طراحی و پیاده سازی Radix 8 Multiplier پرسرعت همراه با کمپرسورهای 8.2

چکیده:

این مقاله به نحوه اجراء و پیاده سازی Multiplier موازی باعملکرد بالا اشاره دارد Booth 4Radix- Multiplier همراه با کمپرسورهای 3.2 و Radix-8 Booth Multiplierکمپرسورهای 4.2 نیز در این مقاله معرفی شده اند در ادامه، طراحیِ کمپرسور 8.2 ارائه شده و با کمپرسور 4.2 مقایسه گردیده است این طرح از یک الگوی N×M برخوردار بوده که N می تواند تا بیش از 126 بیت را به خود اختصاص دهد . از Carry Look Ahead Adder نیز به عنوان یک Adderنهایی جهت بالا بردن سرعت عملیات استفاده می شود در نهایت، بهبود عملکردِ این ضرب کننده ها با اجراء فیلتر Fir مرتبه بالا، ارزیابی شده و از برنامهVHDL همراه با شبیه سازی Model Sim Se 6.4 درMentor Graphics استفاده شده و در مجموع، با کمک Xilinx Ise 9.2i همراه با Spartan 3 Fpga ، تمام پیاده سازی ها، انجام گردیده است

کلمات کلیدی: FPGA ؛ HDL؛ نگاهی به پیشرونده نگاه دارنده؛ نگهدارنده هزینه حمل؛ والاس درخت؛ رمزگذاری غرفه

مقدمه:

با رشد سریعِ سیستم های ارتباطی و مولتی مدیا، پردازشِ واقعیِ سیگنال ها و پردازش داده های گسترده نیز مورد توجه قرار گرفته است. Multiplier یک عامل اصلیِ پردازش سیگنال دیجیتال نظیر فیلترینگ است اکثر روش های پردازش سیگنال از توابع غیر خطی نظیرDCT یا DWT استفاده می کنند با توجه به اینکه این توابع از کاربردهای گسترده ای برخوردار است لذا سرعت آن ها به عنوان یک عامل مهم، تعیین کننده عملکردِ محاسبات خواهد بود از آنجایی که Multiplier نیاز به تاخیر زیادی در میان بلوک های عملیاتی در سیستم دیجیتال دارد لذا مسیر بحرانی، را می توان با کمک Multiplier تعیین نمود علاوه بر این، Multiplier ، ،فضای زیاد و انرژی

فراوانی را صرف می کند لذا طراحیِ Multiplier ها با قابلیت سرعت بالا و مصرف پایین انرژی ، یکی از حوزه های مورد توجه محققان بوده است.

سرعتِ تکثیر را می توان با کاهش تعداد محصولات ، افزایش داد در میان روش های متعدد اجراء Multiplier های پرسرعت، دو رویکرد اصلی به نام های الگوریتم Booth و کمپرسورهایWallace Tree مطرح شده و در این مقاله سعی شده تا این دو روش بیشتر معرفی شوند

نخستین Multiplier از الگوریتم Radix-4 Booth همراه با کمپرسور های 3.2 استفاده نموده گرچه Multiplier دوم از الگوریتم Radix-8 Booth همراه با کمپرسور های 4.2 استفاده می کند این طرح از یک الگوی n×m برخوردار بوده که n می تواند تا بیش از 126 بیت را به خود اختصاص دهد تعداد محصولات جزئی به صورت n/2 در الگوریتم Radix-4 Booth بوده گرچه این تعداد در الگوریتم Radix-8 Booth به n/3 کاهش می یابد کمپرسور Wallace Tree از Carry Save Adders (Csa) برای توسعه محصولات جزئی استفاده می کند که این امر زمان و فضای تراشه را کاهش می دهد به منظور افزایش سرعت عملیات، Carry-Look-Ahead (Cla) به عنوان Adder نهایی استفاده شده است

Multiplier :

تکثیر یکی از دشوار ترین عملیات ها در پردازشگرهای محاسباتی نظیر ALU محسوب شده لذا در طراحی یک الگو با کمک تراشه مخصوص، به این عامل نیاز است. در این رابطه دو نوع معماری مطرح شده اند: Multiplier های موازی/سری و قابل پیکر بندی .

