任务报告书

报告类型：设计&测试报告

任务名称：12-bit 位宽流水线快速模乘器模块实现

姓　　名：XXXXX

日　　期：202X 年 X 月

目录

一、任务内容及分析

1.1 任务内容

1. 任务说明

根据任务要求实现一个 12-bit 位宽的流水线快速模乘器模块。模乘器输入 a、b 两个 12-bit 位宽的乘数，给定模数 q=3329，在固定周期内输出 a*b mod q。模块 IO 接口如表 1 所示，要求当模乘器输入使能有效时，a、b 两个乘数输入，随后模乘器进行计算，在模乘器计算过程中，busy 信号指示高电平为忙状态，否则为低电平指示空闲状态。在经过固定周期后，done 信号由低变高，指示计算完成，同时输出结果。

表 1. 模乘器模块 IO 接口

Pin Name	Width	Type	Description
clk	1	Input	系统时钟
rst_n	1	Input	系统复位，低电平有效
en	1	Input	输入使能，高电平有效
a	12	Input	输入乘数 a
b	12	Input	输入乘数 b
busy	1	Output	模乘器忙状态指示，高电平指示忙
done	1	Output	计算完成信号，高电平有效
r	12	Output	输出结果 r

2. 任务要求

- 使用 verilog 代码完成上述模乘器结构的实现；
- 编写 testbench，完成功能仿真，实现固定周期模乘输出；
- 撰写设计与仿真报告。

1.2 任务分析

1. 核心功能

1）实现 12-bit 无符号整数的模乘运算：r = (a×b) mod 3329。

2）支持流水线结构，在固定周期（5 个时钟周期）内完成计算。

3）输入使能信号（en）有效后，输出忙状态（busy）指示计算中，完成后输出单周期完成信号（done）。

2. 性能要求

1）固定周期：从 en 有效到 done 有效需要严格经过 5 个时钟周期。

2）资源优化：避免使用除法器，可以采用巴雷特约减算法（Barrett Reduction）优化模运算。

3. 测试要求

1）波形验证：观察信号是否在固定周期内完成各级流水线对应的功能，重点查看 busy、en 和 done 的波形是否满足任务要求。

2）数据验证：最终输出 r 要验证是否与理论计算值匹配。

3）时序资源分析：利用软件分析至 Implementation 阶段，查看资源利用情况，并进行时序分析。

二、模块设计

2.1 Barrett reduction 算法

1. 公式

$$r = x - \left\lfloor \frac{x \cdot \mu}{2^{24}} \right\rfloor \cdot q \quad \left(\mu = \left\lfloor \frac{2^{24}}{q} \right\rfloor \right)$$

在此模乘器中，其中 x=a×b，q=3329，μ=5039。

2. 算法优势

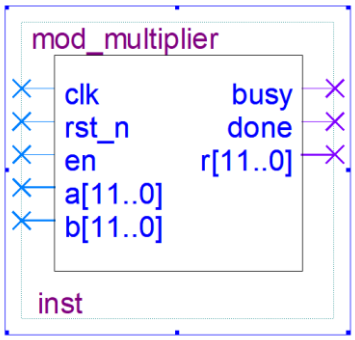
从计算形式来看，Montgomery 方法需要将输入的整数转成“蒙哥马利形式”，

最后还需要将结果从蒙哥马利形式转回正常表达；Barrett 方法则不需要这种复杂的形式转换，所以在此模块中设计采用 Barrett reduction 算法来实现模乘器。

