数字ASK调制系统设计报告

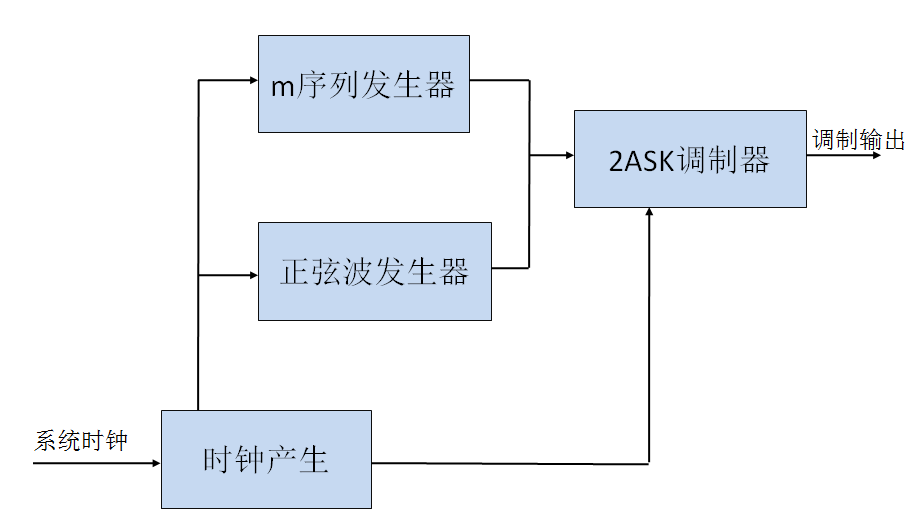
# 设计思路

根据题目要求，系统时钟为32 MHz，系统基带信号速率为3.2 kbps，正弦波信号为10kHz，每个周期16个样点，所以系统内部还需要3.2 kHz和160 kHz两个时钟，由一个专门的模块来产生分频时钟信号。除此以外还需要一个m序列发生模块，一个正弦信号发生模块和一个2ASK模块。每个模块的设计思路如下：

1. 内部时钟产生模块：通过计数分频的方式，产生内部模块所需的3.2 kHz和160 kHz频率的时钟。
2. m序列产生模块：使用线性移位的方式，产生周期为 的 m 序列，反馈线由给出。
3. 正弦载波信号产生模块：在模块内存储正弦信号的16个采样值，用160 kHz 的时钟驱动，依次输出，这样就产生了10kHz，每个周期16个样点的正弦信号。
4. 2ASK调制模块：使用组合逻辑来调制，用数字基带信号来控制正弦幅值。当数字基带信号为1时，使正弦幅值为128；当数字基带信号为0时，使正弦幅值为0。再将已调值以160 kHz的时钟锁存到输出端口，对外部屏蔽竞争——冒险带来的毛刺。

所有模块中的时序部分都使用异步、高电平复位。

# 系统结构



**图1 系统框图**

分别产生正弦波和m序列，送入调制器产生2ASK信号。

# 时序说明

分频模块采用计数器计数翻转输出的方法分频。对于3.2 kHz的输出时钟，计数器的最大值为 ，当计数器达到最大值时翻转时钟，并且计数器清零重新计数。当系统复位时，计数器清零，输出时钟为低电平。同理对于160 kHz的时钟计数器的最大值为。

m序列发生模块，输入时钟为3.2kHz，m序列的阶数为5，输出序列的周期为。

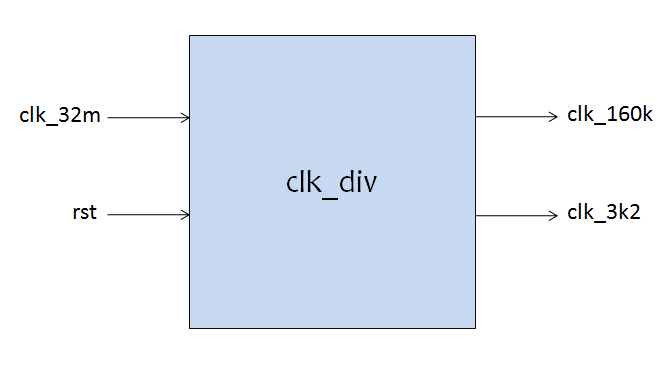
正弦波发生模块当系统复位时输出128，否则由160 kHz时钟上升沿驱动依次输出采样值。

