实 验 报 告

课程名称：	数字系统设计
学生姓名：	陈永炜 **201630257370**
学生姓名：	邢靖 **201630258803**
学生姓名：	敖梓茗 **201630257165**
学生姓名：	蒲尧 **201630258438**
学生专业：	信息工程
开课学期：	**2018——2019** 第一学期

电子与信息学院

2018 年 9 月 制

目录

交织器的设计

一、设计卷积交织器的目的

在数字传输系统中，因为存在噪声、信道衰落等干扰因素，会使传输的信号发生错误，产生误码。虽然数字信号的传输为了防止误码而会进行信道编码，增加传输码的冗余，例如增加监督位等来克服信号在信道传输过程中的错误，但这种检错纠错能力是有限的。例如当出现突发错误，出现大片连续误码时，这时信道的纠错是无能为力的。而卷积交织器可以将原来的信息码打乱，这时尽管出现大面积突发性错误，这些可以通过解交织器来进行分散，从而将大面积的错误较为平均地分散到不同的码段，利于信道纠错的实现。

二、卷积交织的原理

卷积交织的原理其实是编码与解码，实际表现是“延迟”。发送端使用交织器，对原信号序列通过某种逻辑进行打乱后发送，接收端通过反逻辑对打乱后的信息序列进行重新排列，恢复出原信号序列。

解交织的结果是恢复出来的序列和原信号序列内容一致，但有一个延迟，体现在恢复出来的信号序列开头有若干个“0”，“0”的个数取决于交织器的规格。

2.1 卷积交织器的实现

卷积交织有两种实现方法，一种是直接使用移位寄存器，另一种是使用 RAM 来实现类似于移位寄存器的功能。

假设要发送的序列为 ABCDEFGHIXXX……，X 表示无效信息，示例使用三支路、延迟 M=1 的移位寄存器实现，原理图如下图 1：

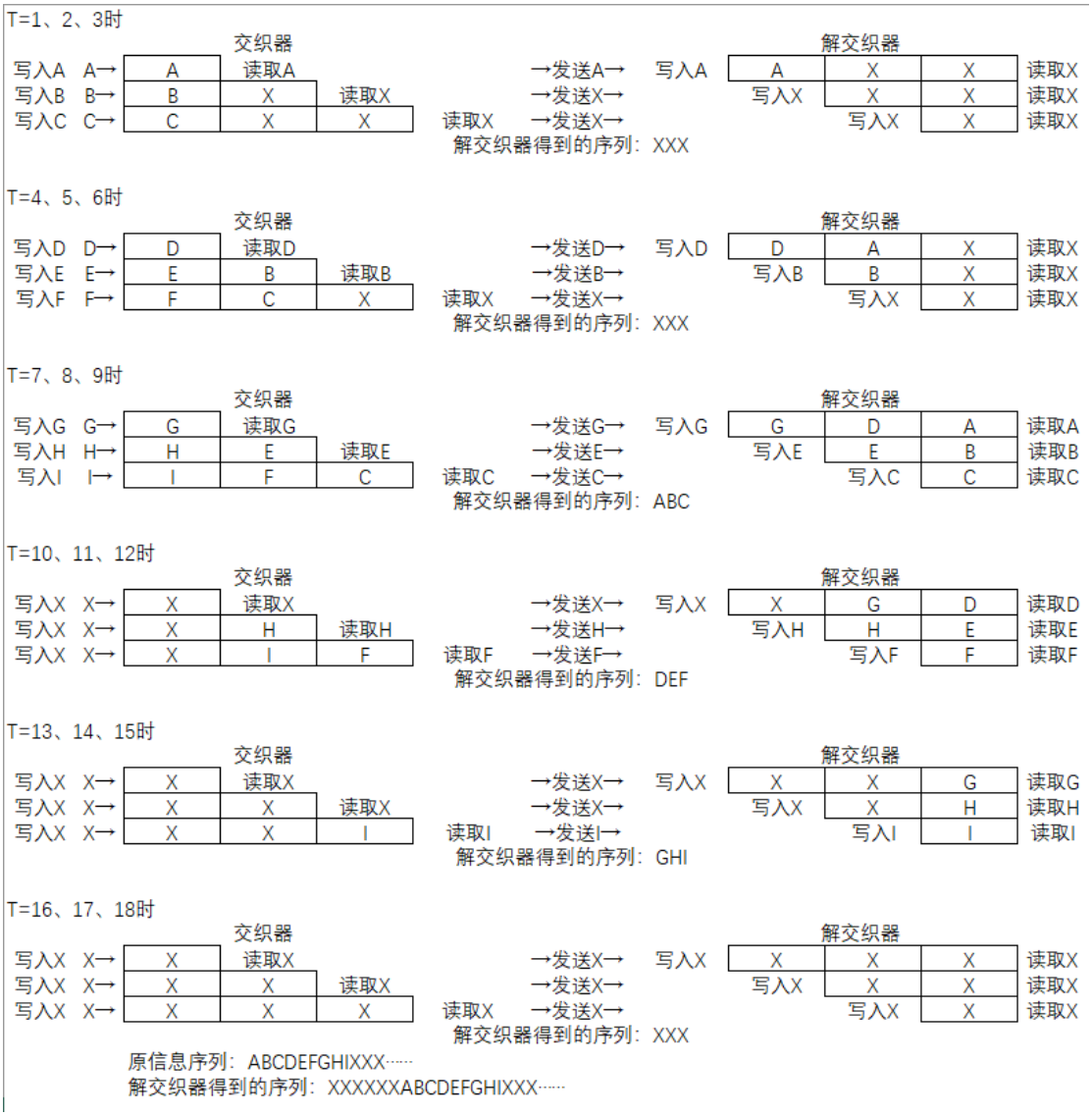


图 1 卷积交织原理

由结果可见，原信息序列 ABCDEFGHIXXX……经过交织器后序列被打乱，而经过解交织器后恢复序列包含了原信号序列的全部信息，体现了编码与解码的原理。而恢复序列和原信息序列相比，序列前端多了“XXXXXX……”，体现了交织器延迟的特性。利用移位寄存器在硬件上实现延时虽然在理论上很简单，需要延时 T 个单位只需要 T 级寄存器；但当系统变得庞大，需要的交织器的支路数、延时数变大时，需要的移位寄存器也会相应地增多，而实验板不一定能提供这么多的资源。而且每个时钟的到来，都会有大量存在于移位寄存器中的数据移动，会使整个系统的功耗剧烈升高。为了克服移位寄存器的缺点，我们选择用 RAM 来模拟移位寄存器的功能。定义一个长度为 4 的 RAM，模拟移位寄存器的功能如下：

例如对信息序列“ABCD”进行延时：

T=1 时，将 A 数据写入 1 单元，读出 2 单元数据，结果为空。

T=2 时，将 B 数据写入 2 单元，读出 3 单元数据，结果为空。

T=3 时，将 C 数据写入 3 单元，读出 4 单元数据，结果为空。

T=4 时，将 D 数据写入 4 单元，读出 1 单元数据，结果为 A。

刚好延迟了 4 个时钟。如此循环，序列“ABCD”就延迟了 4 个时钟后输出，和 4

级移位寄存器功能相同，而使用 RAM 只涉及数据的读和写，克服了移位寄存器大量数据移位的缺点，因此利用 RAM 来模拟移位寄存器所需要的功耗则小得多，所以我们最终选择用 RAM 来实现交织器和解交织器的功能。

