“华为杯”

中国研究生创“芯”大赛

设计方案

作品名称： 基于AMBA总线的CRC运算核IP设计

团队名称： 视野

参赛队员： 李小飞 袁余涵 卢云龙

指导教师： 周云

**摘要**

基于AMBA总线的CRC运算核IP可通过配置数据进行片外数据的读取，然后进行CRC码的运算或校验。IP数据的读写采用根据需求定制的高性能的200MHz的32位AXI总线，在CRC运算模块中，我们对其逻辑进行了优化，使其能在500MHz的高频下进行CRC码的计算，并且与AXI总线的读取速度进行了匹配。对于两个时钟域的跨时钟信号传输，采用了握手信号传输控制信号，FIFO传输数据信号。IP同时设置有一中断模块，在IP完成运算/校验或者AXI总线发生读写错误时发出中断信号使软件能够查询IP的工作状态。最后我们搭建了Testbench仿真平台和UVM验证平台对IP进行了仿真及验证，确保设计功能的正确性。

目录

[**摘要** 1](#_Toc11778746)

[**1 整体设计** 3](#_Toc11778747)

[**2 工作流程** 4](#_Toc11778748)

[**3 模块设计** 4](#_Toc11778749)

[3.1 CRC模块 4](#_Toc11778750)

[3.2 AXI模块 9](#_Toc11778751)

[3.3 跨时钟域同步模块 10](#_Toc11778752)

[3.4 中断模块 11](#_Toc11778753)

[**4 仿真结果** 12](#_Toc11778754)

**1 整体设计**

通过对赛题三：基于AMBA总线的CRC运算核IP设计的描述的分析，我们将IP分为：CRC运算模块、AXI总线模块、跨时钟域同步模块和中断模块共4大模块，总体框架如图1所示。



图1 IP总体框架

其中，AXI3\_SLV为AXI协议的slave接口，负责接收配置数据（源端地址、目的地址、数据长度、crc模式和工作模式等），并将相应的数据发送给CRC模块和AXI3\_MST模块。AXI3\_MST为AXI协议的master接口，负责读取待处理的原始数据，如果IP设定为运算模式，则将原始数据与CRC码一起写入目的地址。CRC模块负责对原始数据进行计算。对于跨时钟域的处理，原始数据从AXI3\_MST到CRC模块，采用异步FIFO进行同步，对于如图中所示的虚线部分的跨时钟域信号的传输采用握手协议进行同步。中断模块的信号来源包括AXI3\_SLV模块的读写错误、AXI3\_MST模块的读写错误和CRC模块完成校验、运算结果已写入到目的地址的信号，中断模块在接收到这些信号后会发出1bit的中断请求信号idle，然后软件可以通过AXI3\_SLV接口查询中断来源或校验结果。

IP核的工作流程为：外部模块将配置数据写入AXI3\_SLV模块，AXI3\_SLV模块在完成接受配置数据后，发出使能信号，然后AXI3\_MST模块接收配置数据，并开始从源地址读取原始数据存入FIFO。如果CRC工作模式为运算模式，则CRC模块异步读取原始数据进行处理，同时AXI3\_MST模块又同步的读取原始数据并写到目的地址，待CRC模块完成运算后，将CRC码写到目的地址的原始数据之后。如果CRC工作模式为校验，则CRC模块异步读取原始数据进行处理，并与CRC码相比较，将校验结果写入AXI3\_SLV模块的寄存器之中，并发出中断信号，供软件查询结果。

**2 工作流程**

IP具有两种工作模式：运算模式和校验模式。

当设定为运算模式时，工作流程为：外部模块将配置数据写入AXI3\_SLV模块的寄存器之中，在完成写入之后，AXI3\_SLV发出使能信号，使CRC模块和AXI3\_MST模块读入相应的配置数据。然后AXI3\_MST模块开始根据配置数据读取待处理的原始数据，并将其存入FIFO之中。当FIFO不再为空时，CRC模块开始读入原始数据并进行运算，与此同时，AXI3\_MST模块写数据部分同步地从FIFO中读取原始数据并写入到目的地址。当CRC模块完成运算时，发出使能信号到AXI3\_MST模块，由其将运算结果写入到目的地址的原始数据之后。然后AXI3\_MST模块发出完成信号给中断模块，中断模块接着产生中断信号，使软件可以查询运算结果。

