طراحی و پیاده‌سازی ضرب‌کنندهٔ ماتریس توسط Verilog

احمد سلیمی [۱]، کیمیا نوربخش [۱]، ساعی سعادت [۱]، علیرضا حسین‌پور [۱]
[۱] دانشگاه صنعتی شریف، دانشکده مهندسی کامپیوتر

۱. مقدمه

هدف این پروژه، طراحی، شبیه‌سازی و سنتز یک ضرب‌کنندهٔ ماتریس با زبان وریلاگ است. از کاربردهای چنین ضرب‌کننده‌ای می‌توان به آموزش مدل‌های شبکه‌های عصبی [۱] اشاره کرد. از آن‌جایی که ماتریس وزن‌ها در شبکه‌های عصبی ابعاد بالایی دارند، طراحی یک ضرب‌کننده که هم بتواند در زمان کوتاه پاسخ را آماده کند و هم از نظر سخت افزاری بهینه باشد، از اهمیت بالایی برخوردار است.

روشی که در این پروژه به کاربرده شده است، تقسیم بندی ماتریس به بلوک‌های مربعی و استفاده از خاصیت بلوکی در ضرب ماتریس‌ها است. ساختار این ضرب‌کننده از ۴ لایه تشکیل شده است. بالاترین لایه، لایه ضرب کننده ماتریس موازی است، لایه بعدی ضرب‌کننده ماتریس سطری و ستونی است. لایه سوم ضرب‌کننده ماتریس ترتیبی و لایه آخر ضرب‌کننده و جمع‌کننده floating point است.

در ادامه و در بخش ۲، ابتدا به معماری سیستم و نحوه طراحی ماژول‌های آن پرداخته می‌شود. در بخش ۳، به نحوه انجام شبیه‌سازی‌ها و نتایج حاصل از آن‌ها پرداخته شده است و در نهایت، در بخش ۴، به نحوه انجام عملیات سنتز این سیستم روی FPGA و نتایج حاصل اشاره شده‌است.

۲. معماری سیستم

معماری این سیستم، از سه لایهٔ اصلی تشکیل شده‌است. در ادامه، معماری و جزئیات هر یک از این لایه‌ها، توضیح داده شده است. همچنین این سیستم توسط زبان Verilog پیاده سازی شده‌است که کدهای مربوطه، در پوشهٔ src از ریپازیتوری گیت‌هاب [۲] این پروژه، قابل دسترسی است.

برای ارتباط بین تمامی ماژول‌ها، برای اطمینان از این که ورودی و خروجی‌ها هنگام استفاده شدن تغییر نمی‌کنند و مقدار صحیحی دارند، برای هر کدام دو سیگنال stable و acknowledge در نظر گرفته شده‌است. نحوه‌ی استفاده از آن‌ها بدین گونه است که ماژولی که مقدار را دارد و می‌خواهد آن‌را پاس بدهد، با استفاده از سیگنال stable به ماژول گیرنده اعلام می‌کند که ورودی آماده‌ی استفاده است، سپس ماژول گیرنده با استفاده از سیگنال acknowledge اعلام می‌کند که ورودی را با موفقیت دریافت کرده و ماژول فرستنده می‌تواند آن را تغییر دهد.

۲.۱. *ضرب‌کنندهٔ ماتریس ترتیبی*

در این ماژول مانند ضرب ماتریسی عادی، دو ماتریس $m \times m$ در هم ضرب می‌شوند. برای به دست آوردن درایه ij حاصل‌ضرب، باید سطر i ام ماتریس اول در ستون j ام ماتریس دوم ضرب شود. برای این موضوع به ازای هر $۱ \leq i \leq m, ۱ \leq j \leq m$، داریم:

$$R_{ij} = \sum_{k=۰}^{m} A_{ik} \times B_{kj}$$

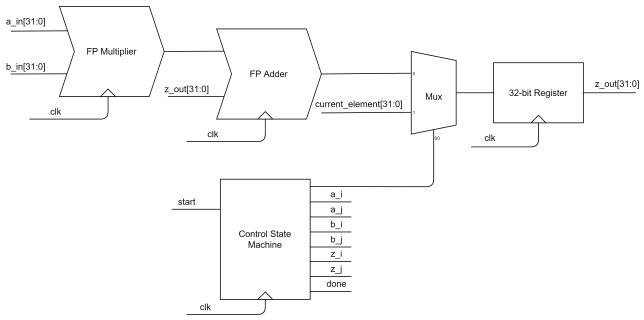
که در آن، R ماتریس $m \times m$ حاصل‌ضرب است.