2ASK模块在系统复位时输出为128，在正常情况下则根据基带型号输出。如果基带信号时0则输出128，如果基带信号是1则输出为正弦波发生器模块的输出。

# 模块说明

1. 内部时钟产生模块

模块框图：



**图2 内部时钟产生模块框图**

引脚说明：

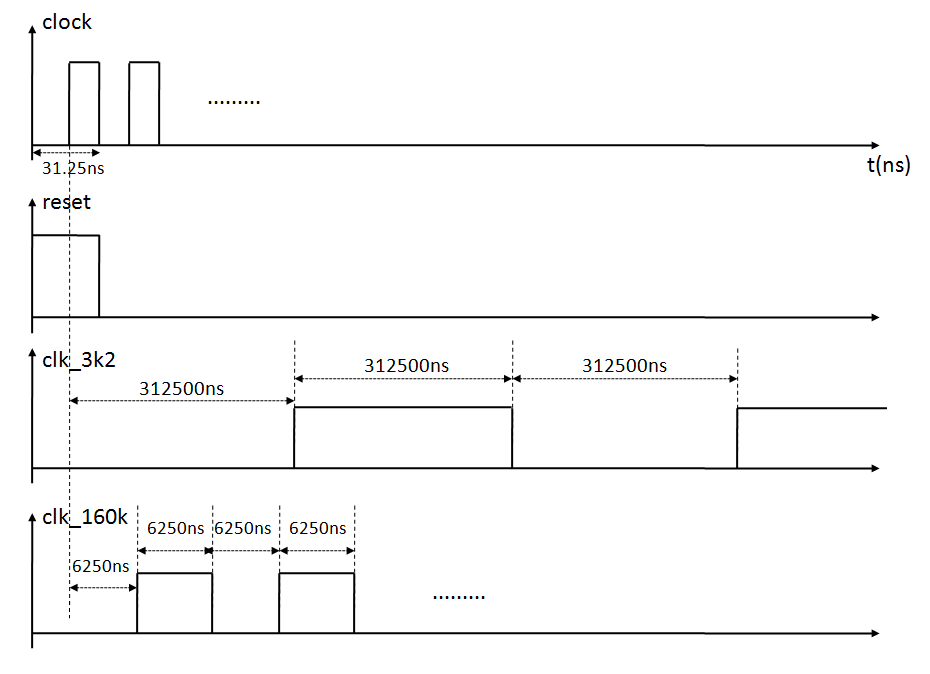
clk\_32m ：系统时钟32 MHz输入

rst ：系统reset复位信号输入

clk\_160k ： 160 kHz时钟输出，周期6 250 ns（对应于Clk\_V2和Clk\_V3）

clk\_3k2 ：3.2 kHz时钟输出，周期312 500 ns（对应于Clk\_V1）

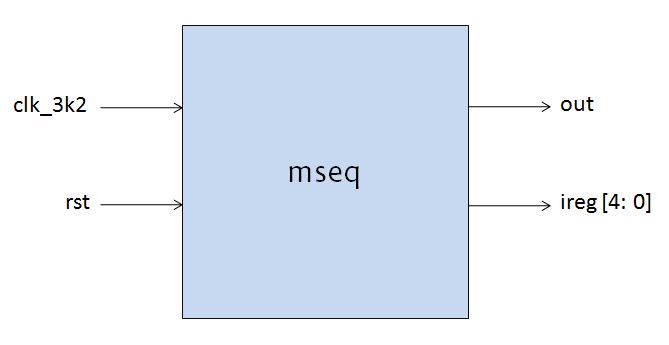
时序图：



**图3 内部时钟产生模块时序图**

1. m序列产生模块

模块框图：



**图4 m序列产生模块框图**

引脚说明：

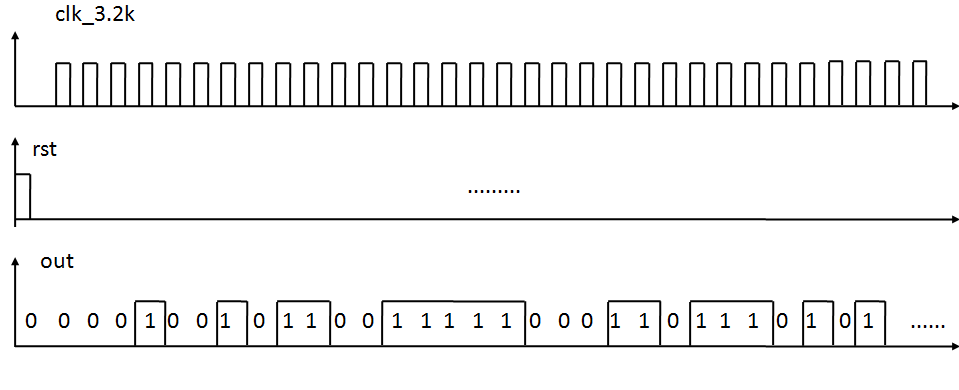
