲/۲
⋮	⋮
N	√n

پیاده‌سازی با حافظه

آدرس	داده
۰	۰
۱	۱
۲	۱/۴
۳	۱/۷
۴	۲
۵	۲/۲
⋮	⋮
N	√n

- **دقت محاسبه:**
  - اعدادی که در جدول نیستند (۱/۵):
    - گرد کردن
    - کوتاه کردن
    - تقریب خطی
    - دقیق‌تر
    - سخت‌افزار اضافی



پیاده‌سازی با حافظه

	Y	X	Current RD-		CURRENT RD+		
DX.Y	HGF	EDCBA	abcdei	fghj	abcdei	fghj	RD
D0.0	000	00000	100111	0100	011000	1011	same
D0.1	001	00000	100111	1001	011000	1001	flip
D0.2	010	00000	100111	0101	011000	0101	flip
D0.3	011	00000	100111	0011	011000	1100	flip
D0.4	100	00000	100111	0010	011000	1101	same
D0.5	101	00000	100111	1010	011000	1010	flip
D0.6	110	00000	100111	0110	011000	0110	flip
D0.7	111	00000	100111	0001	011000	1110	same
D1.0	000	00001	011101	0100	100010	1011	same
D1.1	001	00001	011101	1001	100010	1001	flip
D1.2	010	00001	011101	0101	100010	0101	flip
D1.3	011	00001	011101	0011	100010	1100	flip
D1.4	100	00001	011101	0010	100010	1101	same
D1.5	101	00001	011101	1010	100010	1010	flip
D1.6	110	00001	011101	0110	100010	0110	flip
D1.7	111	00001	011101	0001	100010	1110	same
D2.0	000	00010	101101	0100	010010	1011	same
D2.1	001	00010	101101	1001	010010	1001	flip
D2.2	010	00010	101101	0101	010010	0101	flip
D2.3	011	00010	101101	0011	010010	1100	flip
D2.4	100	00010	101101	0010	010010	1101	same
D2.5	101	00010	101101	1010	010010	1010	flip
D2.6	110	00010	101101	0110	010010	0110	flip
D2.7	111	00010	101101	0001	010010	1110	same
D3.0	000	00011	110001	1011	110001	0100	flip
D3.1	001	00011	110001	1001	110001	1001	same
D3.2	010	00011	110001	0101	110001	0101	same
D3.3	011	00011	110001	1100	110001	0011	same
D3.4	100	00011	110001	1101	110001	0010	flip
D3.5	101	00011	110001	1010	110001	1010	same
D3.6	110	00011	110001	0110	110001	0110	same
D3.7	111	00011	110001	1110	110001	0001	flip

- **مبدل کد:**
  - ۸ بیت به ۱۰ بیت:
    - ۸ بیتی: آدرس
    - ۱۰ بیتی: داده

پیاده‌سازی با حافظه

```
Begin
T = T0
i = 0
curr_sol = init_sol
curr_cost = COST(curr_sol)
while (T > Tmin)
   while (stopping criterion is not met)
    i = i + 1
    trial_sol = TRY_MOVE(curr_sol)
    trial_cost = COST(trial_sol)
    Δcost = trial_cost – curr_cost
    if (Δcost < 0)
                curr_cost = trial_cost
                curr_sol = MOVE(curr_sol)
    else

                r = RANDOM(0,1)
                if (r < e −Δcost/T)
                     curr_cost = trial_cost
                     curr_sol = MOVE(curr_sol)
    T = α · T
end
```

- **توابع دو متغیره:**
  - مثال:
  - تابع نمایی در الگوریتم simulated annealing

پیاده‌سازی با حافظه

- **توابع دو متغیره:**
  - مثال: دو عدد چهاربیتی
  - چسباندن دو عدد

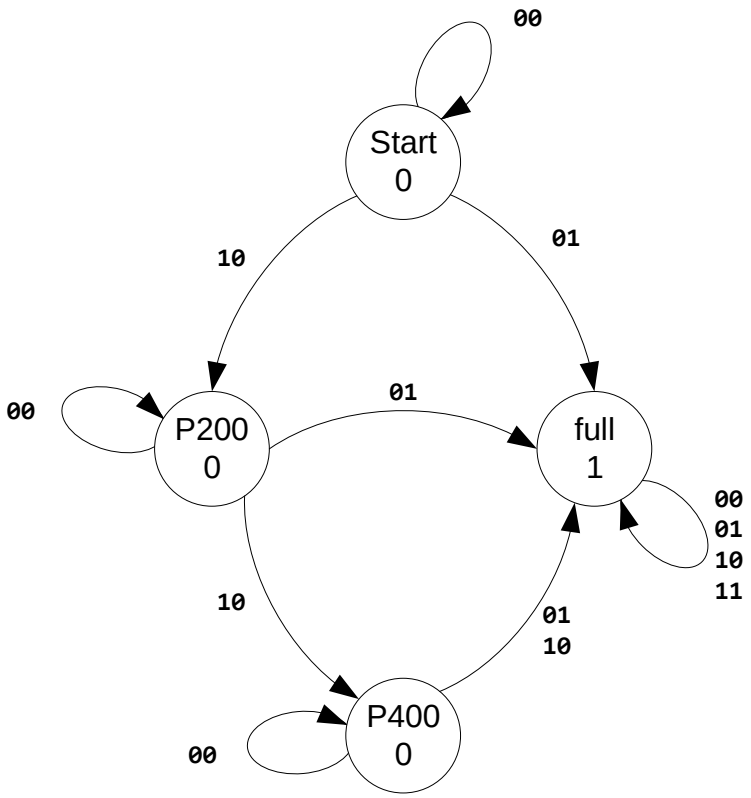
01H	۱
02H	۱
03H	۱
⋮	⋮
B0H	۰
B1H	۰/۰۰۰۰۲
⋮	⋮
7CH	۰/۵۵۸۰۴
⋮	⋮
FFH	۰/۳۶۷۸۸

پیاده‌سازی ماشین حالت با حافظه

پیاده‌سازی ماشین حالت با حافظه

- ماشین فروش شکلات:

حالت جاری			ورودی ها		حالت بعدی		خروجی
	S1	S0	D	P	S1	S0	DOOR
(start)	0	0	0	0	0	0	0
			0	1	1	1	0
			1	0	0	1	0
			1	1	X	X	X
(P200)	0	1	0	0	0	1	0
			0	1	1	1	0
			1	0	1	0	0
			1	1	X	X	X
(P400)	1	0	0	0	1	0	0
			0	1	1	1	0
			1	0	1	1	0
			1	1	X	X	X
(full)	1	1	0	0	1	1	1
			0	1	1	1	1
			1	1	1	1	1
			1	0	1	1	1



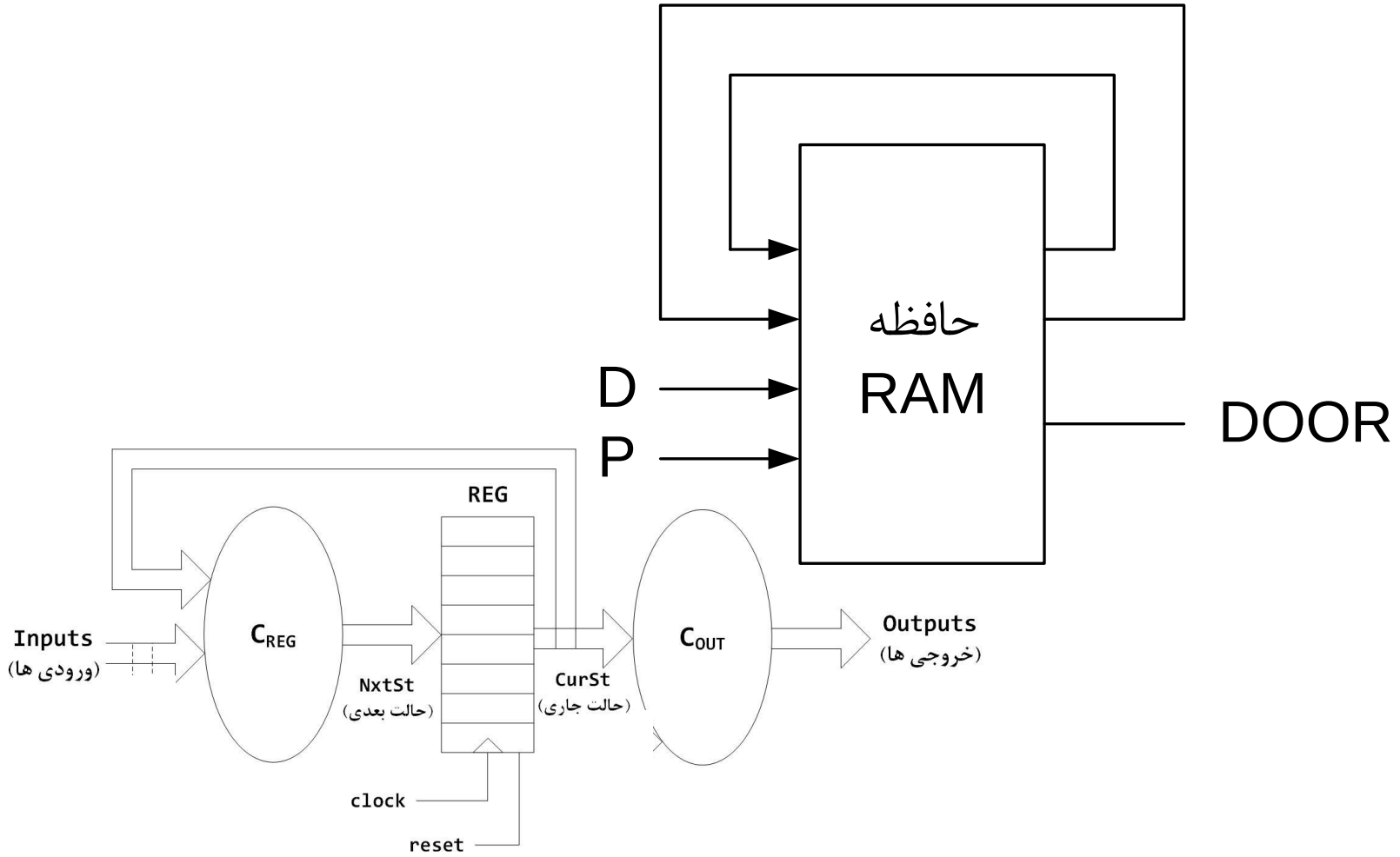
پیاده‌سازی ماشین حالت با حافظه

- ماشین فروش شکلات:

حالت جاری			ورودی ها		حالت بعدی		خروجی
	S1	S0	D	P	S1	S0	DOOR
(start)	0	0	0	0	0	0	0
			0	1	1	1	0
			1	0	0	1	0
			1	1	x	X	X
(P200)	0	1	0	0	0	1	0
			0	1	1	1	0
			1	0	1	0	0
			1	1	X	X	X
(P400)	1	0	0	0	1	0	0
			0	1	1	1	0
			1	0	1	1	0
			1	1	X	X	X
(full)	1	1	0	0	1	1	1
			0	1	1	1	1
			1	1	1	1	1
			1	0	1	1	1

آدرس	محتوای حافظه		
0000	0	0	0
0001	1	1	0
0010	0	1	0
0011	0	0	0
0100	0	1	0
0101	1	1	0
0110	1	0	0
0111	0	0	0
1000	1	0	0
1001	1	1	0
1010	1	1	0
1011	0	0	0
1100	1	1	1
1101	1	1	1
1110	1	1	1
1111	1	1	1

پیاده‌سازی ماشین حالت با حافظه



پیاده‌سازی ماشین حالت با استفاده از پردازندهٔ نهفته

پیاده‌سازی ماشین حالت با پردازندهٔ نهفته

- **پردازنده + حافظه = ماشین حالت**
  - تغییر که از حالتی به حالت دیگر بسته به مقادیر خوانده شده از حافظه یا مقادیر ورودی‌ها
  - محاسبه مقادیر و ارسال به حافظه یا تولید خروجی‌ها

پیاده‌سازی ماشین حالت با پردازندهٔ نهفته

- **مزایا:**
  - ❑ صرفه‌جویی در منابع اگر پردازنده (سخت یا نرم) داریم
    - به خصوص برای FSMهای بزرگ و پیچیده
  - ❑ سهولت اشکال‌زدایی
    - Time to Market ← کمتر

