文章编号:1671-4598(2023)12-0251-07 DOI:10.16526/j.cnki.11-4762/tp.2023.12.037 中图分类号:TP27 文献标识码:A

基于高层次综合的可编程晶振自适应配置技术

迟东明,徐铭泽
(上海航天电子有限公司,上海 201821)

摘要:可编程晶振以其输出频率可编程的特性,非常适合应用在码速率可变的弹、箭或卫星测控通信系统中,但使用传统的基于RTL编程方式对可编程晶振进行配置存在抽象层级低、代码实现复杂、开发效率低、调试难度大等缺点;针对上述问题提出了一种基于高层次综合技术的可编程晶振自适应配置方案;利用高层次综合技术实现了晶振参数的浮点计算及IIC总线通信,并以此为基础实现了可编程晶振的自适应配置;经测试,相对于传统RTL实现方式,使用高层次综合技术实现可编程晶振的参数计算和配置,其代码量仅为前者的1/10至1/20,开发周期仅为前者的1/4至1/10,且代码运行效率并无明显降低;结果表明,高层次综合技术不仅能够有效提高FPGA在算法开发、数据处理等领域的设计、开发效率,在FPGA接口和时序设计方面也有较好的应用和推广价值。

关键词:高层次综合;IIC;FPGA;可编程晶振;自适应

Adaptive Configuration Technology for Programmable Crystals Based on High Level Synthesis

CHI Dongming, XU Mingze
(Shanghai Aerospace Electronics Co., Ltd., Shanghai 201821, China)

Abstract: Programmable crystal has the characterics of its output frequency programmable, it is ideal for use in bullet, arrow or satellite measurement and control communication systems with variable code rates, but programming methods based on traditional register transfer level (RTL) are used to configure the programmable crystal, it has disadvantages such as low abstraction level, complex code implementation, low development efficiency and difficulty in debugging. A programmable crystal adaptive configuration scheme based on high level synthesis technology is proposed to address above problems; the floating point calculation of crystal parameters and Inter-Integrated Circuit (IIC) bus communication are realized using the high level synthesis technology, on this basis, the adaptive configuration of programmable crystal is realized. After the experimental testing, the high level synthesis technology is used to achieve the parameter calculation and configuration of programmable crystal oscillators, the code quantity of the high level synthesis technique is only 1/10 to 1/20 that of traditional RTL implementations, and the development cycle is only 1/4 to 1/10 of the former, with no significant reduction in code running efficiency. The results show that the high-level synthesis technology can not only effectively improve the design and development efficiency of FPGA in the fields of algorithm development and data processing, but also has a good application and promotion value in FPGA interface and timing design.

Keywords: high level synthesis; IIC; FPGA; programmable crystal; adaptive

0 引言

随着弹、箭、卫星测控技术的发展,其通信系统结构也变得更加复杂。为了尽可能提高通信系统的适应性,相关的测控分系统往往需要支持多种码速率通信方式,如从Kbps级至Mbps级宽码率范围内的动态可调,这就要求通信系统相关的时钟电路能够满足多码率的时钟需求。使用普通固定频率的时钟往往需要复杂时钟倍频、分频以及锁相环电路的配合,不仅集成度低,而且系统结构复杂,可编程晶振的出现则可以很好地满足以上需求。

可编程晶振具有极宽的频率范围和超精细的频率分辨率,非常适合应用于需要动态频率调整或非整数相关速率的多速率操作的应用场景。以Silicon Labs公司的SI598[1]为例,SI598是Silicon Labs公司推出的一款输出频率范围为10~810 MHz的可编程晶振,频率可编程分辨率可达28 ppt,其内部集成有第三代DSPLL基础的DCO,通过内部固定频率的晶振进行驱动,可输出高稳定性、超低抖动的时钟信号,时钟相位抖动RMS仅为0.5 ps。该芯片提供IIC[2-3]总线用于对其内部的寄存器进行读写,从而实现输出频率的调整。

从SI598的特性来看,其非常适合为测控系统提供可编程时钟,从而为测控系统的多码率通信提供硬件支持。弹、箭、卫星测控系统中大部分是以FPGA(field programmable gate array,现场可编程门阵列)作为前端核心处理芯片,而为了能够在保持高集成度的同时实现对可编程晶振的配置,

收稿日期:2023-05-25; **修回日期**:2023-06-14。

作者简介:迟东明(1981-),男,硕士,高级工程师。

引用格式:迟东明,徐铭泽.基于高层次综合的可编程晶振自适应配置技术[J].计算机测量与控制,2023,31(12):251-257.

就需要将可编程晶振的配置功能集成到 FPGA 中，而不是围绕可编程晶振配置功能再引入一套独立的嵌入式系统。

目前基于 FPGA 的可编程晶振配置功能的实现大多采取传统的 HDL（hardware description language，硬件描述语言）编程方式，使用 Verilog HDL、VHDL 等硬件描述语言来实现相关配置功能。但使用这种方式实现可编程晶振的配置功能时往往会遇到浮点运算及状态控制代码实现复杂、软件版本控制难度较大等一系列困难（原因在章节 1 可编程晶振配置原理中有具体描述）。

为了解决以上问题，本文引入了高层次综合技术[4-6]（HLS，high-level synthesis）来实现 SI598 的参数计算和配置。高层次综合技术简单来说，就是将用户使用高级语言（如 C、C++、System C 等）描述的设计通过专用的高层次综合工具转换为能够使用常规 FPGA 综合工具综合的 RTL（Register Transfer Level，寄存器传输级）代码（如 Verilog、VHDL 等）的一项技术。

