VHDL

مقدمه

پیچیدگی وحجم سیستمهای دیجیتال هر روز درحال گسترش است و ابزارهای طراحی سخت افزارنیزهمزمان با اضافه شدن این پیچیدگی رو به گسترش است. روشهای طراحی سنتی و استفاده از قلم و کاغذ جای خود را به برنامه های کامپیوتری و نرم افزارهای طراحی داده اند.جدیدترین متدهای طراحی سخت افزار زبانهای توصیف سخت افزار یا HDL(Hardware Description Language) می باشد.

HDLها به منظور توصیف سخت افزار برای اهداف شبیه سازی مدل کردن وتست کردن طراحی و مستند سازی بکار برده می شوند. این زبانها پیاده سازی و شبیه سازی رفتار ذاتی و جزییات یک سیستم را به خوبی پشتیبانی می کنند. زبانهای HDL دارای یک مجموعه ساده ای از دستورات و Symbol ها برای جایگزینی شماتیک و دیاگرامهای سیستم دیجیتال هستند.

یکی از جدیدترین روشهای VHDL HDL می باشد.

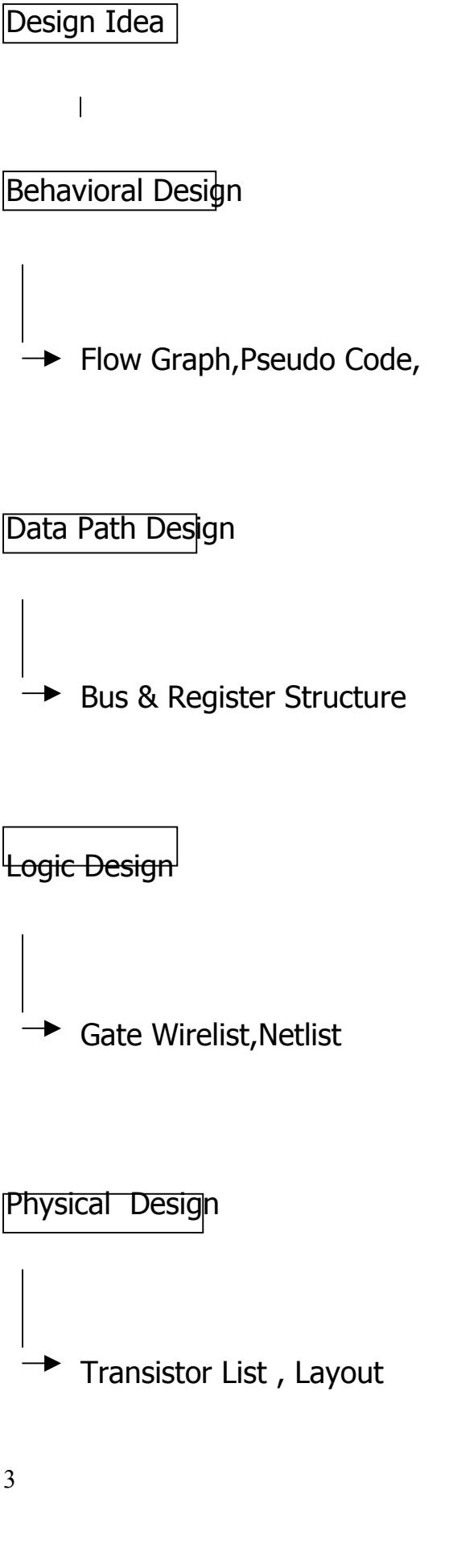
بدنبال تحقیقات برای استانداردسازی طراحی ومستندسازی سازمان دفاع آمریکا در سال ۱۹۸۳ ملزومات مورد نیاز برای استانداردسازی VHSIC Hardware Description Language را فراهم آورد.

کار روی VHDL در تابستان ۱۹۸۳ با همکاری IBM Texas Instruments Intermetrics شروع شد وشش ماه بعد VHDL 2.0 به بازار عرضه شد.این نسخه از زبان VHDL دستورات Sequentional را پشتیبانی نمی کرد و تنها قادر به شبیه سازی سیستمهای Concurrent بود.

در سال ۱۹۸۵ استاندارد سازی VHDL به IEEE واگذار شد و اولین استاندارد آن توسط IEEE در سال ۱۹۸۷ ایجاد گردید .

تعدادی از زبانهای HDL و شبیه سازی سخت افزار عبارتند از : HDL , AHDL, CDL,CONLAN, ISPS,TEGAS ,

پروسه طراحی سخت افزار


Design Idea
Behavioral Design
Flow Graph,Pseudo Code,
Data Path Design
Bus & Register Structure
Logic Design
Gate Wirelist,Netlist
Physical Design
Transistor List , Layout

Manufacturing

|

شکل ۱

Chip or Board

شکل زیر یک پروسه عمو می طراحی سخت افزار را نمایش می دهد در مرحله اول مشخص کردن هدف سیستم قبل از طراحی می باشد. در این دیاگرام نشان داده شده است که بعد از هر مرحله طراح نتیجه طراحی را بررسی می کند و اطلاعات جدیدی را به آن اضافه می کند و به مرحله بعد می رود.

پس از مشخص شدن هدف طراحی لازم است طراح یک مدل رفتاری برای شرح عملکرد مدار تعریف کند.

رفتار مدار در این مرحله بوسیله یک فلوچارت بلوک دیاگرام یا یک سود کد می تواند بیان شود.

در این مرحله طراح ورودی خروجی را بدون توجه به معماری مربوطه مشخص و تعریف می کند.مرحله بعد مشخص کردن Data path است. در این مرحله طراح رجیسترهای و واحد های منطقی مورد نیاز را مشخص می کند.

اجزای سیستم ممکن است توسط گذرگاههای یک طرفه یا دو طرفه به یکدیگر متصل می شوند. همچنین یک پروسیجر نیز برای کنترل جریان داده بین اجزا از طریق Bus ها باید در این مرحله تعریف گردد. نتیجه این فاز از طراحی بدست آوردن معماری سیستم با مشخص شدن کنترل جریان داده است.

طراحی منطقی مرحله بعدی درپروسه طراحی مدارات دیجیتال است و شامل بکارگیری گیتهای منطقی و فلیپ فلاپها برای پیاده سازی رجیسترها گذرگاهها و واحدهای منطقی کنترل جریان است . نتیجه این فاز بدست آوردن یک لیست از گیتهای منطقی و فلیپ فلاپهای مورد نیاز است.

فاز بعدی طراحی تبدیل لیست بدست آمده گیتها و فلیپ فلاپهای بدست آمده در مرحله قبل به مجموعه ای از ترانزیستورها یا لی اوت فیزیکی می باشد. فاز آخر در پروسه طراحی ایجاد یک سیستم برنامه پذیر یا یک IC می باشد.

مفاهیم در VHDL:

موجودیت (Entity) :

به هر یک از Component های سخت افزاری یا اجزا یک سیستم را می توان به عنوان یک موجودیت در VHDL تعریف کرد . هر موجودیت دارای دو دسته خصوصیات است.