Multiplier موازی/سری:

Multiplier های سری در طراحی سیستم-تراشه یاSoc استفاده می شوند با توسعه فناوری، هسته های هوشمند تر و بلوک های کارامد تر در Soc ترکیب شده که این امر، افزایش اتلاف انرژی و فضای ارتباطی بیشتری را به دنبال خواهد داشت.افزایش تراکمِ ماژول های تراشه سبب ایجاد باس هایی متصل به این ماژول ها شده که مشکلاتی را به دنبال خواهد داشت و جهت غلبه بر چنین مشکلاتی، به تازگی تکنیک های نوینی برای انتقال داده

ها در یک لینک سریِ پرسرعت به جای باس های سنتی و قدیمی، معرفی شده اند و از اینطریق می توان داده های گسترده تری را در شبکهِ روتینگ انتقال داد.

Multiplier سری-سری:

لینک سری بر روی تراشه قادر به انتقال داده ها بر حسب گیگابایت/ثانیه، بوده لذا زمانی که ماژول مقصد، در شرایط محاسباتی قرار داشته و پردازش دیجیتالیِ سیگنال ها خاتمه می یابد لذا واحد پردازش داده ها(با پیچیدگیِ کم) به عنوا ماژول مقصد، محاسبات جزوئی را بر روی مجموعه داده ها با سرعت بالا انجام داده گرچه داده ها ممکن است با کمک Multiplier سری-سری به عنوان یک ماژول مقصد در Soc بافر شوند. واحدِ محاسبات با پیچیدگیِ کم ، بخشی از Multiplier سری-سری را تشکیل داده ومحاسبات جزئی را بر روی مجموعه داده های سری با سرعت بالا، انجام می دهد

این واحد، به عنوان یک بافر عمل نموده و با پردازش داده ها، عملیات تکثیر آن ها با سرعت کمتر و با کمک یک Multiplier موازی، انجام خواهد گرفت یکی از مشکلات اصلی در این رابطه، کاهش تاخیر مسیر بحرانی در واحدِ پیش محاسبات بوده که معمولا بر حسب گیگابایت بر ثانیه، تمام محاسبات انجام می شود این طرح پیشنهادی می تواند Multiplier ها را در Aprogrammable Gate Array Fpg ، قرار ادده و با توجه به حضور بلاک های منطقی Clb، Multiplier های قرار ادده شده و بلاک های حافظه با Serializer تلفیق شده و عملیات انتقال داده های سری را به منظور کاهش مشکلات ارتباطی، تسهیل می بخشد

انبارهِ سری بر اساس طرح جدید و با سرعت بسیار بالای نمونه برداری از داده ها(بیش از 4 گیگ)توسعه می یابد این انباره از کانترهای 1 برای تقویت بیت ها در هر شرایطِ ماتریسPp استفاده نموده که این امر باعث کاهش زمان تاخیر بحرانی و کم شدن فضا خواهد شد کانتر(شمارشگر) فوق از پیچیدگی های کمی برخوردار بوده اما خروجی ها به راحتی هماهنگ نشده لذا پیش از اتمام عملیات نهایی، زمان تاخیر افزایش خواهد یافت. خروجیِ مناسب شمارشگر باید بعد از آنالیز زمان تاخیر(بخش 5) خوانده شود ارتفاع ماتریس Pp بعد از بافرینگ توسط شمارشگرها، به صورت لگاریتمی کاهش می یابد.