高层次综合技术相对于传统的 RTL 实现方式具有设计效率高[7]、便于阅读和维护等优势。现阶段，高层次综合技术还处于不断发展和完善的过程中，虽然在异构计算、人工智能等领域已经得到了广泛的应用，但是目前还无法完全替代传统的基于 RTL 的开发过程。比如高层次综合技术目前还是主要应用于视频、图像、编解码、机器学习等基于数据处理的领域[8-11]，但由于其所使用的高级语言的特性，对于接口及时序描述能力较弱，很难做到 RTL 代码对时序的精确描述。

鉴于以上原因，使用高层次综合技术实现对 SI598 的参数计算和配置，首先需要实现通过 IIC 总线对芯片内部的寄存器进行读、写操作，然后再根据新的频率字及当前状态实现参数的自适应计算。基于高层次综合技术的 IIC 总线芯片寄存器读、写功能实现和相关时序控制是其中的设计难点。下面首先介绍可编程晶振的配置原理，然后讨论 IIC 总线的接口和时序设计。

1 可编程晶振配置原理

SI598 内部寄存器结构如图 1 所示，其内部有 8 个寄存器，用于存储配置参数或实现对晶振的控制，所有寄存器位宽均为 8 bit。为了实现 SI598 输出频率可编程，需要对其内部的相关参数进行重新计算，涉及到的主要配置参数为：*RFREQ*（倍频系数，位宽为 38 bit，其中高 10 bit 为整数，低 28 bit 为小数）、*HSDIV*（分频系数 1，位宽为 3 bit）、*N*1（分频系数 2，位宽为 7 bit）。因为以上参数大部分位宽都大于 8 bit，因此被拆分为若干个子参数被存储在不同的寄存器中。

为了使 SI598 按照指定频率输出时钟，首先需要对其特性参数和频率字进行计算，其配置输出频率的主要计算过程如下（假设需要晶振输出的频率为 $f_{out-new}$，当前晶振输出频率为 f_{out}）：

1）通过 IIC 总线读取 SI598 内部寄存器，获取当前的 *RFREQ*、*HSDIV* 和 *N*1 值，且 *RFREQ* 除以 2^{28}。

Register	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
7	High Speed/ N1 Dividers	HS_DIV[2:0]			N1[6:2]				
8	Reference Frequency	N1[1:0]			RFREQ[37:32]				
9	Reference Frequency	RFREQ[31:24]							
10	Reference Frequency	RFREQ[23:16]							
11	Reference Frequency	RFREQ[15:8]							
12	Reference Frequency	RFREQ[7:0]							
135	NewFreq/ Freeze/ Memory Control	Reserved		New Freq	Freeze M	Freeze VCADC	Reserved		RECALL
137	Freeze DCO	Reserved			Freeze DCO	Reserved			

图 1 SI598 内部寄存器结构

2）计算 f_{XTAL}。其中：$f_{XTAL}=\dfrac{f_{out}\times HSDIV\times N1}{RFREQ}$。

3）计算 f_{DCO_new}。其中。其中 *HSDIV* 取值范围为 9 或 11，*N*1 的取值范围为 1～128 的偶数（如 2，4，6，8…，128，这里也包括 1），通过搜索 *HSDIV* 与 *N*1 的各种组合使 f_{DCO_new}保持在 4 850～5 670 MHz 之间。如果存在多个有效的 *HSDIV* 与 *N*1 的组合，则需要选择最大的 *HSDIV* 和最小的 *N*1，以便降低芯片功耗。

4）计算 $RFREQ_{new}$。其中 $RFREQ_{new}=\dfrac{f_{DCO\_new}}{f_{XTAL}}$。而写入 *RFREQ* 寄存器的值为 $RFREQ_{new}$乘以 2^{28}后四舍五入的整数部分。

从以上计算过程可以看出，SI598 相关参数 *RFREQ*、f_{XTAL}、f_{DCO}等的计算不仅涉及到很多浮点计算步骤，而且还涉及到参数的组合、搜索过程。如果使用传统的 RTL 方式实现以上计算和搜索过程，其代码结构及状态控制会非常复杂，这将导致开发周期过长、调试工作量也会非常大，不利于产品的快速开发。

为了降低实现难度，一种折中的解决办法是将参数的搜索及计算过程由人工完成，FPGA 仅保存输出频点的计算结果（如 $N1_{new}$、$HSDIV_{new}$、$RFREQ_{new}$等参数）。当需要输出不同频点时，FPGA 仅需选取事先保存在其内部的相应的参数集，这样就省掉了浮点计算和参数搜索过程，降低了设计难度。但是这种实现方式会遇到以下的问题：由于不同的 SI598 内部晶体之间特性不完全一致，这就导致其内部的 *RFREQ*，甚至 *HSDIV* 和 *N*1 都会存在差异（即不同的 SI598，其 *RFREQ*、*HSDIV* 和 *N*1 不同）。因此在对不同的 SI598 进行配置时，都需要针对其特定的 *RFREQ*、*HSDIV* 和 *N*1 值进行单独计算。这就造成了即使输出相同的频点，针对不同 SI598 所使用的配置参数也不尽相同。对于像航天、军工等对软件版本、流程控制要求非常严格的领域，这无疑增加了软件管理成本，而且也很容易造成低层次质量问题。因此需要实现对频率的自适应配置，SI598 配置参数的计算及配置都需要在 FPGA 内实现。