کاهش مدت زمان چرخهٔ طراحی

کاهش مدت زمان چرخهٔ طراحی

- **چرخهٔ طراحی:**
  - تکرار مراحل به دفعات بسیار زیاد
    - برای اطمینان از صحت مدار
    - برای اطمینان از براوردن محدودیت‌ها
  - کاهش زمان چرخه با کاهش زمان اجرای ابزارها

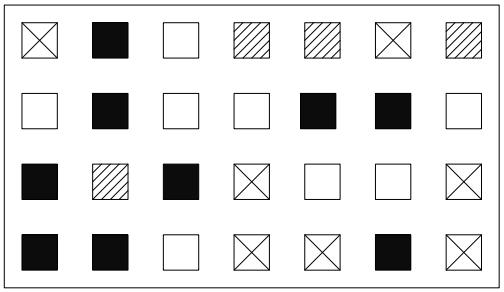
کاهش مدت زمان چرخهٔ طراحی

- تکنیک‌ها:
  - طراحی سلسله‌مراتبی
    - جلوگیری از مسطح‌سازی
    - علت:
      - پردازش چند مدار کوچک به جای یک مدار پیچیده
    - ← از دست دادن مزایای مسطح سازی
    - راه حل:
      - حفظ سلسله مراتب در مراحل اولیهٔ (بررسی فقط درستی عملکرد)
      - سپس مسطح سازی

کاهش مدت زمان چرخهٔ طراحی

- تکنیک‌ها:
  - جاسازی (floorplanning) بلوک‌ها

طراحی مسطح

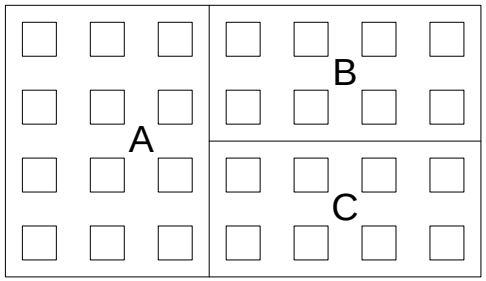


بلوک های منطقی بلوک A

بلوک های منطقی بلوک B

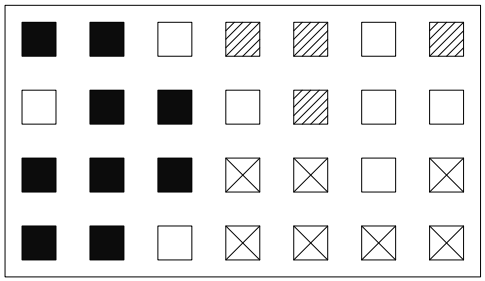
بلوک های منطقی بلوک C

floorplanning



)الف(

جایابی با floorplanning



)ب(

بلوک های منطقی بلوک A

بلوک های منطقی بلوک B

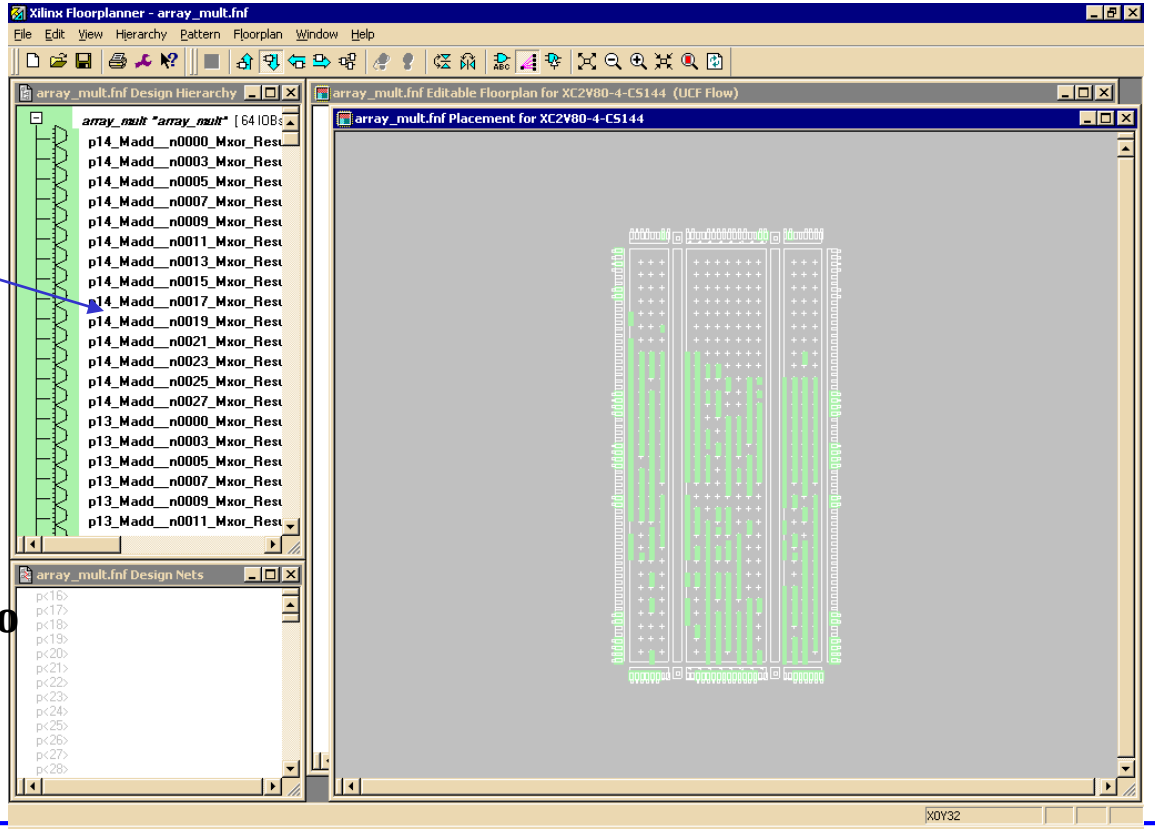
بلوک های منطقی بلوک C

کاهش مدت زمان چرخهٔ طراحی

- تکنیک‌ها:
  - جاسازی (floorplanning) بلوک‌ها
    - گاهی نتیجه را هم بهتر می‌کند

