REDC3329_pipeline设计文档

李远政 U202211014

2025年9月

目录

1 模块要求

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。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。

2 模块输入输出说明

表 1: IO接口表

端口	方向	位宽	描述
clk	Input	1	系统时钟信号。
rst_n	Input	1	低电平有效的异步复位信号。
en	Input	1	使能信号。当 en 为高时，模块在下一个时钟周期锁存输入 a 和 b。
a	Input	12	输入操作数 A。
b	Input	12	输入操作数 B。
busy	Output	1	忙信号。当模块正在处理数据时（从 en 拉高到 done 拉高之间），该信号为高。
done	Output	1	完成信号。当一次模乘计算完成，且输出 r 有效时，该信号拉高一个时钟周期。
r	Output	12	计算结果，即 (a * b) mod 3329。

3 算法理论与算法验证

3.1 Montgomery模乘算法

可以观察到题目中所给出的3329是一个质数，所以可以使用蒙哥马利算法简化取模的计算过程。算法的基本原理是通过一系列的转换将本来对一个普通数N的取模转换为对一个2的幂次R的取模过程，从而可以通过移位算法节省大量计算资源。为了达到对一个数T转换模数的目的，先构造REDC(a,b,R)函数解决$(ab)\cdot R^{-1}(modN)$的问题,这个函数是存在简便算法的。然后反复调用这个函数来实现模乘的过程。具体步骤如下：

3.1.1 REDC函数的设计

构造函数的基本思路是将大数T加上一个适当的数值m，使其能够被R整除，并且结果能够落在0到2N的范围内，并且仍然保证对N同余的特性。这样就可以直接使用简单的数据选择器实现对于N的取模。具体步骤如下：首先构造加数:

$$m = N[T(modR)N'](modR)$$

其中：

$$N \cdot N' = -1(modR)$$

相加之后得到t：

$$t = T+m = T+N[T(modR)N'](modR) = T+[(NN')(modR)][(T(modR))](modR) = T-T(modR)$$

将t再modR的情况下进行化简：

$$t(modR) = T + [(NN')(modR)][(T(modR))](modR) = T - T(modR)$$

从上面的变换中，不难看出这个t是可以被R整除的。接下来将t除以R得到：

$$u = \frac{T+m}{R}$$

现在来估算u的范围：

$$T < N \cdot R$$

$$N[T(modR)N'](modR) < N \cdot R$$

$$0 \leq t < 2N$$

因此可以直接通过比较t和N的大小来实现对N的取模。

3.1.2 REDC函数的乘法性质

因为直接调用REDC函数会带有R的逆元，无法直接得到结果，所以通过多次调用转换来实现模乘的过程。具体步骤如下：
1.转换输入a,b为蒙哥马利域下的数：

$$a_{mont} = REDC(a, R^2)$$

$$b_{mont} = REDC(b, R^2)$$

2.计算a,b的乘积：

$$t = REDC(a_{mont}, b_{mont})$$

3.将结果转换回普通域下的数：

$$r = REDC(t, 1)$$

这样就刚好通过两部约去了最开始引入的R的次方影响，得到了正确的结果。

4 电路设计

4.1 REDC3329整体架构

整体结构在计算部分采用流水线的设计按照设计原理的要求分为三个阶段，分别是输入转换阶段，乘法计算阶段，输出转换阶段。每个阶段都使用寄存器锁存数据，并且通过控制逻辑实现各个阶段的流水线操作。具体结构如下图所示：

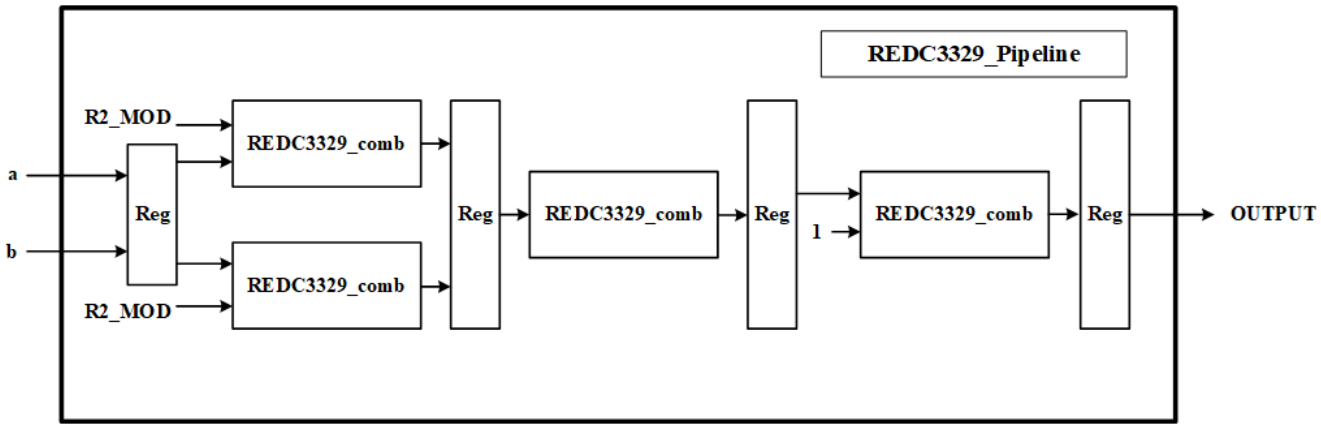


图 1: 流水线模乘器结构图

4.2 REDC模块设计

REDC模块的设计是整个模乘器的核心部分，主要负责实现蒙哥马利模乘算法中的REDC函数。该模块接受一个24-bit的输入T，并输出一个12-bit的结果r，同时还需要处理一些中间变量和状态信号。具体设计如下：