2 可编程晶振配置模块的高层次综合设计与实现

引入高层次综合技术后的 SI598 自适应配置模块的原理

如图 2 所示，主要分为配置模块、IIC 读写函数两部分。其中 IIC 读写函数相当于底层驱动程序，用于实现 SI598 寄存器的读、写操作及生成相关 IIC 接口时序。IIC 读写函数分为两个子函数：IIC 读函数（以下简称 iic _ read _ byte（））、IIC 写函数（以下简称 iic _ write _ byte（））。iic _ read _ byte（）负责从指定的 SI598 寄存器地址读取 1 字节内容并返回读取内容，而 iic _ write _ byte（）负责将 1 字节数据写入 SI598 指定地址的寄存器中。配置模块负责 SI598 配置参数的计算、搜索及 SI598 的具体配置。该模块通过调用 IIC 读写函数实现对 SI598 的寄存器读取和写入。

除了以上两个功能模块外，设计中还引用了两种 FPGA 原语：IOBUF、STARTUP。IOBUF 原语用于实现 SI598 配置模块与 SI598 的 IIC 接口的连接，其实现原理在章节 2.1IIC 接口实现中有具体描述；STARTUP 原语用于为 SI598 配置模块提供工作时钟及复位信号。

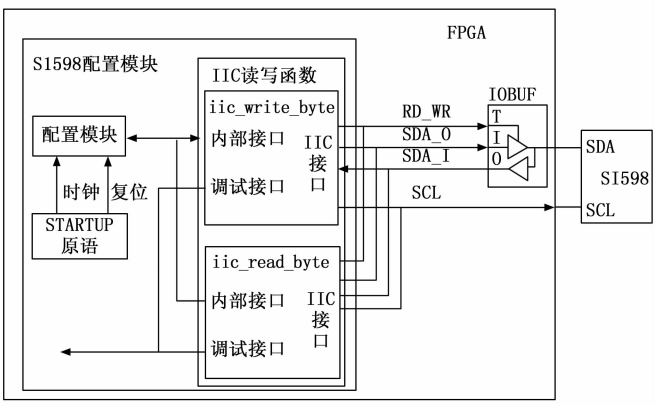

图 2　SI598 配置模块原理框图

由于晶振配置模块本身也需要相应的时钟及复位信号才能正常工作，而这些信号正常情况下是不能由被配置的可编程晶振提供（可编程晶振在配置过程中时钟可能会中断），这就需要额外的时钟源为晶振配置模块提供时钟。

需要说明的是，SI598 配置模块正常情况下应该是系统正常加电后第一个运行的模块，在其完成晶振配置，可编程晶振输出正确时钟后，其他相关模块才能正常工作。如果为了保证 SI598 配置模块正常工作在 FPGA 外部再增加一个晶振势必会增加硬件成本并降低系统集成度，因此该设计中使用了 FPGA 的 STARTUP 原语作为晶振配置模块的时钟源。STARTUP 原语是 FPGA 配置电路相关的专用原语，可以为用户逻辑提供配置时钟及其他配置相关的状态信号。以 Xilinx 公司 7 系列 FPGA 的 STARTUP 原语——STARTUPE2[12] 为例，其对外接口如图 3 所示。其中 CFGMCLK 是 FPGA 内部振荡器时钟输出信号，典型值约为 65 MHz，该信号为 FPGA 配置电路提供时钟，也可以作为用户逻辑的工作时钟；EOS 为 FPGA 启动完成信号，当 FPGA 启动完成后会置高。以上两个信号可以作为可编程晶振配置模块的工作时钟和复位信号。

这里简要描述一下 STARTUP 原语及 SI598 配置模块

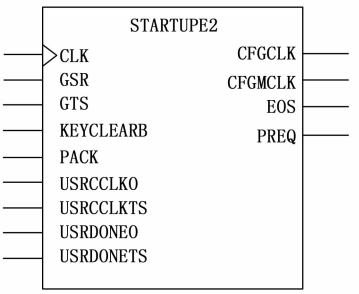

图 3　STARTUPE2 对外接口

在加电后的工作过程：

系统加电后 FPGA 首先进行配置文件的加载，从配置 FLASH 中读取配置比特流，此时 STARTUP 的 CFGMCLK 开始输出配置时钟。因为此时 FPGA 还未完成配置，EOS 为低，SI598 配置模块处于复位状态。当 FPGA 正确完成配置后，EOS 为高，SI598 配置模块开始进行晶振的配置。晶振配置完成后，SI598 输出配置完成信号（该信号在章节 2.1IIC 接口设计有具体描述），对其他用户逻辑进行复位。

2.1　IIC 接口设计

IIC 总线（inter-integrated circuit，集成电路总线）是 PHILIPS 公司推出的一种常用的芯片间互联的双向二线制同步串行总线，相比并行总线连线少，结构简单，且可实现多主机系统所需的裁决和高低速设备同步等功能，是一种高性能的串行总线。

IIC 标准[2] 定义了 2 个接口：SCL、SDA。其中 SCL 为时钟信号，由主设备提供，从设备以该信号为时钟接收从主设备发送的数据；SDA 为双向数据信号，主设备通过 SDA 向从设备发送数据也可以通过该信号读取从设备的数据或应答。为了实现与 SI598 的 IIC 接口互连，首先需要实现 IIC 接口及相关时序。