clk\_3k2 ：3.2 kHz时钟输入

rst ：系统reset复位信号输入

out ：m序列输出

ireg ：内部5个寄存器的状态输出

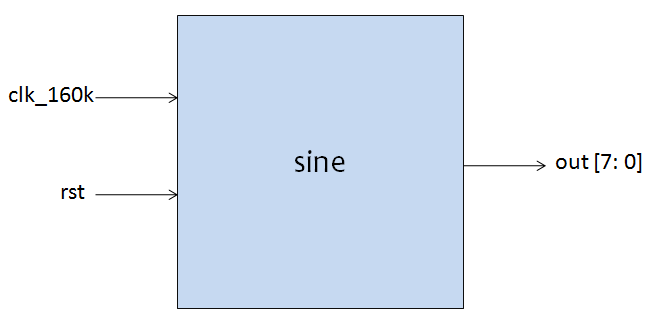
时序说明：



**图5 m序列产生模块时序图**

1. 正弦载波信号产生模块

模块框图：



**图6 正弦波产生模块框图**

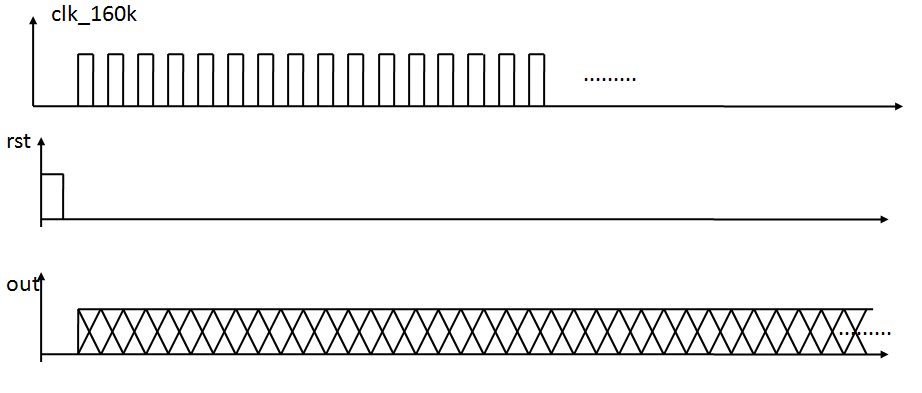
引脚说明：

clk\_160k ：160 kHz时钟输入

rst ：系统reset复位信号输入

out ：正弦波数字量输出（8位）

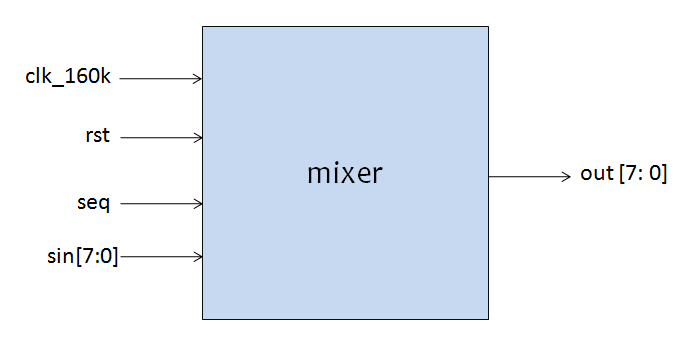
时序说明：



**图7 正弦波产生模块时序图**

1. 2ASK调制模块

模块框图：



**图8 2ASK调制模块框图**

引脚说明：

clk\_160k ：160 kHz时钟输入

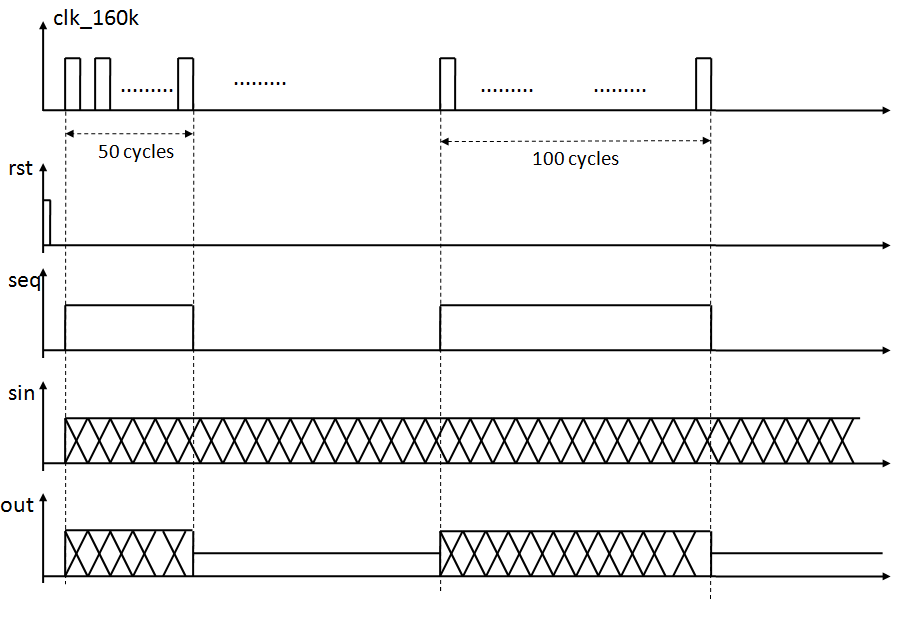