所以本系统使用 ram 来实现交织器与解交织器。具体来说，就是将 RAM 分成 12 路，第一路无延时。第二路延时 M 时间，需要 M+1 个存储单元。第三路延时 2M 时间，需要 2M+1 个存储单元。第四路延时 3M 时间，需要 3M+1 个存储单元。如此类推，第 i 路延时（i-1）M 时间，需要（i-1）M+1 个寄存器；而解交织器相反，第十二路无延时。第十一路延时 M，第十路延时 2M，如此类推。下面介绍数据的流动，以交织器为例，每个时钟，数据进入不同的 RAM 支路，我们可以设置一个 countram（1 至 12）进行循环计数，每个时钟 countram 就加 1，数据进入第 countram 路。确定了数据进入 RAM 的哪一支路后，还要确定数据进入这路 RAM 中的哪一个储存单元，故 RAM 的每个支路都要一个设置一个计数器 coun1-count12，来控制读写哪一个单元，利用上面讨论的 RAM 移位法实现延时，读的地址要比写的地址靠前一位，因为数据读出的时候同时要写入数据，所以 RAM 要用双口 RAM 实现。

三、模块设计

基于上述原理，下面我们设计交织深度 B=12，符号周期 M=17，采用 RAM 法实现的交织/解交织器。

3.1 交织器的设计

交织器的设计首先需要产生读写地址，根据上述的读写地址，我们可以选用 RAM 中相应的寄存器进行读写，由于 RAM 在某个时间只能完成一个读或写的操作，因此需要读写控制模块控制 RAM 的读写信号，最后再将各个模块组合起来得到交织器。

综上所述，交织器的设计可分为读写地址模块，读写控制模块，RAM 模块和总的综合模块。

读写地址模块

根据交织器的原理，若在当前时钟周期在第 i 条支路的第 k 个地址写入数据，则同时需要在第 k+1 个地址读出数据，而后下一个时钟周期需要在第 i+1 条支路上进行操作。操作如下图 2 所示。

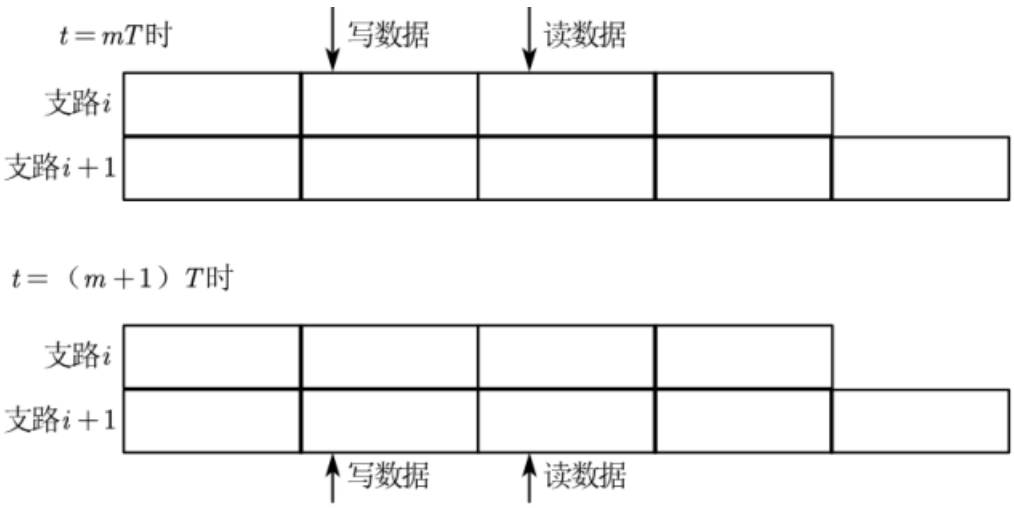


图 2 读写地址模块

当写数据的地址为该条支路的最末位时，读数据的地址变为该条支路的首位。由于设计的交织深度为 12，符号周期为 17，因此支路共有 12 条，每条支路的地址数比上一条支路增加 17。

在各个通道中，第 0 通道无延时；第 1 通道延时 17*1 个时钟周期；第 2 通道延时 17*2 个时钟周期..... 依此类推，第 11 通道延时 17*11 个时钟周期。总共所需存储单元数为 1+ 18+ ....+188=1134，相应的要用到地址总线为 11b。也就是说要用到 2k 的 RAM。

总结为数学计算公式，设第 i 通道的基地址为 bi(baseaddress)，尾地址为 ci，各通道的变址为 ai，则 RAM 的读写地址的变化规律为：第 i 通道读地址:

$$rd\_add = ai + bi \quad (1)$$

第 i 通道写地址:

$$wr\_add = ai + bi - 1;\ 当\ ai \neq 0 \quad (2)$$

$$wr\_add = ci;当\ ai = 0; \quad (3)$$

①交织器读写地址模块的代码如下：

```
------------------------------------------------------------------
--read & write address of interleaver module
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity AddrGen is
port(clk:in std_logic;
    AddrR:out integer range 0 to 2047;
    AddrW:out integer range 0 to 2047
    );
end AddrGen;

architecture behav of AddrGen is
    signal count:integer range 0 to 11:=0;
```

```
begin

--count process
CNT:process(clk)
variable c:integer range 0 to 11:=0;
begin
    if(rising_edge(clk)) then
        c:=(c+1)mod 12;
    end if;
    count<=c;
end process;

--address process
--ai:change address,bi:base address,ci:tail address
--AddrR<=ai+bi;
--AddrW<=ai+bi-1;when ai isn't 0;
--AddrW<=ci;when ai is 0;

addr:process(clk)
    constant a0:integer:=0;
    constant a1:integer:=1;
    constant c1:integer:=18;
    constant a2:integer:=19;
    constant c2:integer:=53;
    constant a3:integer:=54;
    constant c3:integer:=105;
    constant a4:integer:=106;
    constant c4:integer:=174;
    constant a5:integer:=175;
    constant c5:integer:=260;
    constant a6:integer:=261;
    constant c6:integer:=363;
    constant a7:integer:=364;
    constant c7:integer:=483;
    constant a8:integer:=484;
    constant c8:integer:=620;
    constant a9:integer:=621;
    constant c9:integer:=774;
    constant a10:integer:=775;
    constant c10:integer:=945;
    constant a11:integer:=946;
    constant c11:integer:=1133;
    variable b1:integer range 0 to 17:=0;
    variable b2:integer range 0 to 34:=0;
```

```
    variable b3:integer range 0 to 51:=0;
    variable b4:integer range 0 to 68:=0;
    variable b5:integer range 0 to 85:=0;
    variable b6:integer range 0 to 102:=0;
    variable b7:integer range 0 to 119:=0;
    variable b8:integer range 0 to 136:=0;
    variable b9:integer range 0 to 153:=0;
    variable b10:integer range 0 to 170:=0;
    variable b11:integer range 0 to 187:=0;

begin

if(rising_edge(clk)) then
    case count is
    when 0=> AddrW<=a0;AddrR<=a0;
    when 1=>
        b1:=(b1+1)mod 18;
        if(b1=0) then
        AddrW<=c1;
        AddrR<=a1;
        else
        AddrR<=a1+b1;
        AddrW<=a1+b1-1;
        end if;
    when 2=>
        b2:=(b2+1)mod 35;
        if(b2=0) then
        AddrW<=c2;
        AddrR<=a2;
        else
        AddrR<=a2+b2;
        AddrW<=a2+b2-1;
        end if;
    when 3=>
        b3:=(b3+1)mod 52;
        if(b3=0) then
        AddrW<=c3;
        AddrR<=a3;
        else
        AddrR<=a3+b3;
        AddrW<=a3+b3-1;
        end if;
    when 4=>
        b4:=(b4+1)mod 69;
```

```
        if(b4=0) then
        AddrW<=c4;
        AddrR<=a4;
        else
        AddrR<=a4+b4;
        AddrW<=a4+b4-1;
        end if;
    when 5=>
        b5:=(b5+1)mod 86;
        if(b5=0) then
        AddrW<=c5;
        AddrR<=a5;
        else
        AddrR<=a5+b5;
        AddrW<=a5+b5-1;
        end if;
    when 6=>
        b6:=(b6+1)mod 103;
        if(b6=0) then
        AddrW<=c6;
        AddrR<=a6;
        else
        AddrR<=a6+b6;
        AddrW<=a6+b6-1;
        end if;
    when 7=>
        b7:=(b7+1)mod 120;
        if(b7=0) then
        AddrW<=c7;
        AddrR<=a7;
        else
        AddrR<=a7+b7;
        AddrW<=a7+b7-1;
        end if;
    when 8=>
        b8:=(b8+1)mod 137;
        if(b8=0) then
        AddrW<=c8;
        AddrR<=a8;
        else
        AddrR<=a8+b8;
        AddrW<=a8+b8-1;
        end if;
    when 9=>
```

```
        b9:=(b9+1)mod 154;
        if(b9=0) then
        AddrW<=c9;
        AddrR<=a9;
        else
        AddrR<=a9+b9;
        AddrW<=a9+b9-1;
        end if;
    when 10=>
        b10:=(b10+1)mod 171;
        if(b10=0) then
        AddrW<=c10;
        AddrR<=a10;
        else
        AddrR<=a10+b10;
        AddrW<=a10+b10-1;
        end if;
    when 11=>
    b11:=(b11+1)mod 188;
        if(b11=0) then
        AddrW<=c11;
        AddrR<=a11;
        else
        AddrR<=a11+b11;
        AddrW<=a11+b11-1;
        end if;
    end case;
end if;
end process;
end behav;
-------------------------------------------------------------------------
```