در این ماژول برای محاسبه جمع و ضرب‌ها، از ماژول‌های جمع‌کننده و ضرب‌کننده اعشاری [۳] استفاده شده‌است.

ماژول ضرب‌کنندهٔ ماتریس ترتیبی [۴] این فرایند را در قالب یک ماشین حالت انجام می‌دهد. برای محاسبه درایه i, j ام، یک accumulator برای نگه‌داری جواب نهایی در نظر گرفته شده‌است و سپس به ازای هر k، ابتدا با استفاده از ماژول FP_multiplier حاصل $A_{ik} \times B_{kj}$ محاسبه شده و با استفاده از ماژول FP_Adder ، به ازای k های مختلف جواب به‌روزرسانی می‌شود.

شکل ۱ بلوک دیاگرام ضرب‌کنندهٔ ماتریس ترتیبی را نشان می‌دهد. باید توجه کرد که حافظه‌ای که حاوی ماتریس‌های ورودی و ماتریس جواب است، در خارج این

[1] Neural Networks
[2] https://github.com/kimianoorbakhsh/Verilog-Matrix-Multiplier
[3] floatng point adder and multiplier
[4] sequential matrix multiplier



شکل ۱: بلوک دیاگرام ضرب‌کنندهٔ ماتریس ترتیبی.

ماژول قرار دارد. در نتیجه، واحد کنترل در این ماشین حالت محدود [۵] اندیس‌های درایه‌های موردنیاز خود، یعنی $a_i, a_j, b_i, b_j, z_i, z_j$ را تعیین می‌کند، و مقادیر مربوط به هر درایه در ماتریس‌های ورودی در a_{in} و b_{in} قراره گرفته، و مقدار z_{out} نیز در ماتریس جواب قرار داده‌می‌شود.

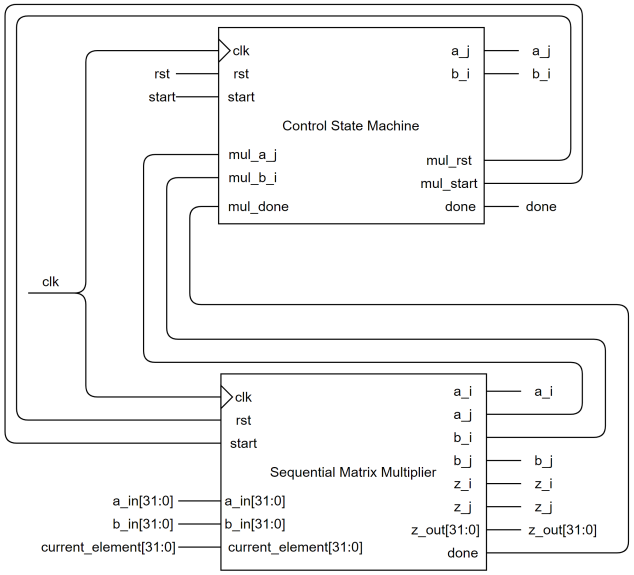
زمان اجرای ضرب ماتریس در این ماژول، $O(m^3)$ است. همچنین مساحت مورد استفاده $O(1)$ است.

۲.۲. *ضرب‌کنندهٔ ماتریس سطری در ستونی*

وظیفهٔ این ماژول، این است که با استفاده از یک ماژول ضرب‌کنندهٔ ماتریس ترتیبی، حاصل‌ضرب یک ماتریس سطری $m \times n$ در یک ماتریس ستونی $n \times m$ را محاسبه‌کند. حاصل این ضرب، یک ماتریس $m \times m$ خواهد بود.
برای انجام این کار، ابتدا ماتریس $m \times n$ به $\lceil \frac{n}{m} \rceil$ ماتریس $m \times m$ تقسیم‌بندی می‌شود. مشابهاً ماتریس $n \times m$ نیز به $\lceil \frac{n}{m} \rceil$ ماتریس $m \times m$ تقسیم‌بندی می‌شود. حال با استفاده از ماژول ضرب‌کننده ماتریس ترتیبی و با توجه به این که قاعده ضرب بلوکی در ماتریس‌ها برقرار است، هر یک از این ماتریس‌های $m \times m$ به مانند یک عدد در نظر گرفته می‌شود و ماتریس‌های متناظر به ترتیب در هم ضرب می‌شوند.