在 Barrett 约减法中，可以通过预计算的 μ 将除法转换为乘法和移位，相较于除法，效率得到一定的提升，并且避免了硬件除法器的开销。

2.2 系统架构设计

1. 模块框图



2. 寄存器及流水线设计

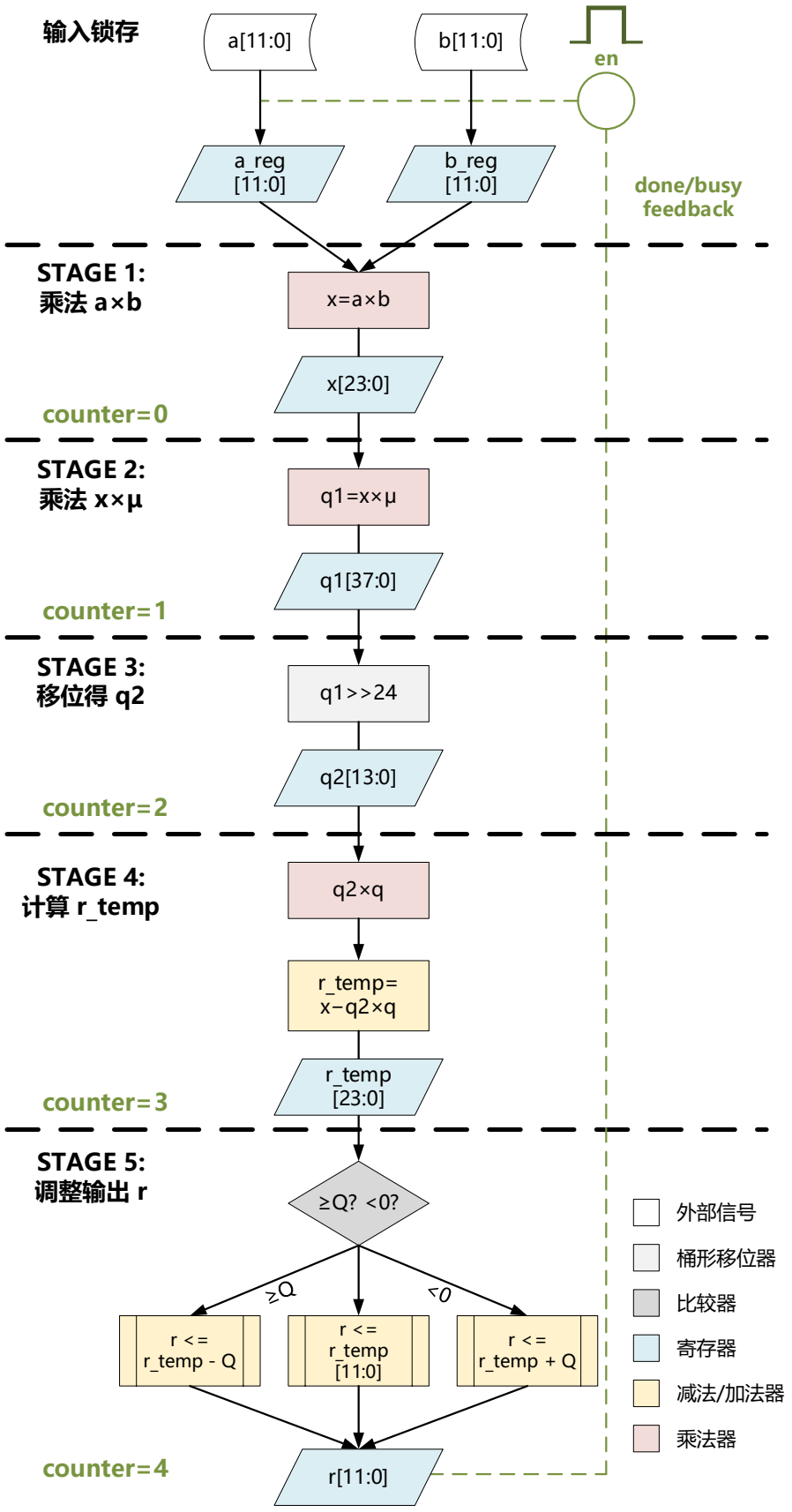
1）将模乘器运算划分为五级流水线在固定周期内实现，具体内容如下：

阶段	寄存器	寄存器位数	操作
1	x	24	计算 x=a×b
2	q1	38	计算 x×μ
3	q2	14	取高 14 位（等效右移 24 位）
4	r_temp	24	计算余数 r_temp=x−q2×q
5	r	12	调整余数至[0, q)范围