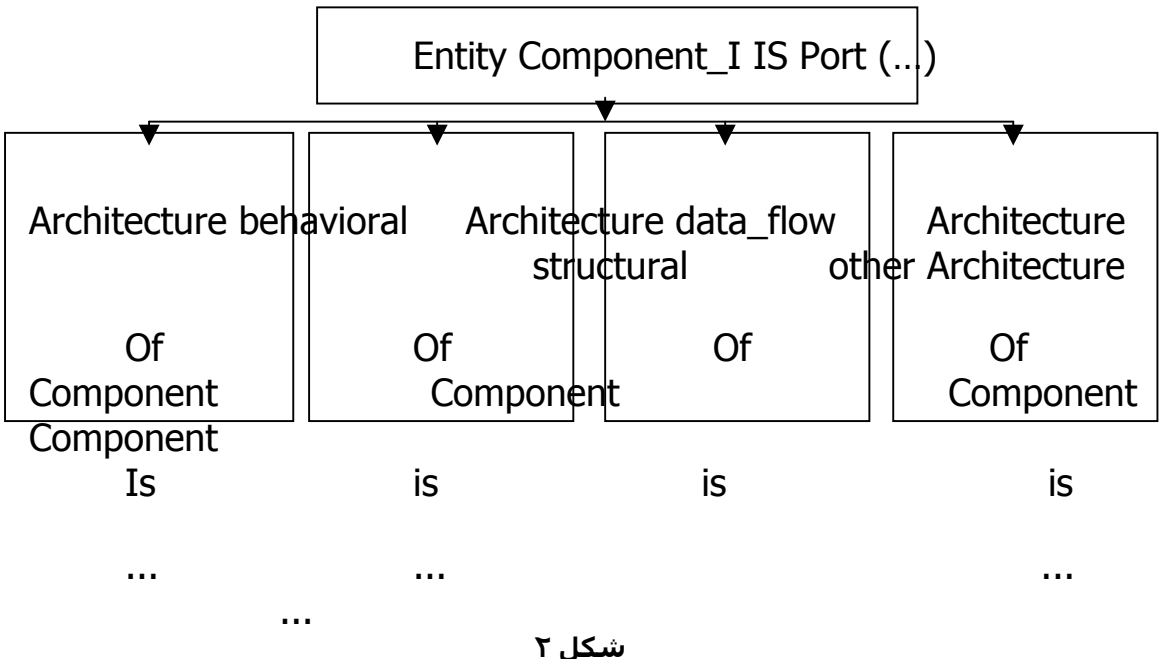
۱- خصوصیات خارجی : که قابل رویت برای کاربران می باشد . خصوصیات خارجی به دو دسته تقسیم می شوند :

الف – ورودی و خروجیهای یک موجودیت (Interface)

ب - پارامترهای عمومی : شروط و قیود دیگری ممکن است برای اجزا سیستم تعریف شوند که مربوط به رفتار سخت افزار و خصوصیات آن و یا تاثیرات محیطی تاثیرگذار می باشد.

۲- خصوصیات داخلی : (Architecture) که از آن به عنوان عملکرد یک موجودیت یاد می شود.

یک موجودیت می تواند دارای چند تعریف Architecture باشد .



شکل ۲

در زیر نحوه تعریف یک موجودیت و معماری آن آمده است :

```
Entity Component_I IS
        Generic (Declaration Generic Parameter)
        Port (Input & Output Declaration )
End Component_I
Architecture Architecture_name OF  Component_I IS
        Declaration
Begin
```

Body Architecture;

End Architecture

۳- Package ها :

غالبا پیش می آید که برای تشریح یک گروه از تعاریف بکار برده می شود. بعنوان مثال نوعهای مختلف داده تعریف شده توسط کاربر و سربارگذاری عملیات این نوعها برای یک Component .

این تعاریف می توانند همراه هم تحت عنوان یک Package بکار برده شوند. نحوه تعریف یک Package در زیر آمده است

Package Package_name IS

Component Declaration

Sub_program Declaration

End Package

Package Body Package_name IS

Type Definition

Sub_program

End Package_name

۴- کتابخانه ها و Binding

VHDL اجازه می دهد که از کتابخانه های مختلف موجود یا ایجاد شده توسط کاربر و همچنین اتصال Sub Component ها به یکدیگر (Binding) برای طراحی استفاده کنید .

```
Library Library_name ;

Configuration Confoguration_name OF Component IS

    Binding OF Entities AND Architectures

    Specifying Parameters OF Design

    Binding Component OF a Library TO Subcomponents

End Configuration
```

مثالی برای VHDL

تعریف گیت NAND :

۱- تعریف موجودیت :

```
Entity nand IS

    Port (a,b :  In vlbit ; y : out vlbit)

End nand ;
```

در این تعریف مشخص می شود که گیت NAND دارای ۲ ورودی از نوع VLBIT (یک متغیر تک بیتی با مقادیر ۰ ، ۱ ،X،Z) و یک خروجی از همان نوع است. b,a به ترتیب ورودی و y خروجی است.

۲- تعریف رفتار گیت NAND

تعریف عملیات یک موجودیت با تعاریف Architecture مشخص می شود بعنوان مثال رفتار یک NAND برای ایجاد خروجی a nand b به صورت زیر تعریف می شود.

```
Architecture Behavioral OF nand IS

    y <= a

End Behavioral
```

در این مثال y یک سیگنال است و هرگاه a و b تغییر کند y نیز تغییر خواهد کرد و اصطلاحا Drive می شود و عملیات NAND در یک حلقه ناپایان تا زمانیکه شبیه سازی متوقف نشده است اجرا می شود ولی عملیات انتساب y <= a and b تنها هنگام تغییر a و b انجام می گیرد.

انواع داده در VHDL :

یک زبان تشریح و شبیه سازی سخت افزار در Level های مختلف نیاز به انواع داده علاوه بر نوع داده منطقی (۰،۱) دارد. انواع داده در VHDL دارای انواع داده تعریف شده است از قبیل نوع داده Integer ممیز شناور ، نوع داده شمارشی و انواع داده قابل تعریف توسط کاربر (User Defined) . همچنین دارای آرایه ها و رکوردها می باشد.

VHDL دارای ساختار بسیار سطح بالایی برای برنامه نویسان مهیا کرده است . بعنوان مثال همانند C++ قابلیت سربارگذاری عملگرها و تعریف عملیات جدید برای انواع داده های تعریف شده توسط کاربر می باشد.

نوع شمارشی : یک نوع شمارش با تمام مقادیری که یک متغیر از این نوع ممکن است داشته باشد تعریف می شود بعنوان مثال

```
Type VLBIT IS ('0','1','Z','X')
```

در این تعریف نوع شمارشی VLBIT با مقادیر ۰و۱وx وy تعریف می شود و متغیرهای از نوع VLBIT می تواند یکی از این ۴ مقدار را داشته باشد.

۲- **ارایه ها** : در VHDL می توان انواع آرایه ها از نوع های مختلف تعریف کرد بعنوان مثال

```
Signal a : Vlbit (31 downto 0) ;
```

نحوه استفاده از آرایه ها طی مثال زیر شرح داده می شود

```
Entity nand32 IS

        Port (a,b : in vlbit_1d(31 downto 0);y : out vlbit (31 downto
0));

End nand32

 Architecture Behavioral OF nand32 IS

Begin

        Y <= a nand b ;

End Behavioral
```

-a و b در مثال بالا یک آرایه ۳۲ بیتی از نوع vlbit هستند و می تواند بعنوان یک Bus ، ۳۲ بیتی یا یک عدد صحیح ۴ بیتی مورد استفاده قرار گیرد.