这里使用了 Xilinx 公司的 VIVADO HLS 高层次综合工具作为该模块的开发平台。VIVADOHLS 支持用户使用 C、C++以及 SystemC 等高级语言进行设计，后续以 C 为例对 IIC 接口及功能实现进行描述。

虽然 VIVADO HLS 提供了对双向端口的支持，但要实现精确的读、写时序控制仍较困难，无法直接与 IIC 总线互连。为了能够实现对 SDA 端口读写时序的精确控制，SI598 配置模块外部使用了 1 个 IOBUF 连接到 SI598 的 SDA。IOBUF 是 FPGA 的 IO 端口用于控制 IO 输入、输出状态的原语，其包含有 3 个接口：I、O 和 T，其中 I 为输入，O 为输出，T 为三态控制，当 T 有效时，相应的 FPGA 接口表现为高阻态，此时内部逻辑可以通过 IOBUF 读取接口上的数据。IOBUF 采用了如图 2 所示的连接方式，其中 FPGA 作为主设备，而 SI598 作为从设备。RD _ WR、SDA _ O、SCL 作为主设备输出，其中 RD _ WR 连接 IOBUF 的 T 端，用于控制相关 IO 的输入、输出状态，当 RD _ WR 为 0 时，IOBUF 处于输出状态，SDA _ O 上的数据通过 IOBUF 输出

到 SDA 总线上；当 RD _ WR 为 1 时，IOBUF 处于输入状态，SDA 总线上从设备的应答信号或数据通过 IOBUF 输出到 SDA _ I 上，而 SDA _ I 作为主设备输入，用于接收 SI598 的应答信号及返回的数据。

为了使 RD _ WR、SDA _ O、SCL 等端口能够同步输出数据，在代码实现上，将这 3 个端口合并成 1 个 3 bit 的无符号整型指针，这样可以保证在对该指针进行赋值时，相关端口的同步输出。同时输出接口需要使用寄存器用于保持当前状态，直到数据再次更新，而为了便于观察模块输出的指令，也需要对外端口具备一些指示信号（如数据有效指示）。这里用到了 VIVADO HLS 的约束语句，约束语句用于指示高层次综合器在指定的语句或函数使用哪种综合策略或映射为哪种硬件资源，从而指导综合器按照预期的方式对代码进行综合。

SI598 配置模块对外接口相关代码及约束如下：

```
void hls_iic_block(
uint3 * iic_rdwr_sdo_scl,
uint1 * iic_sda_i,
uint47 * iic_wr_buf,
uint32 * freq_word,
uint1 * finish)
{
pragma HLS INTERFACE ap_none register port=finish
pragma HLS INTERFACE ap_vld register port=iic_rdwr_sdo
_scl
pragma HLS INTERFACE ap_vld register port=iic_sda_i
pragma HLS INTERFACE ap_vld register port=iic_wr_buf
```

接口参数均使用了指针类型，主要目的是为了减少参数传递造成的延时并尽量减少对 FPGA 的资源占用。其中 * iic _ rdwr _ sdo _ scl 是 RD _ WR、SDA _ O、SCL 合并后的 uint3 类型指针，* iic _ sda _ i 是 uint1 类型指针，* iic _ wr _ buf 是 uint47 类型指针，该接口为调试接口，SI598 配置模块通过 IIC 总线发送的所有寄存器读、写指令都会同步输出到该接口上，用户也可以通过该接口用于判断芯片的工作状态或进行问题定位；* freq _ word 是 uint32 类型指针，用于表示 SI598 的目标频率控制字，SI598 配置模块通过该接口读取目标频率字并进行频率控制字的计算；* finish 是 uint1 类型指针，用于指示配置过程完成情况，当所有配置工作完成后，SI598 配置模块将该接口置高，通知其他逻辑晶振配置过程完成。其中 * iic _ rdwr _ sdo _ scl、* iic _ wr _ buf、* finish 为输出信号，而 * iic _ sda _ i、* freq _ word 为输入信号。

为了对函数对外接口进行必要的约束，以便 VIVADOHLS 按照用户要求生成对外接口，在主函数开始部分增加了一些接口约束语句。其中“pragma HLS INTERFACE”表示对函数接口部分进行约束；* iic _ rdwr _ sdo _ scl 和 * iic _ wr _ buf 的接口采用了 ap _ vld 接口形式，ap _ vld 表示相关接口需要绑定数据有效指示信号。当 VIVADOHLS 相关接口进行综合时会为该信号绑定一个数据有效指示信号，以便于其他模块判断输出数据的有效性，或者便于使用逻辑分析仪通过数据有效指示信号抓取信号用于调试；* iic _ sda _ i、* freq _ word、* finish 的接口则采用了 ap _ none 接口形式，ap _ none 表示相关接口不绑定任何握手或指示信号。

图 4 是配置模块经过 VIVADO HLS 综合后实际的对外接口情况。其中前缀为 ap _ 的接口是 VIVADO HLS 工具为该模块增加的模块级握手信号，握手信号协议采用的是 ap _ ctrl _ hs，该协议是模块级设计默认采用的协议，主要是增加了一些为保证模块正常工作所需的时钟、复位、使能信号及模块当前工作状态指示信号等接口。而 ap _ * 下方的接口则是 VIVADO HLS 根据模块接口定义及其约束生成的自定义接口，这部分接口与 SI598 配置模块对外接口相关代码及约束是一一对应的。

