ضرب کننده ی تقریبی



**محقق : نیما مخملی**

**استاد درس: دکتر حاتم عبدلی**

فهرست  
  
مقدمه   
*معرفی* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  
  
*معماری های معروف* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -  
  
*تحلیل و بررسی الگوریتم ها* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -   
  
*الگوریتم انتخاب شده* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -   
الگوریتم اول  
الگوریتم دوم  
  
  
  
  
*منابع* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -   
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
***مقدمه***  
  
  
  
  
  
در مقیاس نانو بهبود عملکرد مدارها و سیستم های دیحیتال به طور فزاینده ای دشوار میشود. بهره وری انرژی یکی از دغدغه های اصلی در طراحی سیستم های دیجیتال است محاسبات با پردازش چندرسانه ای (صوت, ویدئو,گرافیک و تصویر), تشخیص,جستجو,یادگیری ماشین و داده کاوی به طور فزاینده ای سنگین میشود.  
طراحان شروع به جستجوی راه حل های جدید برای محاسبه کارآمد کردند.  
راه حل ها توسط محاسبات تقریبی ارائه شده است. یک ویژگی مشترک این است که یک نتیجه کامل ضروری نیست و یک نتیجه تقریبی یا کمتر از بهینه کافی است.  
محاسبات تقریبی سخت افزار مورد نیاز برای طراحی سیستم را در مقایسه با محاسبات دقیق کاهش می دهد. جمع کننده ها و ضرب کننده ها واحدهای اساسی در هر مدار دیجیتالی هستند که برای انجام محاسبات استفاده می شوند.  
با روند رو به رشد سریع در مقیاس گذاری تا مقیاس نانومتر، مدارهای حسابی باید با توان کم، اندازه فشرده و تاخیر انتشار کمتر اجرا شوند. اینها دلایل تحقق بلوک های جمع کننده و ضرب کننده با استفاده از محاسبات تقریبی هستند. ضریب ها اجزای کلیدی بسیاری از سیستم های دیجیتال با کارایی بالا هستند. عملکرد یک سیستم به طور کلی توسط عملکرد ضریب تعیین می شود، زیرا ضریب معمولاً کندترین عنصر در سیستم است و معمولاً مساحت و توان بیشتری و تأخیر طولانی را مصرف می کند. بنابراین، طراحی ضریب توان کم بخش مهمی در طراحی سیستم VLSI کم مصرف بوده است.

#### محاسبات تقریبی چیست؟ محاسبات تقریبی یک تکنیک خوب محاسباتی هست که یک نتیجه ی احتمالا نادرست را به جای نتیجه ی دقیق و تضمین شده را ایجاد میکند. این روش محاسبات اغلب برای سیستم هایی که هدف آنها یک نتیجه ی تقریبی هست به کار میرود. ضرب کننده های تقریبی !؟ ضرب کننده های تقریبی به طور وسیع برای محاسبات کارآمد انرژی در برنامه هایی که تحمل ذاتی در برابر عدم دقت را نشان میدهند به کار گرفته میشوند. چالش هایی که با آن مواجه هستیم عملکرد , ضریب , مساحت و تاخیر هستند که باعث میشود انتخاب یک ضرب کننده تقریبی بسیار سخت و چالش برانگیز باشد. ضرب کننده های تقریبی , مدارهای بسیار پرکاربرد در حوزه ی طراحی دیجیتال هستند که با کاهش دقت محاسبات اولا باعث کاهش مصرف توان دوما باعث افزایش سرعت پردازش و سوم کاهش مساحت سخت افزاری میشوند. من از تکنیک هایی برای این مقاله استفاده کرده ام که کاربرد اصلی آنها در حوزه های پردازش تصویر , یادگیری ماشین و سامانه های نهفته هست. هدف اصلی ما فدا کردن دقت برای کارایی بیشتر هست.

**معماری های ضرب کننده های تقریبی**   
روش های مختلفی برای طراحی ضرب کننده های تقریبی وجود دارد که چند مورد از آنها را در ادامه شرح کوتاهی خواهم داد.  
  
)Approximate Compressors(  
  
فشرده کننده های تقریبی  
در این روش کاهش تعداد سطوح مدار و تاخیر محاسباتی در ضرب کننده ها استفاده میشوند. در این معماری بیت ها به طور تقریبی جمع میشوند که دقت را کاهش میدهد اما عملکرد بهبود پیدا خواهد کرد**.**وقتی دو عدد باینری را در هم ضرب میکنیم حاصل ضرب شامل چندین ضرب جزئی است که در نهایت باهم جمع میشوند.  
در ضرب کننده های استاندارد تمام این جزئیات محاسبه میشوند. اما در ضرب کننده های تقریبی مبتنی بر حذف بیت کم اهمیت برخی ازاین ضرب های جزئی حذف یا تقریب زده میشوند.  
   
در طراحی ضرب‌کننده‌ها، حاصل‌ضرب دو عدد شامل مجموعه‌ای از بیت‌های جزئی است که باید در چندین مرحله با هم جمع شوند.  
اگر این مراحل به‌صورت دقیق انجام شوند، تأخیر و مصرف توان بالا خواهد بود.  
فشرده‌سازهای تقریبی برخی از بیت‌های با وزن کمتر را تقریب زده یا حذف می‌کنند، بنابراین مدار سریع‌تر و کم‌مصرف‌تر می‌شود  
  
فشرده‌سازها معمولاً در طراحی مدارهای دیجیتال برای کاهش تعداد جمع‌کننده‌های مورد نیاز استفاده می‌شوند.  
که 4 یا 5 بیت را دریافت کرده و 2 خروجی تولید می‌کنند  
معروف‌ترین نوع فشرده‌سازها **(4:2) و (5:2)**

### **فشرده‌ساز دقیق (Exact Compressor)**

### 🔸 یک **(4:2) Exact Compressor** دقیق، چهار بیت ورودی را می‌گیرد و آنها را به دو بیت خروجی Sum و Carry تبدیل می‌کند. 🔸 برای محاسبه‌ی این دو خروجی، نیاز به چندین گیت منطقی (جمع‌کننده‌های کامل و نیمه) داریم که منجر به افزایش توان و تأخیر می‌شود. **روش های فشرده سازی تقریبی** **حذف برخی از بیت‌های کم‌اهمیت (Bit Truncation)**

🔹 در این روش، برخی از ورودی‌های فشرده‌ساز نادیده گرفته می‌شوند تا سخت‌افزار کاهش یابد.  
🔹 این روش خطای نسبی بالایی دارد اما مصرف توان و تأخیر را کاهش می‌دهد.