在 modelsim 中对该模块进行仿真，运行结果图如下图 3：

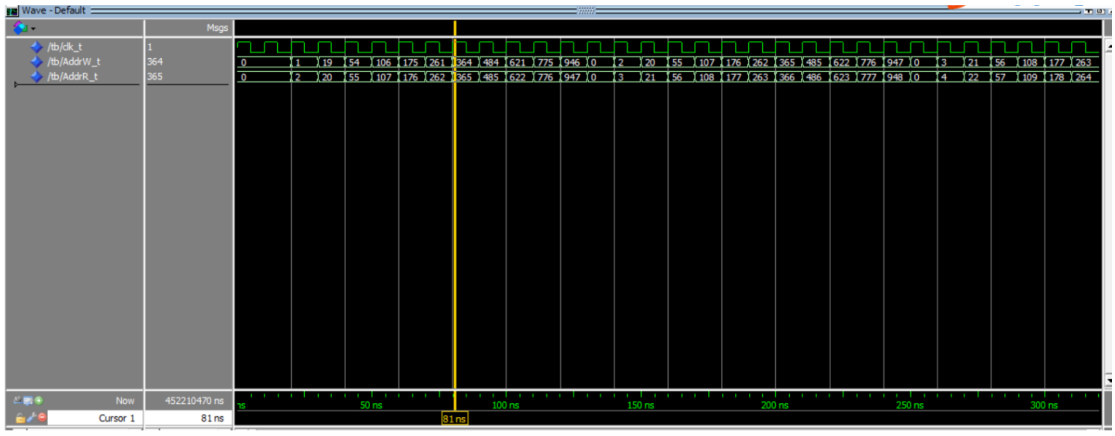


图 3 交织器读写地址

可以看出读地址依次为 0，1,19,54,106,175,261…,

而写地址依次为 0,2,20,55,107,176,262…仿真结果与预期相吻合。

②读写控制模块

由前所述，我们需要在一个时钟周期内完成读写操作，由于 RAM 一个时间段内仅能完成一个操作，因此需要一个读写控制模块产生相应的读写控制信号，以便 RAM 完成读写控制操作。

在这里，我们在时钟信号的零电平段进行写操作，在 1 电平段进行读操作。该模块代码如下：

```
----------------------------------------------------------------------
--read & write control module
library ieee;
use ieee.std_logic_1164.all;
entity WRCon is
port(clk:in std_logic;
    WE,RE:out std_logic);
end entity;
architecture behav of WRcon is
begin
    WE<=not clk;
    RE<=clk;
end behav;
----------------------------------------------------------------------
```

在 modelsim 中进行仿真，实验结果如下图 4

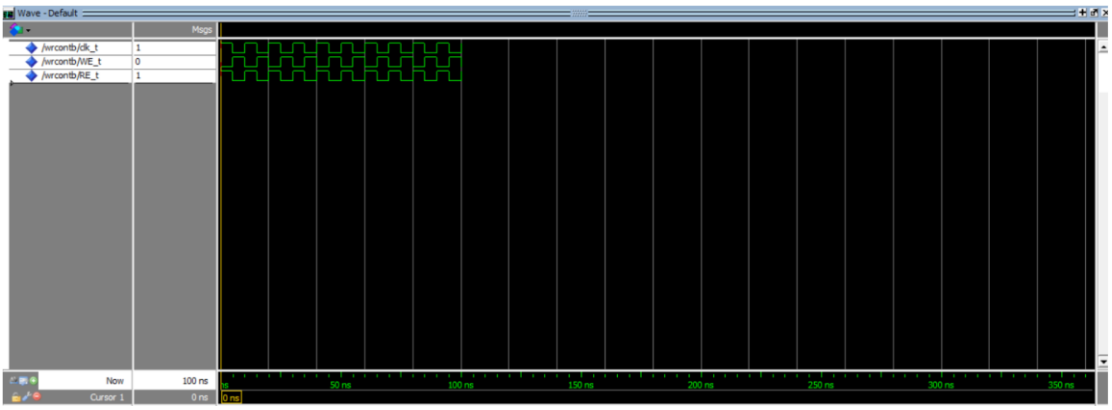


图 4 交织器读写控制

可以看到读写信号的有效时间不相重合，有效避免了读写冲突。

③RAM 模块

将读写信号和地址输入到 RAM 模块中，就可以对 RAM 中相应的地址进行读写操作。由于 B=12，M=17，因此总的地址单元为 B（B-1）M=2244 个，同时我们采用 8 位的数据进行读写，因此确定 RAM 个规格为 2k*8。RAM 模块的代码如下：

```
------------------------------------------------------------------
--ram module
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity RAM is
port(AddrR:in integer range 0 to 2047;
    AddrW:in integer range 0 to 2047;
    DataIn:in std_logic_vector(7 downto 0);
    DataOut:out std_logic_vector(7 downto 0);
    CS,RE,WE:in std_logic);
end RAM;
architecture behav of RAM is
begin
process(AddrW,RE,WE)
type ram_array is array(0 to 2047) of std_logic_vector(7 downto 0);
variable mem:ram_array;
begin
if(CS='0') then
    if(RE='0') then DataOut<=mem(AddrR);
    elsif(WE='0') then mem(AddrW):=DataIn;
    end if;
end if;
end process;
end behav;
------------------------------------------------------------------
```

④数据锁存模块

为了防止信息丢失和竞争冒险，我们在 RAM 的输出端加了数据锁存器，代码如下

```
------------------------------------------------------------------
--data lock module
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity DataLock is
port(Data:in std_logic_vector(7 downto 0);
    DataLock:out std_logic_vector(7 downto 0);
    RE:in std_logic);
end entity;
architecture behav of DataLock is
begin
process(Data,RE)
begin
```

```
if(RE='0') then DataLock<=Data;
end if;
end process;
end behav;
-------------------------------------------------------------
```