- اندیسهای آرایه می توانند از هر عددی شروع شود.

۳- **رکوردها** : در VHDL امکان تعریف نوع رکورد وجود دارد :

```
Type rec_exm IS Record

        A: Integer

        .

        .

End Record
```

۴- **فایلها** : VHDL دارای نوع داده فایل نیز می باشد که میتوان خروجی های یک تابع را در آن قرار داد یا ورودیهای یک تابع یا زیر برنامه از داخل یک File تامین کرد .

تعریف نوع فایل Type Logic_data IS File OF Character

تعریف متغیر از نوع فایل FILE Input_file : Logic_data

۵- نوع داده **Time** : یکی از انواع داده ای که در VHDL است و یکی از تمایزهای آن با دیگر زبانهاست نوع داده Time است و می توان متغیر از نوع Time تعریف کرد.

مدل کردن تاخیر زمانی :

یکی از قابلیتهای زبان VHDL قابلیت زمانبندی در سطوح مختلف می باشد.

VHDL به طراح اجازه می دهد که مقدار دهی سیگنالها را زمانبندی کند و نسبت دهی مقادیر را به سیگنالها با تاخیر زمانی واقعی انجام می دهد .(همانند آنچه در سخت افزار دیده شود)

```
Entity nand IS
        Port (a,b : in vlbit ; y : out vlbit);
End nand;
Architecture Behavioral OF nand IS
Begin
        Y <= a nand b after 2ns;
End Behavioral
```

در این مثال بعد از هر تغییری که در a و b یا هر دو ایجاد شود مقدار a nand b بعد از ns ۲ به y منتقل می شود.

پارامترهای عمومی :

به همراه تعریف هر Entity می توان خصوصیات عمومی نیز برای آن تعریف کرد . هرگاه پارامترهای عمومی تعریف شوند در هر یک از نمونه هایی که از آن موجودیت گرفته می شود این پارامترها در نظر گرفته خواهد شد.

از طرفی تغییر رفتار داخلی و محدودیتها را تنها با تغییر این پارامترهای عمومی می توان در موجودیت تعریف شده اعمال کرد.

مثال

```
Entity nand  IS
        Generic (Delay : Time := 2 ns);
        Port (a,b : in vlbit ; y : out vlbit);
End nand;
Architecture Behavioral OF nand IS
Begin
        Y <= a nand b after Delay ;
End Behavioral;
```

متغیر Delay جز پارامترهای عمومی موجودیت nand تعریف شده است و نوع آن از نوع زمان و مقدار آن ns ۲ است.

انواع عملگرها در VHDL:

در قسمت قبل تعریف انواع داده شرح داده شده در این قسمت عملگرهای زبان VHDL که روی انواع داده استاندارد تعریف شده اند شرح می دهیم .

- **عملگرهای منطقی**

Not , Xnor, Xor, Nor , Nand , On, And عملگرهای منطقی تعریف شده توسط VHDL است که روی دادههای بیتی و منطقی و بردارهای بیتی ، منطقی کار می کنند.

- **عملگرهای مقایسی ای**

عملگرهای مقایسه ای روی عملوند های یکسان تعریف می شوند و نتیجه عمل آن یک مقدار منطقی True یا False است.

عملگرهای مقایسه ای عبارتند از > ، < ، =>، <= ، /= ، equal ، Not equal ، Less than ، Less than or equal

- **عملگرهای شیفت**

SLL (شیفت منطقی به چپ)

SLA (شیفت منطقی به راست)

SRL (شیفت ریاضی به چپ)

SRA (شیفت ریاضی به راست)

ROL (چرخش به چپ)

ROR (چرخش به راست)

- **عملگر های ریاضی**

عملگرهای * ، / ، mod، rem،**،ABS،+،-،...

/ و * برای عملوندهای حقیقی و صحیح تعریف می شوند.

mod ، Rem برای عملوندهای Integer تعریف شده اند.

** عملگر توان است.

ساختار کنترلی :

برای پیاده سازی دستورات ترتیبی و همچنین دستورات شرطی و حلقه ها در VHDL ساختارهای کنترلی در نظر گرفته شده است.

ساختارهای شرطی :

ساختار IF :

```
IF شرط THEN
    Statement 1
ELSE
    Statement 2
END IF
```

ساختار IF ELSEIF

```
IF شرط THEN
```

```
        Statement 1
ELSEIF
        Statement 2
ELSEIF
        .
        .
END IF
```

ساختار شرطی SELECT ... WHEN

```
SELECT x CASE
        WHEN CASE 1
        WHEN  CAES 2
        .
        .
END SELECT
```

ساختار حلقه ها :

انواع حلقه ها در زبان VHDL پشتیبانی می شوند مانند FOR یا LOOP

```
LOOP1 : Statement1 LOOP
        .
        .
        Statements loop body
END LOOP LOOP1;
```

```
LOOP1 : FOR  I IN 5 TO 25 LOOP
        Sequential Statement
END LOOP1
```

ضرب اعداد بدون علامت :

ضرب باینری دو عدد باینری بسیار ساده شبیه به ضرب اعداد مبنای ۱۰ میباشد به این ترتیب که در مضروب فیه از کم ارزشترین بیت شروع میکنیم و آن بیت را در تمام بیت های مضروب ضرب میکنیم و حاصل ضرب را نگهداری میکنیم و این عمل را برای تمام بیت های دیگر مضروب فیه انجام میدهیم

مثال :

مضروب 1000

مضروب فیه 1001

1000

0000

0000

1000

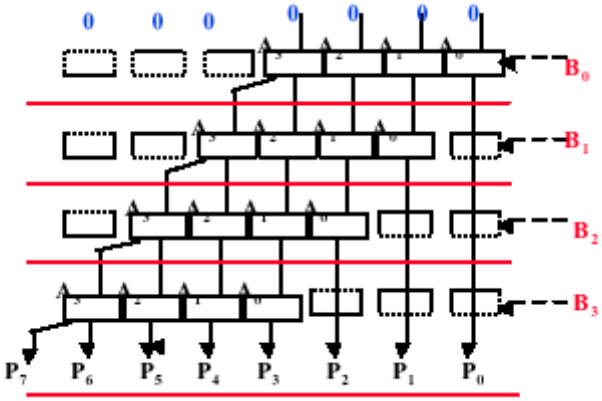
01001000

در ضرب باینری اگر مضروب فیه 0 باشد حاصلضرب صفر و اگر مضروب فیه 1 باشد حاصل برابر با مضروب خواهد بود به این معنی که اگر مضروب 0 باشد حاصل 0 و اگر 1 باشد حاصل 1خواهد شد

توجه کنید که حاصلضرب دو عدد N بیتی وMبیتی M+N بیت خواهد شد

در ادامه ۴ الگوریتم ضرب رامعرفی میکنیم

در حالت عادی روش ضرب دودویی به این صورت است که بیت های مضروب فیه از سمت راست بررسی میشوند و به ازای هر بیت ۱ مقدار مضروب یکی به چپ شیفت داده میشود و به نتیجه حاصلضرب اضافه میگردد و در صورتیکه بیت در مضروب فیه یافت شود بازای هر بیت 0 تنها یک شیفت در نظر گرفته میشود که هنگام رسیدن به رقم 1 بعدی به مضروب اعمال میگردد نهایتا حاصل جمع به دست آمده همان ضرب عدد ۲ است