Multiplier موازی-موازی:

در الگوریتمِ Multiplier موازی-سری،مولفه های سری باعث کاهش فضای تراشه سیلیکونی شده و دو مولفه ثابت ارائه شده اندN,M.

$$a(m) = am-1 \ldots\ldots a0$$
$$b(n) = bn-1 \ldots\ldots b0 \qquad (1)$$

مولفهQ نیز بدین صورت تعیین می شود

$$Q(m+n) = \sum \sum a^i b^i 2^{i+j} \qquad (2)$$

Multiplier ها نقش مهمی را در پردازش دیجیتالی سیگنال ها و بسیاری از موارد دیگر ، دارند با رشد و توسعه فناوری، بسیاری از محققان به دنبال طراحی Multiplier هایی با سرعت بالا، مصرف انرژی کم و بازدهی بیشتر بوده که این امر مستلزم استفاده از VLSI است.یکی از روش های مرسوم تکثیر،الگوریتم Add And Shift است در Multiplier های موازی، تعداد محصولات جزئی به عنوان پارامترهای اصلی، تعیین کننده عملکرد Multiplier است به منظور کاهش تعداد این محصولات، الگوریتم Modified Booth ارائه شده است از الگوریتمWallace Tree نیز برای کاهش تعداد عملیات ها می توان استفاده نمود حال با ترکیب این دو روش، می توان مزیت های هر دو را در یک Multiplier خلاصه نمود با این حال، حضور تعداد کثیری از محصولات فرعی باعث کاهش سرعت و افزایش فضای تراشه شده که علت آن، عدم هماهنگی ساختار بوده و افزایش مصرف انرژی هم به خاطر زیاد شدن ارتباطات در مسیر روتینگ صورت گرفته است لذا انتخاب یکMultiplier موازی یا سری تنها به نوع کاربرد بستگی دارد در این مقاله، الگوریتم های تکثیر و معماری، ارائه شده و بر حسب سرعت، قدرت، فضا و ترکیب ماتریس ها، با یکدیگر مقایسه شده اند

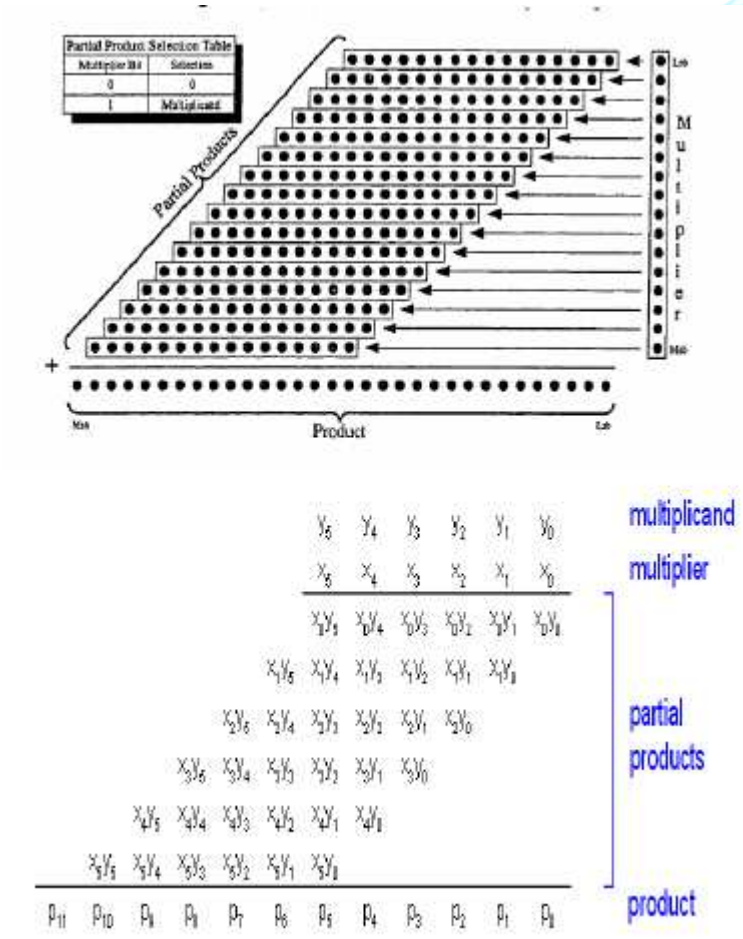
Multiplier های مختلف:

همانطور که می دانید، دو عملگر در عملیات تکثیر وجود دارد: یکی مضروب و دیگری Multiplier . در یک سیستم دودویی، می توان از Multiplier های متنوعی استفاده نمود یک Multiplier دودویی نیز از عملیات

ساده Add And Shift استفاده می کند در صنعت الکترونیک-دیجیتال، Multiplier های فراوانی دیده شده که برخی از آن ها بدین صورت می باشند:

Array Multiplier:

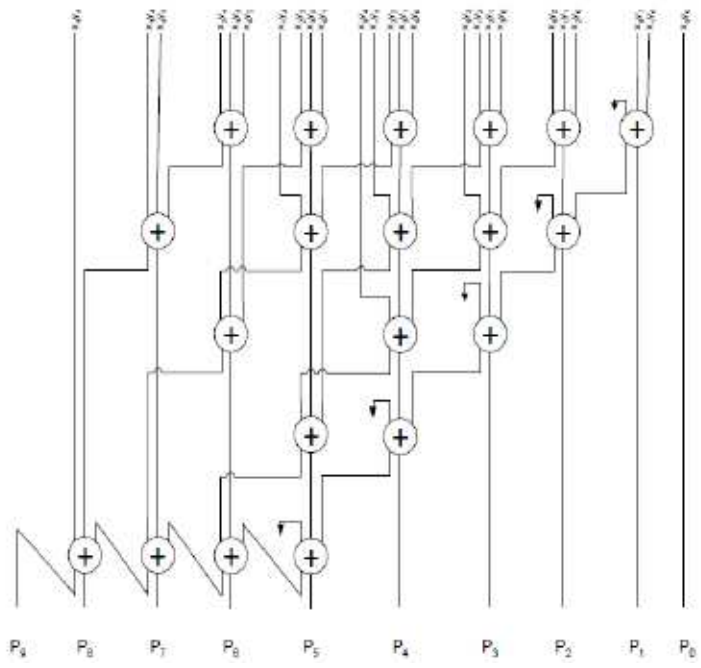
در شکل 1 می توانید این نوع Multiplier را مشاهده نمائید به این Multiplier ، Array می گویند زیرا دارای آرایه ای از .Adders هاست یک Array Multiplier از عملیات Add Shift در یک 22 دودویی استفاده می کند شکل زیر، یک Multiplier 4×4 را نشان می دهد Digital Multiplication از توالیِ زیادی بر روی محصولات جانبی استفاده نموده و محصول نهایی نیز یک عامل کلیدی در میان سایر الگوهای تکثیر به شمار خواهد رفت.



شکل 1: Array Multiplier

Wallace Tree Multiplier:

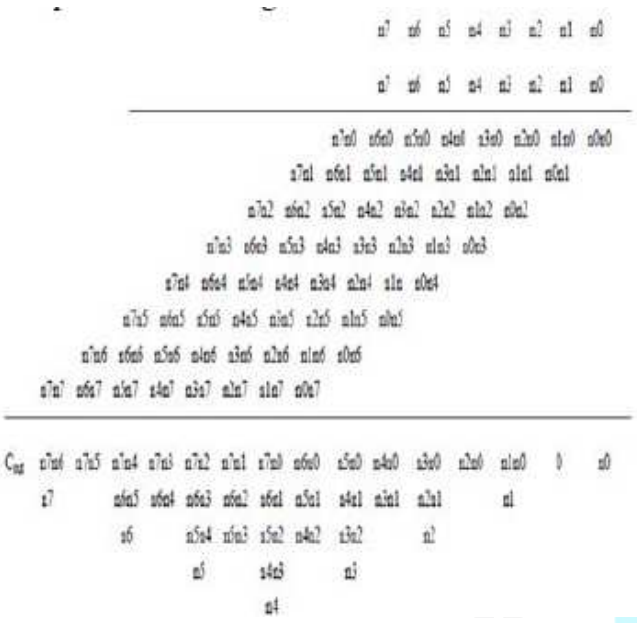
طرح های متعددی با هدف بهبود سرعت Multiplier موازی طی سال های گذشته طراحی شده اند Wallace یکی از ویژگی های مهم Multiplier موازی محسوب شده و این Multiplier ها دارای بیت های بالاتری هستنددر معماری Wallace Tree ،تمام بیت های محصولات جزئی در هر ستون در کنار یکدیگر قرار گرفته اند از مجموعه مشخصِ شمارشگر ها جهت کاهش حجم ماتریس جدید استفاده شده است یکی از رایج ترین شمارشگرها، شمارشگر 3.2 بوده که یکFull Adder است مزیتِ Wallace Tree، سرعت آن است زیرا افزودن محصولات جزئی، به صورت .O (Logn) است در ادامه می توانید نمودار کلیِ Multiplier Wallace Tree 4 بیتی را مشاهده نمائید همانطور که در این نمودار می توان دید ، تعدادی محصول جزئی به بلوک Wallace Tree اضافه شده که در نتیجه بیت های بیشتری به Final Fast Adder (Cra) افزوده گردیده است



شکل2: معماری Wallace Tree

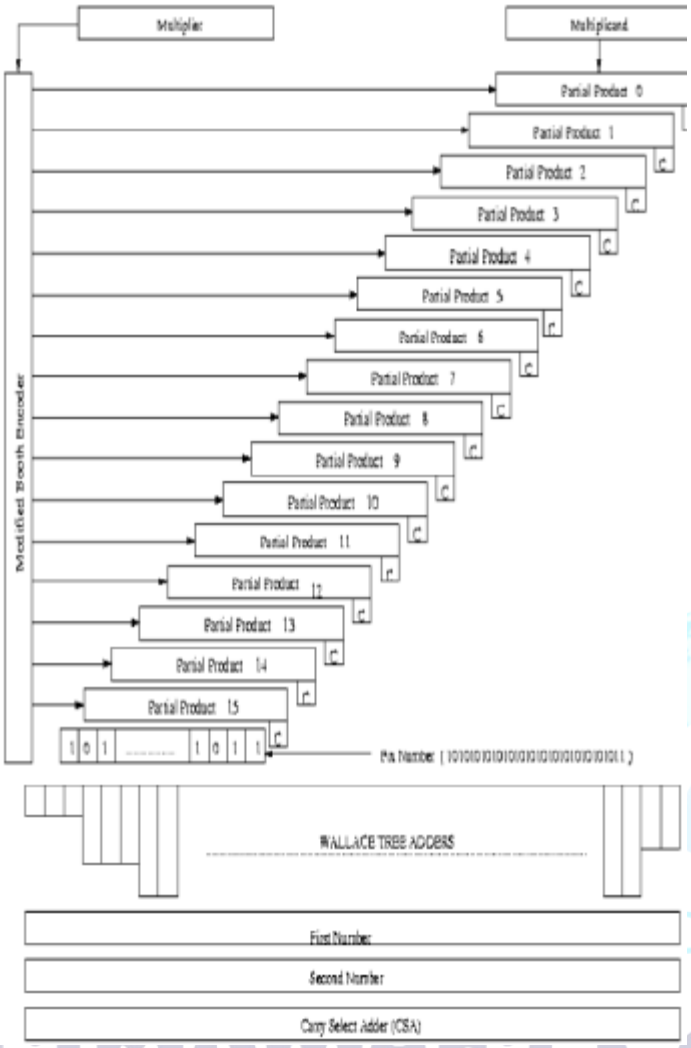
با توجه به اینکه Wallace Tree یک روش جمع بندی است لذا می توان از ان در کنار Multiplier .Array و با هر نوع .Array Booth استفاده نمود در ادامه، اجراء یک Squarer 8 بیتی با استفاده از Wallace Tree