شکل ۲ بلوک دیاگرام ضرب‌کنندهٔ ماتریس سطری در ستونی است. اندیس‌های a_i و b_j مستقیماً از ضرب‌کنندهٔ ماتریس ترتیبی حاصل می‌شوند، اما دیگر

[5] Finite State Machine



شکل ۲: بلوک دیاگرام ضرب‌کنندهٔ ماتریس سطری در ستونی.

اندیس‌ها بصورت

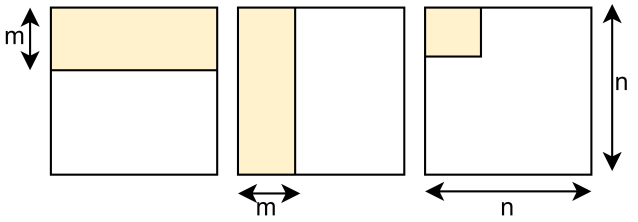
$$a_j = mk + a_{j\text{seq}}$$
$$b_i = mk + b_{i\text{seq}}$$

محاسبه می‌شوند، که در آن، k مشخص می‌کند که چندمین ماتریس $m \times m$ در هر لحظه در حال محاسبه ضرب است، و $a_{j\text{seq}}$ و $b_{i\text{seq}}$ اندیس‌هایی هستند که ضرب‌کنندهٔ ماتریس ترتیبی تعیین کرده‌است.

زمان اجرای ضرب ماتریس در این ماژول، $O(nm^2)$ است. همچنین از آنجایی که در این ماژول صرفا از یک ضرب‌کنندهٔ ترتیبی استفاده شده‌است، مساحت مورد استفاده $O(1)$ است.

۲.۳. *ضرب‌کنندهٔ ماتریس موازی*

این ماژول دو ماتریس $n \times n$ را به صورت موازی در هم ضرب می‌کند. به این صورت که ابتدا این ماتریس، به $\lceil \frac{n}{m} \rceil^2$ ماتریس $m \times m$ تقسیم‌بندی می‌شود. سپس مطابق شکل ۳، هر بلوک $m \times m$ از ماتریس حاصل‌ضرب، توسط یک ضرب‌کنندهٔ ماتریس سطری در ستونی، بصورت موازی با دیگر بلوک‌های حاصل‌ضرب، محاسبه می‌شود. در نتیجه، در این لایه، به تعداد $\lceil \frac{n}{m} \rceil^2$، ضرب‌کنندهٔ ماتریس سطری در ستونی ساخته می‌شود.



شکل ۳: روش محاسبهٔ حاصل‌ضرب یک بلوک از ماتریس جواب، توسط یک ضرب‌کنندهٔ ماتریس سطری در ستونی.

شکل ۴ بلوک دیاگرام ضرب‌کنندهٔ ماتریس موازی را نمایش می‌دهد. حافظهٔ ماتریس‌های ورودی و خروجی در این ماژول موجود است، اما به علت پیچیدگی اتصالات مربوط به این حافظه‌ها، از نمایش آن‌ها در بلوک دیاگرام صرف نظر شده‌است.

زمان اجرای ضرب ماتریس در این ماژول، معادل زمان اجرای یک ضرب‌کنندهٔ سطری در ستونی است که برابر $O(nm^۲)$ است. همچنین از آنجایی که در این ماژول از $(\frac{n}{m})^۲$ ماژول ضرب‌کنندهٔ سطری در ستونی استفاده شده‌است، مساحت مورد استفاده $O((\frac{n}{m})^۲)$ است.