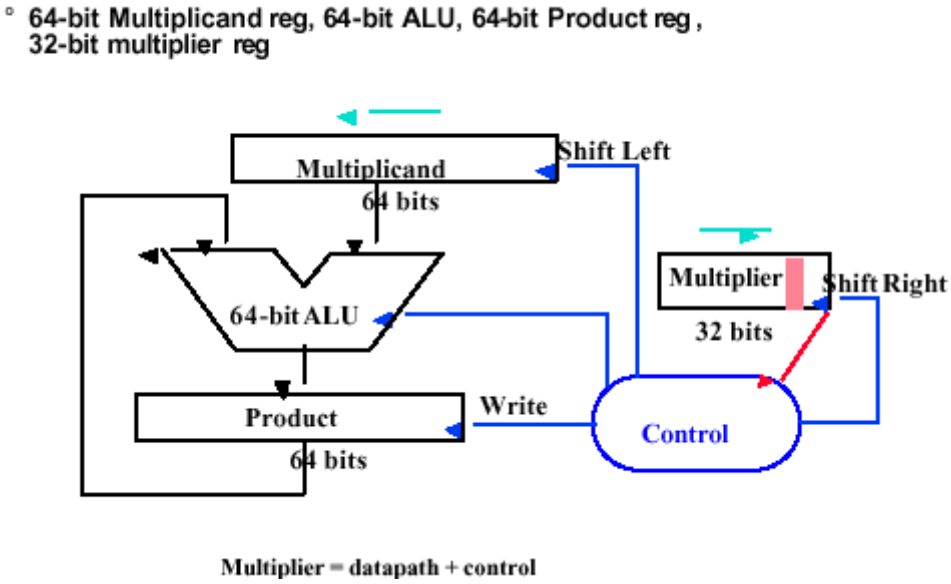
شکل ۳

در مدار بالا در هر مرحله ضرب یک بیت مضروب فیه با تک تک بیتهای مضروب And میشود (به ترتیب از بیت کم ارزش به با ارزش) و در هر مرحله مضروب یک شیفت به چپ داده میشود به گونه ای که در هر مرحله A*2^I محاسبه میشود

پیاده سازی با این روش به مدارپیچیده ای نیاز دارد و به ازای بیتهای زیاد سخت افزار آن بسیار بزرگ و پیچیده خواهد شد

ضرب کننده Shift-Add جهت اعداد بدون علامت

شماتیک سخت افزار مدار در شکل زیر آمده است :



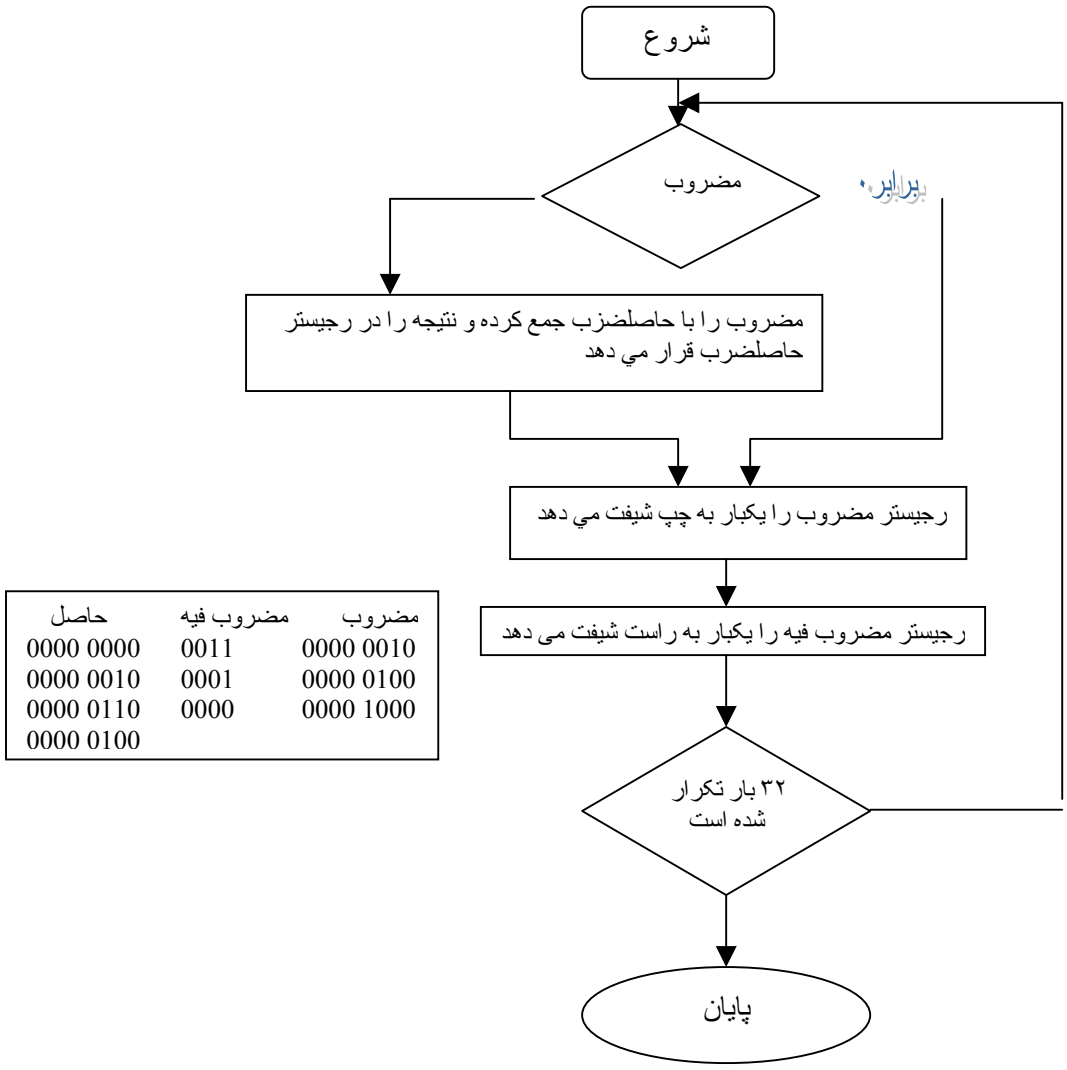
شکل ۴

در این معماری به یک رجیستر ۶۴ بیتی برای مضروب و یک ALU ۶۴ بیتی و رجیستر حاصلضرب ۶۴ بیتی و یک رجیستر ۳۲ بیتی برای نگهداری مضروب فیه نیاز است

مدار کنترل این معماری عملیات شیفت رجیستر های مضروب و مضروب فیه و عمل نوشتن حاصلضرب را در رجیستر مربوطه انجام میدهد

الگوریتم ضرب (نسخه ۱) :

الگوریتم محاسبه ضرب به روش Shift-Add در زیر آمده است :



مضروب	مضروب فیه	حاصل
0000 0010	0011	0000 0000
0000 0100	0001	0000 0010
0000 1000	0000	0000 0110
		0000 0100