rst ：系统reset复位信号输入

seq ：待调制信号即m序列

sin ：正弦信号输入（8位）

out ：ASK输出（8位）

时序说明：



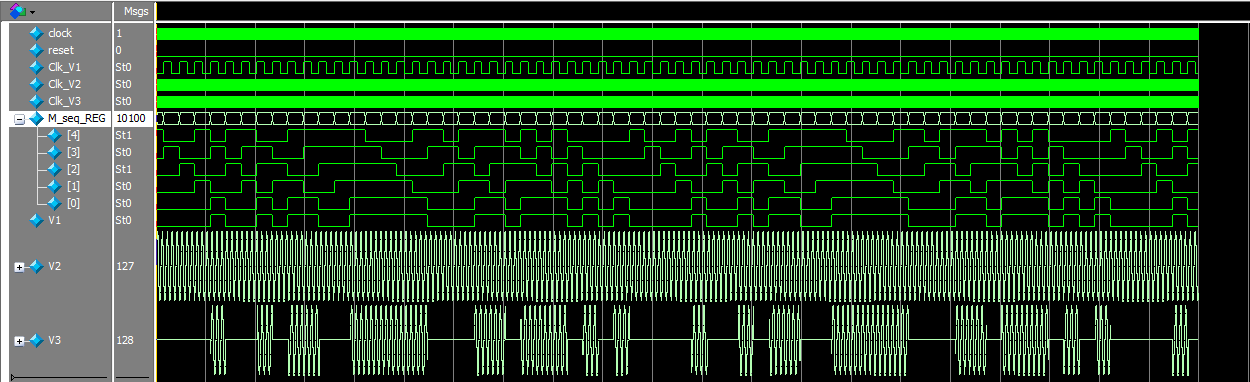
**图9 2ASK调制模块时序图**

# 综合及仿真

器件综合结果：

|  |  |
| --- | --- |
| Top-level Entity Name | Simple\_Communication\_Sys |
| Family | Cyclone II |
| Total logic elements | 60 |
| Total combinational functions | 56 |
| Dedicated logic registers | 45 |
| Total registers | 45 |
| Total pins | 27 |
| Total virtual pins | 0 |
| Total memory bits | 0 |
| Embedded Multiplier 9-bit elements | 0 |
| Total PLLs | 0 |

RTL仿真结果：



仿真结果分析：

由仿真结果可得出，Clk\_V1的输出为3.2kHz的方波，Clk\_V2的输出为160kHz的方波，Clk\_V3的输出为160kHz的方波。V1的输出为周期是31的m序列，V2的输出为10kHz的正弦波，V3输出2ASK波形。M\_seq\_REG输出的是m序列发生器内部的寄存信号。输出结果与预期结果一致。完成了设计任务的全部内容。

# 结论

根据器件综合结果和RTL仿真结果可知，该2ASK调制系统以极少的硬件资源实现了完整的2ASK系统功能。

# 附加说明

1. 本项目进行硬件描述所用的语言为SystemVerilog。
2. 本项目源码文件编码格式为UTF-8，在竞赛所用计算机的Quartus软件中能正确识别。不过为了防止在其他较旧电脑上出现显示问题，在源码压缩包中我们提供了不含中文注释的版本。
3. Quartus工程文件名为ask.qpf。