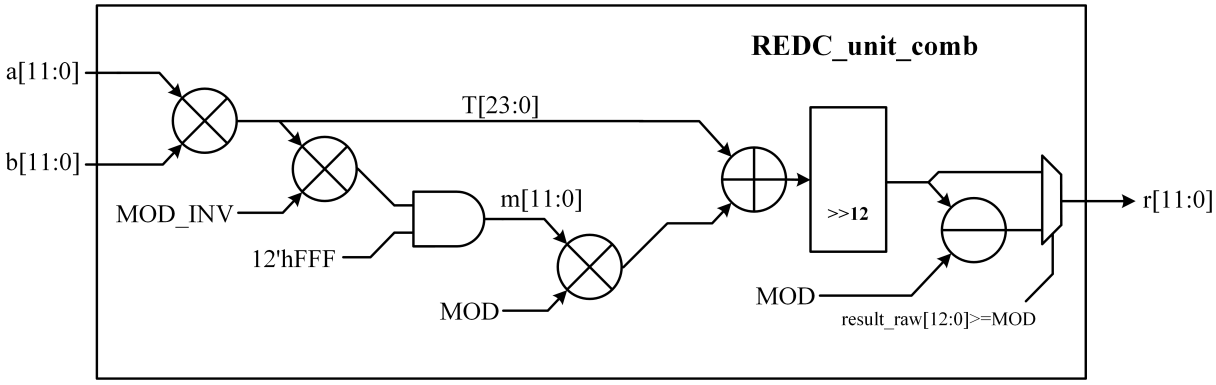


图 2: REDC3329_unit_comb结构图

5 功能验证

5.1 仿真验证

因为12bit输入范围较小，可以通过穷举所有输入的方式来验证设计的正确性。编写了一个简单的testbench来实现该功能。将theroy信号进行四个周期的打拍，然后和r进行对比，如果不相等则说明设计有误。为了方便观测结果，在仿真平台中添加wrong信号，将两个信号进行异或，如果出现错误的计算结果可以直接观测到wrong信号置1。仿真结果显示在所有的12-bit位宽输入下，wrong信号在复位完成之后的计算过程中始终为低电平，说明设计的结果和理论结果完全一致，验证了设计的正确性。