### **استفاده از گیت‌های تقریبی (Approximate Logic Gates)**

🔹 در این روش، گیت‌های منطقی مورد استفاده در طراحی فشرده‌ساز تغییر داده می‌شوند تا به جای انجام عملیات دقیق، مقدار خروجی را تقریبی محاسبه کنند.

### **تقریب در تولید بیت‌های Carry و Sum**

🔹 در برخی موارد، بیت‌های Carry یا Sum را به‌جای محاسبه‌ی دقیق، به مقدار ثابتی نگاشت می‌کنند تا پردازش سریع‌تر انجام شود.

|  |  |  |
| --- | --- | --- |
| **فشرده‌ساز تقریبی (Approximate)** | **فشرده‌ساز دقیق (Exact)** | **ویژگی** |
| **کمتر** | **زیاد** | **تعداد گیت‌ها** |
| **پایین** | **بالا** | **توان مصرفی** |
| **بیشتر** | **کمتر** | **سرعت** |
| **کمتر** | **بالا** | **دقت** |
| **پردازش تصویر، یادگیری ماشین** | **پردازش‌های دقیق** | **کاربرد** |

**کاربردهای مهم فشرده‌سازهای تقریبی**

**پردازش تصویر و ویدئو:**  
در بسیاری از پردازش‌های تصویر، نیازی به دقت محاسباتی 100٪ نیست. استفاده از فشرده‌سازهای تقریبی باعث کاهش مصرف توان در پردازش‌های بلادرنگ (Real-time) می‌شود.

**شبکه‌های عصبی و یادگیری ماشین:**  
مدل‌های یادگیری عمیق معمولاً با داده‌های نویزی کار می‌کنند و خطای جزئی در محاسبات مشکلی ایجاد نمی‌کند. بنابراین استفاده از فشرده‌سازهای تقریبی باعث افزایش سرعت پردازش می‌شود.

**پردازنده‌های کم‌مصرف و سیستم‌های IoT:**  
در پردازنده‌های تعبیه‌شده (Embedded Systems) که مصرف توان مهم است، از این روش برای بهینه‌سازی مصرف انرژی استفاده می‌شود.

#### ****)Truncation Multipliers(**** در این روش کاهش تعداد سطوح مدار و تاخیر محاسباتی در ضرب کننده ها استفاده میشوند. در این معماری بیت ها به طور تقریبی جمع میشوند که دقت را کاهش میدهد اما عملکرد بهبود پیدا خواهد کرد. وقتی دو عدد باینری را در هم ضرب میکنیم حاصل ضرب شامل چندین ضرب جزئی است که در نهایت باهم جمع میشوند. در ضرب کننده های استاندارد تمام این جزئیات محاسبه میشوند. اما در ضرب کننده های تقریبی مبتنی بر حذف بیت کم اهمیت برخی ازاین ضرب های جزئی حذف یا تقریب زده میشوند. معایب : افزایش خطای محاسباتی در برخی موارد دقت کمتر نسبت به ضرب کننده های دقیق یا معمولی مزایا : کاهش مصرف توان و تاخیر در پردازش کاهش تعداد ترانزیستور ها کاربرد ها : پردازش تصویر یادگیری ماشین حالا میخواهیم چند تکنیک برای حذف بیت های کم اهمیت ارائه دهیم. (simple trunction) حذف مستقیم : در این روش، برخی از بیت‌های حاصل‌ضرب که مقدار آنها کوچک است، مستقیماً نادیده گرفته می‌شوند. ساده ترین راه برای انجام این کار این است که آخرین بیت های خروجی را صفر در نظر بگیریم. این تکنیک مزایایی دارد مانند کاهش تاخیر و مساحت مدار اما خطای محاسباتی نسبی زیادی دارد. (error-based Trunction) دراین روش به جای حذف ساده بیت ها مقدار آنها گرد یا به مقدار دیگری تقریب زده میشود. میتوان مقدار حذف شده را با مقدار ثابتی جایگزین کرد. مزایای این روش کاهش مصرف توان و تاخیر و از معایب آن میتوان به نیاز داشتن به مدار اضافی برای گرد کردن اشاره کرد. (Error Compensation) برخی از روش های پیشرفته بعد از حذف بیت های کم اهمیت مقدار جبرانی به خروجی اضافه میکنند تا خطای حذف را کاهش دهند. مقدار جبرانی معمولا به صورت یک مقدار ثابت یا یک تابع از ورودی ها انتخاب میشود. مزایای آن کاهش میزان خطا و بهینه تر بودن نسبت به حذف مستقیم هست و معایب آن پیچیدگی بالای آن.

|  |  |  |
| --- | --- | --- |
| معایب | مزایا | روش |
| خطای محاسباتی بالا | سادگی , کاهش توان و کاهش تاخیر | حذف مستقیم |
| نیاز به سخت افزار اضافی برای گرد کردن | دقت بالاتر از حذف مستقیم | گرد کردن |
| پیچیدگی سخت افزاری بیشتر | کاهش چشمگیر خطا | تصحیح خطا |

#### 1011

#### × 1101

#### -------------

#### 1011 (1011 × 1)

#### 0000 (1011 × 0)

#### 1011 (1011 × 1, shifted)

#### 1011 (1011 × 1, shifted again)

#### -------------

#### 10001111 (143 در دهدهی)

#### 

#### 1011

#### × 1101

#### -------------

#### 1010 (تقریب زده‌شده 1011 × 1)

#### 0000

#### 1010

#### 1010

#### -------------

#### 10001010 (138 در دهدهی)

کاربردهای این الگوریتم   
  
 **پردازش تصویر:** در بسیاری از الگوریتم‌های پردازش تصویر، مقدار دقیق پیکسل‌ها خیلی مهم نیست، بنابراین این روش در فشرده‌سازی تصویر و فیلترگذاری کاربرد دارد.