بهبود سرعت با جاسازی

- **Floorplanner window:**
  - ❑ Floorplanner → View/edit placed design



LEs

- **Green rectangles: mapped components to CLBs**

بهبود سرعت با جاسازی

- محدودیت زمانی: ۲۵ ns
- گزارش ابزار جایابی و مسیریابی:

Timing constraint: TS_P2P = MAXDELAY FROM TIMEGRP "PADS" TO TIMEGRP "PADS" 25 ns ;

20135312 items analyzed, 11 timing errors detected. (11 setup errors, 0 hold errors)

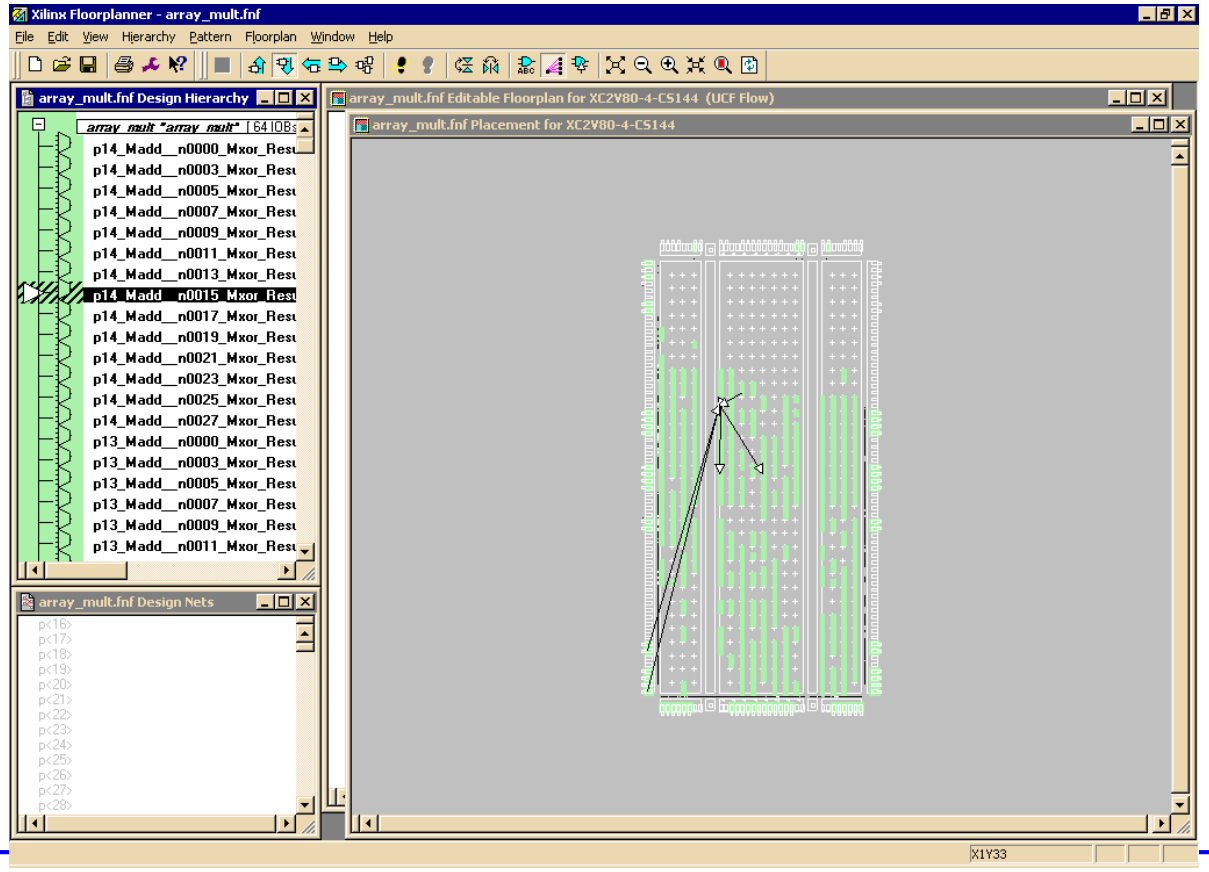
Maximum delay is 31.128ns.

بهبود سرعت با جاسازی

Slack: -6.128ns (requirement - data path)
Source: y<0> (PAD)
Destination: p<30> (PAD)
Requirement: 25.000ns
Data Path Delay: 31.128ns (Levels of Logic = 31)