جهت فشرده سازی عملیات اضافه سازی، نمایش داده شده است شکل 2، یک معماری Wallace Tree را نشان می دهد



شکل3: عملیات 8 بیتی Squarer

Wallace، یک ویژگیِ مهم Multiplier موازی است مزیت ان، استفاده از Multiplier های بیش از 16 بیت بوده و در معماری Wallace Tree ، تمام بیت های محصولات جانبی در هر ستون در کنار یکدیگر و بدون هرگونه مشکلی قرار گرفته اند در شکل 3، این طرح نشان داده شده است عملیات 8 بیتی در شکل 4 موجود است در ادامه، مجموعه دیگری از شمارشگرها جهت کاهش حجم ماتریس جدید ، مورد استفاده قرار می گیرد . یکی از رایج ترین شمارشگرها، شمارشگر 3.2 بوده که Adder. Full است مزیتِ Wallace Tree، سرعت آن است زیرا افزودن محصولات جزئی .O (Logn) انجام می شود در ادامه می توانید نمودار کلیِ Wallace Multiplier Tree 4 بیتی را مشاهده نمائید همانطور که در این نمودار می توان دید ، تعدادی محصول جزئی به بلوک Wallace Tree اضافه شده که در نتیجه بیت های بیشتری به Final Fast Adder (Cra) افزوده گردیده است



شکل4: تکثیر 32 بیتی با استفاده از و **Booth** Wallace Tree

: **Radix 2 Booth** Multiplier

الگوریتم Booth ، روشی برای اضافه کردنِ اعداد صحیحِ دوجزئی به صورت دو علامته بوده و مطابق با این روش، Strings Of 0's در این Multiplier نیاز به هیچ عملیات اضافی ندارند اما تغییر در String Of 1's به طریق $2^{k+1} - 2^{m}$. صورت می گیرد الگوریتم Booth ابتدا، Multiplier را ثبت می کند هر بیت در فرمت ثبت شده در Multiplier دارای 3 مقدار است:0-1-2 . فرض کنید می خواهیم عددی را ضربدر 0کمپرسور 0 نمائیم این

عدد ، اختلاف بین 1000 و 00010 را لحاظ می کند عمل تکثیر 0کمپرسور 0 تنها با مجموع محصولات مربوطه، صورت می گیرد

در یک عملیات تکثیر استاندارد، 3 عامل افزوده شده برای دستیابی به رشته ای از 3 .1's نیاز است این عمل را می توان با کمک عملیات تفریق نیز انجام داد در جدول 1، ضرب 0110 با کمک قوانین مربوطه نشان داده شده است

Q_n	Q_{n+1}	Recoded bits	Operation performed
0	0	0	Shift
0	1	+1	Add M
1	0	-1	Subtract M
1	1	0	Shift

جدول1: قوانین ثبت Radix 2

به منظور ایجاد Multiplier ثبت شده برای Radix 2، مراحل زیر باید انجام شوند.

- افزودن Multiplier همراه با 0 به Lsb
- گروه بندی دو بیت به طریق مختلف
- ثبت عدد با کمک جدول فوق

اکنون مثالی را با مضروب 8 بیتی به صورت 11011001 و Multiplier به صورت 0کمپرسور 00010 در نظر بگیرید

Multiplicand	11011001
Multiplier	011100010
	↓↓↓↓↓↓↓↓↓
Recoded multiplier	+1 0 0 -1 0 0 +1 -1
	000100111
	111011001
	000000000
	000000000
	000100111
	000000000
	000000000
	111011001
Product	0000001001001001

الگوریتم اصلاح شده Booth برای Radix 4:

یکی از شیوه های توسعه Multiplier پرسرعت، افزایش قابلیت موازی بودن آن ها بوده که این امر به کاهش مراحل محاسباتی، کمک می کند نسخه اصلیِ الگوریتم Booth از دو عیب برخوردار است:

1. تعداد عملیات جمع-تفریق و عملیات تقسیم، متغیر بوده و چندان سنخیتی در طراحیِ Multiplier های موازی ندارد
2. این الگوریتم ، تنها در .1's های مجزا، چندان اثر بخش نبوده و می توان با کمک .Radix 4 اصلاح شده،بر این مشکل غلبه نمود.