**پردازش صوتی:** در بسیاری از الگوریتم‌های پردازش صوت، می‌توان با تقریب زدن مقادیر ضرب، مصرف توان را کاهش داد.

**یادگیری ماشین و هوش مصنوعی:** بسیاری از شبکه‌های عصبی نیازی به دقت بالا در محاسبات ندارند، بنابراین استفاده از ضرب‌کننده‌های تقریبی باعث افزایش کارایی می‌شود.

**سیستم‌های تعبیه‌شده کم‌مصرف:** این روش در میکروکنترلرها و سیستم‌های IoT برای بهینه‌سازی مصرف توان کاربرد دارد.

#### ****(Segment-Based Multiplication)****

در این روش، عملیات ضرب به بخش‌های کوچکتری تقسیم می‌شود و محاسبات تقریبی روی این بخش‌ها انجام می‌شود. این تکنیک باعث کاهش مصرف توان و پیچیدگی مدار می‌شود

🔹 **ضرب مبتنی بر بخش‌بندی (Segment-Based Multiplication)** یک روش تقریبی برای پیاده‌سازی ضرب‌کننده‌ها است که عددهای ورودی را به بخش‌های کوچک‌تر تقسیم کرده و سپس روی این بخش‌ها به‌طور جداگانه عملیات انجام می‌دهد.

🔹 این روش با کاهش پیچیدگی محاسباتی، سرعت پردازش را افزایش می‌دهد و توان مصرفی را کاهش می‌دهد.

**چرا از ضرب مبتنی بر بخش‌بندی استفاده می‌کنیم؟**  
  
 **کاهش پیچیدگی محاسباتی:**  
به‌جای ضرب مستقیم دو عدد بزرگ، آنها را به چند بخش کوچک‌تر تقسیم می‌کنیم و سپس روی این بخش‌ها محاسبات را انجام می‌دهیم.

**کاهش مصرف توان و تأخیر:**  
محاسبات ساده‌تر باعث کاهش تعداد گیت‌های منطقی مورد نیاز و در نتیجه کاهش مصرف انرژی می‌شود.

**افزایش کارایی در پردازش‌های تقریبی:**  
در بسیاری از کاربردهای پردازشی، نیاز به دقت 100٪ نیست، بنابراین می‌توان از این روش برای افزایش سرعت و کاهش هزینه استفاده کرد.  
  
  
  
در روش **Segment-Based Multiplication**، عددهای ورودی به چندین **بخش** (Segment) تقسیم می‌شوند و سپس این بخش‌ها به‌طور جداگانه پردازش می‌شوند.

**مراحل کلی فرآیند ضرب مبتنی بر بخش‌بندی:**

**تقسیم عددهای ورودی:**

* عددهای ورودی را به چندین بخش کوچک‌تر تقسیم می‌کنیم.
* برای مثال، اگر عدد ورودی 8 بیتی باشد، می‌توان آن را به دو بخش 4 بیتی تقسیم کرد.

**محاسبه‌ی حاصل‌ضرب بخش‌ها:**

* هر بخش با بخش متناظر خود در عدد دیگر ضرب می‌شود.
* در برخی روش‌های تقریبی، بخش‌های کم‌اهمیت نادیده گرفته می‌شوند.

**جمع کردن حاصل‌ضرب‌های جزئی:**

* حاصل‌ضرب‌های به‌دست‌آمده از بخش‌های مختلف را با هم جمع می‌کنیم تا حاصل‌ضرب نهایی تولید شود.

**روش‌های مختلف در ضرب مبتنی بر بخش‌بندی**

🔹 روش‌های مختلفی برای تقسیم عدد و انجام محاسبات وجود دارند که دقت و عملکرد متفاوتی ارائه می‌دهند. برخی از مهم‌ترین روش‌ها عبارتند از:

**روش حذف بخش‌های کم‌اهمیت (Truncated Segment Multiplication)**

🔹 در این روش، برخی از بخش‌های کم‌اهمیت عدد ورودی نادیده گرفته می‌شوند تا محاسبات سریع‌تر انجام شود.  
 **مزایا:** کاهش توان مصرفی  
 **معایب:** افزایش خطای محاسباتی

**روش تقریب در ضرب بخش‌ها (Approximate Segment Computation)**

🔹 در این روش، از مدارهای تقریبی برای ضرب بخش‌های مختلف عدد استفاده می‌شود.  
 **مزایا:** بهینه‌سازی سرعت و توان  
 **معایب:** ممکن است دقت کاهش یابد