شکل ۵

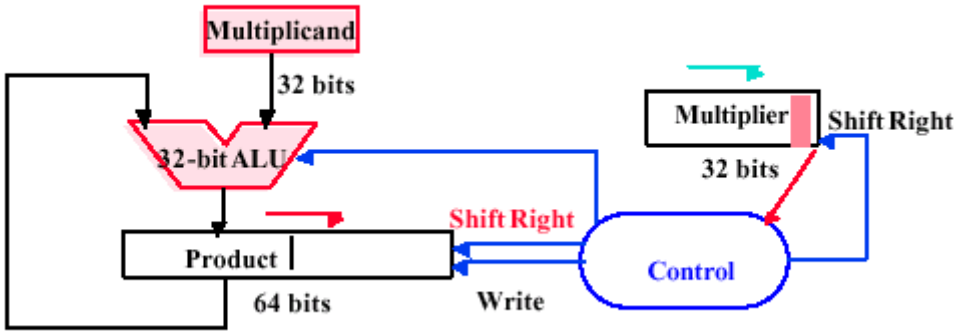
ملاحظاتی بر الگوریتم ضرب نسخه ۱ :

از آنجا که به ازای هر شیفت به چپ مضروب به چپ یک 0 به بیت کم ارزش وارد میشود بطور متوسط نیمی از بیتها همواره 0 میباشد لذا جمع کننده ۶۴ بیتی عملا هدر میرود

سوال : بررسی کنید که اگر بجای شیفت مضروب به چپ حاصلضرب را به راست شیفت دهیم آیا نتیجه درست خواهد بود ؟

ضرب نسخه ۲ :

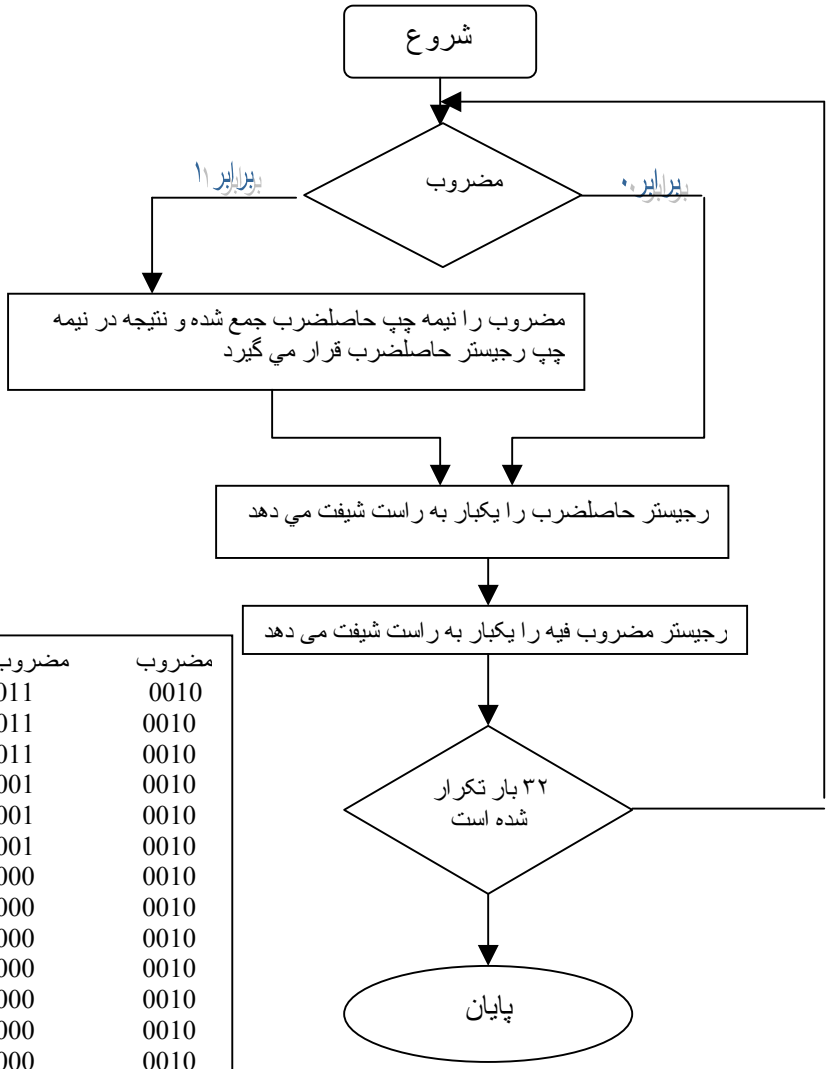
شماتیک سخت افزار ضرب نسخه ۲ در زیر آمده است :



شکل ۶

در این از یک رجیستر ۳۲ بیتی برای مضروب و یک رجیستر ۳۲ بیتی جهت مضروب فیه و یک رجیستر حاصلضرب ۶۴ بیتی و ALU ۳۲ بیتی استفاده میشود

الگوریتم ضرب نسخه ۲ :



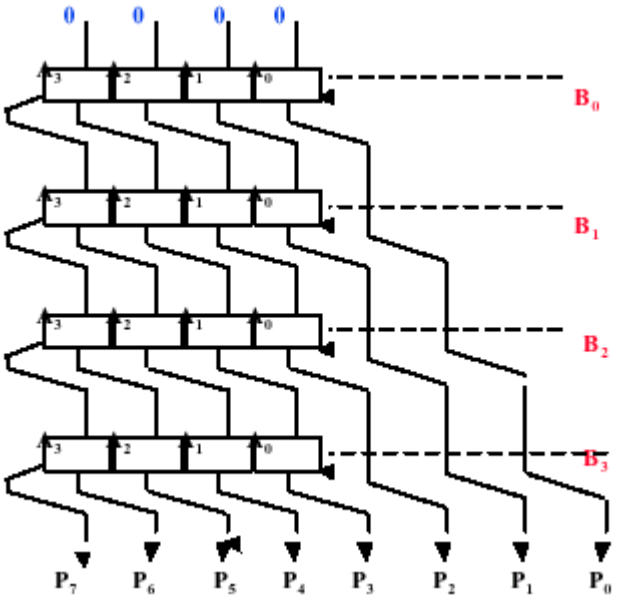
مضروب	مضروب فیه	حاصل
0010	0011	0000 0000
0010	0011	1:0010 0000
0010	0011	2:0001 0000
0010	0001	3:0001 0000
0010	0001	1:0011 0000
0010	0001	2:0001 1000
0010	0000	3:0001 1000
0010	0000	1:0001 1000
0010	0000	2:0000 1100
0010	0000	3:0000 1100
0010	0000	1:0000 1100
0010	0000	2:0000 0110
0010	0000	3:0000 0110

شکل ۷

- در این الگوریتم تیز تعدادی از بیتهای حاصلضرب بدون استفاده خواهد بود
- رجیستر حاصلضرب به اندازه تعداد بیتها در مضروب فیه بیتهایش هدر میرود
- در این الگوریتم رجیستر مضروب و حاصلضرب ترکیب میشوند