الگوریتم Booth که رشته های 3 بیتی را اسکن می کند بدین صورت عمل خواهد کرد:

- ✓ توسعه موقعیت بیت 1 در صورت نیاز برای تضمین مقدار n
- ✓ اضافه کردن 0 به سمت راست Multiplier LSB
- ✓ مطابق با مقدار هر بردار،هر محصول جزئی باید به صورت 0, +M,-M, +2M or -2M باشد

مقادیر منفیBدارای 2's بوده و در این مقاله از Carry-look-ahead (CLA) استفاده شدتکثیرM با تغییر M از 1 بیت به سمت چپ، انجام شده لذا در هر حالت، در طراحی یک Multiplier موازیِ n بیتی، باید از n/2 محصول جزئی استفاده نمود

محصولات جزئی بر اساس فرمول زیر محاسبه می شوند

$$Z_n = -2 \times B_{n+1} + B_n + B_{n-1} \qquad (3)$$

که B اشاره دارد به Multiplier

Multiplicand 1 0 0 0 0 0 0 1

Multiplier 0 1 1 1 1 1 1 0 0

+2 0 0 -2

0000000011111110

00000000000000

000000000000

1100000010

Product 1100000101111110

جدول 2: قوانین کدگذاریِ **Radix 4** اصلاح شده

Device parameter	Usage of 8:2 compressor	Usage in % 8:2 compressor	Usage of 8:2 compressor	Usage in % 8:2 compressor	Total available
Number of Slices	2121	45	2181	46	4656
Number of 4 input LUTs	3943	42	4044	43	9312
Number of IOs	128		128		
Number of bonded IOBs	128	55	128	55	232

جدول3: مقایسه Multiplier نرمال و اصلاح شده

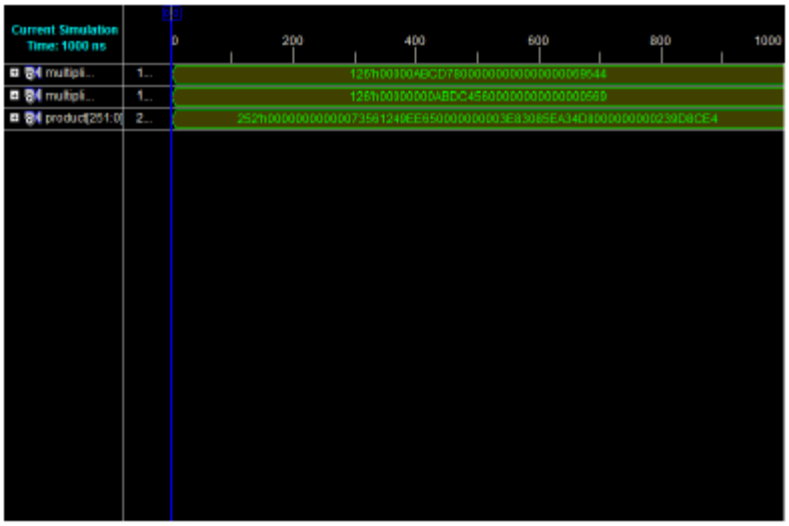
تاخیر کلی در Multiplier اصلاح شده و نرمال:

92.458ns (54.205ns logic, 38.253ns route) (58.6% logic, 41.4% route)

115.924ns (64.207ns logic, 51.717ns route) (55.4% logic, 44.6% route)