⑤交织器模块（调用①②③④）

最后将上述模块连接起来合成一个总的交织器模块，代码如下

```
-------------------------------------------------------------
--InterLeaver module
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity InterLeaver is
port(clk:in std_logic;
    DataIn:in std_logic_vector(7 downto 0);
    DataOut:out std_logic_vector(7 downto 0));
end entity;

architecture stru of InterLeaver is
    --read & write address of interleaver module
    component AddrGen is
    port(clk:in std_logic;
        AddrR:out integer range 0 to 2047;
        AddrW:out integer range 0 to 2047
        );
    end component;
    --ram module
    component RAM is
    port(AddrR:in integer range 0 to 2047;
        AddrW:in integer range 0 to 2047;
        DataIn:in std_logic_vector(7 downto 0);
        DataOut:out std_logic_vector(7 downto 0);
        CS,RE,WE:in std_logic);
    end component;
    --read & write control module
    component WRcon is
    port(clk:in std_logic;
        WE,RE:out std_logic);
    end component;
    --data lock module
    component DataLock is
```

```
    port(Data:in std_logic_vector(7 downto 0);
        DataLock:out std_logic_vector(7 downto 0);
        RE:in std_logic);
    end component;
signal adr,adw:integer range 0 to 2047;
signal we,re:std_logic;
signal d:std_logic_vector(7 downto 0);
begin
    u0:AddrGen port map(clk,adr,adw);
    u1:WRcon port map(clk,we,re);
    u2:RAM port map(adr,adw,DataIn,d,'0',re,we);
    u3:DataLock port map(d,DataOut,re);
end stru;
-------------------------------------------------------------------
```

为了进行仿真验证，我们将输入数据设为从 00000000 递增到 11111111 的序列，在 modelsim 中仿真时设置数据输出格式为 10 进制，观察其输出。仿真结果图如下：

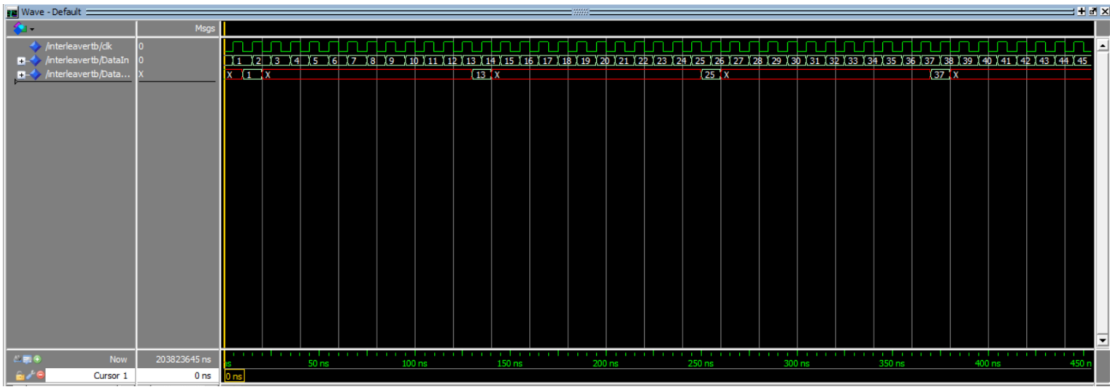


图 5 开始仿真时结果

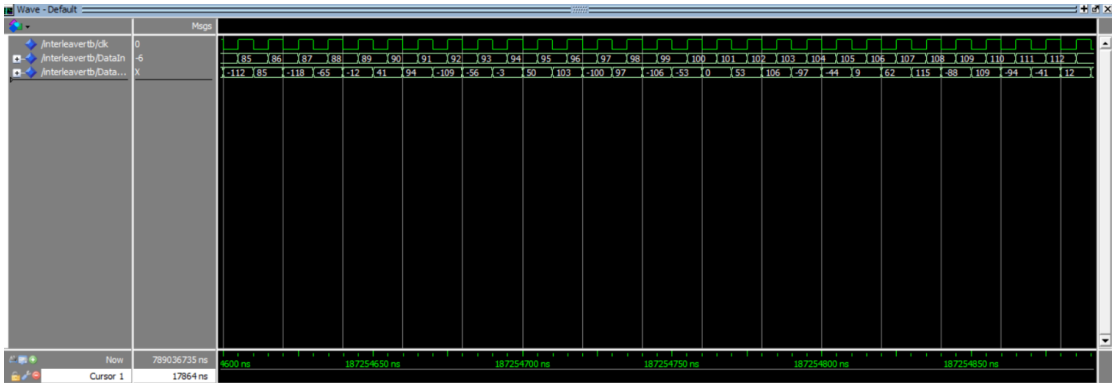


图 6 中间仿真时结果

可以看到在开始的一段时间内仅有第一条支路（即无延时的支路）有输出，其他支路由于还未写入数据，无输出。过了一段时间后，每条支路都有输出，输出的序列为乱序。

3.2 时钟延时模块

⑥时钟延时模块

该部分是在连接交织器和解交织器时所用到的延时模块，产生的延时时钟信号输送给解交织器。代码如下：

```
----------------------------------------------------------------
--ClockDelay module
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity ClockDelay is
port(clk:in std_logic;
    clk_de:out std_logic);
end ClockDelay;
architecture behav of ClockDelay is
signal c,d:std_logic:='0';
begin
clk_de<=clk and d;
process(clk)
begin
if(clk'event and clk='1') then
    if(c='0') then
        c<='1';
    end if;
elsif(clk'event and clk='0') then
    if(c='1') then
        d<='1';
    end if;
end if;
end process;
end behav;
----------------------------------------------------------------
```

3.3 解交织器

解交织器与交织器的原理相同，为了实现正确译码，经过交织器和解交织器的数据的延时应该相同，才能按顺序输出序列，因此解交织器的支路应该与交织器实现延时互补，具体实现方式如下图 7：