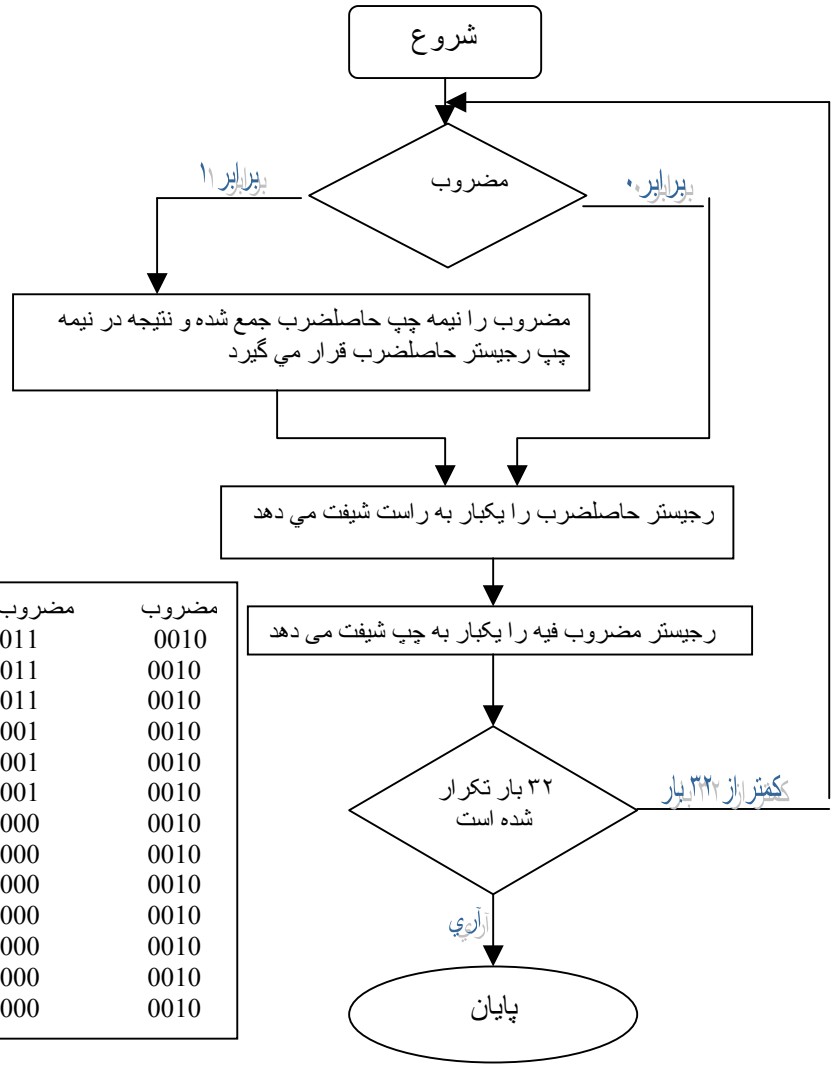
شیفت به راست حاصلضرب بجای شیفت به چپ مضروب

شماتیک زیر عملیات شیفت به راست حاصلضرب را نمایش میدهد



شکل ۸

الگوریتم عملیاتی که مدار فوق انجام می دهد در زیر آمده است :



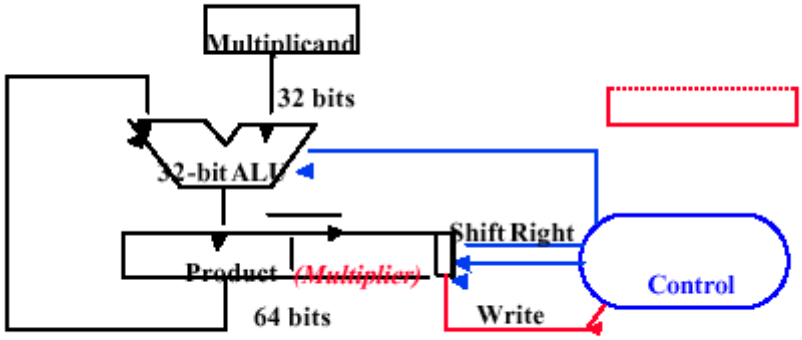
مضروب	مضروب فیه	حاصل
0010	0011	0000 0000
0010	0011	1:0010 0000
0010	0011	2:0001 0000
0010	0001	3:0001 0000
0010	0001	1:0011 0000
0010	0001	2:0001 1000
0010	0000	3:0001 1000
0010	0000	1:0001 1000
0010	0000	2:0000 1100
0010	0000	3:0000 1100
0010	0000	1:0000 1100
0010	0000	2:0000 0110
0010	0000	3:0000 0110

شکل ۹

- در صورتیکه عمل شیفت به راست صورت گیرد نیز تعداد زیادی از بیتهای حاصلضرب بدون استفاده خواهد بود

ضرب نسخه ۳ :

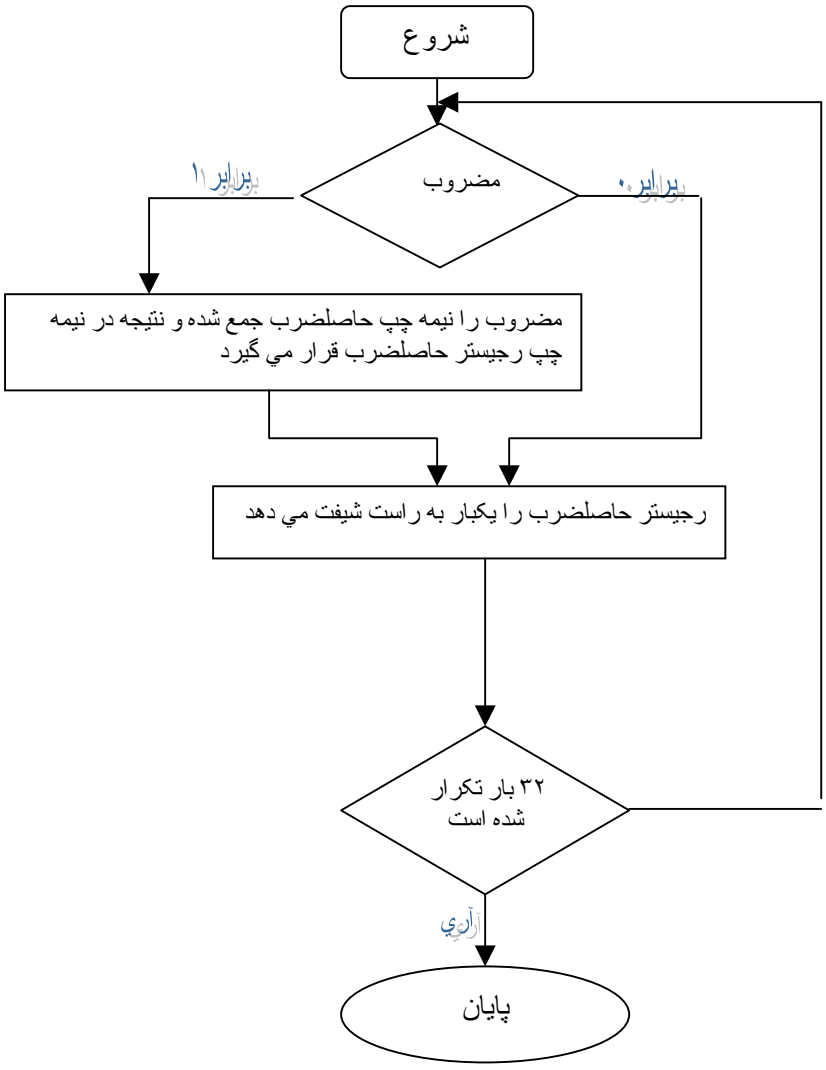
شماتیک سخت افزار ضرب نسخه ۳ در زیر آمده است :