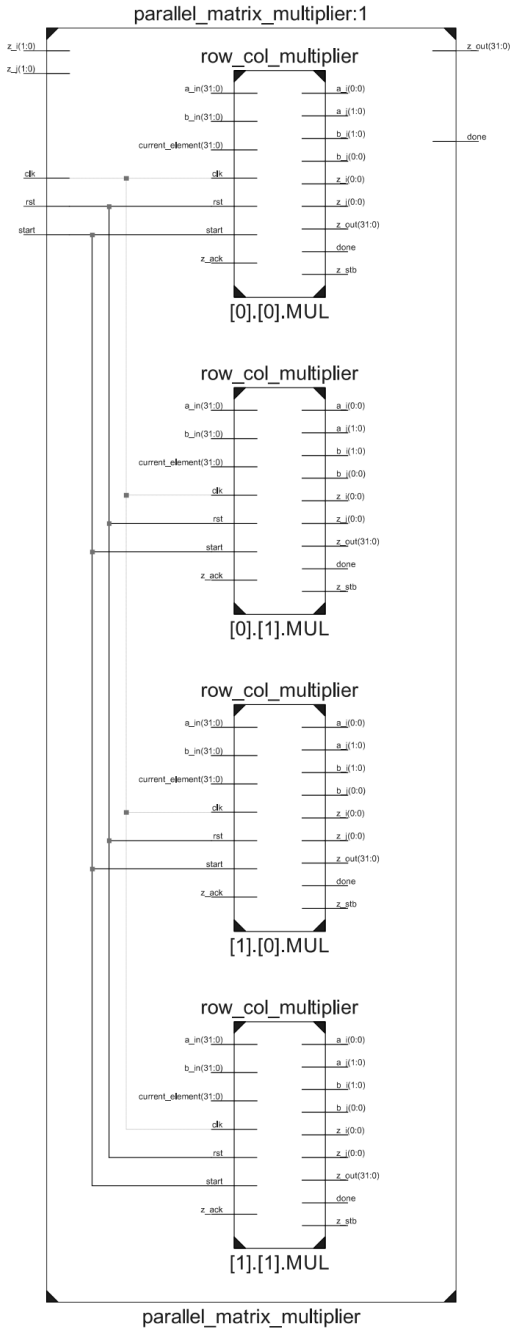
در نهایت، ساختار درختی کل سیستم در شکل ۵ نمایش داده‌شده است.

۳. شبیه‌سازی و نتایج

برای هر یک از ماژول‌های شرح داده شده در بخش قبل، یک Testbench نوشته شده‌است. مقادیر ورودی، در قالب فایل‌های باینری در پوشهٔ data موجود اند و در ابتدای اجرای شبیه‌سازی، از فایل‌های مربوطه خوانده شده و در حافظه نوشته می‌شوند. همچنین پس از اجرای عملیات ضرب ماتریس، خروجی حاصل در فایل sim_out.bin نوشته می‌شود.

برای بررسی و مقایسهٔ پاسخ‌های حاصل از شبیه‌سازی، یک مدل طلایی توسط زبان پایتون نوشته شده‌است که ورودی‌ها و خروجی حاصل از شبیه‌سازی را دریافت کرده، و پاسخ صحیح را با پاسخ به دست آمده مقایسه می‌کند. اجرای مدل طلایی، توسط دستور زیر قابل انجام است.

```
python gold_standard/model.py data/<input_a address> data/<input_b address> <sim_out.bin address> <n> <k> <m>
```