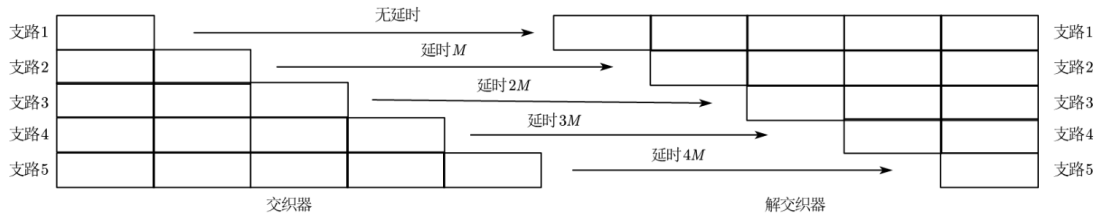


图 7 交织解交织关系

⑦解交织读写地址模块

由于解交织器可以看成是交织器的逆，实现方式二者相同，可以复用交织器的模块，此处二者相同的模块部分不再赘述，仅描述解交织器中读写地址模块。这部分的读写地址方式与交织器刚好相反，具体代码如下：

```
-----------------------------------------------------------------
--read & write address of deinterleaver module
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity DeAddrGen is
port(clk:in std_logic;
    AddrR:out integer range 0 to 2047;
    AddrW:out integer range 0 to 2047
    );
end DeAddrGen;
architecture behav of DeAddrGen is
    signal count:integer range 0 to 11:=0;
begin

--count process
C:process(clk)
variable c:integer range 0 to 11:=0;
begin
    if(rising_edge(clk)) then
        c:=(c+1)mod 12;
    end if;
    count<=c;
end process;

addr:process(clk)
    constant a11:integer:=0;
    constant a10:integer:=1;
    constant c10:integer:=18;
    constant a9:integer:=19;
    constant c9:integer:=53;
    constant a8:integer:=54;
    constant c8:integer:=105;
```

```
    constant a7:integer:=106;
    constant c7:integer:=174;
    constant a6:integer:=175;
    constant c6:integer:=260;
    constant a5:integer:=261;
    constant c5:integer:=363;
    constant a4:integer:=364;
    constant c4:integer:=483;
    constant a3:integer:=484;
    constant c3:integer:=620;
    constant a2:integer:=621;
    constant c2:integer:=774;
    constant a1:integer:=775;
    constant c1:integer:=945;
    constant a0:integer:=946;
    constant c0:integer:=1133;
    variable b10:integer range 0 to 17:=0;
    variable b9:integer range 0 to 34:=0;
    variable b8:integer range 0 to 51:=0;
    variable b7:integer range 0 to 68:=0;
    variable b6:integer range 0 to 85:=0;
    variable b5:integer range 0 to 102:=0;
    variable b4:integer range 0 to 119:=0;
    variable b3:integer range 0 to 136:=0;
    variable b2:integer range 0 to 153:=0;
    variable b1:integer range 0 to 170:=0;
    variable b0:integer range 0 to 187:=0;
begin
if(rising_edge(clk)) then
    case count is
    when 0=>
        b0:=(b0+1)mod 188;
        if(b0=0) then
        AddrW<=c0;
        AddrR<=a0;
        else
        AddrR<=a0+b0;
        AddrW<=a0+b0-1;
        end if;
    when 1=>
        b1:=(b1+1)mod 171;
        if(b1=0) then
        AddrW<=c1;
        AddrR<=a1;
```

```
        else
        AddrR<=a1+b1;
        AddrW<=a1+b1-1;
        end if;
    when 2=>
        b2:=(b2+1)mod 154;
        if(b2=0) then
        AddrW<=c2;
        AddrR<=a2;
        else
        AddrR<=a2+b2;
        AddrW<=a2+b2-1;
        end if;
    when 3=>
        b3:=(b3+1)mod 137;
        if(b3=0) then
        AddrW<=c3;
        AddrR<=a3;
        else
        AddrR<=a3+b3;
        AddrW<=a3+b3-1;
        end if;
    when 4=>
        b4:=(b4+1)mod 120;
        if(b4=0) then
        AddrW<=c4;
        AddrR<=a4;
        else
        AddrR<=a4+b4;
        AddrW<=a4+b4-1;
        end if;
    when 5=>
        b5:=(b5+1)mod 103;
        if(b5=0) then
        AddrW<=c5;
        AddrR<=a5;
        else
        AddrR<=a5+b5;
        AddrW<=a5+b5-1;
        end if;
    when 6=>
        b6:=(b6+1)mod 86;
        if(b6=0) then
        AddrW<=c6;
```

```
        AddrR<=a6;
        else
        AddrR<=a6+b6;
        AddrW<=a6+b6-1;
        end if;
    when 7=>
        b7:=(b7+1)mod 69;
        if(b7=0) then
        AddrW<=c7;
        AddrR<=a7;
        else
        AddrR<=a7+b7;
        AddrW<=a7+b7-1;
        end if;
    when 8=>
        b8:=(b8+1)mod 52;
        if(b8=0) then
        AddrW<=c8;
        AddrR<=a8;
        else
        AddrR<=a8+b8;
        AddrW<=a8+b8-1;
        end if;
    when 9=>
        b9:=(b9+1)mod 35;
        if(b9=0) then
        AddrW<=c9;
        AddrR<=a9;
        else
        AddrR<=a9+b9;
        AddrW<=a9+b9-1;
        end if;
    when 10=>
        b10:=(b10+1)mod 18;
        if(b10=0) then
        AddrW<=c10;
        AddrR<=a10;
        else
        AddrR<=a10+b10;
        AddrW<=a10+b10-1;
        end if;
    when 11=>
        AddrW<=a11;AddrR<=a11;
    end case;
```

```
end if;
end process;
end behav;
-------------------------------------------------------------------
```

⑧解交织器模块（调用②③④⑦）

将 4 个模块集成在一个实体中得到解交织器模块，代码如下：

```
-------------------------------------------------------------------
-- DeInterLeaver module
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity DeInterLeaver is
port(clk:in std_logic;
    DataIn:in std_logic_vector(7 downto 0);
    DataOut:out std_logic_vector(7 downto 0));
end entity;
architecture stru of DeInterLeaver is
    --read & write address of deinterleaver module
    component DeAddrGen is
    port(clk:in std_logic;
        AddrR:out integer range 0 to 2047;
        AddrW:out integer range 0 to 2047
        );
    end component;
    --ram module
    component RAM is
    port(AddrR:in integer range 0 to 2047;
        AddrW:in integer range 0 to 2047;
        DataIn:in std_logic_vector(7 downto 0);
        DataOut:out std_logic_vector(7 downto 0);
        CS,RE,WE:in std_logic);
    end component;
    --read & write control module
    component WRcon is
        port(clk:in std_logic;
        WE,RE:out std_logic);
    end component;
    --data lock module
    component DataLock is
        port(Data:in std_logic_vector(7 downto 0);
        DataLock:out std_logic_vector(7 downto 0);
        RE:in std_logic);
```

```
    end component;
signal adr,adw:integer range 0 to 2047;
signal we,re:std_logic;
signal d:std_logic_vector(7 downto 0);
begin
  u0:DeAddrGen port map(clk,adr,adw);
  u1:WRcon port map(clk,we,re);
  u2:RAM port map(adr,adw,DataIn,d,'0',re,we);
  u3:DataLock port map(d,DataOut,re);
end stru;
--------------------------------------------------------------------
```

3.4 交织解交织综合模块

⑨交织解交织综合模块（调用⑤⑥⑧）

这里将交织、延时、解交织连接在一起，代码如下：

```
--------------------------------------------------------------------
-- InterLeaver & DeInterleaver synthetical Model
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity In_DeModel is
port(clk:in std_logic;
    DataIn:in std_logic_vector(7 downto 0);
    DataOut:Out std_logic_vector(7 downto 0));
end entity;
architecture stru of In_DeModel is
    --InterLeaver module
    component InterLeaver is
    port(clk:in std_logic;
        DataIn:in std_logic_vector(7 downto 0);
        DataOut:out std_logic_vector(7 downto 0));
    end component;
    component DeInterLeaver is
    port(clk:in std_logic;
        DataIn:in std_logic_vector(7 downto 0);
        DataOut:out std_logic_vector(7 downto 0));
    end component;
    component ClockDelay is
    port(clk:in std_logic;
        clk_de:out std_logic);
    end component;
```

```
    signal d:std_logic_vector(7 downto 0);
    signal clk_de:std_logic;
begin
    u0:InterLeaver port map(clk,DataIn,d);
    u1:ClockDelay port map(clk,clk_de);
    u2:DeInterLeaver port map(clk_de,d,DataOut);
end stru;
-------------------------------------------------------------------
```

⑩交织解交织综合模块 TestBench

产生时钟和输入信号，代码如下：

```
-------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

ENTITY In_DeModel_vhd_tst IS
END In_DeModel_vhd_tst;
ARCHITECTURE In_DeModel_arch OF In_DeModel_vhd_tst IS
-- constants
-- signals
SIGNAL clk : STD_LOGIC;
SIGNAL DataIn : STD_LOGIC_VECTOR(7 DOWNTO 0):="00000000";
SIGNAL DataOut : STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL DataOut1 : STD_LOGIC_VECTOR(7 DOWNTO 0);
COMPONENT In_DeModel
    PORT (
    clk : IN STD_LOGIC;
    DataIn : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    DataOut : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
    );
END COMPONENT;

component InterLeaver
port(clk:in std_logic;
    DataIn:in std_logic_vector(7 downto 0);
    DataOut:out std_logic_vector(7 downto 0));
end component;

BEGIN
    i1 : In_DeModel
    PORT MAP (
    clk => clk,
```

```
    DataIn => DataIn,
    DataOut => DataOut
    );
    i2 : InterLeaver
    port map(
    clk => clk,
    DataIn => DataIn,
    DataOut => DataOut1
);
--clock signal
clk_gen: PROCESS
constant period:time:=20 ns;
BEGIN
    clk<='0';
    wait for period/2;
    clk<='1';
    wait for period/2;
END PROCESS clk_gen;
--input signal
DataIn_gen:process(clk)
begin
if(rising_edge(clk))then
    DataIn<=DataIn+'1';
end if;
end process DataIn_gen;

END In_DeModel_arch;
--------------------------------------------------------------------
```