**روش ضرب بخش‌های وزن‌دار (Weighted Segment Multiplication)**

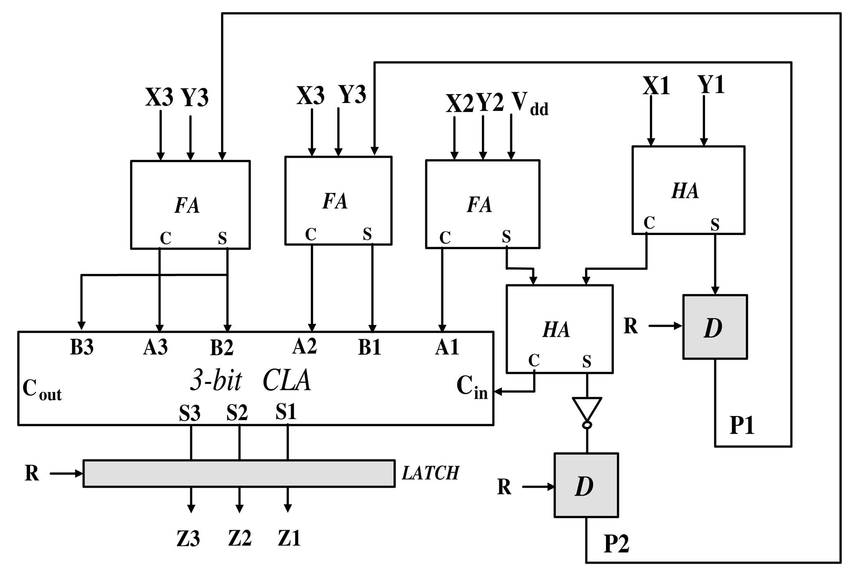
🔹 در این روش، هر بخش از عدد ورودی وزن خاصی دارد و هنگام محاسبه‌ی حاصل‌ضرب، این وزن‌ها لحاظ می‌شوند.  
 **مزایا:** بهبود دقت نسبت به روش‌های دیگر  
 **معایب:** پیچیدگی سخت‌افزاری بیشتر

#### تحلیل الگوریتم ها "بررسی جامع الگوریتم‌های بهینه‌سازی در طراحی ضرب‌کننده‌های تقریبی: تکنیک‌های کاهش پیچیدگی، افزایش سرعت و کاهش مصرف توان. در این مطالعه، روش‌های مختلفی مانند حذف بیت‌های کم‌اهمیت، جایگزینی عملگرها، تقریب لگاریتمی و سایر تکنیک‌های مدرن بررسی شده‌اند. هدف این الگوریتم‌ها دستیابی به طراحی‌های کم‌مصرف و کارآمد برای کاربردهایی مانند پردازش سیگنال، یادگیری ماشین و سامانه‌های تعبیه‌شده است. با بهره‌گیری از این روش‌ها، می‌توان تعادلی میان دقت، سرعت و هزینه سخت‌افزاری ایجاد کرد." الگوریتم ****لگاریتمی (Logarithmic Multiplication)**** یکی از روش‌های تقریبی برای ضرب دو عدد است که به جای انجام ضرب مستقیم، از ویژگی‌های ریاضی لگاریتم‌ها برای تبدیل عملیات ضرب به جمع استفاده می‌کند. در این الگوریتم، دو عدد ورودی ابتدا به لگاریتم تبدیل می‌شوند، سپس این لگاریتم‌ها با هم جمع می‌شوند و در نهایت معکوس لگاریتم حاصل محاسبه می‌شود تا نتیجه ضرب به دست آید. مزایای الگوریتم:

* **افزایش سرعت**: جمع معمولاً سریع‌تر از ضرب است. بنابراین، تبدیل ضرب به جمع می‌تواند به کاهش زمان محاسبات کمک کند.
* **کاهش پیچیدگی محاسبات**: در بعضی شرایط، الگوریتم لگاریتمی می‌تواند پیچیدگی محاسبات را کاهش دهد.

### **معایب:**

* **کاهش دقت**: هنگام تبدیل به لگاریتم و سپس بازگشت به نتیجه، ممکن است دقت محاسبات کاهش یابد. زیرا خود فرآیند تبدیل به لگاریتم و سپس معکوس کردن آن ممکن است با خطا همراه باشد.
* **نیاز به محاسبات اضافی**: محاسبه لگاریتم و تابع نمایی نیاز به منابع اضافی دارد که می‌تواند در سیستم‌های با منابع محدود مشکل‌ساز باشد.
* **الگوریتم نادیده گرفتن بیت‌های کم‌اهمیت (Truncation)** **یکی از روش‌های کاهش پیچیدگی و سرعت بخشیدن به عملیات ضرب در طراحی سخت‌افزار است. در این روش، به جای اینکه تمام بیت‌های ورودی در محاسبات ضرب مورد توجه قرار گیرند، فقط بخش‌های خاصی از آن‌ها که اهمیت کمتری دارند (یعنی بیت‌های کم‌اهمیت) نادیده گرفته می‌شوند. این کار باعث می‌شود که مصرف منابع کاهش یابد و سرعت افزایش یابد، اما در عین حال دقت ضرب کاهش می‌یابد.**