Multiplier Bits				Recoded Operation on multiplicand, X
Y_{i+2}	Y_{i+1}	Y_i	Y_{i-1}	
0	0	0	0	0X
0	0	0	1	+X
0	0	1	0	+X
0	0	1	1	+2X
0	1	0	0	+2X
0	1	0	1	+3X
0	1	1	0	+3X
0	1	1	1	+4X
1	0	0	0	-4X
1	0	0	1	-3X
1	0	1	0	-3X
1	0	1	1	-2X
1	1	0	0	-2X
1	1	0	1	-1X
1	1	1	0	-1X
1	1	1	1	0X

جدول 4: کدگذاری مجدد **radix-8** در جدول



شکل 5: نتایج شبیه سازی کمپرسور های 8.2 radix 8

نتایج شبیه سازی:

در radix 8 همراه با کمپرسور های 8.2، نتایج شبیه سازی در شکل 5 نشان داده شده است

مضروب

=00"&x"0000000ABDC456000000000000000569"
Multiplier="00"&x"0000ABCD780000000000000006954
4"
Product=00000000000073561249EE650000000003E83085
EA34D8000000000239D8CE4

نتیجه گیری:

در این مقاله، طراحی و پیاده سازی دو Multiplier موازی با عملکرد بالا ، ارائه گردیده که Multiplier نخست از الگوریتم Booth4Radix با کمپرسور های 3.2 و دیگری از الگوریتم Booth8Radix با کمپرسور های 4.2 استفاده نموده اند هردو طراحی بر روی اجرا شد Multiplier استفاده کننده از Booth4Radix با کمپرسور های 3.2، در مقایسه با نوع دیگر، ازتجهیزات کمتری استفاده نموده است در عین حال، Multiplier که از Booth8Radix و کمپرسور های 8.2 استفاده کرده نسبت به دیگری، پرسرعت تر است همچنین استفاده از کمپرسور Booth8Radix با کمپرسور های 8.2 همراه با فیلترFIR باعث بهبود سرعت نسبت به نوع دیگر الگوریتم، شده است .

VI. REFERENCES

[1] Aparna P R, Nisha Thomas, "Design and Implementation of a High Performance Multiplier using HDL", IEEE Transactions, vol.20, pp.: 401-408, 08 Feb. 2009.

[2] Dong-Wook Kim, Young-Ho Seo, "A New VLSI Architecture of Parallel Multiplier-Accumulator based on Radix-2 Modified Booth Algorithm", Very Large Scale Integration (VLSI) Systems, IEEE Transactions, vol.18, pp.: 201-208, 04 Feb. 2010.

[3] Prasanna Raj P, Rao, Ravi, "VLSI Design and Analysis of Multipliers for Low Power", Intelligent Information Hiding and Multimedia Signal Processing, Fifth International Conference, pp.: 1354-1357, Sept. 2009.

[4] Lakshmanan, Masuri Othman and Mohamad Alauddin Mohd.Ali, "High Performance Parallel Multiplier using Wallace-Booth Algorithm", Semiconductor Electronics, IEEE International Conference , pp.: 433- 436, Dec. 2002.

[5] Jan M Rabaey, "Digital Integrated Circuits, A Design Perspective", Prentice Hall, Dec.1995.

[6] Louis P. Rubin field, "A Proof of the Modified Booth's Algorithm for Multiplication", Computers, IEEE Transactions, vol.24, pp.: 1014-1015, Oct. 1975.

[7] Rajendra Katti, "A Modified Booth Algorithm for High Radix Fixed point Multiplication", Very Large Scale Integration (VLSI) Systems, IEEE Transactions, vol. 2, pp.: 522-524, Dec. 1994.

[8] C. S. Wallace, "A Suggestion for a Fast Multiplier", Electronic Computers, IEEE Transactions, vol.13, Page(s): 14-17, Feb. 1964.

[9] Hussin R et al, "An Efficient Modified Booth Multiplier Architecture", IEEE International Conference, pp.:1-4, 2008.