شکل ۱۰

در این الگوریتم رجیستر مضروب ۳۲ بیتی و ALU ۳۲ بیتی و حاصلضرب ۶۴ بیتی خواهد بود و نیازی به رجیستر مضروب فیه نیست

الگوریتم ضرب نسخه ۳ در زیر آمده است :



شکل ۱۱

ملاحظاتی بر ضرب نسخه ۳ :

- این الگوریتم دارای دو مرحله کمتر از دو الگوریتم قبل میباشد
- دو رجیستر مضروب فیه و حاصلضرب با یکدیگر ترکیب شده اند
- بیتهای بلا استفاده در الگوریتم های قبل در این الگوریتم حذف شده اند
- الگوریتم هایی که در بالا بررسی شد در مورد اعداد بدون علامت بود حال اگر در نظر بگیریم که اعداد علامتدار باشند چه راه حلهایی میتوان ارایه داد :
  - ساده ترین روش محاسبه بدون در نظر گرفتن علامت ها و اعمال علامت روی محاسبه حاصلضرب
  - بکار بردن مکمل ۲ برای تبدیل اعداد منفی
  - استفاده از الگوریتم بوت

ضرب بوت :

مثال : محاسبه 6 * 2 :

0010

x 0110

+ 0000 shift (0 in multiplier)

+ 0010 add (1 in multiplier)

+ 0010 add (1 in multiplier)

+ 0000 shift (0 in multiplier)

00001100

توجه : در اکثر موارد مقادیر یکسانی را برای یک عملیات میتوان از طریق مختلف جمع و تفریق محاسبه کرد به عنوان مثال :

6 =-2 + 8

0110= - 0010 + 0100 = 1110 + 0100

0010

x 0110

0000 shift (0 in multiplier)

– 0010 sub (first 1 in multpl.)

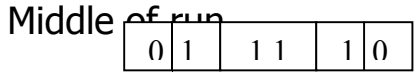
0000 shift (mid string of 1s)

0010 add (prior step had last)

00001100

الگوریتم بوت :

اساس روش بوت به این ترتیب است که رشته ای از صفر ها در ضرب کننده فقط عملیات شیفت را دارند ولی رشته ای از یک ها از ارزش 2 ^ m تا 2 ^ k باعث اضافه شدن 2 –2 ^ k+1 به حاصل جمع میانی میگردد m ^



عملیات	مثال	شرح	بیت سمت راستی	بیت جاری
تفریق	**0001111000**	**در ابتدای اجرا**	**۰**	**۱**
هیچ	**0001111000**	**در میانه اجرا**	**۱**	**۱**
جمع	**0001111000**	**در پایان اجرا**	**۱**	**۰**
هیچ	**0001111000**	**در میانه اجرا**	**۱**	**۰**

برای پیاده نمودن این روش از سمت راست مضروب بررسی می شود و با رسیدن به اولین یک مضروب از حاصل جع میانی کم می شود و با رسیدن به صفر بعدی که قبل از آن یک وجود داشته است مضروب با حاصل جمع میانی جمع می گردد ولی در هر صورت در بررسی هر بیت مضروب عمل شیفت حاصل جمع به راست صورت می گیرد

مثال برلی ضرب بوت :

محاسبه 3- * 2 :

	عملیات	مضروب	حاصلضرب	بعدی؟
0	مقدار دهی اولیه	0010	0000 1101 0	10 -> sub
1a	P=p-m	1110	+ 1110 1110 1101 0	Shift p (sign ext.)
1b		0010	1111 0110 1 +0010	01 -> add
2a			0001 0110 1	Shift p
2b		0010	0000 1011 0 +1110	10 -> sub
3a		0010	1110 1011 0	Shift p
3b		0010	1111 0101 1	11 -> no op

4a			1111 0101 1	Shift
4b		0010	1111 1010 1	Done

محاسبه 7 * 2 :

	عملیات	مضروب	حاصلضرب	بعدی؟
0	مقدار دهی اولیه	0010	0000 0111 0	10 -> sub
1a	P=p-m	1110	+1110 1110 0111 0	Shift p
1b		0010	1111 0011 1	11 -> no op,shift
2		0010	1111 1001 1	11 -> no op,shift
3		0010	1111 1100 1	01 -> add
4a		0010	+0010 0001 1100 1	Shift

4b		0010	0000 1110 0	Done

: instructions

Instraction	example	Meaning command	
and	And $1,$2,$3	$1=$2 & $3	3reg. Operand;logical and
or	Or $1,$2,$3	$1=$2 | $3	3reg. Operand;logical or
xor	Xor $1,$2,$3	$1=$2 xor $3	3reg. Operand;logical Xor
nor	Nor $1,$2,$3	$1=~($2 | $3)	3reg. Operand;logical nor
And immediate	Andi $1,$2,10	$1=$2 & 10	Logical and reg.,constant
Or immediate	Ori $1,$2,10	$1=$2 | 10	Logical or reg. constant
Xor immediate	Xori $1,$2,10	$1=~$2 & ~10	Logical xor reg. ,constant
Shift left logical	Sll $1,$2,10	$1=$2 << 10	Shift left by constant

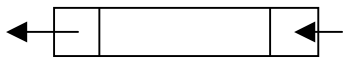
Shift right logical	Srl $1,$2,10	$1=$2 >> 10	Shift right by constant
Shift right arithm.	Sra $1,$2,10	$1=$2 << $3	Shift right (sign extend)
Shift left logical	Sllv $1,$2,$3	$1=$2 << $3	Shift left by variable
Shift right logical	Arlv $1,$2,$3	$1=$2 >> $3	Shift right by variable
Shift right arithm	Srav $1,$2,$3	$1=$2 >> $3	Shift right arith. By variable

شیفت دهنده ها :

۱- منطقی :

- شیفت منطقی به چپ

در این عمل آخرین بیت (با ارزشترین بیت) بیرون می رود و باقی بیتها به اندازه یک بیت به چپ رفته و کم ارزشترین بیت 0 میشود

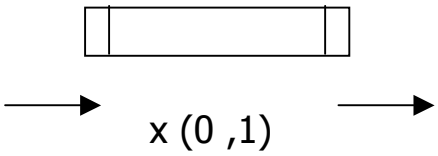


msb lsb

X 0 or 1 0

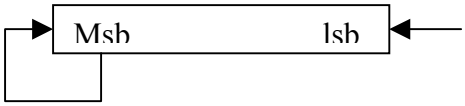
- **شیفت منطقی به راست**

در این عمل بیت کم ارزش بیرون رفته و بیتهای دیگر یک بیت به راست منتقل می شوند و بیت با ارزش بیشتر 0 می شود



0

۲-ریاضی



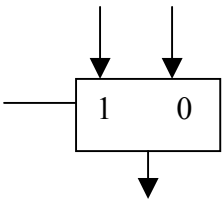
- مدارات فوق تنها یک شیفت را نمایش می دهد ولی در عمل ممکن است تا ۳۲ بیت شیفت انجام گیرد

پیاده سازی شیفت دهنده ها :