* ابتدا باید مشخص کنید که چه تعداد از بیت‌ها در عملیات ضرب باید نادیده گرفته شوند. این تعداد معمولاً به اندازه‌ای انتخاب می‌شود که تفاوت زیادی در نتیجه نهایی ایجاد نکند.
* به عنوان مثال، اگر شما یک عدد 16 بیتی دارید و تنها 8 بیت اول آن اهمیت دارند، می‌توانید 8 بیت پایین‌تر را نادیده بگیرید.
* بعد از انتخاب تعداد بیت‌های مهم، بیت‌های کم‌اهمیت را از داده ورودی حذف می‌کنید. این کار می‌تواند به سادگی با استفاده از عملگرهای شیفت انجام شود.
* به عنوان مثال، اگر شما یک عدد 16 بیتی دارید و فقط 8 بیت اول آن را استفاده می‌کنید، می‌توانید با یک شیفت منطقی 8 بیتی، 8 بیت کم‌اهمیت را حذف کنید.
* بعد از نادیده گرفتن بیت‌های کم‌اهمیت، عملیات ضرب را تنها بر روی بیت‌های باقی‌مانده انجام می‌دهید.
* به این ترتیب، این عملیات نسبت به ضرب واقعی بسیار سریع‌تر انجام خواهد شد زیرا تعداد بیت‌ها کاهش یافته است.
* در نهایت، نتیجه حاصل از ضرب را می‌توان به نحوی بازسازی کرد تا آن را با فرمت اصلی تطابق دهد. این معمولاً شامل افزودن صفرها یا صفر کردن بیت‌های اضافی در نتیجه است.
* **مزایای الگوریتم:**
* **افزایش سرعت**:
  + نادیده گرفتن بیت‌های کم‌اهمیت باعث کاهش حجم محاسبات می‌شود. چون عملیات ضرب بر روی تعداد کمتری از بیت‌ها انجام می‌شود، زمان محاسبه کاهش می‌یابد.
* **کاهش مصرف منابع**:
  + با کاهش تعداد بیت‌ها، نیاز به منابع سخت‌افزاری کمتر می‌شود. به ویژه در طراحی‌های سخت‌افزاری دیجیتال که تعداد بیت‌ها می‌تواند تأثیر زیادی بر مصرف توان و مساحت مدار بگذارد.
* **سادگی طراحی**:
  + این روش می‌تواند طراحی را ساده‌تر کند زیرا نیاز به پیاده‌سازی و بهینه‌سازی الگوریتم‌های پیچیده‌تر نیست. به سادگی می‌توانید بیت‌های اضافی را حذف کرده و عملیات ضرب را بر روی بخش کوچکتری از داده‌ها انجام دهید.
* **معایب الگوریتم:**
* **کاهش دقت**:
  + بزرگترین عیب این روش کاهش دقت است. نادیده گرفتن بیت‌های کم‌اهمیت می‌تواند منجر به از دست رفتن بخشی از اطلاعات و در نتیجه خطا در نتیجه ضرب شود.
  + این خطا ممکن است در برخی از کاربردها قابل چشم‌پوشی باشد، اما در مواردی که دقت بسیار مهم است، این روش کارایی نخواهد داشت.
* **حساسیت به مقدار ورودی**:
  + اینکه کدام بیت‌ها کم‌اهمیت هستند و باید نادیده گرفته شوند، ممکن است به مقدار ورودی بستگی داشته باشد. در برخی ورودی‌ها، بیت‌های کم‌اهمیت ممکن است تأثیر زیادی در نتیجه نهایی داشته باشند.
* **محدودیت در استفاده**:
  + این الگوریتم برای کاربردهایی مناسب است که دقت نهایی آن‌ها قابل چشم‌پوشی باشد. در بسیاری از کاربردهای حساس به دقت، استفاده از این روش توصیه نمی‌شود.

الگوریتم **جاگذاری OR به جای XOR**   
  
یکی از روش‌های کاهش پیچیدگی در محاسبات ضرب است. این روش به طور خاص در طراحی‌های سخت‌افزاری و زمانی که هدف کاهش مصرف منابع یا افزایش سرعت عملیات است، استفاده می‌شود. در این الگوریتم، به جای استفاده از عملیات XOR که به طور معمول در ضرب‌های دقیق‌تری مانند ضرب‌های دو بیتی یا استفاده در الگوریتم‌های کدگذاری به کار می‌رود، از عملیات OR استفاده می‌شود. این جایگزینی ممکن است باعث کاهش دقت در نتیجه شود، اما در عوض سرعت را افزایش می‌دهد و طراحی ساده‌تری به همراه دارد.

### مراحل الگوریتم جاگذاری OR به جای XOR:

* **تعریف مشکل**:
  + هدف اصلی الگوریتم کاهش پیچیدگی در محاسبات ضرب است. معمولاً در الگوریتم‌های ضرب استاندارد، از عملیات XOR به منظور انجام محاسبات خاص در هنگام ترکیب بیت‌ها استفاده می‌شود.
  + با این حال، برای سرعت بخشیدن به این فرآیند و استفاده کمتر از منابع، می‌توانیم XOR را با OR جایگزین کنیم. این عمل باعث افزایش سرعت محاسبات در سیستم‌هایی می‌شود که نیاز به دقت بالا ندارند.
* **جاگذاری OR به جای XOR**:
  + در مرحله این الگوریتم، به سادگی تمام عملیات‌های XOR که در ضرب‌های معمولی بکار می‌روند، با عملیات OR جایگزین می‌شوند.
  + این تغییر باعث می‌شود که محاسبات سریع‌تر انجام شوند، زیرا OR نسبت به XOR هزینه کمتری دارد.
  + به عنوان مثال، در یک ضرب معمولی دو عدد بیتی، ممکن است بخواهید از XOR برای اعمال اختلاف‌ها و   
    تنظیم بیت‌ها استفاده کنید. اما با استفاده از OR، می‌توانید این فرآیند را با سرعت بالاتر انجام دهید.
* **اجرای عملیات ضرب**:
  + پس از جایگزینی XOR با OR، عملیات ضرب را انجام می‌دهید. در این مرحله، با توجه به اینکه OR استفاده می‌شود، نتیجه ممکن است دقیق نباشد، زیرا OR اطلاعات دقیقی درباره تفاوت بین بیت‌ها به شما نمی‌دهد.
  + با این حال، این روش برای سیستم‌هایی که نیاز به دقت کامل ندارند و بیشتر به سرعت و مصرف کم منابع اهمیت می‌دهند، مناسب است.
* **تحلیل نتیجه و محدودیت‌ها**:
  + با استفاده از OR به جای XOR، احتمالاً دقت کاهش خواهد یافت. این کاهش دقت می‌تواند در برخی سیستم‌ها به مشکلاتی منجر شود، اما در کاربردهای خاص مانند پردازش‌های سریع در سخت‌افزار، این کاهش دقت معمولاً قابل چشم‌پوشی است.
  + اگر دقت برای شما اهمیت زیادی ندارد (مثلاً در شبیه‌سازی‌هایی که نیاز به سرعت بالا دارند)، این روش می‌تواند مؤثر باشد.