为了验证结果是否正确，我们在 modelsim 中将交织器和解交织器相连，输入数据为 00000000 到 11111111 的循环递增序列，观察交织器和解交织器的输出。仿真结果如下：

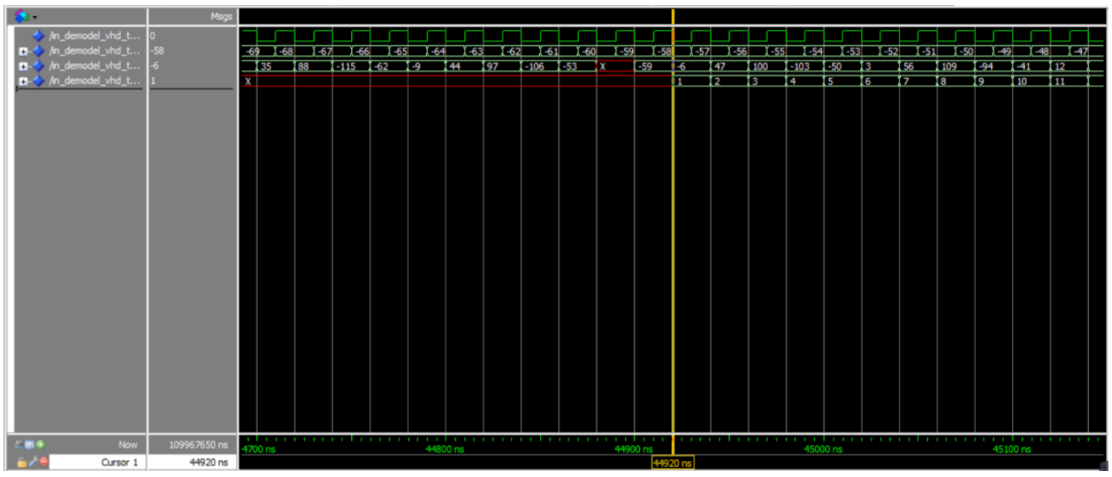
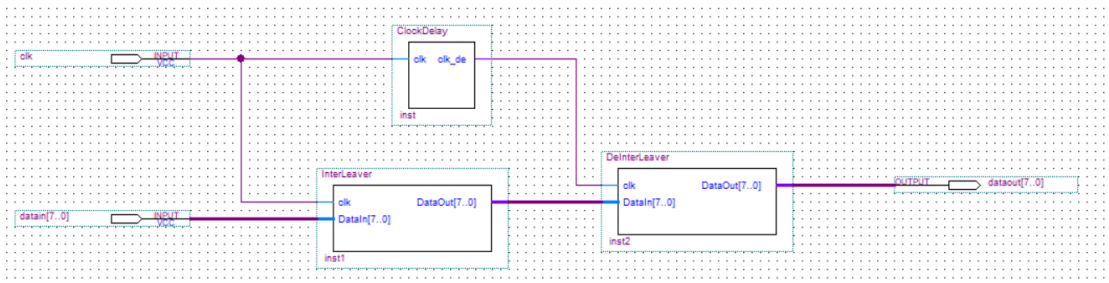


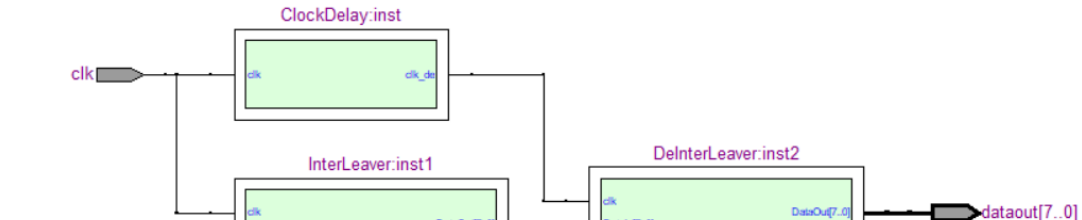
图 8 综合交织解交织

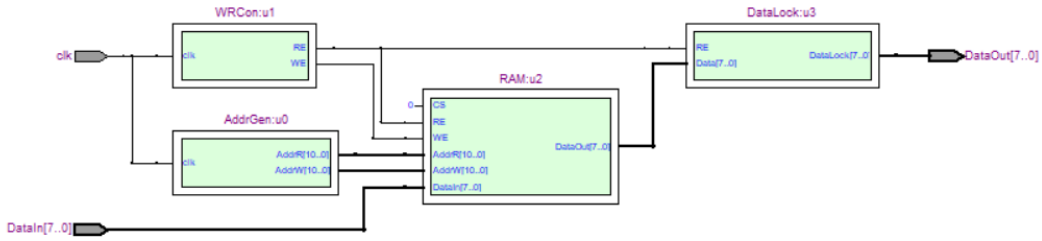
可以看到在一定的延时以后才有原序列的输出。由于每个时钟周期为 20ns，因此该延时一定大于 12*11*17*20=44880ns，由于延时器的存在，实际输出还要再往后延时 40ns，总数为 44920ns，与实验结果相一致。

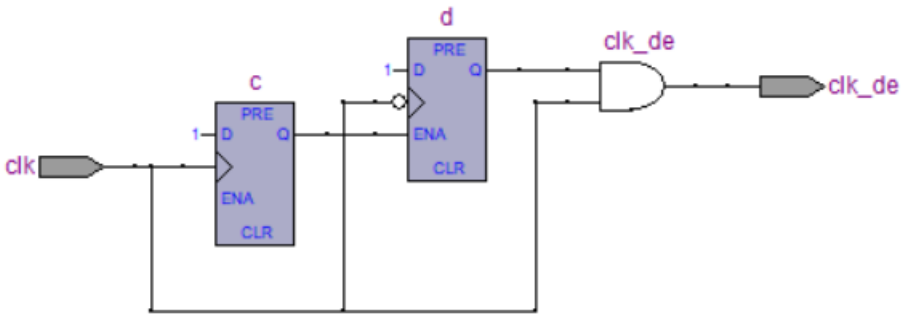
电路图如下（由于延迟较多，而 QuartusII 时钟信号最大为 1μs 所以采用 Modelsim 仿真）：

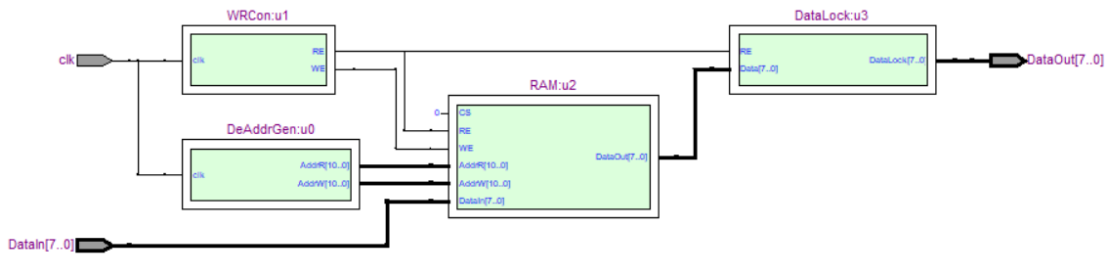


RTL 图（依次为总体图，interleaver、delay、deinterleaver 模块）：









四、设计中遇到的问题及解决方法

1.交织原理不太清楚。解决方法：查找资料，图形展示

2.RAM 大小的计算。解决方法：通过计算Σ（17*i+1）=1134,(i=0~11)得 RAM 需要 2K。

3.读写地址的计算。解决方法：将交织器理论总结为数学计算公式，读写地址分别为

rd_add=ai+bi,

wr_add=ai+bi-1(ai≠0)，

wr_add=ci（ai=0）

4.仿真时，modelsim 结果是 16 进制数，看起来不方便。解决方法：查资料，在需要显示十进制的信号处，右键，选择 radix，在弹出的窗口中选择需要的进制（十进制 decimal）即可。