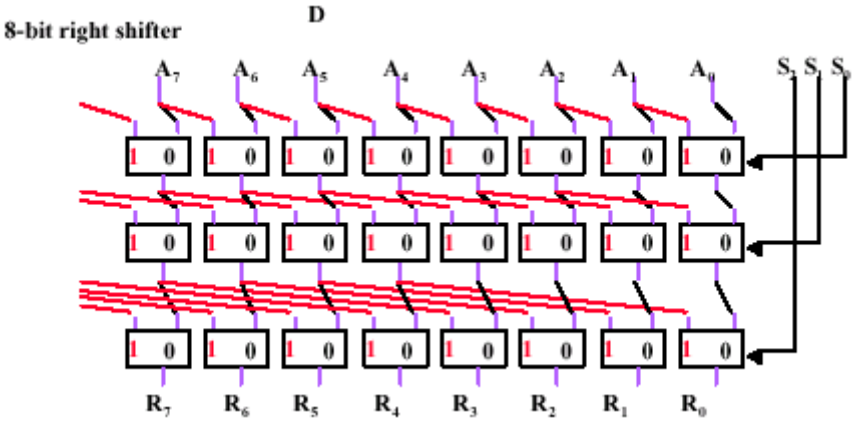
شیفت دهنده های متشکل از مالتی پلکستر ها :

در شکل زیر شماتیکی از از شیفت دهنده های تشکیل شده از Mux2-1 نمایش داده شده است

:



Mux2-1



شکل ۱۲

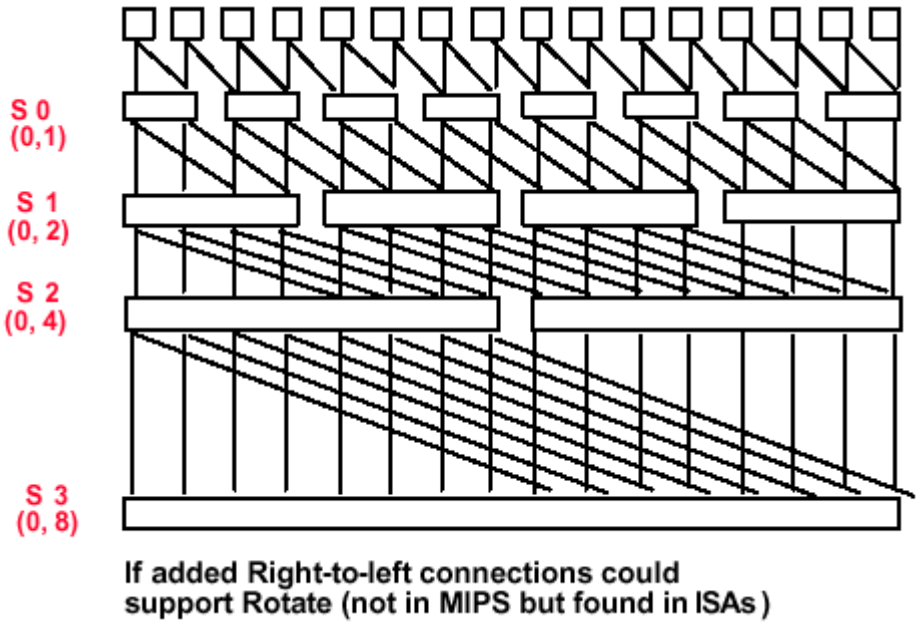
سیگنال های S0,S1,S2 تعداد شیفت ها را مشخص میکنند

S2	S1	S0	R0	R1	R2	R3	R4	R5	R6	R7
0	0	0	A0	A1	A2	A3	A4	A5	A6	A7
0	0	1	A1	A2	A3	A4	A5	A6	A7	X
0	1	0	A2	A3	A4	A5	A6	A7	X	X
0	1	1	A3	A4	A5	A6	A7	X	X	X

- برای دیگر مقادیر S2,S1,S0 جدول راکامل کنید

توجه : اگر از Mux4-1 استفاده شود میتوان شیفت رجیستر با امکان شیفت به راست و چپ داشت

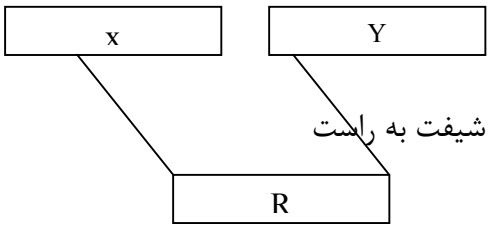
مثال :



شکل ۱۳

شیفت دهنده های فانل :

شماتیک آن در زیر آمده است :



شکل ۱۴

- Shift A by I bit y 32

  x 32

  Sa=shift right amount

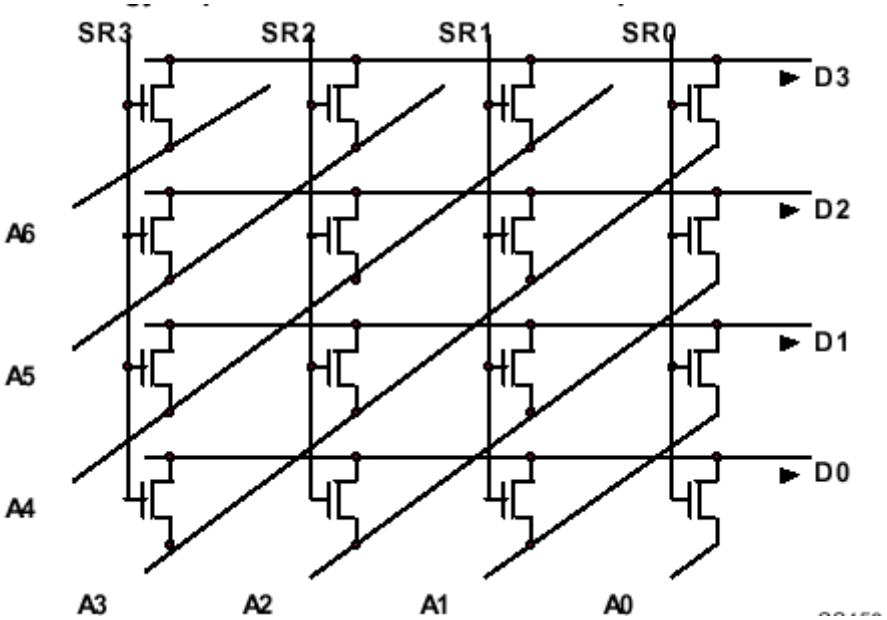
Shift right

- Logical y=0 , x=A , sa=i
- Arithmetic ? y=_ , x=_ , sa=_
- Rotate ? y=_ ,x=_ , sa=_
- Left shifts ? y=_ , x=_ , sa=_ r

  32

شیفت دهنده های بارل :

در پیاده سازی این نوع شیفت دهنده ها از تکنیک سوئیچ استفاده میشود و با توجه به نوع سوئیچ پیاده سازی متفاوت خواهد بود در زیر شماتیکی برای یک نوع شیفت دهنده بارل که از ترانزیستور استفاده کرده است آمده است :



شکل ۱۶

D3	D2	D1	D0	شرط
A3	A2	A1	A0	Sr0
A4	A3	A2	A1	Sr1
A5	A4	A3	A2	Sr2
A6	A5	A4	A3	Sr3