2）除上述寄存器外，模块内还定义了如下寄存器：

寄存器	寄存器位数	作用
a_reg, b_reg	12	输入锁存寄存器
counter	3	流水线阶段计数器

3. 具体数据通路流程图


输入锁存
a[11:0]
b[11:0]
en
a_reg [11:0]
b_reg [11:0]
done/busy feedback
STAGE 1: 乘法 a×b
x=a×b
x[23:0]
counter=0
STAGE 2: 乘法 x×μ
q1=x×μ
q1[37:0]
counter=1
STAGE 3: 移位得 q2
q1>>24
q2[13:0]
counter=2
STAGE 4: 计算 r_temp
q2×q
r_temp= x−q2×q
r_temp [23:0]
counter=3
STAGE 5: 调整输出 r
≥Q? <0?
≥Q
<0
r <= r_temp - Q
r <= r_temp [11:0]
r <= r_temp + Q
counter=4
r[11:0]
外部信号
桶形移位器
比较器
寄存器
减法/加法器
乘法器

三、验证方案

3.1 测试策略

1. 典型用例设计（5 组）

测试用例	输入（a,b）	预期输出（r）	验证目标
零输入测试	（0,0）	0	基础功能验证
最小正整数测试	（1,1）	1	基本运算正确性
边界值测试	（3328,3328）	1	负数特性（3328 ≡ -1 mod 3329）
大数运算测试	（3000,3000）	1713	中间值溢出处理
溢出测试	（4095,4095）	852	输入 12 位最大值验证运算不溢出

2. 随机测试（1000 组）

在 0-3328 范围内随机输入 a,b 进行测试，每完成 100 组测试就打印一次完成测试进度，最终打印随机测试的 1000 组通过率及耗时。

3. 全遍历测试（11082241 组）

在 0-3328 范围输入 a,b 进行 3329*3329 全遍历测试，每完成 1000 组测试就打印一次完成测试进度，最终打印全遍历测试的 11082241 组通过率及总耗时。

3.2 测试结果

1. 功能正确性验证

1) 五组典型用例：

```
/SIM 3> run
# Pass: a=   0, b=   0, res=   0
# Pass: a=   1, b=   1, res=   1
# Pass: a=3328, b=3328, res=   1
# Pass: a=3000, b=3000, res=1713
# Pass: a=4095, b=4095, res= 852
# ** Note: $finish    : D:/FPGA/mod_multiplier/simulation/modelsim/
work/tb_mod.v(53)
#    Time: 345 ns  Iteration: 0  Instance: /tb_mod_multiplier
# 1
```

随机测试：

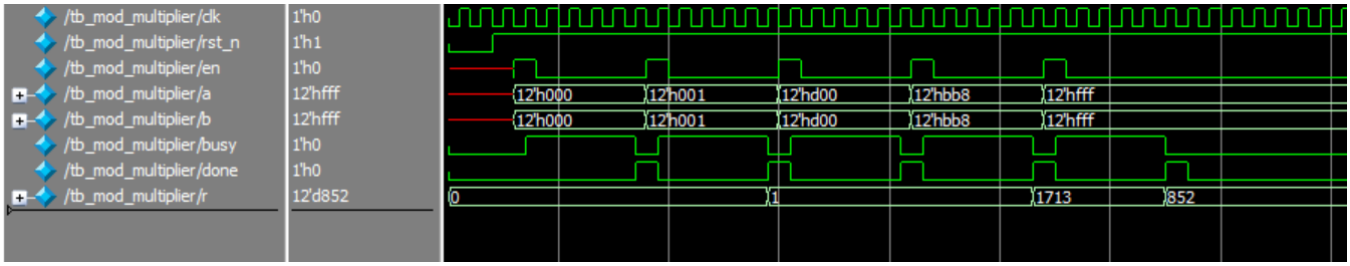
```
VSIM 6> run
# ===== 基础功能测试开始 =====
# ===== 基础功能测试完成 =====
# 测试用例数: 5, 错误数: 0
# ===== 随机测试开始 (1000次) =====
# 已完成 0 次随机测试...
# 已完成 100 次随机测试...
# 已完成 200 次随机测试...
# 已完成 300 次随机测试...
# 已完成 400 次随机测试...
# 已完成 500 次随机测试...
# 已完成 600 次随机测试...
# 已完成 700 次随机测试...
# 已完成 800 次随机测试...
# 已完成 900 次随机测试...
# ===== 随机测试完成 =====
# 测试用例数: 1000, 错误数: 0
# ===== 部分遍历测试开始 (0-100) =====
# ===== 部分遍历测试完成 =====
# 测试用例数: 10201, 错误数: 0
# 耗时: 0.61 ms
#
# ===== 测试总结 =====
# 总测试用例数: 11206
# 总错误数: 0
# 测试通过率: 100.00%
# *** 所有测试通过 ***
```