五、设计总计

邢靖：本次设计让我了解了什么是交织器以及如何实现交织器。其中用 ram 寄存器循环计数以实现延时功能这样节省了大量空间是本次设计的一大亮点。对 vhdl 的代码格式以及 testbench 的书写的掌握更加熟练

敖梓茗：交织器的原理其实并不难，但我上网找原理资料的时候发现大部分人都只用文字晦涩地说出来，并没有生动形象地说清楚，所以我在课设报告中用图表的方式把交织器原理重新叙述了一遍，希望给以后选这个题目的师弟师妹们一个参考，让他们能对交织器有个直观形象的理解。VHDL 是一门硬件描述语言，是把我们内心想法实现出来的一种工具，当我们内心有了构想，明白了原理，把东西做出来也是顺理成章的事情。当然在实现的过程中会有许许多多的问题，需要我们耗费大量的精力去检查与纠错。但是在课设中运用与复习知识，在学中做，在做中学，又何尝不是一种乐趣?

陈永炜：本次课程设计的难点在于读写控制逻辑的产生，而攻克该难点的关键在于对原理的理解。和队友查询大量资料后，终于对卷积交织的原理有了形象化的理解。但在确定地址时，我们不知道如何产生地址循环效果，即在地址大于本支路的末位地址时，重新指向开头的地址。多方求解未果后，我们联想到了数字信号处理课上的求模运算，可以将结果在 0 至 17 内循环，解决了循环问题。在仿真阶段，我们有重新复习了时钟序列的产生方法。同时为了直观地感受交织器与解交织器的输出与输入之间的关系，我们运用了顺序序列作为输入数据，观察输出与输入之间的关系，更好地感受交织器的效果。

另外，我们对重要的功能模块都进行了仿真，并将仿真结果与理论结果进行了比较，加深了对交织器理论和 VHDL 代码的理解，也感受到了这门课程的趣味，此次课程设计让人收获良多。

蒲尧：本次实验我们选择的是交织器，一是觉得网上参考资料较多，二是觉得和我们通信原理所学的编码译码关系甚密。

在准备阶段我们查找了许多关于交织器原理的资料，但由于大同小异都是晦涩的文字描述，我们对其没有全面深入的了解。在和队友的合作思考下，我们了解了其大致编码译码方式，敖梓茗同学的示意图更是加深了我们的理解。

编码方面，我们大致将交织器和解交织器分为读写控制（读写不同时，避免干扰）、读写地址控制（写入地址在读取地址前一位）、RAM（选择 2k*8 暂存地址和数据信息）、数据锁存（锁存数据，消除竞争冒险）4 个模块。本实验的关键部分是交织器和解交织器的读写地址控制模块，因为我们选择的深度 B=12，每个信号延时 M=17，所以交织器第 i 路（0~11）延时 i*M；与此相反解交织器第 i 路（0~11）延时（11-i）*M。另外一个关键点是 RAM 的设计，由于 QuartusⅡ对 RAM 编译时还要对程序综合，生成各种门电路，占据内存较大，有时会编译不成功，所以我们按照书上的教学用 QuartusⅡ调用 Modelsim 进行编译仿真，果真速度提高，仿真成功。

总结这次课设，我对交织解交织经历了从一无所知到理解深刻的过程。对 VHDL 代码的编写、对 Altera 公司两大利器 QuartusⅡ和 Modelsim 的使用让我把数字系统课程设计这门课学到的东西运用到了较高水平。课程设计果真是让我们理论指导实践，实践体现理论，收获满足感的课程啊！

附录 A 全部源代码

交织器模块（InterLeaver.vhd）

```
----------------------------------------------------------------------
--InterLeaver module
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity InterLeaver is
port(clk:in std_logic;
    DataIn:in std_logic_vector(7 downto 0);
    DataOut:out std_logic_vector(7 downto 0));
end entity;

architecture stru of InterLeaver is
    --read & write address of interleaver module
    component AddrGen is
    port(clk:in std_logic;
        AddrR:out integer range 0 to 2047;
        AddrW:out integer range 0 to 2047
        );
    end component;
    --ram module
    component RAM is
    port(AddrR:in integer range 0 to 2047;
        AddrW:in integer range 0 to 2047;
        DataIn:in std_logic_vector(7 downto 0);
        DataOut:out std_logic_vector(7 downto 0);
        CS,RE,WE:in std_logic);
    end component;
    --read & write control module
    component WRcon is
    port(clk:in std_logic;
        WE,RE:out std_logic);
    end component;
    --data lock module
    component DataLock is
    port(Data:in std_logic_vector(7 downto 0);
        DataLock:out std_logic_vector(7 downto 0);
        RE:in std_logic);
    end component;
signal adr,adw:integer range 0 to 2047;
signal we,re:std_logic;
```

```
signal d:std_logic_vector(7 downto 0);
begin
    u0:AddrGen port map(clk,adr,adw);
    u1:WRcon port map(clk,we,re);
    u2:RAM port map(adr,adw,DataIn,d,'0',re,we);
    u3:DataLock port map(d,DataOut,re);
end stru;
-------------------------------------------------------------------
```