### مزایای الگوریتم:

* **افزایش سرعت**:
  + OR معمولاً سریع‌تر از XOR است. استفاده از OR به جای XOR باعث افزایش سرعت عملیات می‌شود، زیرا OR نیاز به عملیات پیچیده‌تری نسبت به XOR ندارد.
* **کاهش مصرف منابع**:
  + OR معمولاً منابع کمتری نسبت به XOR مصرف می‌کند. این ویژگی می‌تواند به کاهش مصرف توان و مساحت مدار در سیستم‌های سخت‌افزاری کمک کند.
* **سادگی طراحی**:
  + با استفاده از OR به جای XOR، طراحی مدارهای سخت‌افزاری ساده‌تر و قابل فهم‌تر می‌شود.

### معایب الگوریتم:

* **کاهش دقت**:
  + بزرگ‌ترین عیب این روش، کاهش دقت نتیجه است. با استفاده از OR، بخشی از دقت ضرب از دست می‌رود، زیرا OR فقط وجود حداقل یک 1 را بررسی می‌کند و تفاوت‌های خاص بین بیت‌ها را نادیده می‌گیرد.
  + این ممکن است در برخی موارد باعث خطا در محاسبات نهایی شود.
* **محدودیت در کاربرد**:
  + این روش بیشتر در کاربردهایی مفید است که دقت کمتری در نتیجه ضرب مورد نیاز باشد. در صورتی که دقت بالا ضروری باشد، این روش مناسب نخواهد بود.

### مثال ساده:

* فرض کنید می‌خواهیم ضرب دو عدد 4 بیتی A=1010A = 1010A=1010 و B=1101B = 1101B=1101 را انجام دهیم.

#### معمولاً با استفاده از XOR:

* در الگوریتم‌های معمولی، شما از XOR برای تطبیق و انجام عملیات‌های خاص روی بیت‌ها استفاده می‌کنید. به طور خاص در برخی الگوریتم‌های ضرب (مانند ضرب‌های دقیق با بیت‌های تکمیلی)، XOR ممکن است به کار رود.

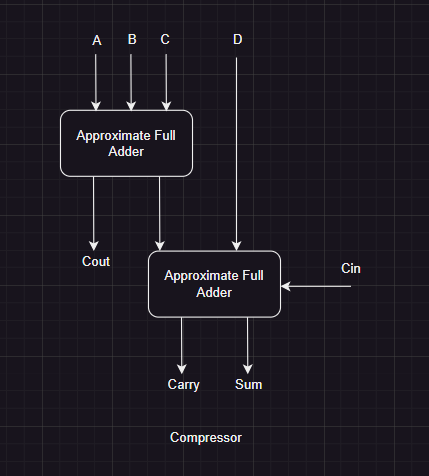
#### با استفاده از OR:

* حالا، اگر شما XOR را با OR جایگزین کنید، محاسبات به سادگی انجام می‌شود:
  + 1010 OR 1101 = 1111
  + 1111 همان نتیجه نهایی است که بدون در نظر گرفتن دقت دقیق به دست می‌آید، اما در اینجا عملیات سریع‌تری انجام شده است.

و اما !  
الگوریتم انتخاب شده در این مقاله   
  
شرح الگوریتم اول   
  
جمع کننده های تقریبی ...  
  
در این بحث قصد دارم تا تقریب های مختلف اعمال شده برای جمع کننده تقریبی را مورد بحث قرار دهیم.  
1) یک تقریب نمیتواند به صورت دلخواه انجام شود. ما باید مطمئن شویم که ساده سازی حاصل باید حداقل خطاها را در جدول درستی FA ایجاد کند در اینجا تقریب برای هردو Half\_Adder , Full\_Adder با جایگزین کردن گیت xor با or انجام میشود زیرا xor تمایل به اشغال فضای بیشتری دارد و تاخیر طولانی ایجاد میکند. این تقریب منجر به یک خطا در محاسبه جمع میشود  
پس از تقریب، معادلات نیم جمع کننده به این صورت میشود   
 **half\_adder : Sum = A+B Carry = A.B**

در جمع کننده کامل، هر یک از گیت های XOR با گیت های OR جایگزین میشود این منجر به خطا در 2 مورد آخر از 8 مورد می شود.   
  
 A = 0,B = 0,Cin = 0 and A = 1,B = 1,Cin = 1   
 **full\_adder : P = A+B Sum = P Xor C Carry = AB+BC**  
  
 جدول صدق جمع کننده کامل نشان می دهد که Sum = Cout برای شش مورد از هشت حالت، به جز ترکیب های ورودی A = 0، B = 0، Cin = 0 و A = 1، B = 1، Cin = 1. با این تقریب sum دو خطا دارد و حمل برای همه موارد خطا ندارد  
 جدول درستی برای full\_adder نشان میدهد که Sum = `Cout برای شش خروجی از هشت تا .  
رهگیری نزدیک جدول صدق جمع کننده کامل نشان می دهد که Cout=A برای شش مورد از هشت مورد یا Cout =B برای 6 مورد. بنابراین دراین تقریب مجموع مانند تقریب قبلی با سه خطا و Cout=A محاسبه می شود. برای کاربردهای مقاوم به خطا، می‌توانیم با Sum=B با اجازه دادن یک خطا بیشتر گسترش دهیم. و Cout=A

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| iiii | iiii | iii | iii | ii | ii | i | i | Exact output | Exact output | input | input | input |
| Cout | Sum | Cout | Sum | Cout | Sum | Cout | Sum | Cout | Sum | Cin | B | A |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |

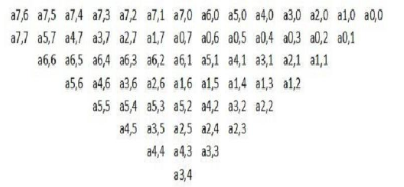
تعداد محصولات جزئی در فرآیند ضرب را می توان با استفاده از کمپرسورها کاهش داد. به طور کلی اندازه کمپرسور را می توان به صورت m:n نشان داد، جایی که m نشان دهنده تعداد بیت های ورودی و n نشان دهنده تعداد بیت های خروجی است. کمپرسورها در دو نوع کمپرسورهای درجه پایین و کمپرسورهای درجه بالا هستند. در این طرح ضریب تقریبی ما از کمپرسور تقریبی مرتبه پایین (4:2) برای کاهش محصولات جزئی استفاده کردیم. کمپرسور تقریبی مرتبه پایین با دو جمع کننده کامل تقریبی طراحی شده است.  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
برای حفظ حداقل اختلاف خطا، در محاسبات مجموع دروازه XOR را می توان با یک گیت OR جایگزین کرد. این تقریب منجر به 5 خطا از 16 مورد می شود.  
Carry = Cin , Sum= (A⊕B) l+ (C⊕D)l Cout=A.B+C.D

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Absolute differende | Approximate compressor | Approximate compressor | exact compressor | exact compressor | in | in | in | in |
| output | Carry | Sum | Carry | Sum | D | C | B | A |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |

**برای نمونه طبق نتیجه یک آزمایش که در اینترنت مشاهده شده میتوانیم از لحاظ زمانی هم آنها را اینگونه مقایسه کنیم.**

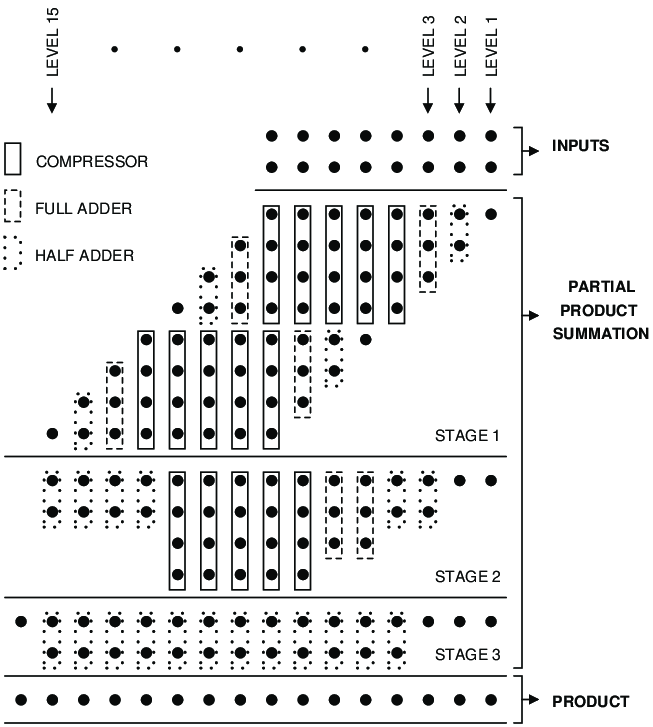
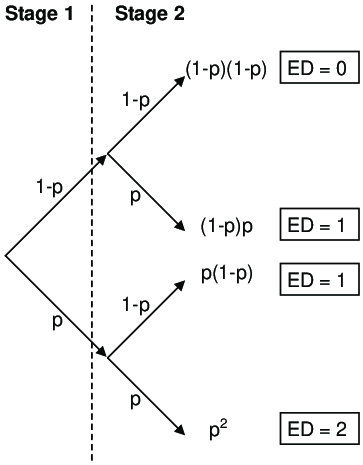
|  |  |  |
| --- | --- | --- |
| **Area in terms of LUTs** | **Delay in ns** | **Compressor Type** |
| **4** | **6.582** | **Exact 4:2 Compressor** |
| **2** | **5.103** | **Approximate**  **compressor 4 : 2**  **compressor** |

***Dadda*** *درمقایسه با سایر تکنیک های پیاده سازی برای فرآیند ضرب، تکنیک دادا برای فرآیند ضرب تعداد مراحل جمع کننده را کاهش می دهد. در این تکنیک از جمع کننده های نیمه و جمع کننده کامل برای جمع بندی محصولات جزئی استفاده می شود. با توجه به این پیچیدگی سخت افزار کاهش می یابد.*

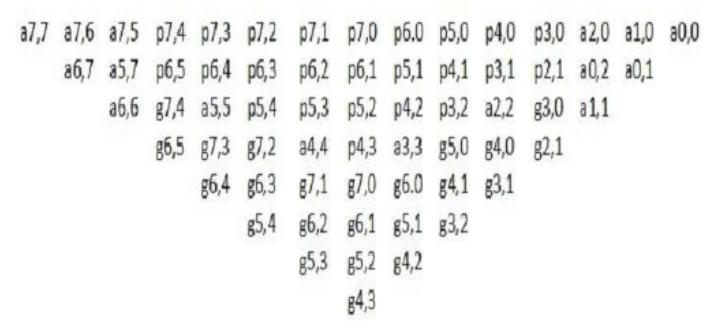
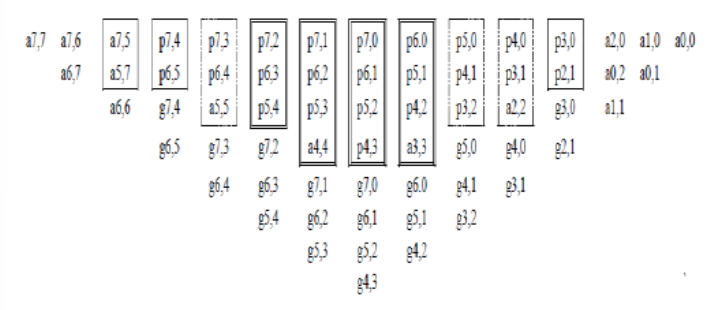
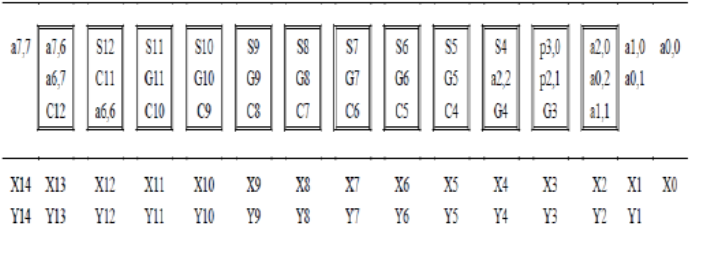
برای توضیح این مقاله از یک ضرب کننده 8در8 استفاده میکنیم   
تکنیک های تقریب در مراحل مختلف فرآیند ضرب اعمال می شوند. تکنیک‌های تقریب متفاوتی برای جمع‌بندی محصولات جزئی پیشنهاد شده‌اند. فرآیند ضرب اساساً به سه مرحله تقسیم می شود 1. تولید ضرب های جزئی 2. کاهش در درخت محصول جزئی با اعمال کمپرسور تقریبی 3. جمع محصولات جزئی تولید حاصلضرب جزئی: در این مقاله ما یک ضرب 8 بیتی و یک را در نظر گرفتیم. ضریب 8 بیتی فرض کنید a مضرب و b ضریب است. نمایش ریاضی ضرب و ضریب به شرح زیر است.  
  