全遍历测试：

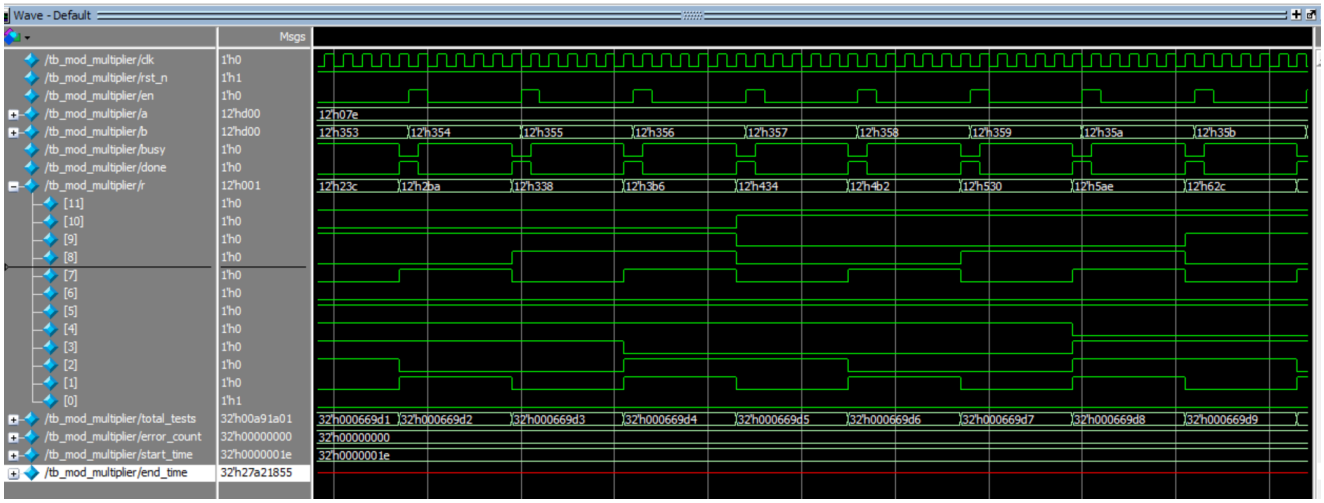
```
# 进度: 10860000/11082241 (98.0%)
# 进度: 10870000/11082241 (98.1%)
# 进度: 10880000/11082241 (98.2%)
# 进度: 10890000/11082241 (98.3%)
# 进度: 10900000/11082241 (98.4%)
# 进度: 10910000/11082241 (98.4%)
# 进度: 10920000/11082241 (98.5%)
# 进度: 10930000/11082241 (98.6%)
# 进度: 10940000/11082241 (98.7%)
# 进度: 10950000/11082241 (98.8%)
# 进度: 10960000/11082241 (98.9%)
# 进度: 10970000/11082241 (99.0%)
# 进度: 10980000/11082241 (99.1%)
# 进度: 10990000/11082241 (99.2%)
# 进度: 11000000/11082241 (99.3%)
# 进度: 11010000/11082241 (99.3%)
# 进度: 11020000/11082241 (99.4%)
# 进度: 11030000/11082241 (99.5%)
# 进度: 11040000/11082241 (99.6%)
# 进度: 11050000/11082241 (99.7%)
# 进度: 11060000/11082241 (99.8%)
# 进度: 11070000/11082241 (99.9%)
# 进度: 11080000/11082241 (100.0%)
# ===== 全遍历测试完成 =====
# 测试用例数: 11082241, 错误数: 0
# 总耗时: 664.93 ms
#
# ===== 测试总结 =====
# 总测试用例数: 11082241
# 总错误数: 0
# 测试通过率: 100.00%
# *** 所有测试通过 ***
```