当设定为校验模式时，工作流程为：外部模块将配置数据写入AXI3\_SLV模块的寄存器之中，在完成写入之后，AXI3\_SLV发出使能信号，使CRC模块和AXI3\_MST模块读入相应的配置数据。然后AXI3\_MST模块开始根据配置数据读取待处理的原始数据，并将其存入FIFO之中。当FIFO不再为空时，CRC模块开始读入原始数据并进行运算，完成运算之后与附在原始数据之后的CRC码相比较，当完成比较后，通过两级触发器将结果同步到AXI3\_MST模块，同时发出使能信号到中断模块，中断模块接着产生中断信号，使软件可以查询校验结果。

**3 模块设计**

3.1 CRC模块

3.1.1 CRC运算原理

循环冗余校验（英语：Cyclic redundancy check，通称“CRC”）是一种根据网络数据包或计算机文件等数据产生简短固定位数校验码的一种散列函数，主要用来检测或校验数据传输或者保存后可能出现的错误。生成的数字在传输或者存储之前计算出来并且附加到数据后面，然后接收方进行检验确定数据是否发生变化。由于本函数易于用二进制的计算机硬件使用、容易进行数学分析并且尤其善于检测传输通道干扰引起的错误，因此获得广泛应用。

循环冗余校验同其他差错检测方式一样，通过在要传输的k比特数据D后添加(n-k)比特冗余位(又称帧检验序列，Frame Check Sequence，FCS)F形成n比特的传输帧T，再将其发送出去。

特别的，循环冗余校验提供一个预先设定的(n-k+1)比特整数P，并且要求添加的(n-k)比特F满足：

T mod P == 0 ……(1)

其中 T =2n-kD + F

基于上述要求，实际应用时，发送方和接收方按以下方式通信：

1. 发送方和接收方在通信前，约定好预设整数P。

2. 发送方在发送前根据数据D确定满足(1)式的F，生成CRC码，T 即为数据位D与校验位F的拼接，发送T。

3. 接收方收到CRC码 T，进行 result = T mod P 运算，当且仅当result = 0时接收方认为没有差错。

校验位F使用模二运算求解，模二运算采用无进位的二进制加法，恰好为异或(XOR)操作。(以下运算均为模2运算)

CRC码 T 需要满足(1)式，即(2n-kD + F)/P结果为某一整数，对此表达式进行恒等变换，可得：

(2n-kD + F)/P = 2n-kD / P + F / P …… (2)

继续对等式中2n-kD / P 进行恒等变换，将其整数部分 Q 分离，即 Q=(2n-kD - R)/P，有

2n-kD / P = Q + R / P …… (3)

将(3)式带入(2)式 得到：

(2n-kD + F) / P = Q + R / P+ F / P …… (4)

由于采用无进位的二进制加法(等价于XOR操作)，因此当我们令 F = R 时，有

(2n-kD + F) / P = Q + R / P+ F / P = Q …… (5)

当Q为整数时，T =(2n-kD + F)满足T mod P == 0。

故F = R = 2n-kD mod P。

实际上用二进制模2运算进行2n-kD (相当于D左移n-k位)的运算。意思就是在D后面补n-k个0。现在D就变成了n位。然后用2n-kD除以P，但是在除法运算中的加减法用的是模二运算，即异或。最终得到的余数即是校验码。

一般CRC计算有几种方法，例如串行、并行，还有查表法。

首先介绍串行算法，以CRC-16为例。

首先,判断D的首位。若为1，与P进行模2运算；为0则不与P进行模2运算，即可跳过此次运算直接跳到后面的非零位。另外，P的首位必定是1,每次模2运算后，结果的首位必定被移出，因此只需考虑P的后16位。构造1个16位的CRC寄存器C，取D前16位存入C，每次运算将寄存器首位抛弃后左移1位。D的后1位移入寄存器中即串行输入1位数据，寄存器C就实时更新为移位后的状态。