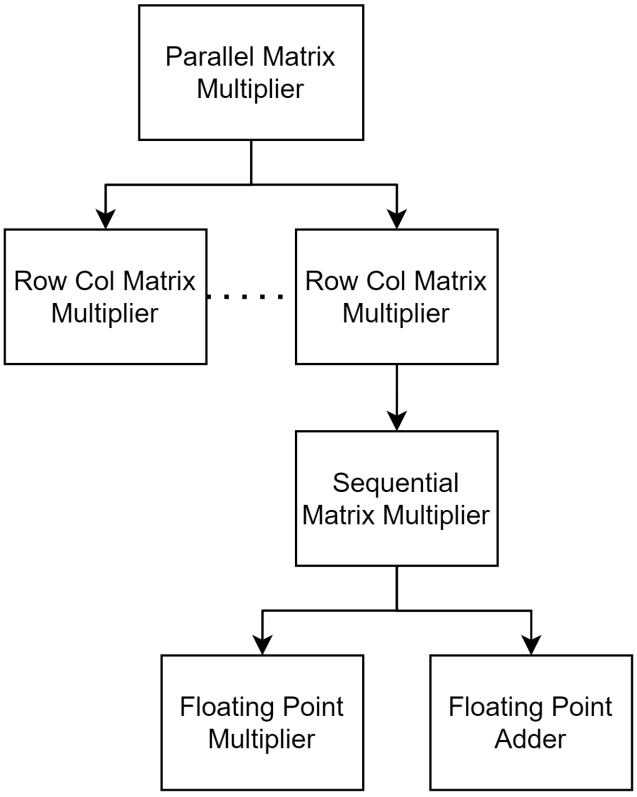


شکل ۴: بلوک دیاگرام ضرب‌کنندهٔ ماتریس موازی.

در ادامه به هر کدام از Testbenchها به تفصیل پرداخته می‌شود.

۳.۱. *ضرب‌کنندهٔ ماتریس ترتیبی*

برای آزمایش و شبیه‌سازی این ماژول، دو ماتریس ۴×۴ به صورت زیر در هم ضرب شدند.



شکل ۵: ساختار درختی سیستم.

$$\begin{bmatrix} ۱ & ۲ & ۳ & ۴ \\ ۵ & ۶ & ۷ & ۸ \\ ۹ & ۱۰ & ۱۱ & ۱۲ \\ ۱۳ & ۱۴ & ۱۵ & ۱۶ \end{bmatrix} \cdot \begin{bmatrix} ۲ & ۰ & ۰ & ۰ \\ ۰ & ۲ & ۰ & ۰ \\ ۰ & ۰ & ۲ & ۰ \\ ۰ & ۰ & ۰ & ۲ \end{bmatrix} = \begin{bmatrix} ۲ & ۴ & ۶ & ۸ \\ ۱۰ & ۱۲ & ۱۴ & ۱۶ \\ ۱۸ & ۲۰ & ۲۲ & ۲۴ \\ ۲۶ & ۲۸ & ۳۰ & ۳۲ \end{bmatrix} \quad (۱)$$

در این Testbench، ابتدا ورودی‌ها از فایل‌های مربوطه خوانده شده، سپس سیگنال start مربوط به ضرب‌کنندهٔ ماتریس ترتیبی فعال می‌شود. پس از فعال شدن سیگنال done در ماژول ضرب‌کننده، نتیجهٔ حاصل در فایل خروجی نوشته می‌شود. نتیجهٔ اجرای مدل طلایی نیز بصورت زیر است.

```
> python .\gold_standard\model.py .\data\
    square_input_a.bin .\data\square_input_b.bin
    .\sim_out.bin 4 4 4
A:
[[ 1.  2.  3.  4.]
[ 5.  6.  7.  8.]
[ 9. 10. 11. 12.]
[13. 14. 15. 16.]]
B:
[[2. 0. 0. 0.]
[0. 2. 0. 0.]
[0. 0. 2. 0.]
[0. 0. 0. 2.]]
Actual:
[[ 2.  4.  6.  8.]
[10. 12. 14. 16.]
[18. 20. 22. 24.]
[26. 28. 30. 32.]]
Expected:
[[ 2.  4.  6.  8.]
[10. 12. 14. 16.]
[18. 20. 22. 24.]
[26. 28. 30. 32.]]
True
```

۳.۲. *ضرب‌کنندهٔ ماتریس سطری در ستونی*

برای آزمایش و شبیه‌سازی این ماژول، دو ماتریس ۸×۴ و ۴×۸ به صورت زیر در هم ضرب شدند.