  
a =

b =

am,n=am. bn



**کاهش در درخت حاصلضرب جزئی:**  
 فرض کنید اگر بیت های ضریب و مضرب n=4 داشته باشیم، حاصلضرب جزئی ممکن است از 0 تا 15 باشد. اگر n=8 باشد، شرایط محصول جزئی از 0 تا 63 متغیر است. برای n=16، محصولات جزئی از 0-255 متغیر است، یعنی با افزایش تعداد بیت ها در ورودی ها، تعداد محصولات جزئی نیز به شدت افزایش می یابد. برای جمع همه محصولات جزئی، ضریب به تعداد بیشتری از جمع کننده های نیمه و کامل نیاز دارد که در واقع پیچیدگی سخت افزاری مدار ضریب را افزایش می دهد. بنابراین نیاز به کاهش محصولات جزئی در فرآیند ضرب وجود دارد. کاهش اصطلاحات از نظر آماری قابل انجام است. احتمال بدست آوردن حاصل ضرب جزئی باید 1 باشد.  
 کاهش جزئی محصولات را می توان به صورت ستونی انجام داد. ستونی که بیش از سه محصول جزئی دارد قابل تغییر است. در این معماری ستون 3 به ستون 11 حاصل های جزئی تغییر یافته است. حاصل های جزئی am،n و an،m با هم ترکیب می شوند تا عبارت را تولید کنند. عبارت های تولید شده به صورت Pm,n و gm,n نمایش داده می شوند.

  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
***در این ساختار فقط در مرحله اول به 3 کمپرسور تقریبی و 3 عدد جمع کننده نیم و کامل نیاز داشتیم. تعداد جمع کننده های کامل مورد نیاز به شدت کاهش می یابد***  
  
**نتایج سنتز ضرایب تقریبی برای n=8 طراحی شده اند و با استفاده از Xilinx Artix-7 fpga نگاشت شده اند.  
از گزارش های سنتز , مساحت , تاخیر زمانی , توان دینامیکی و توان استاتیک به شرح جدول هست**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PDP p j | Power in m W | Delay in ns | Area in terms of LUTs | Approximate  Multipliers |
| 310.206 | 78 | 3.977 | 23 | Approximation 1 |
| 348.261 | 87 | 4.003 | 26 | Approximation 2 |
| 217.224 | 56 | 3.879 | 19 | Approximation 3 |
| 189.899 | 53 | 3.853 | 16 | Approximation 4 |

اگر بتوان تقریب بالا را برای صرفه جویی در توان و تأخیر بیشتر تحمل کرد، باید از تقریب 4 استفاده کرد. تقریب 4 43% صرفه جویی در مساحت و 68% صرفه جویی در مصرف انرژی را نسبت به تقریب 1 ارائه می دهد. جدول مقایسه جامعی از ضریب های تقریبی را ارائه می دهد تا ایده ای از مبادله بین مساحت، توان و تاخیر بدست آورید. برای کاربردهایی که در آنها صرفه جویی در مصرف انرژی بالا با مساحت کمتر مورد نظر است، می توان از Approximation 4 استفاده کرد. برای صرفه جویی در مصرف برق با عملکرد بهتر، تقریب II پیشنهاد می شود.  
  
**جمعبندی**   
**در این بخش، ضریب‌های تقریبی مختلفی ارائه کرده‌ایم، برای جمع حاصل ضرب های جزئی، تکنیک‌های تقریب متفاوتی روی جمع‌کننده‌ها اعمال شده‌اند. تقریب 3 و 4 به کاهش قابل توجهی در مساحت و مصرف توان در مقایسه با تقریبی 1دست می‌یابند. طرح‌های ضرب‌کننده پیشنهادی را می‌توان در برنامه‌هایی با حداقل افت کیفیت خروجی و در عین حال صرفه‌جویی قابل توجهی در توان و مساحت مورد استفاده قرار داد.**  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
منابع   
  
آموزش های ویدیویی  
  
[https://youtu.be/DllIOr2NHAY?si=74VZYVNqIanMq7Zg/](https://youtu.be/DllIOr2NHAY?si=74VZYVNqIanMq7Zg)  
  
  
  
  
مقالات

suganthi venkatachalam and Seok -Bum ko,Senor Member,IEEE "Design of Power and Area efficient Approximate Multipliers"

Pooja Rathee,RekhaYadav "Approximate Compressors for Multiplication"International journal on Recent and Innovation

<https://www.researchgate.net/figure/mplementation-metrics-comparison-of-proposed-and-exciting-4-2-compression-design-in_fig3_339751840>