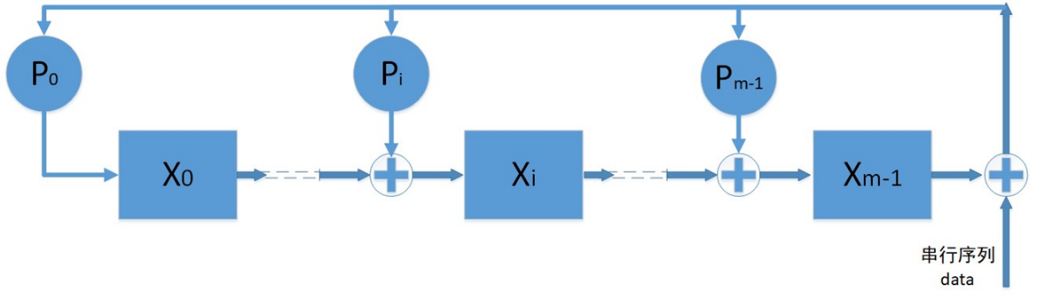


图2 CRC 串行算法示意图

上图为串行算法对应的硬件电路，CRC寄存器X在最高位的反馈下与P进行异或。

但是串行算法每次只能进入一个数据，速度非常慢，所以一般采用并行算法。

并行算法实际上就是先以串行为输入，采用递推算法，直接得出最后的公式。

假设CRC的码长是m，如果用X = [xm-1，… ，x1，x0]T表示LFSR各寄存器的第j个状态值，用X′ = [xm-1′，… ，x1′，x0′ ]T表示第(j+ 1)个状态值；把生成多项式写成P={pm，pm-1，p0}；d表示串行进入LFSR的1比特码字；同时用⮾表示异或运算，用⊕表示位与运算，两者均是模二运算；则可以得到下列方程组

（1）

数据长度为多少，则将此式递推多少遍，就可以得到最后的公式了。

3.1.2 CRC模块设计

本题目中总线时钟为200MHz，crc工作时钟为500MHz。由于crc结果需要上一个crc结果和现在的data\_in计算，因此，为了充分利用两个时钟之间的大小比例关系，我们决定在crc模块里每2个或者每3个时钟计算出一个crc结果。

经过讨论，我们决定选择每2个时钟计算出一个crc结果，2个时钟完成计算的话，我们可以把32bit数据拆分成2个16bit数据，每个时钟计算一个16bit数据，正好2个时钟完成32bit数据的计算。如图3所示。并行计算16bit数据的复杂程度远远小于计算32bit数据，这大大减少了组合逻辑的数量，减轻了设计压力。

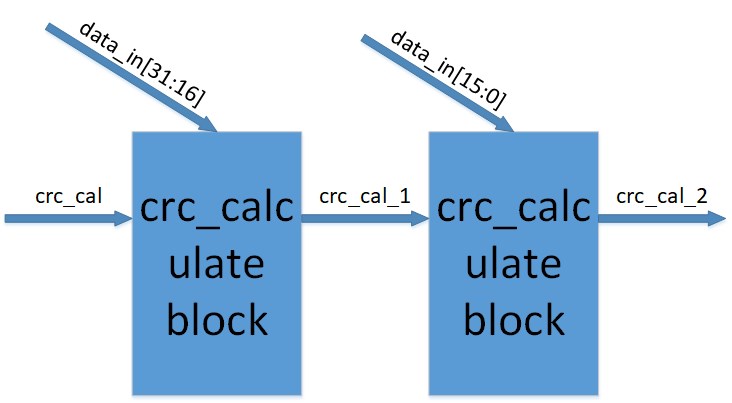


图3 CRC计算原理图

进一步的分析流程，我们发现，在计算一个32bit数据的crc结果时，两次crc计算所用的公式相同。因此，我们可以使用迭代法，即两次计算均使用同一模块。如图4所示，CRC\_clk0、CRC\_clk1、CRC\_clk2其实是同一变量，只是为了区分不同时钟时间，才命名不同。在两次计算过程中，均使用同一计算模块进行crc计算，这大大减小了逻辑资源使用量。