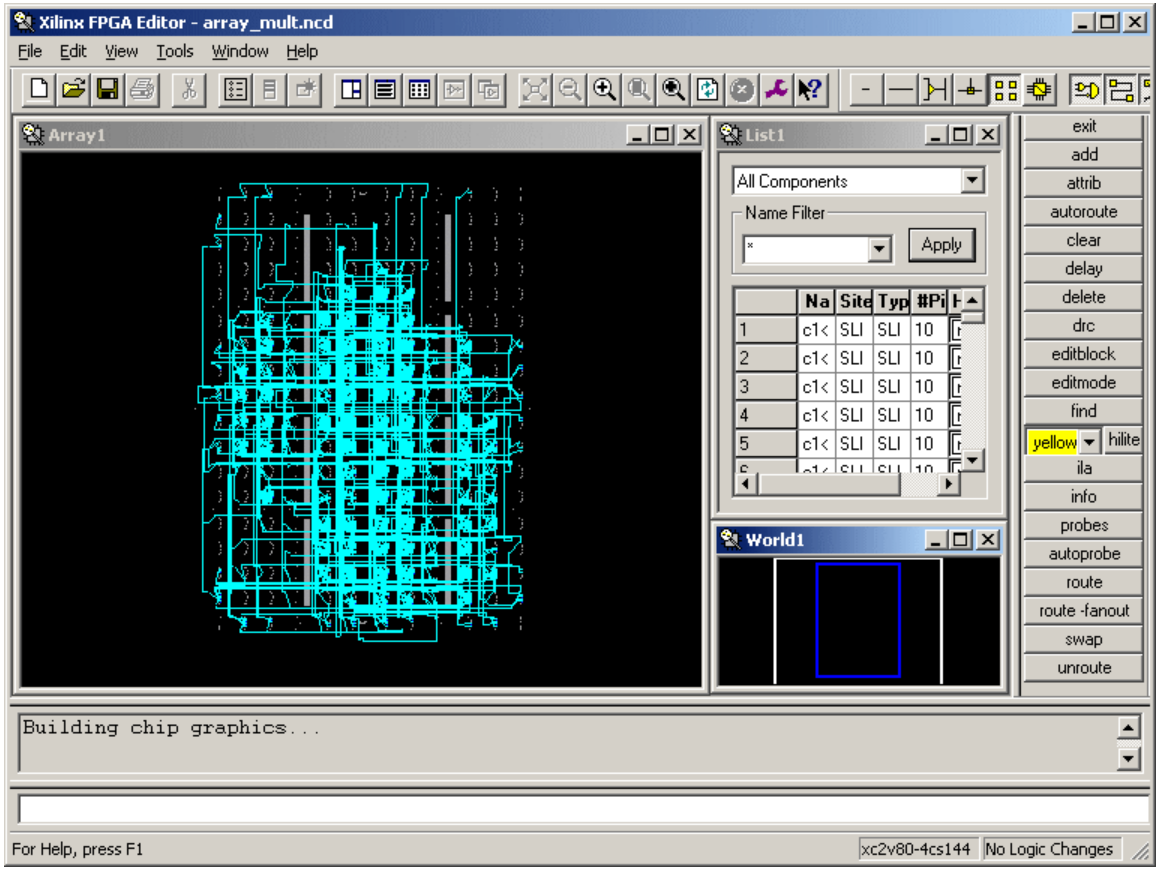
Rat's nest

- **If you click on a component in the deign hierarchy window, its rat's nest is shown.**