# 设计源码

// ======================= 顶层模块 =======================

/\*

\* 顶层模块

\* 功能：连接各个子模块

\* 输入：160kHz 的时钟信号，复位信号

\* 输出：各个模块所用的时钟信号，m 序列，m 序列的寄存器状态，正弦载波信号，已调信号

\*/

**module** Simple\_Communication\_Sys**(**

**input** logic clock**,**

**input** logic reset**,**

**output** logic Clk\_V1**,**

**output** logic Clk\_V2**,**

**output** logic Clk\_V3**,**

**output** logic **[**4**:**0**]** M\_seq\_REG**,**

**output** logic V1**,**

**output** logic **[**7**:**0**]** V2**,**

**output** logic **[**7**:**0**]** V3

**);**

// 内部时钟产生模块

clk\_div clk\_div0**(**

**.**rst**(**reset**),**

**.**clk\_32m**(**clock**),**

**.**clk\_160k**(**Clk\_V2**),**

**.**clk\_3k2**(**Clk\_V1**)**

**);**

always\_comb Clk\_V3 **=** Clk\_V2**;**

// m 序列产生模块

mseq mseq\_u0**(**

**.**clk\_3k2**(**Clk\_V1**),**

**.**rst**(**reset**),**

**.**out**(**V1**),**

**.**ireg**(**M\_seq\_REG**)**

**);**

// 正弦基带信号产生模块

sine sine\_u0**(**

**.**clk\_160k**(**Clk\_V2**),**

**.**rst**(**reset**),**

**.**out**(**V2**)**

**);**

// 2ASK 调制模块

mixer mixer\_u0**(**

**.**clk\_160k**(**Clk\_V2**),**

**.**rst**(**reset**),**

**.**seq**(**V1**),**

**.**sin**(**V2**),**

**.**out**(**V3**)**

**);**

endmodule

// ======================= 内部时钟产生模块 =======================

/\*

\* 内部时钟产生模块

\* 功能：通过计数分频，产生内部模块所需的各种频率的时钟

\* 输入：32MHz 的时钟信号，复位信号

\* 输出：160kHz 和 3.2kHz 的时钟信号

\*/

**module** clk\_div**(**

**input** logic rst**,** // 复位信号

**input** logic clk\_32m**,** // 32 MHz 时钟

**output** logic clk\_160k**,** // 160 kHZ 时钟

**output** logic clk\_3k2 // 3.2 kHz 时钟

**);**

**parameter** MAX0**=**13'd4999**,** MAX1**=**7'd99**;** // 计数器最大值

bit **[**12**:**0**]** acc0**;** // 计数器 0

bit **[** 6**:**0**]** acc1**;** // 计数器 1

/\*

\* 过程名：proc\_clk\_3k2

\* 类型：触发器逻辑，异步复位（高电平有效）

\* 功能：3.2kHz 时钟的计数器

\*/

always\_ff **@(posedge** clk\_32m **or** **posedge** rst**)** **begin** **:** proc\_clk\_3k2

**if(**rst**)** acc0 **<=** '0**;**

**else** acc0 **<=** **(**acc0 **==** MAX0**)** **?** '0 **:** acc0 **+** 1**;**

**end**

/\*

\* 过程名：proc\_clk\_160k

\* 类型：触发器逻辑，异步复位（高电平有效）

\* 功能：160kHz 时钟的计数器

\*/

always\_ff **@(posedge** clk\_32m **or** **posedge** rst**)** **begin** **:** proc\_clk\_160k

**if(**rst**)** acc1 **<=** '0**;**

**else** acc1 **<=** **(**acc1 **==** MAX1**)** **?** '0 **:** acc1**+**1**;**

**end**

/\*

\* 过程名：proc\_output

\* 类型：触发器逻辑，异步复位（高电平有效）

\* 功能：通过计数器的值来产生输出的时钟信号

\*/

always\_ff **@** **(posedge** clk\_32m **or** **posedge** rst**)** **begin** **:** proc\_output

**if(**rst**)** **begin**

clk\_160k **<=** 0**;**

clk\_3k2 **<=** 0**;**

**end** **else** **begin**

**if(**acc0 **==** MAX0**)** clk\_3k2 **<=** **!**clk\_3k2**;**

**if(**acc1 **==** MAX1**)** clk\_160k **<=** **!**clk\_160k**;**

**end**

**end**

endmodule

// ======================= m序列产生模块 =======================

/\*

\* m 序列产生模块

\* 功能：使用线性移位的方式，产生周期为 (2^5-1) = 31 的 m 序列

\* 输入：3.2kHz 的时钟信号，复位信号

\* 输出：m 序列，内部状态信号

\*/

**module** mseq**(**

**input** logic clk\_3k2**,** // 3.2kHz 时钟

**input** logic rst**,** // 复位信号

**output** logic **[**4**:**0**]** ireg**,** // 内部寄存器状态

**output** logic out // m 序列

**);**

/\*

\* 过程名：gen\_seq

\* 类型：触发器逻辑，异步复位（高电平有效）

\* 功能：控制产生 m 序列的寄存器的反馈

\*/

always\_ff **@** **(posedge** clk\_3k2 **or** **posedge** rst**)** **begin** **:** gen\_seq