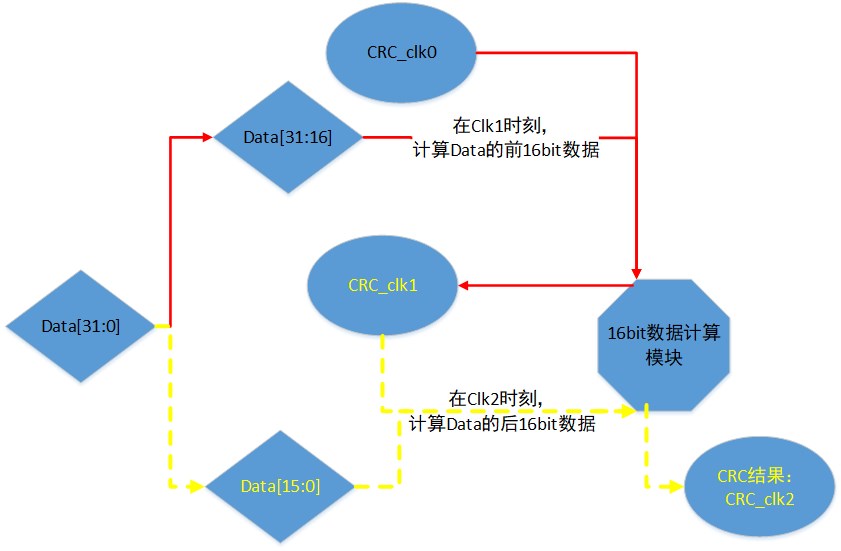


图4 CRC计算流程图

由于给的数据长度范围是1~65535byte，而总线数据位宽为32，所以，在32bit数据的最后，可能会有填不满32bit的情况，所以要进行数据的整理拼接。如图5所示，数据为12、34、56、78、23。实际32bit数据显示为12345678，23000000。我们对它进行整理拼接，由于缺少24位，因此我们补上24’h000000。进行拼接后，数据变为00000012，34567823。整理拼接后的数据更利于我们进行计算，而且不会影响最后输出的结果。

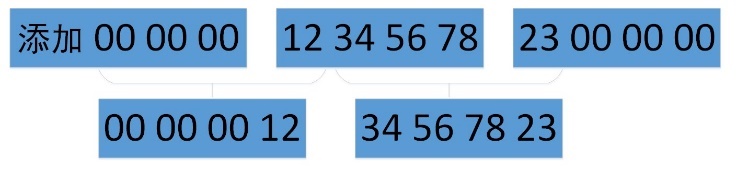


图5 CRC数据拼接示意图

我们尽量减少组合逻辑的层次以提高最大工作频率。由于本题目要求crc模块的时钟要达到500Mhz，因此在crc模块设计中，我们尽可能的使用时序逻辑，并尽量减少组合逻辑的层次，以提高最大工作频率。例如判断数据是否有效时，我们没有简单的用组合逻辑“与或非”，而是使用时序逻辑，这样输出给下一级时，能够尽可能的减小一级组合逻辑层数。

为了实现低功耗，我们采用了门控时钟的设计。

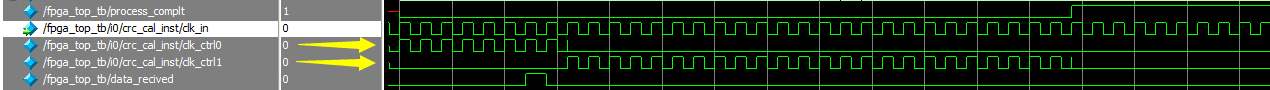
在crc计算模块中，主要分为3个运行状态。

①读取配置，做好计算准备。

②计算以及输出结果。

③计算完成后待机。

鉴于最后的待机状态是没有必要使用时钟的，因此，我们使用了两个门控时钟clk\_ctrl0、clk\_ctrl1。在①状态，只有clk\_ctrl0工作，控制①状态的逻辑运行。在②状态，只有clk\_ctrl1工作，控制②状态的逻辑运行。而在③状态，则没有时钟工作。这样能够大幅降低不必要的功耗。