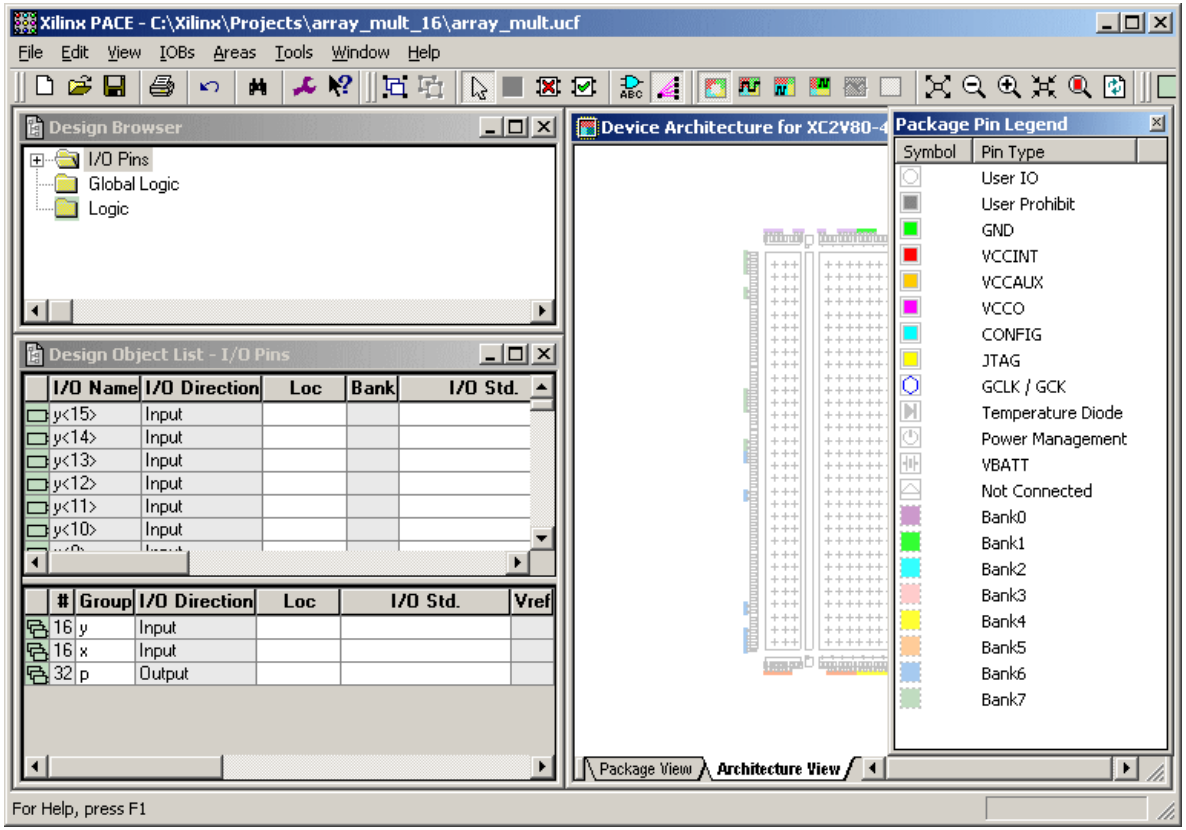
Routing editor view

- **FPGA Editor → View/Edit Routed Design**

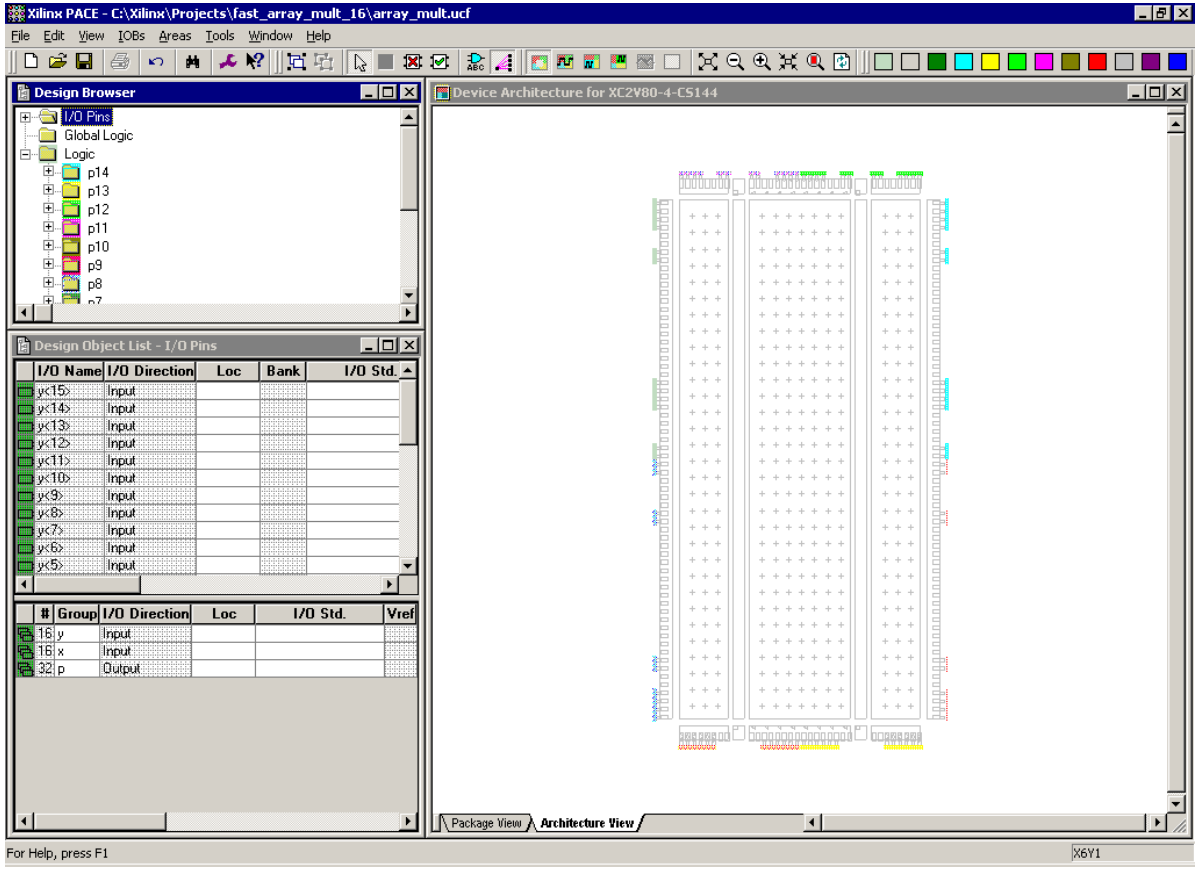


بهبود سرعت با جاسازی

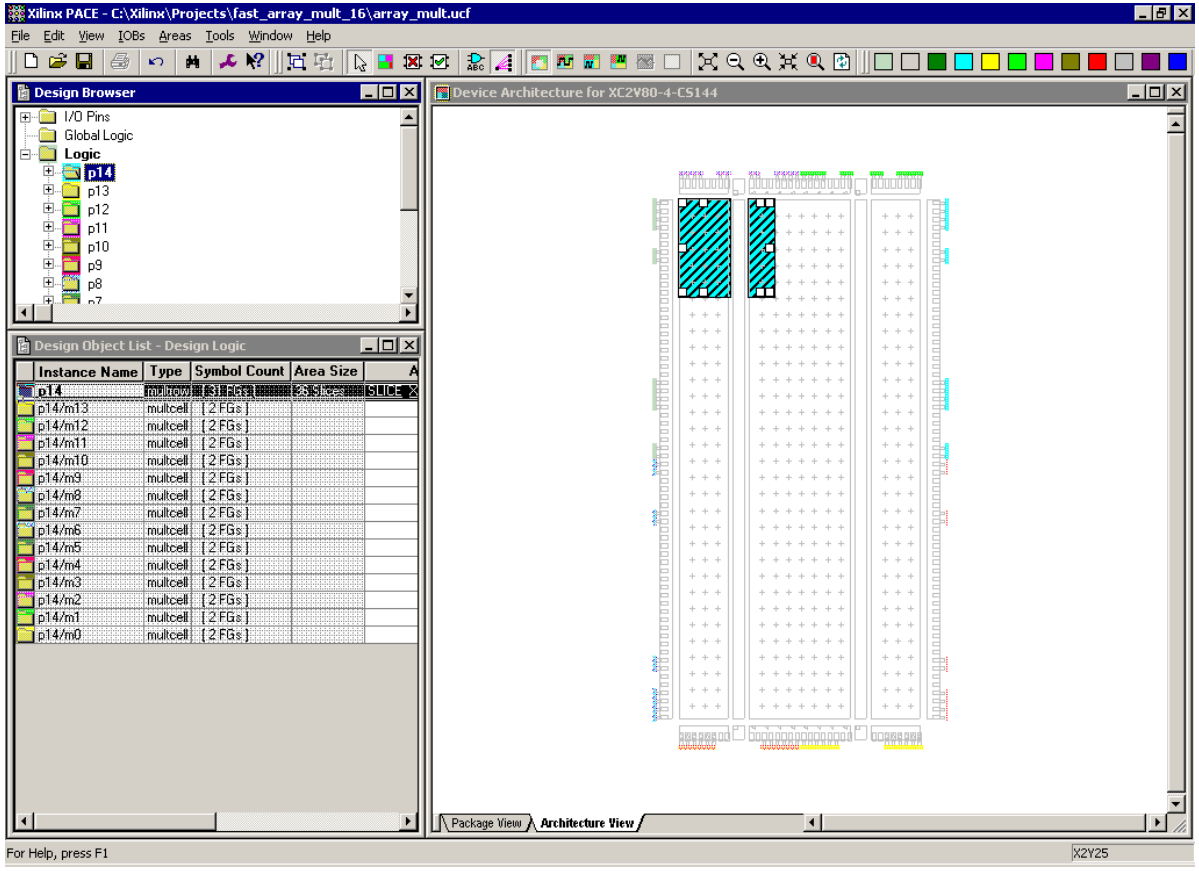
- قرار دادن محدودیت روی جایابی:
  - تعیین جاسازی برای LBها
  - انتساب پایه به ورودی–خروجی‌ها (IOB)