Interface

Summary

RTL Ports	Dir	Bits	Protocol	Source Object	C Type
ap_clk	in	1	ap_ctrl_hs	hls_iic_block	return value
ap_rst	in	1	ap_ctrl_hs	hls_iic_block	return value
ap_start	in	1	ap_ctrl_hs	hls_iic_block	return value
ap_done	out	1	ap_ctrl_hs	hls_iic_block	return value
ap_idle	out	1	ap_ctrl_hs	hls_iic_block	return value
ap_ready	out	1	ap_ctrl_hs	hls_iic_block	return value
iic_rdwr_sdo_scl	out	3	ap_vld	iic_rdwr_sdo_scl	pointer
iic_rdwr_sdo_scl_ap_vld	out	1	ap_vld	iic_rdwr_sdo_scl	pointer
iic_sda_i	in	1	ap_none	iic_sda_i	pointer
iic_wr_buf	out	47	ap_vld	iic_wr_buf	pointer
iic_wr_buf_ap_vld	out	1	ap_vld	iic_wr_buf	pointer
freq_word	in	32	ap_none	freq_word	pointer
finish	out	1	ap_none	finish	pointer

图 4 SI598 配置模块综合后对外接口

2.2 IIC 接口时序设计

根据 IIC 总线协议，IIC 总线指令一般分为写字节指令和读字节指令，可以支持 1 字节或者多字节的读、写操作，为了便于实现，这里主要讨论单字节 IIC 读、写接口时序的实现。为了保证以上指令正确的时序关系，本文将 IIC 总线的读写波形（如 SDA、SCL、RD _ WR）保存到了一个二维数组中，具体实现如下：

通过对 IIC 总线协议的分析，IIC 总线协议主要由如图 5 所示的 4 种基本时序构成：起始位、停止位、写数据和读数据。为了模拟出 SDA 与 SCL 的时序关系，数组中的数据单元使用了 4 bit 宽度的无符号整型数据。如 IIC 数据起始位要求 SCL 为高时，SDA 由高至低，用 4 bit 无符号整型数据描述的话就是 SCL=0x6，SDA=0xC。

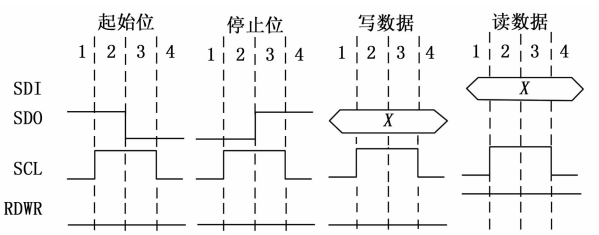

图 5 IIC 协议基本时序

主设备通过 IIC 总线写 1 字节数据至从设备的时序如图 6 所示。其中“S”表示 IIC 数据的起始位，“A”表示从设备在接收到 1 字节数据后给主设备的应答位（低有效），“P”表示 IIC 数据的停止位。而为了输出完整的 IIC 指令，只需要遍历数组中的数据并逐位输出给 SDA、SCL、RD _ WR 即可。对于保存波形的二维数组的大小，如通过 IIC 总线写 1 字节数据，则该二维数组大小为 3 行 * 29 列 * 4 bit =348 bit。

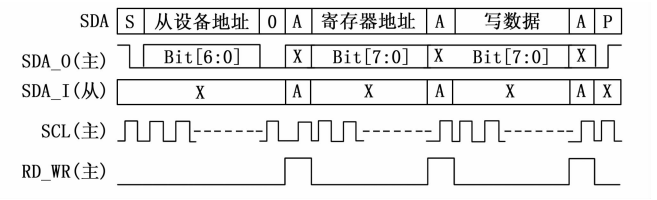

图 6　IIC 总线写字节指令时序关系

主设备通过 IIC 总线从从设备读 1 字节数据的时序如图 7 所示。时序关系与 IIC 写指令基本相同，主要区别是增加了一次从设备地址写入及数据读取过程。对于保存波形的二维数组的大小，如通过 IIC 总线写 1 字节数据，则该二维数组大小为 3 行 * 39 列 * 4 bit=468 bit。

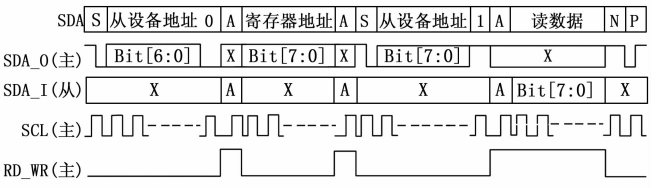

图 7　IIC 总线读字节指令时序关系

根据设计要求，SI598 配置模块的工作时钟为 10 MHz，该频率远大于 SI598 的 IIC 总线通信速率（该芯片 IIC 总线通信速率最高支持 400kbps），为了能够匹配 IIC 总线的通信速率，需要在对 IIC 接口读写时插入一些延时，这里引用了 VIVADO HLS 中的专用延时语句——ap _ wait _ n (num _ clock)，该语句用于指示高层次综合工具延时指定的时钟周期，其中 num _ clock 为时钟周期数，表示延时 num _ clock 个时钟周期。