三种测试类型下经验证均测试通过，输出结果如上，说明利用巴雷特约减法算法设计的 12 位宽模乘器可以较好在要求范围内准确实现取模运算。

2. 时序波形分析



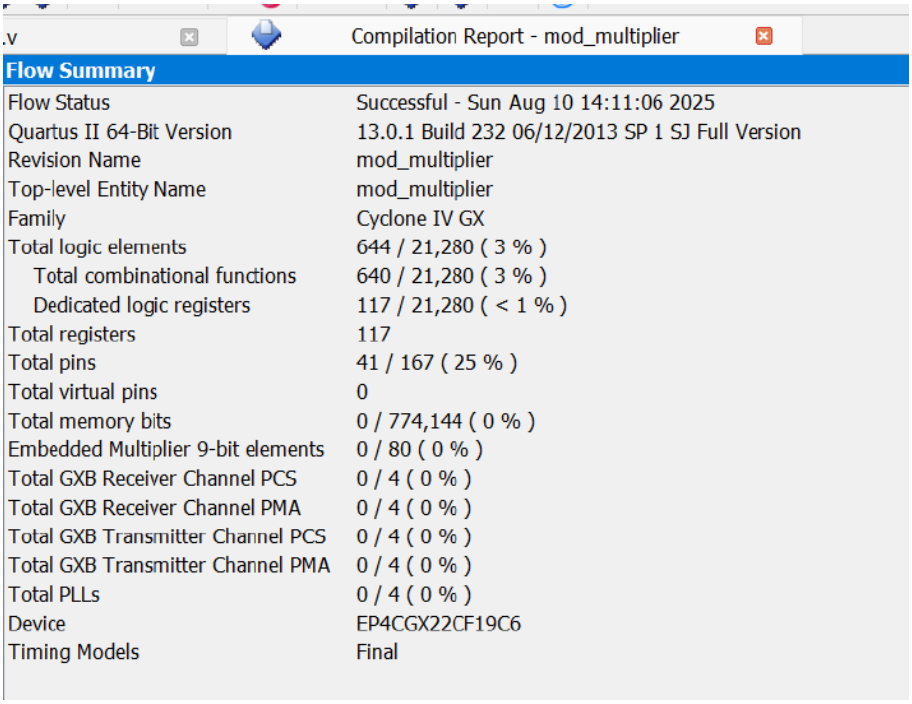
在最后一个测试用例测试下，模乘器的各信号波形结果如上图所示。可以看到 busy 在 en 有效后立即拉高，持续 5 周期，done 在第 5 周期结束时产生单周期脉冲，r 在流水线最后阶段输出正确结果，固定周期流水线功能满足任务要求。



查看全遍历的测试下，模乘器的各信号波形结果如上图所示。可以看到输入信号根据遍历设计跳变，根据 en/busy/done 的信号输入模乘器，模乘器较好地实现了五级流水线模乘的功能。

四、Implementation 分析

4.1 综合实现



.v　　Compilation Report - mod_multiplier

Flow Summary

Flow Status	Successful - Sun Aug 10 14:11:06 2025
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Full Version
Revision Name	mod_multiplier
Top-level Entity Name	mod_multiplier
Family	Cyclone IV GX
Total logic elements	644 / 21,280 (3 %)
Total combinational functions	640 / 21,280 (3 %)
Dedicated logic registers	117 / 21,280 (< 1 %)
Total registers	117
Total pins	41 / 167 (25 %)
Total virtual pins	0
Total memory bits	0 / 774,144 (0 %)
Embedded Multiplier 9-bit elements	0 / 80 (0 %)
Total GXB Receiver Channel PCS	0 / 4 (0 %)
Total GXB Receiver Channel PMA	0 / 4 (0 %)
Total GXB Transmitter Channel PCS	0 / 4 (0 %)
Total GXB Transmitter Channel PMA	0 / 4 (0 %)
Total PLLs	0 / 4 (0 %)
Device	EP4CGX22CF19C6
Timing Models	Final

1) 总体状态：Flow Status:Successful，编译成功无报错

2) 目标芯片：EP4CGX22CF19C6（Cyclone IV GX 系列）

3) 资源使用情况：

项目	使用量/总量	占比	说明
Total logic elements	644/21280	3%	基本逻辑单元，已用很少
Combinational functions	640/21280	3%	组合逻辑资源
Dedicated logic registers	117/21280	<1%	寄存器资源，非常少
Total pins	41/167	25%	芯片引脚已用 41 个
Memory bits	0/774144	0%	没用片内存储器
Embedded Multipliers (9-bit)	0/80	0%	没用硬件乘法器
PLLs	0/4	0%	没用锁相环
GXB 收发器	0/4	0%	没用高速串行收发器