جاسازی

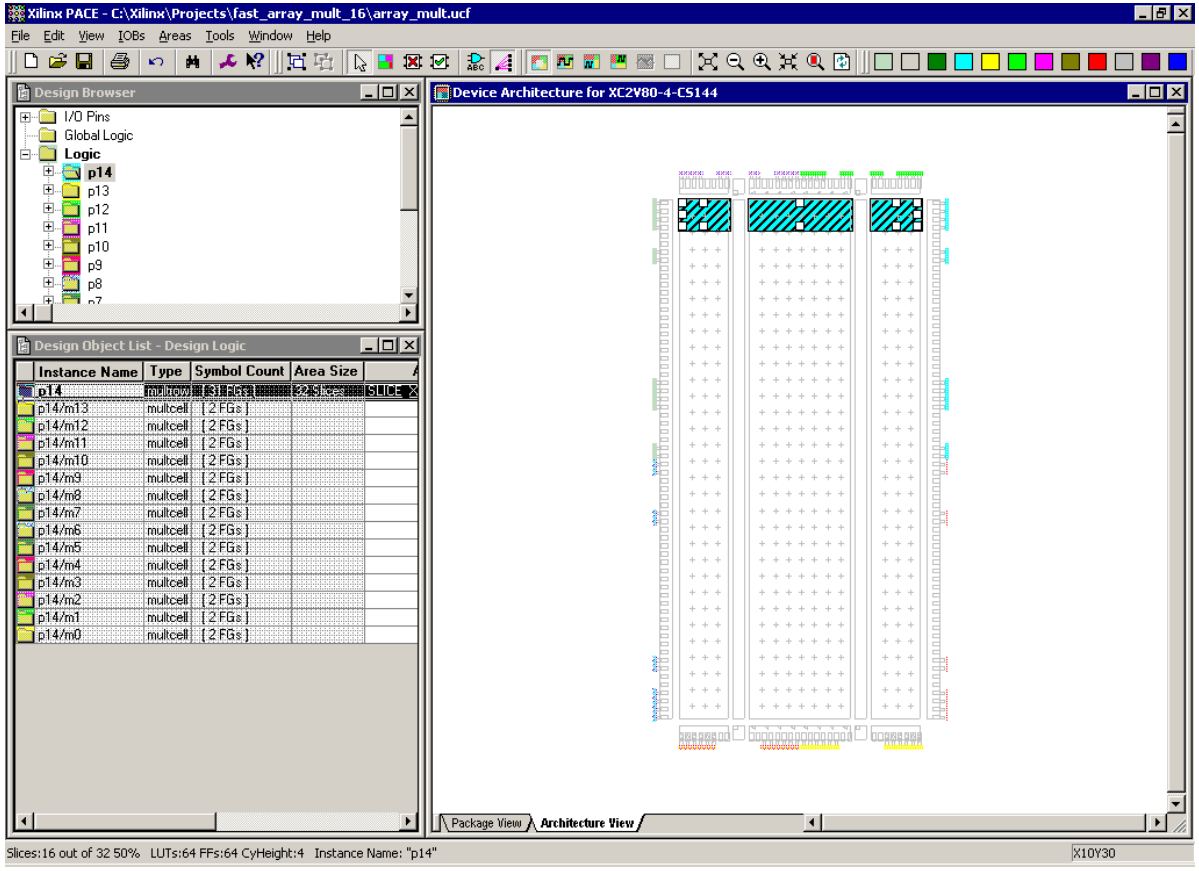

Xilinx PACE - C:\Xilinx\Projects\fast_array_mult_16\array_mult.ucf
File Edit View IOBs Areas Tools Window Help
Design Browser
I/O Pins
Global Logic
Logic
p14
p13
p12
p11
p10
p9
p8
Design Object List - I/O Pins
I/O Name I/O Direction Loc Bank I/O Std.
y<15> Input
y<14> Input
y<13> Input
y<12> Input
y<11> Input
y<10> Input
y<9> Input
y<8> Input
y<7> Input
y<6> Input
y<5> Input
Group I/O Direction Loc I/O Std. Vref
18 y Input
18 x Input
32 p Output
Device Architecture for XC2V80-4-CS144
Package View
Architecture View
For Help, press F1
X6Y1

جاسازی


Xilinx PACE - C:\Xilinx\Projects\fast_array_mult_16\array_mult.ucf
File Edit View IOBs Areas Tools Window Help
Design Browser
I/O Pins
Global Logic
Logic
p14
p13
p12
p11
p10
p9
p8
Design Object List - Design Logic
Instance Name Type Symbol Count Area Size
p14 multcell [31 FGs] 25 Slices SLICE
p14/m13 multcell [2 FGs]
p14/m12 multcell [2 FGs]
p14/m11 multcell [2 FGs]
p14/m10 multcell [2 FGs]
p14/m9 multcell [2 FGs]
p14/m8 multcell [2 FGs]
p14/m7 multcell [2 FGs]
p14/m6 multcell [2 FGs]
p14/m5 multcell [2 FGs]
p14/m4 multcell [2 FGs]
p14/m3 multcell [2 FGs]
p14/m2 multcell [2 FGs]
p14/m1 multcell [2 FGs]
p14/m0 multcell [2 FGs]
Device Architecture for XC2V80-4-CS144
Package View
Architecture View
For Help, press F1
X2Y25