综上所述，IIC 总线写字节指令的 C 语言伪代码如下：

```
uint4 iic_wr_array[3][iic_write_buffer_width]={
{数据},
{时钟},
{读写控制}};
for(loop1=0;loop1<iic_write_buf_width;loop1++)
{ bit_mask=0x8;
for(loop2=0;loop2<4;loop2++)
{
* iic_rdwr_sdo_scl=
(uint3)(((iic_wr_array[2][loop1]&bit_mask)>0)<<2|
(uint3)(((iic_wr_array[0][loop1]&bit_mask)>0)<<1|(uint3)
(((iic_wr_array[1]&bit_mask)>0);
ap_wait_n(延时周期);
if((iic_wr_array[2][loop1])&&(* sda_i))
{
如果为读,则读取 * iic_sda_i 上的数据并保存
}
bit_mask=bit_mask>>1;
}
}
```

其中：iic _ wr _ array 是一个 uint4 类型的二维数组，用于存放 IIC 写指令的整个时序，包括 SDA 数据、SCL 时钟和 SDA 读写控制标志，在输出 IIC 写指令时序时使用了两级 for 循环。根据 IIC 协议，主程序在输出一定数量的数据位后会将 SDA 设置为输入状态，以便监听从设备的应答信号或者接收返回的数据，因此在循环的同时，程序会根据 iic _ wr _ array 的 SDA 读写控制标志将 IOBUF 设置为相应的输入、输出状态，当 IOBUF 为输入状态时，程序会从 * iic _ sda _ i 读取从设备应答信号或数据，并保存到缓冲区中，如果是从设备应答信号则会根据应答信号电平判断此次写入是否成功，以便进行后续操作；如果是从设备返回的数据则会将返回数据输出给 * iic _ wr _ buf 或者上一级程序。IIC 读字节指令基本结构与 IIC 写字节指令类似，这里不再复述。

2.3　SI598 配置参数的浮点计算

从 SI598 的配置过程可知，f_{XTAL}、f_{DCO}、*RFREQ* 等参数的计算过程均涉及到浮点计算。虽然基于 RTL 的实现方式也可以实现浮点运算，同时 FPGA 厂家也提供了一系列的浮点运算 IP，但基于这种实现方式进行浮点运算不仅费时、费力，而且调试难度很大，不利于产品的快速开发。而 VIVADO HLS 支持 IEEE-754 标准的单精度和双精度浮点计算[13-15]，因此可以为 SI598 的配置参数计算提供支持。

IEEE-754 标准的单精度浮点数由 1 bit 符号位、8 bit 指数位以及 23 bit 尾数位构成，而双精度浮点数由 1 bit 符号位、11 bit 指数位以及 52 bit 尾数位构成。由于 SI598 的 RFREQ 是位宽为 38 bit 的频率控制字（低 28 bit 为分数部分，高 10 bit 为整数部分），单精度浮点数无法满足设计需求。为了保证计算精度，这里使用双精度浮点数变量保存 SI598 的浮点参数。由于双精度浮点计算需要耗费较多的 FPGA 硬件资源，为了节省硬件资源，程序使用了尽量少的双精度浮点类型变量，并优化了浮点计算步骤，使得整个参数计算过程使用尽量少的浮点计算，其余变量则根据实际位宽使用 VIVADO HLS 的任意精度整型数据，而浮点数与整型数据的混合计算则使用强制数据类型转换。相关参数计算的相关 C 语言代码如下：

```
double rfreq_fac;//SI598 的 RFREQ_FACTOR
uint16 hs_div_nq_new;//HSDIVnew * N1new
uint16 hs_div_n1_old;//HSDIVold * N1old
double rfreq_new;//RFREQ_new
uint38 rfreq_new_int;//RFREQnew 整数部分
hsdiv_n1_new=hsdiv_new * n1_new;
```

```
hsdiv_n1_old=hsdiv_old * n1_old;
rfreq_fac=(double)( * freq_word)/41943040.0f;
rfreq_fac=rfreq_fac * (double)hsdiv_n1_new/(double)hsdiv_n1_old;
rfreq_new=rfreq_fac * (double)rfreq_old;
rfreq_new_int=(uint38)llround(rfreq_new);
```

其中定义 rfreq _ fac、rfreq _ new 为双精度浮点类型变量，用于存放 RFREQ 计算过程中的中间结果；rfreq _ new _ int 为 uint38 类型的整型数，对应 RFREQ 的 38 位数据。由于 rfreq _ new 是双精度数据类型变量，而 rfreq _ new _ int 是整型数变量，rfreq _ new 到 rfreq _ new _ int 赋值需要进行数据类型转换。这里数据类型转换使用了 llround（）函数，该函数是 VIVADOHLS 的标准 C 函数头文件 math.h 中的数据类型转换函数，用于舍入给定值并将其转换为长整型整数。

2.4 SI598 配置模块的实现

从以上设计过程可以看出，SI598 配置模块主要是建立在 SI598 寄存器读、写操作的基础上，其余诸如配置参数搜索、计算以及频率控制字的读取等操作均可以放置在配置模块的主体结构中实现。SI598 配置模块主程序流程如图 8 所示。