如上图所示，可以看到在仿真结果中，两个门控时钟确实正确产生并正常工作了。

3.2 AXI模块

AXI总线协议是AMBA3.0中新提出的一种高性能的片上总线协议，在此次设计中我们参考AMBA AXI Protocol v1.0标准，并结合IP的实际需求，设计了符合数据传输要求并且逻辑面积尽可能小的AXI协议。

IP中所设计的AXI协议具备5个相对独立的通道：写地址通道、写数据通道、写响应通道、读地址通道和读数据通道。支持incr1/4/8/16 burst传输，并且模块会根据欲发送或接受的数据长度自动依次选择16/8/4/1 burst传输。支持非对齐传输，AXI3\_MST模块支持连续发送5个outstanding地址。由于IP中所有数据的传输为顺序传输，为减少逻辑面积，去除了乱序传输、间插等不需要的功能。

AXI3\_SLV模块中设置有12个位宽为8的寄存器组，存放数据如表一所示。

表1 AXI3\_SLV模块寄存器组配置

|  |  |
| --- | --- |
| 寄存器组 | 存放数据 |
| Reg[0][7:0] | 源地址 |
| Reg[1][7:0] | 源地址 |
| Reg[2][7:0] | 源地址 |
| Reg[3][7:0] | 源地址 |
| Reg[4][7:0] | 目的地址 |
| Reg[5][7:0] | 目的地址 |
| Reg[6][7:0] | 目的地址 |
| Reg[7][7:0] | 目的地址 |
| Reg[8][7:0] | 数据长度 |
| Reg[9][7:0] | 数据长度 |
| Reg[10][7:6] | CRC运算模式 |
| Reg[10][5] | IP工作模式 |
| Reg[11][7:6] | CRC校验结果 |
| Reg[11][5:0] | 中断类型 |

3.3 跨时钟域同步模块

IP设计中共存在3种形式的跨时钟域信号的传输，分别为慢时钟域到快时钟域控制信号的传输、快时钟域到慢时钟域控制信号的传输和慢时钟域到快时钟域数据的传输。

对于慢时钟域到快时钟域控制信号的传输，我们采用的是三级触发器同步进行上升沿检测的方式进行同步。如图6所示。



图 6 慢时钟域到快时钟域控制信号的同步示意图

如图所示，慢时钟域Clock\_s中的信号en\_data通过3级寄存器同步到快时钟域Clock\_f，产生一个快时钟周期的同步信号output，则快时钟域可利用output信号采样data。



图7 快时钟域到慢时钟域控制信号的同步示意图

对于快时钟域到慢时钟域控制信号的传输，我们采用与上面类似的方法，但是区别在于，快时钟域的控制信号持续为高直至慢时钟域采到信号并反馈给快时钟域。时序如图7所示。

如图所示，快时钟域Clock\_f的en\_data通过3级触发器同步为慢时钟域Clock\_s的output，则慢时钟域可利用output信号采样data，然后output同样被3级触发器同步到快时钟域的output\_f，则快时钟域得知慢时钟域已收到信号。

对于原始数据的跨时钟域传播，由于AXI3\_MST模块读数据的速度受到模块外输入信号rvalid信号的影响而不确定，并且与 CRC处理数据的速度也不能完全匹配，所以确定采取异步FIFO的方式进行传输。同时考虑到如果IP处于运算模式，读进来的原始数据同时要发送到目的地址，且读取与发送的速度未必能完全匹配，所以同时对异步FIFO进行同步读取，两者同时控制FIFO的满信号的产生。

3.4 中断模块

中断模块负责产生中断信号，以通知软件查询IP工作状态。中断模块的信号来源共有6个，分别为：AXI3\_SLV模块的读/写错误、AXI3\_MST模块的读/写错误、CRC模块完成校验、AXI3\_MST模块完成CRC码的写入。同时在AXI3\_SLV模块种设置以终端类型寄存器组，其定义中断类型，具体如表2所示。