**if(**rst**)** ireg **<=** 5'b10000**;**

**else** **begin**

ireg**[**4**]** **<=** ireg**[**0**]** **^** ireg**[**2**];**

ireg**[**3**:**0**]** **<=** ireg**[**4**:**1**];**

**end**

**end**

/\*

\* 类型：组合逻辑

\* 功能：将最后一个寄存器作为输出

\*/

always\_comb out **=** ireg**[**0**];**

endmodule

// ======================= 正弦基带信号产生模块 =======================/\*

\* 正弦基带信号产生模块

\* 功能：产生频率为 10kHz 的正弦信号（采样频率为 160 kHz），作为载波

\* 输入：160kHz 的时钟信号，复位信号

\* 输出：正弦信号（8位，无符号整数）

\*/

**module** sine**(**

**input** logic clk\_160k**,** // 160kHz 时钟

**input** logic rst**,** // 复位信号

**output** logic **[**7**:**0**]** out // 正弦值

**);**

bit **[**3**:**0**]** index**;** // 采样值的编号（0 到 15）

const bit **[**0**:**15**][**7**:**0**]** sin\_val **=** **{** // 正弦信号在一个周期内的 16 个采样值

8'd128**,**

8'd179**,**

8'd222**,**

8'd249**,**

8'd254**,**

8'd238**,**

8'd202**,**

8'd154**,**

8'd101**,**

8'd53**,**

8'd17**,**

8'd1**,**

8'd6**,**

8'd33**,**

8'd76**,**

8'd127

**};**

/\*

\* 过程名：change\_index

\* 类型：触发器逻辑，异步复位（高电平有效）

\* 功能：在每个时钟上升沿，将采样值的编号加一

\*/

always\_ff **@** **(posedge** clk\_160k **or** **posedge** rst**)** **begin** **:** change\_index

**if(**rst**)** index **<=** 0**;**

**else** index **<=** index **+** 1**;** // ok(overflow)

**end**

/\*

\* 过程名：set\_value

\* 类型：触发器逻辑，异步复位（高电平有效）

\* 功能：在每个时钟上升沿，将新的正弦信号值锁存到输出端口

\*/

always\_ff **@** **(posedge** clk\_160k **or** **posedge** rst**)** **begin** **:** set\_value

**if(**rst**)** out **<=** sin\_val**[**0**];**

**else** out **<=** sin\_val**[**index**];**

**end**

endmodule

// ======================= 2ASK调制模块 =======================/\*

/\*

\* 2ASK 调制模块

\* 功能：将基带信号调制到正弦载波上

\* 输入：160kHz 的时钟信号，复位信号，正弦信号值，待调制的基带数字信号

\* 输出：2ASK 已调信号（8位，无符号整数）

\*/

**module** mixer**(**

**input** logic clk\_160k**,** // 160kHz 时钟

**input** logic rst**,** // 复位信号

**input** logic seq**,** // 基带数字信号

**input** logic **[**7**:**0**]** sin**,** // 正弦载波信号

**output** logic **[**7**:**0**]** out // 已调信号

**);**

const bit **[**7**:**0**]** sin\_zero **=** 128**;** // 正弦 0 值

bit **[**7**:**0**]** mixed**;**

/\*

\* 过程名：mix

\* 类型：组合逻辑

\* 功能：用基带信号来控制正弦幅值

\*/

always\_comb **begin** **:** mix

**if(**seq**)** mixed **=** sin**;**

**else** mixed **=** sin\_zero**;**

**end**

/\*

\* 过程名：buff

\* 类型：触发器逻辑，异步复位（高电平有效）

\* 功能：锁存已调信号

\*/

always\_ff **@** **(posedge** clk\_160k **or** **posedge** rst**)** **begin** **:** buff

**if(**rst**)** out **<=** sin\_zero**;**

**else** out **<=** mixed**;**

**end**

endmodule