$$A \times B$$

$$= \begin{bmatrix} ۱ & ۲ & ۳ & ۴ & ۵ & ۶ & ۷ & ۸ \\ ۱ & ۲ & ۳ & ۴ & ۵ & ۶ & ۷ & ۸ \\ ۱ & ۲ & ۳ & ۴ & ۵ & ۶ & ۷ & ۸ \\ ۱ & ۲ & ۳ & ۴ & ۵ & ۶ & ۷ & ۸ \end{bmatrix} \cdot \begin{bmatrix} ۱/۵ & ۰ & ۰ & ۰ \\ ۰ & ۱/۵ & ۰ & ۰ \\ ۰ & ۰ & ۱/۵ & ۰ \\ ۰ & ۰ & ۰ & ۱/۵ \\ ۱/۵ & ۰ & ۰ & ۰ \\ ۰ & ۱/۵ & ۰ & ۰ \\ ۰ & ۰ & ۱/۵ & ۰ \\ ۰ & ۰ & ۰ & ۱/۵ \end{bmatrix}$$

$$= \begin{bmatrix} ۹ & ۱۲ & ۱۵ & ۱۸ \\ ۹ & ۱۲ & ۱۵ & ۱۸ \\ ۹ & ۱۲ & ۱۵ & ۱۸ \\ ۹ & ۱۲ & ۱۵ & ۱۸ \end{bmatrix}$$

در این Testbench، ابتدا ورودی‌ها از فایل‌های مربوطه خوانده شده، سپس سیگنال start مربوط به ضرب‌کنندهٔ ماتریس ترتیبی فعال می‌شود. پس از فعال شدن سیگنال done در ماژول ضرب‌کننده، نتیجهٔ حاصل در فایل خروجی نوشته می‌شود. نتیجهٔ اجرای مدل طلایی نیز بصورت زیر است.

```
> python .\gold_standard\model.py .\data\row_input
    .bin .\data\col_input.bin .\sim_out.bin 4 8 4
A:
[[1. 2. 3. 4. 5. 6. 7. 8.]
[1. 2. 3. 4. 5. 6. 7. 8.]
[1. 2. 3. 4. 5. 6. 7. 8.]
[1. 2. 3. 4. 5. 6. 7. 8.]]
B:
[[1.5 0.  0.  0. ]
[0.  1.5 0.  0. ]
[0.  0.  1.5 0. ]
[0.  0.  0.  1.5]
[1.5 0.  0.  0. ]
[0.  1.5 0.  0. ]
[0.  0.  1.5 0. ]
[0.  0.  0.  1.5]]
Actual:
[[ 9. 12. 15. 18.]
[ 9. 12. 15. 18.]
[ 9. 12. 15. 18.]
[ 9. 12. 15. 18.]]
Expected:
[[ 9. 12. 15. 18.]
[ 9. 12. 15. 18.]
[ 9. 12. 15. 18.]
[ 9. 12. 15. 18.]]
True
```

۳.۳. *ضرب‌کنندهٔ ماتریس موازی*

برای آزمایش و شبیه‌سازی این ماژول، همان دو ماتریس رابطه ۱ در هم ضرب شده‌اند. اما تفاوت در این‌جا این است که در این حالت هر یک از ماتریس‌ها، به چهار ماتریس ۲×۲ تقسیم شده‌اند. $(m = ۲)$

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan6
Device	XC6SLX150
Package	FGG484
Speed	-3
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	Verilog
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	☐

شکل ۶: تنظیمات FPGA.

در این Testbench، ابتدا ورودی‌ها از فایل‌های مربوطه خوانده شده، سپس سیگنال start مربوط به ضرب‌کنندهٔ ماتریس ترتیبی فعال می‌شود. پس از فعال شدن سیگنال done در ماژول ضرب‌کننده، نتیجهٔ حاصل در فایل خروجی نوشته می‌شود. نتیجهٔ اجرای مدل طلایی نیز بصورت زیر است.

```
> python .\gold_standard\model.py .\data\
    square_input_a.bin .\data\square_input_b.bin
    .\sim_out.bin 4 4 4
A:
[[ 1.  2.  3.  4.]
[ 5.  6.  7.  8.]
[ 9. 10. 11. 12.]
[13. 14. 15. 16.]]
B:
[[2. 0. 0. 0.]
[0. 2. 0. 0.]
[0. 0. 2. 0.]
[0. 0. 0. 2.]]
Actual:
[[ 2.  4.  6.  8.]
[10. 12. 14. 16.]
[18. 20. 22. 24.]
[26. 28. 30. 32.]]
Expected:
[[ 2.  4.  6.  8.]
[10. 12. 14. 16.]
[18. 20. 22. 24.]
[26. 28. 30. 32.]]
True
```

شکل موج‌ها [6] در بخش ضمیمه‌ها نمایش داده شده‌است.