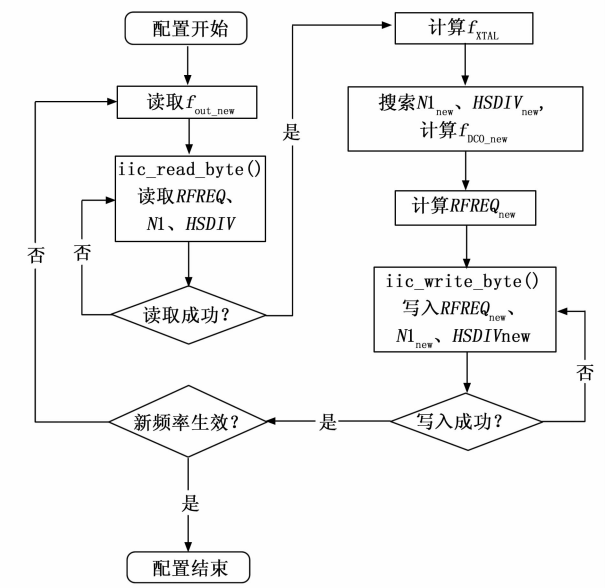

图 8 SI598 配置模块程序流程图

具体的配置过程描述如下：

1）配置开始，配置程序在接收到 STARTUP 原语的时钟和复位信号后开始工作，首先读取新的频率控制字（由 freq _ word 计算得到）。

2）配置程序通过 iic _ read _ byte（）尝试读取 SI598 中 *RFREQ*、*N*1、*HSDIV* 等相关寄存器参数，如果读取成功则进入下一步，如果读取失败则重复读取，如果读取失败次数达到一定值则退出程序，配置失败。

3）配置程序利用步骤 2 中读取的 *RFREQ*、*N*1、*HSDIV* 等参数计算 f_{XTAL}。

4）配置程序根据搜索策略搜索最优 $N1_{new}$、$HSDIV_{new}$ 参数组合，并以此为基础计算 f_{DCO_new}。

5）配置程序根据 f_{DCO_new} 和 f_{XTAL} 计算 $RFREQ_{new}$。

6）配置程序通过 iic _ write _ byte（）尝试将计算出的 $N1_{new}$、$HSDIV_{new}$、$RFREQ_{new}$ 写入 SI598 指定的寄存器，如果写入成功则进入下一步，如果写入失败则重复写入，如果写入失败次数达到一定值则退出程序，配置失败。

7）配置程序通过 iic _ read _ byte（）尝试读取 SI598 相应寄存器相应标志位判断新频率是否已生效，如果已生效则配置结束，输出配置完成信号（对应接口 finish），如果没有生效则进入步骤 1），对芯片重新进行配置。

3 测试结果与分析

上述程序通过 VIVADO HLS 开发环境实现后，再通过高层次综合工具转换为 RTL 级代码并打包为用户自定义 IP，最后由 VIVADO[16] 开发环境的顶层模块调用。

而为了验证该模块的功能，首先对模块进行了功能仿真，随后进行了硬件测试。

3.1 模块功能仿真

图 9 是在 VIVADO 平台上对配置模块进行功能仿真的 IIC 总线时序关系截图，图中是 IIC 总线写指令部分时序关系，时序关系与设计预期一致，配置模块收到由测试激励产生的 ACK 信号后，模块产生预期的有效数据，表明写操作成功。

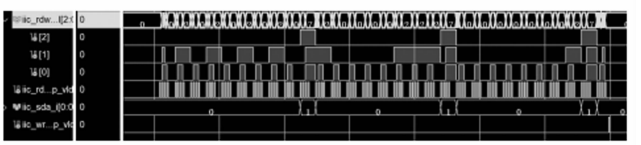

图 9 IIC 总线写操作仿真时序

3.2 硬件测试验证

在硬件测试阶段，该模块相关的硬件平台使用了 Xilinx 公司的 Artix-7 系列 FPGA XC7A200T[17]，而软件开发平台为 VIVADO 和 VIVADO HLS。

使用 VIVADOHLS 对代码进行综合后，模块的资源占用情况如图 10 所示。其中 LUT 使用了 8 607 个，资源占用率约 6%；FF 使用了 2 513 个，资源占用率不到 1%；DSP 模块使用了 12 个，资源占用率约 1%；BlockRAM 使用了 3 个，资源占用率不到 1%。可以看出整个模块的 FPGA 资源占用率不高。

在时序性能方面，VIVADOHLS 提供了不同的优化策略，这里选择了默认的优化策略。经过 VIVADO HLS 综合后的模块运行速率可达 87.5 ns（11.4 MHz），优于设定的工作速率 100 ns（10 MHz），满足设计需求。

硬件测试时，首先在 VIVADOHLS 下将 SI598 配置模块进行打包，封装成 IP，然后在 VIVADO 开发环境下，将包含有 STARTUP 原语和 SI598 配置模块 IP 的设计综合、布局、布线，最终生成 bit 文件。为了验证硬件加电后配置模块的工作情况，这里将 bit 文件转换为 MCS 文件并烧写

Utilization Estimates

Summary

Name	BRAM_18K	DSP48E	FF	LUT
DSP	-	-	-	-
Expression	-	1	0	1912
FIFO	-	-	-	-
Instance	2	11	1719	5882
Memory	1	-	0	0
Multiplexer	-	-	-	813
Register	-	-	794	-
Total	3	12	2513	8607
Available	730	740	269200	129000
Utilization (%)	~0	1	~0	6