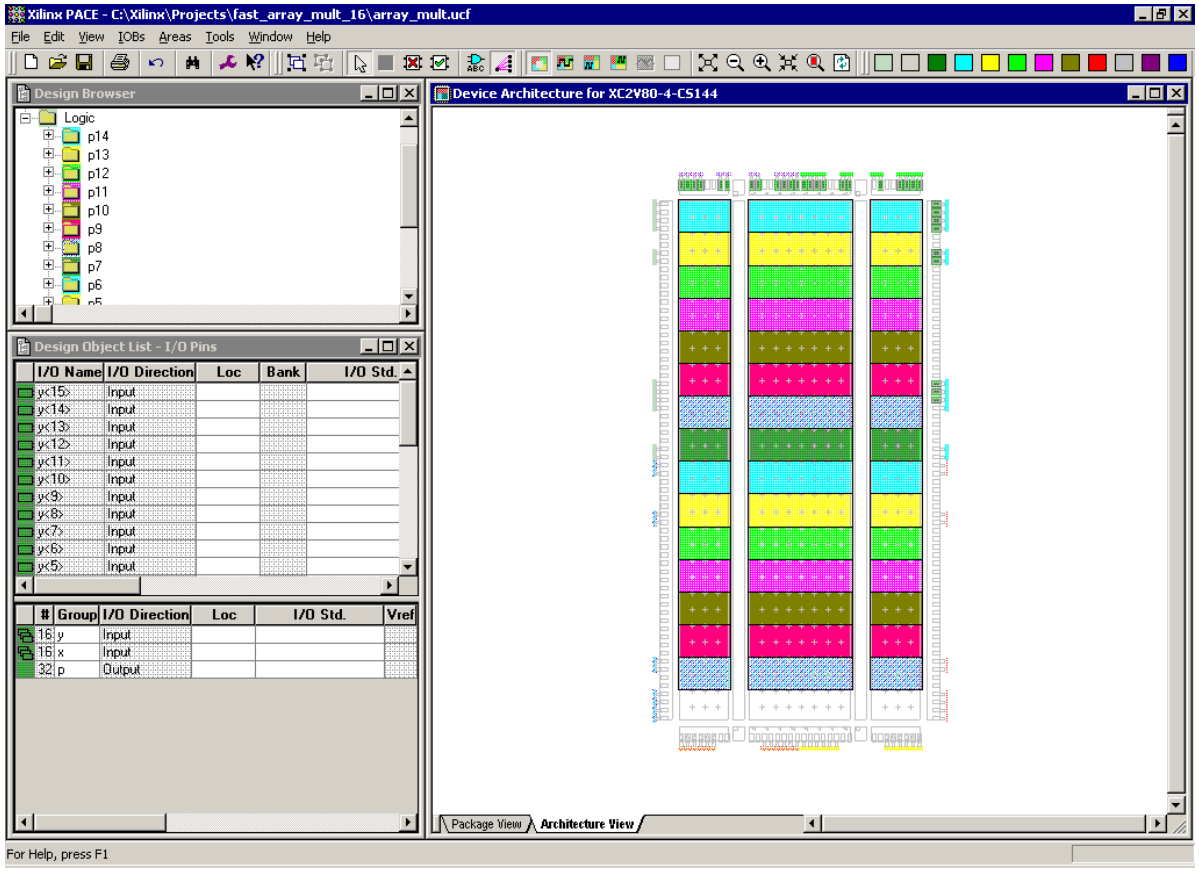
جاسازی

- تغییر شکل پودمان



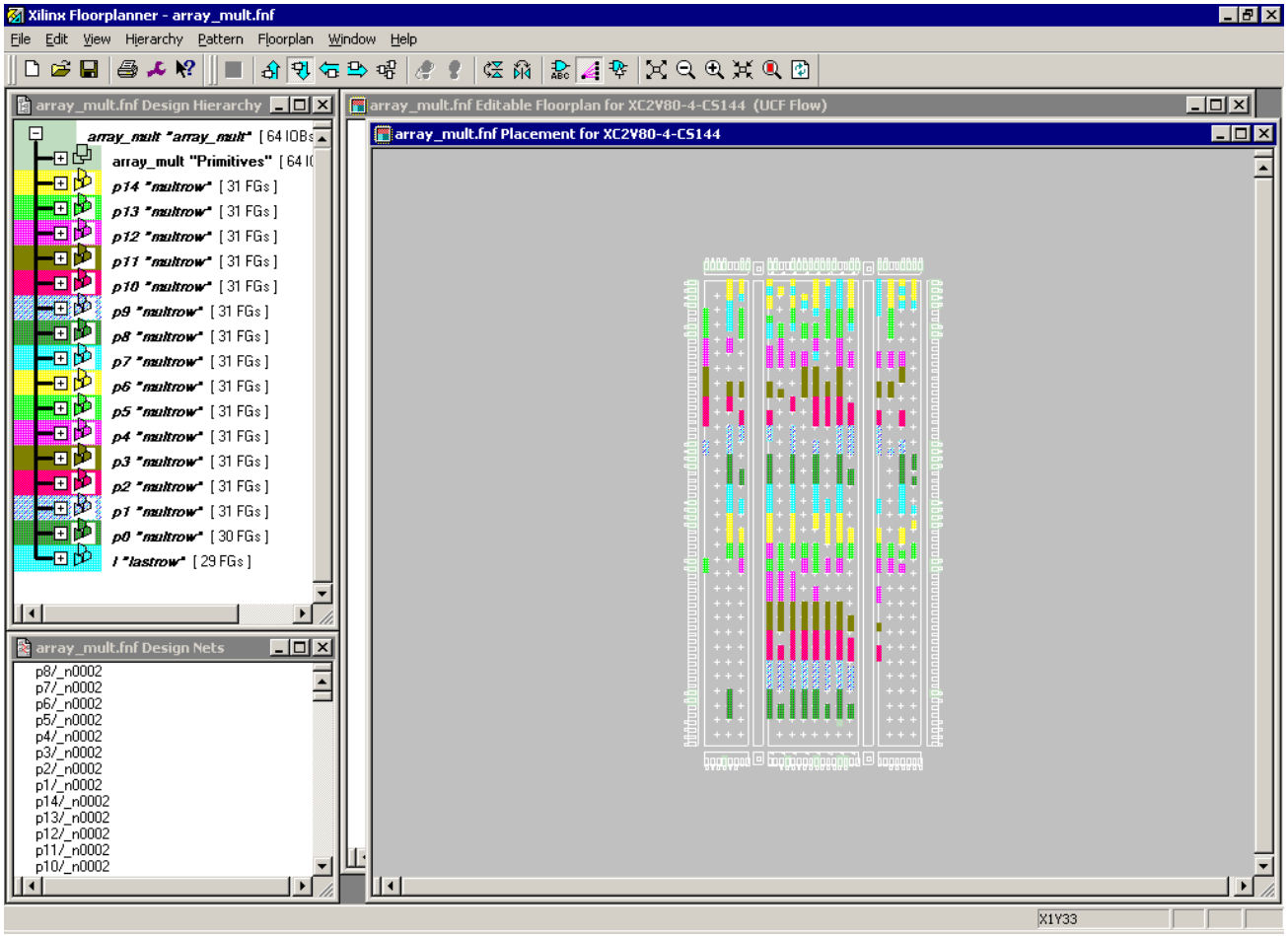
جاسازی

- طرح منظم ← ساختار مداری منظم
  - در طرح مسطح، ابزار فاقد اطلاعات نظم طرح است



جاسازی

- نتیجهٔ جایابی



جاسازی

- گزارش ابزار جایابی و مسیریابی:

19742142 items analyzed, 0 timing errors detected. (0 setup errors, 0 hold errors)

Maximum delay is 29.934ns.

- مقایسه با 31.128 بدون جاسازی