۴. سنتز بر روی FPGA و نتایج

با استفاده از نرم‌افزار XilinxISE، پروژه‌ای برای یک FPGA با تنظیماتی که در شکل ۶ نمایش داده‌شده است، ساخته شد. سپس فایل‌های FP_adder.v، FP_multiplier.v، sequential_matrix_multipiler.v، row_col_multiplier.v و parallel_matrix_multiplier.v به پروژه اضافه شد و ماژول parallel_matrix_multiplier به عنوان System Top در نظر گرفته شد. پس از تعیین منابع پروژه، System Top سنتز شد. در این سنتز، مقدار پارامتر n برابر ۴ و m برابر ۲ در نظر گرفته شد.

در جدول I مقدار استفاده از بخش‌های مختلف FPGA که توسط ابزار سنتز بدست آمده‌است، نشان داده شده است. گزارش زمانی سنتز نیز بصورت زیر می‌باشد.

```
Timing Summary:
---------------
Speed Grade: -3

Minimum period: 15.993ns (Maximum Frequency:
    62.528MHz)
Minimum input arrival time before clock: 2.762ns
Maximum output required time after clock: 5.694ns
Maximum combinational path delay: 7.920ns
```

خروجی‌های مربوط به سنتز، در پوشهٔ synthesis موجود است.

Table I: Device Utilization Summary (estimated values).

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	2932	184304	1%
Number of Slice LUTs	4508	92152	4%
Number of fully used LUT-FF pairs	2261	5179	43%
Number of bonded IOBs	40	338	11%
Number of BUFG/BUFGCTRLs	1	16	6%
Number of DSP48A1s	16	180	8%

۵. نتیجه‌گیری

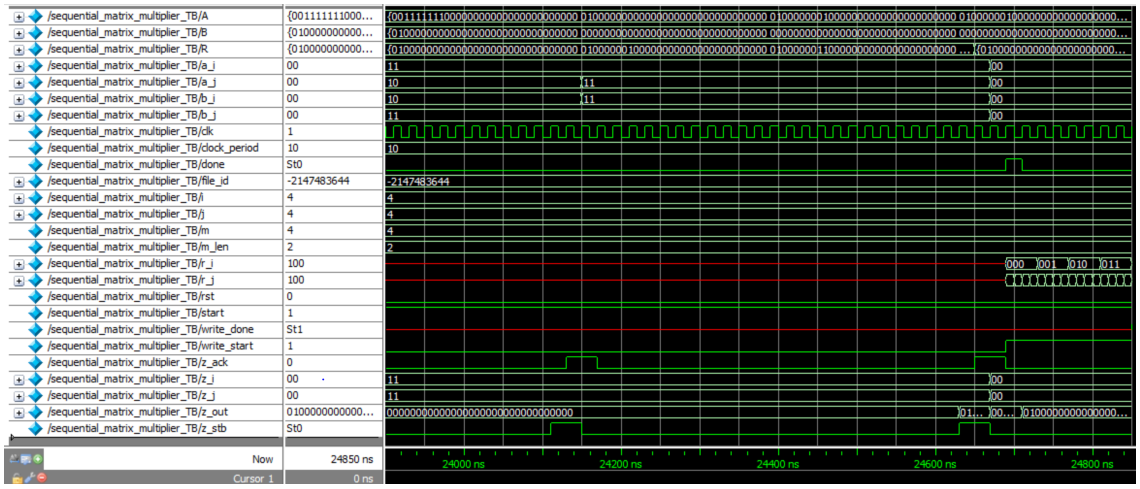
در این پروژه به طراحی، پیاده سازی، شبیه سازی و سنتز یک ضرب‌کننده ماتریسی پرداخته شد. ساختار این ضرب‌کننده از چند لایه مختلف تشکیل شده است که مهم‌ترین آن‌ها دو لایه هستند. لایه اصلی اول، ضرب‌کننده ماتریس ترتیبی است که مانند الگوریتم رایج ضرب ماتریس‌ها، حاصلضرب را به صورت ترتیبی محاسبه می‌کند. لایه اصلی دوم، ضرب‌کننده ماتریس موازی است که با شکاندن ماتریس به زیرـماتریس های کوچکتر، فرایند ضرب ماتریسی را تا حدی به صورت موازی انجام می‌دهد.