4.2 时序分析

1) **Fmax:**

Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	132.52 MHz	132.52 MHz	clk	

2) **无 sdc 约束下，建立时间与保持时间：**

Slow 1200mV 85C Model Setup Summary

	Clock	Slack	End Point TNS
1	clk	-6.546	-380.097

Slow 1200mV 85C Model Hold Summary

	Clock	Slack	End Point TNS
1	clk	0.355	0.000

可以看出建立时间存在违例，加上 SDC 时钟约束后在 100MHz 下进行时序分析，可以看出建立时间不存在违例。

Summary (Setup)

	Clock	Slack	End Point TNS
1	clk	2.014	0.000

3) **多工艺角下：**

Setup Times

	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Refere
1	a[*]	clk	2.331	2.843	Rise	clk
2	b[*]	clk	2.496	2.966	Rise	clk
3	en	clk	2.182	2.278	Rise	clk

Hold Times

	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
1	a[*]	clk	-0.785	-1.384	Rise	clk
2	b[*]	clk	0.277	0.147	Rise	clk
3	en	clk	0.029	-0.029	Rise	clk

Clock to Output Times

	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
1	busy	clk	6.615	6.581	Rise	clk
2	done	clk	6.608	6.577	Rise	clk
3	⌄ r[*]	clk	6.833	6.795	Rise	clk
1	r[0]	clk	6.528	6.478	Rise	clk
2	r[1]	clk	6.273	6.218	Rise	clk
3	r[2]	clk	6.767	6.732	Rise	clk
4	r[3]	clk	6.483	6.439	Rise	clk
5	r[4]	clk	6.513	6.462	Rise	clk
6	r[5]	clk	6.833	6.795	Rise	clk
7	r[6]	clk	6.533	6.473	Rise	clk
8	r[7]	clk	6.511	6.461	Rise	clk
9	r[8]	clk	6.729	6.695	Rise	clk
10	r[9]	clk	6.484	6.432	Rise	clk
11	r[10]	clk	6.546	6.500	Rise	clk
12	r[11]	clk	6.783	6.735	Rise	clk

Minimum Clock to Output Times

	Data Port	Clock Port	Rise	Fall	Clock Edge	Clock Reference
1	busy	clk	3.811	3.837	Rise	clk
2	done	clk	3.806	3.841	Rise	clk
3	⌄ r[*]	clk	3.613	3.615	Rise	clk
1	r[0]	clk	3.755	3.773	Rise	clk
2	r[1]	clk	3.613	3.615	Rise	clk
3	r[2]	clk	3.884	3.927	Rise	clk
4	r[3]	clk	3.715	3.733	Rise	clk
5	r[4]	clk	3.741	3.760	Rise	clk
6	r[5]	clk	3.915	3.963	Rise	clk
7	r[6]	clk	3.754	3.768	Rise	clk
8	r[7]	clk	3.739	3.758	Rise	clk
9	r[8]	clk	3.788	3.878	Rise	clk
10	r[9]	clk	3.713	3.731	Rise	clk
11	r[10]	clk	3.762	3.777	Rise	clk
12	r[11]	clk	3.895	3.934	Rise	clk

可以看出，在输入信号路径上，存在保持时间违例问题，最坏 Slack 有 -1.384 ns。

4.3 优化措施

通过时序分析可以看出，建立时间的违例通过调整时钟约束，适当降低时钟频率即可解决。保持时间的违例可以通过人为增加数据路径的延迟，插入约 1.4ns 的两级寄存器延迟，或调整布局布线解决，同时整体时序收敛可行。

附录

参考文献

[1] CSDN：barret reduction 原理详解及硬件优化 .[在线]. 网址:https://blog.csdn.net/qq_57502075/article/details/130052118
[2] 同态加密：巴雷特模乘(Barrett Modular Multiplication) .[在线]. 网址: 巴雷特模乘(Barrett Modular Multiplication) - 知乎
[3] 大数取模运算 Barrett reduction .[在线]. 网址：https://blog.csdn.net/YKRY35/article/details/79179285
[4] 蒙哥马利约减和 barret 约减算法 .[在线]. 网址：https://blog.csdn.net/weixin_39057744/article/details/121296209.
[5] 夏宇闻 著. Verilog 数字系统设计教程. 电子工业出版社.