表2 中断类型定义

|  |  |
| --- | --- |
| Reg[11][5:0] | 中断类型 |
| 000001 | AXI3\_SLV模块写错误 |
| 000010 | AXI3\_MST模块读错误 |
| 000100 | AXI3\_MST模块写错误 |
| 001000 | AXI3\_SLV模块读错误 |
| 010000 | CRC模块校验完成 |
| 100000 | CRC码已写入目的地址 |

**4 仿真结果**

在完成方案设计及代码编写后，我们对各个小模块及总体都进行了仿真分析。

AXI3部分仿真结果如图8、图9所示。

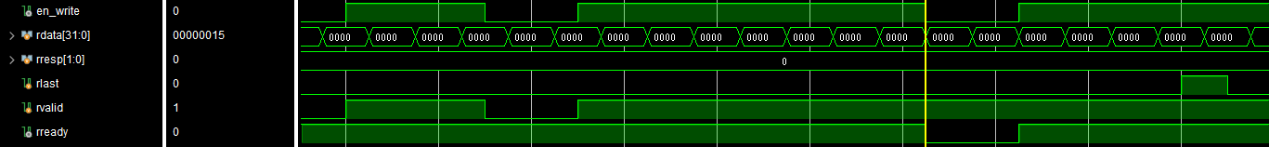


图8 AXI3\_MST从目的地址读原始数据时序图

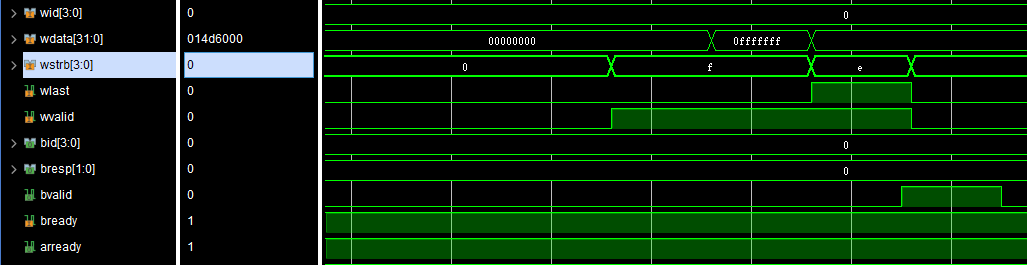


图9 AXI3\_SLV从IP外读入配置数据时序图

异步&同步FIFO的仿真结果如图10所示。

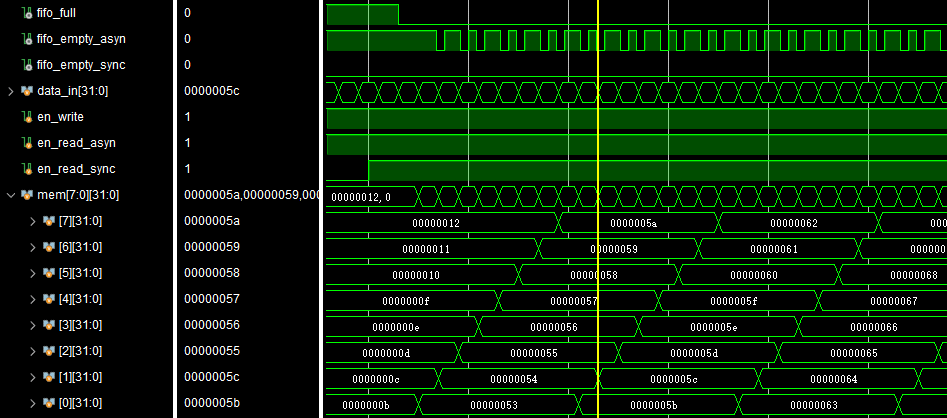


图10 异步&同步FIFO时序图

CRC模块的仿真结果如图11所示。数据长度为5，输入数据为12、34、56、78、23。我们将它拼接为00000012，34567823.拆分成0000、0012、3456、7823。crc\_16ccitt的校验结果分别为0x0000,0x3273,0xde61,0xdf5e。而此工作模式为校验模式，有一个额外输入数据0xdf5e，经比较，二者相等，因此校验结果正确，可见我们的程序正确，能够正确运行。