ویژگی خوب این طراحی این است که می‌توان با تناسب مقادیر m و n، تعادلی بین زمان اجرای ضرب ماتریسی و میزان سخت‌افزار مصرف شده به وجود آورد. به این صورت که به ازای n ثابت، هر چه m بزرگتر باشد، عملیات ضرب به سمت کاملا ترتیبی شدن پیش می‌رود، و هرچه m کوچکتر باشد میزان موازی‌سازی بیشتر می‌شود. در نتیجه، هرچه میزان موازی‌سازی بیشتر شود، مساحت مورد استفاده افزایش می‌یابد، اما سرعت اجرا نیز بیشتر می‌شود.

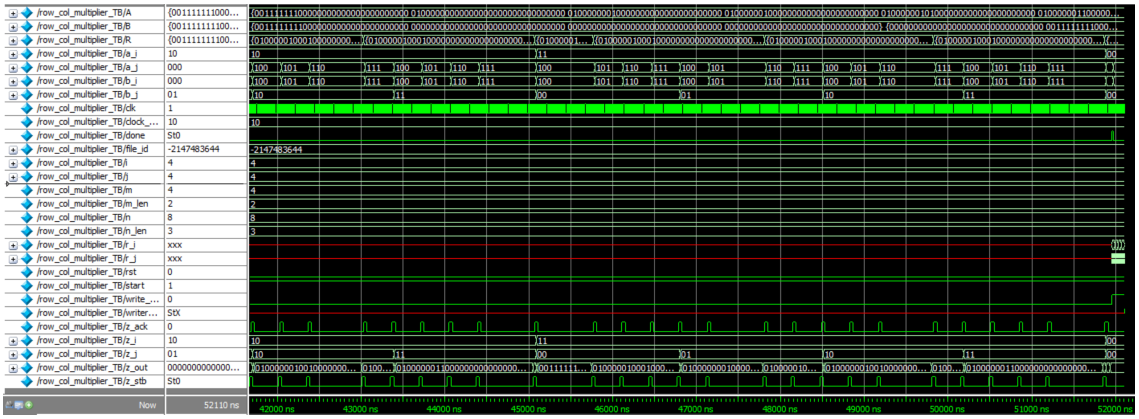
همچنین در نهایت عملیات سنتز انجام شد و مقادیر میزان استفاده از FPGA نیز محاسبه و گزارش شدند. بیشترین میزان فرکانس مدار نیز برابر ۶۲/۵۲۸MH است.

[6] waveforms

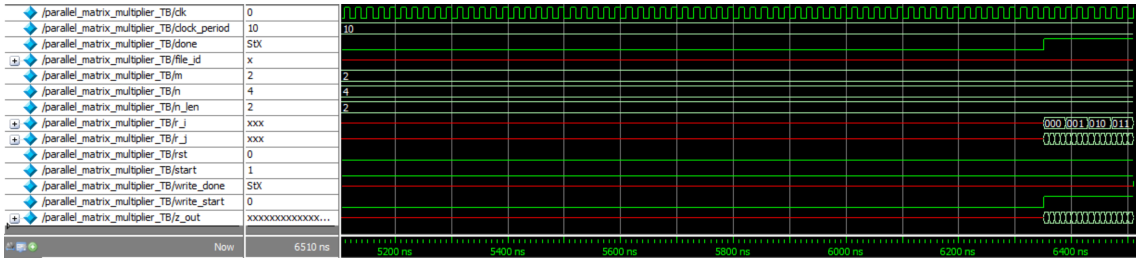
۶. ضمیمه‌ها



شکل ۷: شکل موج شبیه‌سازی ماژول ضرب‌کننده ماتریس ترتیبی



شکل ۸: شکل موج شبیه‌سازی ماژول ضرب‌کننده ماتریس سطری در ستونی



شکل ۹: شکل موج شبیه‌سازی ماژول ضرب‌کننده ماتریس موازی