图 10　SI598 配置模块 FPGA 资源占用情况

到 FPGA 的配置 FLASH 中。经过反复加电测试，SI598 均可以输出指定频率的时钟，可以证明 STARTUP 原语可正常为 SI598 配置模块提供稳定的时钟及复位信号。

图 11 是在实际硬件测试过程中的使用 VIVADO 内置的 ILA（integrated logic analyzer，集成逻辑分析器）抓取的配置模块 IIC 总线相关信号波形，与功能仿真结果一致。

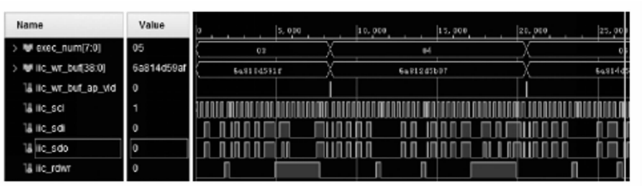

图 11　SI598 配置模块 ILA 截图

为了验证可编程晶振的自适应特性，实际测试中使用了 2 块相同的硬件平台。由于不同 SI598 的内部参数并不相同，需要配置程序针对不同参数进行自适应计算从而输出相同的频率。测试结果表明，对于不同的 SI598 芯片（内置参数不同）均可以使用相同代码输出相同的频率，满足自适应配置要求。

在代码量及开发周期上，整个配置模块的代码量约 500 行，测试激励代码量约 100 行，且整个编码调试过程用时仅 2 天。作为对比，如果采用 RTL 级实现方式，预计代码量约为 5 000 行以上，而编码调试时间预计要 2 周左右。可以看出，利用高层次综合技术有效提高了模块的设计、开发效率。同时相关代码由于使用的是 C 语言实现，代码的阅读、维护以及软件管理方面相对于传统的 RTL 代码有着较大的优势。

4　结束语

目前高层次综合技术主要应用在数据处理[18-19]、数据编解码[20-21]等领域，而在数字接口设计、芯片配置等方面的应用和研究还较少。而本文将高层次综合技术应用于可编程晶振的自适应配置中，解决了芯片配置中涉及到的浮点计算、IIC 接口时序生成、IIC 接口通信等问题，经过简单修改也可以应用于很多其他 IIC 总线芯片的配置过程中，且代码实现效率较高、开发周期短，具有较好的应用和参考价值。

参考文献：

[1] SILICON LABS. Si598/Si599 Datasheet（v1.0）[EB/OL]. Silicon Labs，2011.

[2] NXPSe. UM10204-I2C-Bus Specification and User Manual [Z]. Nxp semiconductors. Version. 6，2014：4.

[3] 王红亮，刘　伟，何少恒，等. IIC 总线和 LVDS 在高速数据传输接口电路中的应用研究 [J]. 计算机测量与控制，2016，24（7）：181－182，186.

[4] 党宏社，王　黎，王晓倩. 基于 Vivado HLS 的 FPGA 开发与应用研究 [J]. 陕西科技大学学报（自然科学版），2015，33（1）：155－159.

[5] CANIS A，CHOI J，ALDHAM M，et al. LegUp：high-level synthesis for FPGA-based processor/accelerator systems [C] //Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays，2011：33－36.

[6] MARTIN G. High-level synthesis：past，present，and future [J]. IEEE Design & Test of Computer，2009，26（4）：18－25.

[7] 高国栋，林　明. 用 Vivado HLS 实现粒子滤波算法的硬件加速 [J]. 江苏科技大学学报（自然科学版），2018，32（2）：245－251.

[8] 林振钰. 基于 ZYNQ 的高清图像显示及检测系统设计 [J]. 计算机测量与控制，2021，29（2）：30－34.

[9] 王　林. 基于 HLS 协议视频监控加密系统优化设计与实现 [J]. 计算机测量与控制，2019，27（7）：169－173.

[10] 朱中杭. 基于 SRCNN 的压缩感知图像超分辨率重建仿真研究 [D]. 哈尔滨：哈尔滨工程大学，2018.

[11] 李　浩. 基于异构多核的高性能视频编码器研究与实现 [D]. 北京：北京邮电大学，2015.

[12] Xilinx. UG470-7 series FPGAs configuration user guide [Z]. Xilinx，2019.

[13] HRICA J. 利用赛灵思 Vivado HLS 实现浮点设计 [J]. 今日电子，2013，1：34－38.

[14] HRICA J. XAPP599-floating-point-design with Vivado HLS（v1.0）[Z]. Xilinx，2012：4－6.

[15] Xilinx. UG902-vivado design suite user guide-high-level synthesis（v2019.1）[Z]. Xilinx，2019.

[16] Xilinx. UG893-Vivado design suite user guide-using the Vivado IDE（v2019.1）[Z]. Xilinx，2019.

[17] Xilinx. DS180-7 series FPGAs data sheet-overview（v2.6）[Z]. Xilinx，2018.

[18] 彭鑫磊，余　乐. 基于高层次综合的卷积神经网络设计与优化方法研究 [J]. 微电子学与计算机，2019，36（8）：63－67.

[19] 褚亭强. 基于 Zynq-7000 的高清视频采集处理软硬件协同设计 [D]. 南京：南京邮电大学，2017.

[20] 王　林，石　焘. 基于 HLS 协议视频监控加密系统优化设计与实现 [J]. 计算机测量与控制，2019，27（7）：169－173，179.

[21] 高李娜. H.264 视频编解码的 FPGA 实现 [D]. 西安：西安电子科技大学，2017.