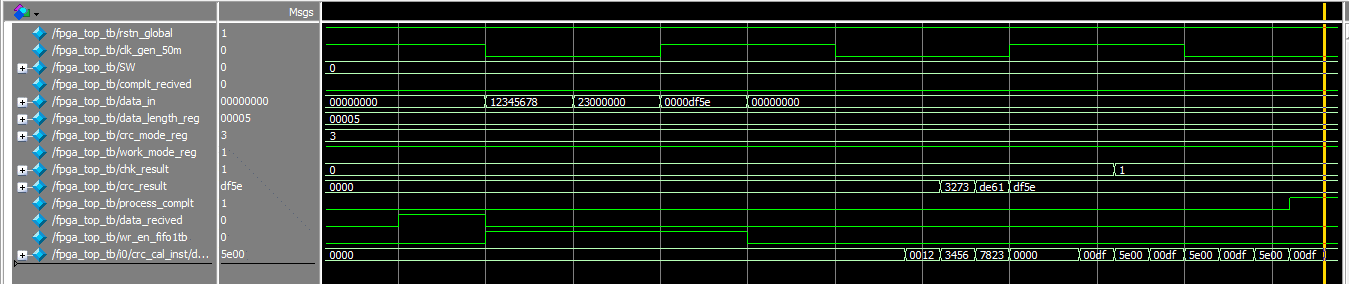


图11 CRC运算模块时序图

对于整体模块的仿真,由于时间原因，尚有一些BUG未能修复，但总体上来说IP的功能可以实现。

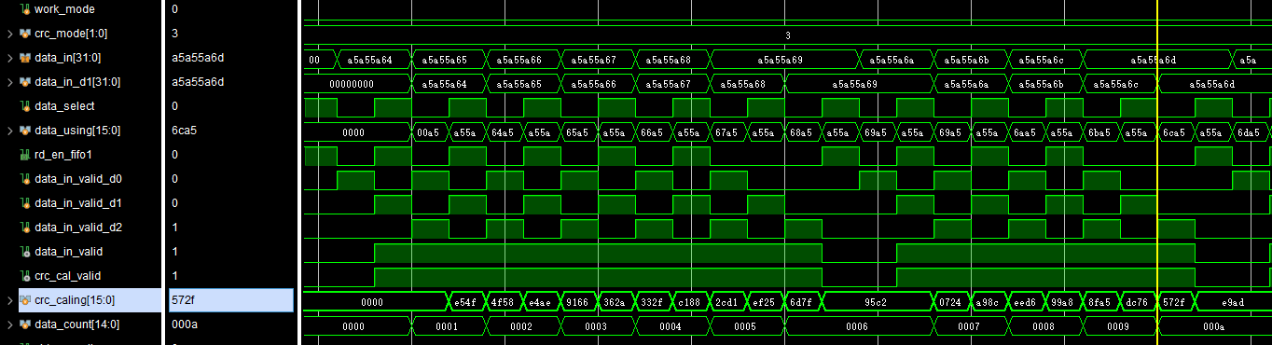


图12 CRC码产生时序图

图12为CRC从FIFO读取原始数据，并逐步产生CRC码的时序图。图中work\_mode为0，代表IP处于运算模式，crc\_mode为11，代表CRC模式为CRC\_CCITT。CRC从第一个32位的数据a5a55a64开始计算CRC码。

图13位AXI3\_MST模块将CRC码附在原始数据之后发送到目的地址时序图。在数据发出之后，产生中断信号intr，随后产生空闲信号idle，代表IP处于空闲状态。

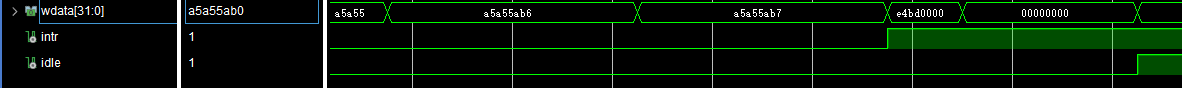


图13 CRC码写入时序图