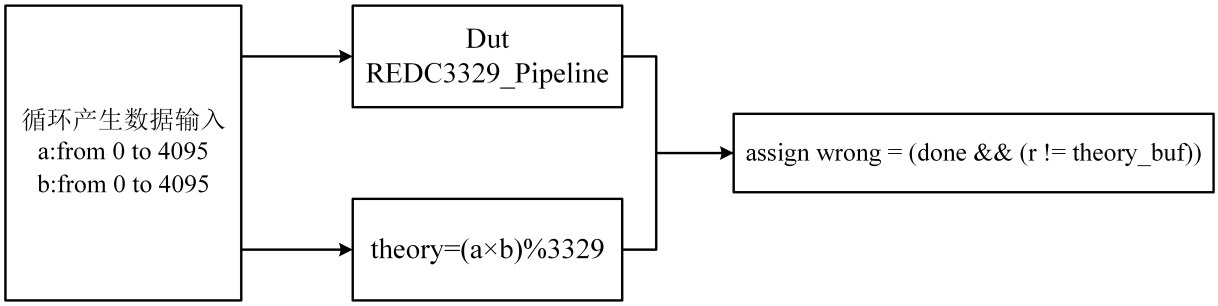


图 3: 仿真流程设计图

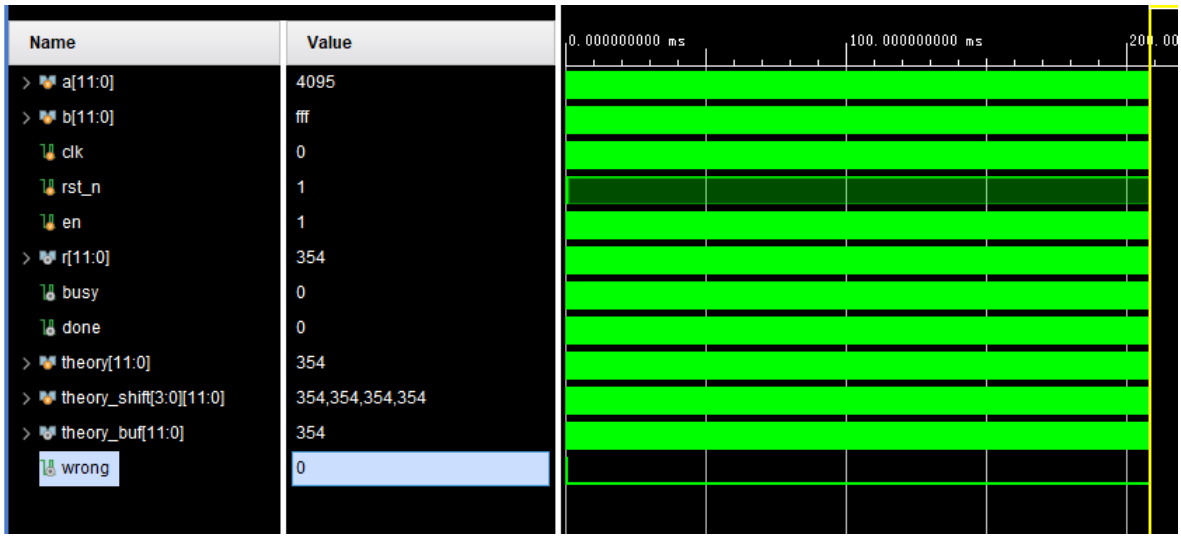


图 4: 仿真波形图

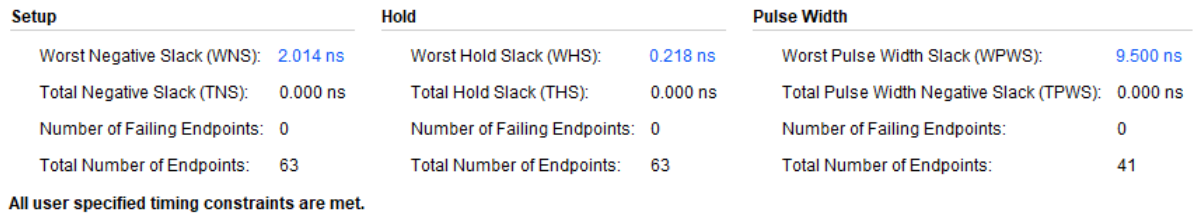
5.2 综合实现

设置实现约束文件指定时钟周期为20ns，暂不考虑外设时序问题。

使用ZYNQ-7020作为芯片型号经过综合和实现之后，最终的时序报告显示在该时钟周期下没有违反时序的情况。资源利用率方面，整体设计使用了较少的逻辑单元和寄存器资源，具体资源利用率如下表所示：

Resource	Utilization	Available	Utilization %
LUT	99	53200	0.19
FF	40	106400	0.04
DSP	11	220	5.00
IO	41	125	32.80
BUFG	1	32	3.13

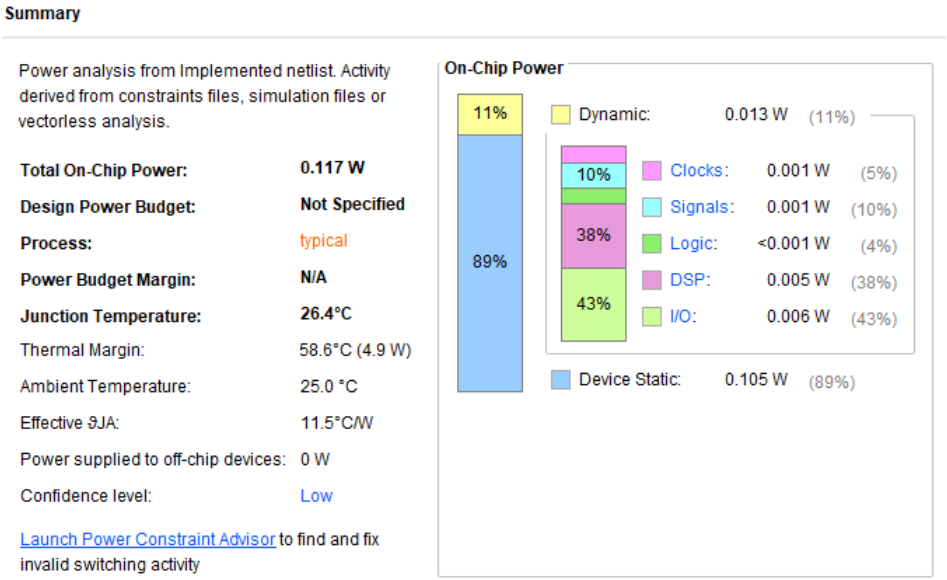
图 5: 资源占用统计



Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	2.014 ns	Worst Hold Slack (WHS):	0.218 ns	Worst Pulse Width Slack (WPWS):	9.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	63	Total Number of Endpoints:	63	Total Number of Endpoints:	41

All user specified timing constraints are met.

图 6: 时序报告



Summary

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.117 W**
Design Power Budget:	**Not Specified**
Process:	typical
Power Budget Margin:	**N/A**
Junction Temperature:	**26.4°C**
Thermal Margin:	58.6°C (4.9 W)
Ambient Temperature:	25.0 °C
Effective ϑJA:	11.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity

图 7: 功耗报告