交织器读写模块(AddrGen.vhd)

```
-------------------------------------------------------------------
--read & write address of interleaver module
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity AddrGen is
port(clk:in std_logic;
    AddrR:out integer range 0 to 2047;
    AddrW:out integer range 0 to 2047
    );
end AddrGen;

architecture behav of AddrGen is
    signal count:integer range 0 to 11:=0;
begin

--count process
CNT:process(clk)
variable c:integer range 0 to 11:=0;
begin
    if(rising_edge(clk)) then
        c:=(c+1)mod 12;
    end if;
    count<=c;
end process;

--address process
--ai:change address,bi:base address,ci:tail address
--AddrR<=ai+bi;
--AddrW<=ai+bi-1;when ai isn't 0;
--AddrW<=ci;when ai is 0;

addr:process(clk)
```

```
    constant a0:integer:=0;
    constant a1:integer:=1;
    constant c1:integer:=18;
    constant a2:integer:=19;
    constant c2:integer:=53;
    constant a3:integer:=54;
    constant c3:integer:=105;
    constant a4:integer:=106;
    constant c4:integer:=174;
    constant a5:integer:=175;
    constant c5:integer:=260;
    constant a6:integer:=261;
    constant c6:integer:=363;
    constant a7:integer:=364;
    constant c7:integer:=483;
    constant a8:integer:=484;
    constant c8:integer:=620;
    constant a9:integer:=621;
    constant c9:integer:=774;
    constant a10:integer:=775;
    constant c10:integer:=945;
    constant a11:integer:=946;
    constant c11:integer:=1133;
    variable b1:integer range 0 to 17:=0;
    variable b2:integer range 0 to 34:=0;
    variable b3:integer range 0 to 51:=0;
    variable b4:integer range 0 to 68:=0;
    variable b5:integer range 0 to 85:=0;
    variable b6:integer range 0 to 102:=0;
    variable b7:integer range 0 to 119:=0;
    variable b8:integer range 0 to 136:=0;
    variable b9:integer range 0 to 153:=0;
    variable b10:integer range 0 to 170:=0;
    variable b11:integer range 0 to 187:=0;

begin

if(rising_edge(clk)) then
    case count is
    when 0=> AddrW<=a0;AddrR<=a0;
    when 1=>
        b1:=(b1+1)mod 18;
        if(b1=0) then
        AddrW<=c1;
```

```
        AddrR<=a1;
        else
        AddrR<=a1+b1;
        AddrW<=a1+b1-1;
        end if;
    when 2=>
        b2:=(b2+1)mod 35;
        if(b2=0) then
        AddrW<=c2;
        AddrR<=a2;
        else
        AddrR<=a2+b2;
        AddrW<=a2+b2-1;
        end if;
    when 3=>
        b3:=(b3+1)mod 52;
        if(b3=0) then
        AddrW<=c3;
        AddrR<=a3;
        else
        AddrR<=a3+b3;
        AddrW<=a3+b3-1;
        end if;
    when 4=>
        b4:=(b4+1)mod 69;
        if(b4=0) then
        AddrW<=c4;
        AddrR<=a4;
        else
        AddrR<=a4+b4;
        AddrW<=a4+b4-1;
        end if;
    when 5=>
        b5:=(b5+1)mod 86;
        if(b5=0) then
        AddrW<=c5;
        AddrR<=a5;
        else
        AddrR<=a5+b5;
        AddrW<=a5+b5-1;
        end if;
    when 6=>
        b6:=(b6+1)mod 103;
        if(b6=0) then
```

```
        AddrW<=c6;
        AddrR<=a6;
        else
        AddrR<=a6+b6;
        AddrW<=a6+b6-1;
        end if;
    when 7=>
        b7:=(b7+1)mod 120;
        if(b7=0) then
        AddrW<=c7;
        AddrR<=a7;
        else
        AddrR<=a7+b7;
        AddrW<=a7+b7-1;
        end if;
    when 8=>
        b8:=(b8+1)mod 137;
        if(b8=0) then
        AddrW<=c8;
        AddrR<=a8;
        else
        AddrR<=a8+b8;
        AddrW<=a8+b8-1;
        end if;
    when 9=>
        b9:=(b9+1)mod 154;
        if(b9=0) then
        AddrW<=c9;
        AddrR<=a9;
        else
        AddrR<=a9+b9;
        AddrW<=a9+b9-1;
        end if;
    when 10=>
        b10:=(b10+1)mod 171;
        if(b10=0) then
        AddrW<=c10;
        AddrR<=a10;
        else
        AddrR<=a10+b10;
        AddrW<=a10+b10-1;
        end if;
    when 11=>
    b11:=(b11+1)mod 188;
```

```
        if(b11=0) then
        AddrW<=c11;
        AddrR<=a11;
        else
        AddrR<=a11+b11;
        AddrW<=a11+b11-1;
        end if;
    end case;
end if;
end process;
end behav;
--------------------------------------------------------------------
```

读写控制模块(WRCon.vhd)

```
--------------------------------------------------------------------
--read & write control module
library ieee;
use ieee.std_logic_1164.all;
entity WRCon is
port(clk:in std_logic;
    WE,RE:out std_logic);
end entity;
architecture behav of WRcon is
begin
    WE<=not clk;
    RE<=clk;
end behav;
--------------------------------------------------------------------
```

RAM 模块(RAM.vhd)

```
--------------------------------------------------------------------
--ram module
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity RAM is
port(AddrR:in integer range 0 to 2047;
    AddrW:in integer range 0 to 2047;
    DataIn:in std_logic_vector(7 downto 0);
    DataOut:out std_logic_vector(7 downto 0);
    CS,RE,WE:in std_logic);
end RAM;
architecture behav of RAM is
begin
```

```
process(AddrW,RE,WE)
type ram_array is array(0 to 2047) of std_logic_vector(7 downto 0);
variable mem:ram_array;
begin
if(CS='0') then
    if(RE='0') then DataOut<=mem(AddrR);
    elsif(WE='0') then mem(AddrW):=DataIn;
    end if;
end if;
end process;
end behav;
--------------------------------------------------------------------
```

数据锁存模块(DataLock.vhd)

```
--------------------------------------------------------------------
--data lock module
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity DataLock is
port(Data:in std_logic_vector(7 downto 0);
    DataLock:out std_logic_vector(7 downto 0);
    RE:in std_logic);
end entity;
architecture behav of DataLock is
begin
process(Data,RE)
begin
if(RE='0') then DataLock<=Data;
end if;
end process;
end behav;
--------------------------------------------------------------------
```

解交织读写地址模块(DeAddrGen.vhd)

```
--------------------------------------------------------------------
--read & write address of deinterleaver module
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity DeAddrGen is
port(clk:in std_logic;
    AddrR:out integer range 0 to 2047;
    AddrW:out integer range 0 to 2047
```

```
    );
end DeAddrGen;
architecture behav of DeAddrGen is
    signal count:integer range 0 to 11:=0;
begin

--count process
C:process(clk)
variable c:integer range 0 to 11:=0;
begin
    if(rising_edge(clk)) then
        c:=(c+1)mod 12;
    end if;
    count<=c;
end process;

addr:process(clk)
    constant a11:integer:=0;
    constant a10:integer:=1;
    constant c10:integer:=18;
    constant a9:integer:=19;
    constant c9:integer:=53;
    constant a8:integer:=54;
    constant c8:integer:=105;
    constant a7:integer:=106;
    constant c7:integer:=174;
    constant a6:integer:=175;
    constant c6:integer:=260;
    constant a5:integer:=261;
    constant c5:integer:=363;
    constant a4:integer:=364;
    constant c4:integer:=483;
    constant a3:integer:=484;
    constant c3:integer:=620;
    constant a2:integer:=621;
    constant c2:integer:=774;
    constant a1:integer:=775;
    constant c1:integer:=945;
    constant a0:integer:=946;
    constant c0:integer:=1133;
    variable b10:integer range 0 to 17:=0;
    variable b9:integer range 0 to 34:=0;
    variable b8:integer range 0 to 51:=0;
    variable b7:integer range 0 to 68:=0;
```

```
    variable b6:integer range 0 to 85:=0;
    variable b5:integer range 0 to 102:=0;
    variable b4:integer range 0 to 119:=0;
    variable b3:integer range 0 to 136:=0;
    variable b2:integer range 0 to 153:=0;
    variable b1:integer range 0 to 170:=0;
    variable b0:integer range 0 to 187:=0;
begin
if(rising_edge(clk)) then
    case count is
    when 0=>
        b0:=(b0+1)mod 188;
        if(b0=0) then
        AddrW<=c0;
        AddrR<=a0;
        else
        AddrR<=a0+b0;
        AddrW<=a0+b0-1;
        end if;
    when 1=>
        b1:=(b1+1)mod 171;
        if(b1=0) then
        AddrW<=c1;
        AddrR<=a1;
        else
        AddrR<=a1+b1;
        AddrW<=a1+b1-1;
        end if;
    when 2=>
        b2:=(b2+1)mod 154;
        if(b2=0) then
        AddrW<=c2;
        AddrR<=a2;
        else
        AddrR<=a2+b2;
        AddrW<=a2+b2-1;
        end if;
    when 3=>
        b3:=(b3+1)mod 137;
        if(b3=0) then
        AddrW<=c3;
        AddrR<=a3;
        else
        AddrR<=a3+b3;
```

```
        AddrW<=a3+b3-1;
        end if;
    when 4=>
        b4:=(b4+1)mod 120;
        if(b4=0) then
        AddrW<=c4;
        AddrR<=a4;
        else
        AddrR<=a4+b4;
        AddrW<=a4+b4-1;
        end if;
    when 5=>
        b5:=(b5+1)mod 103;
        if(b5=0) then
        AddrW<=c5;
        AddrR<=a5;
        else
        AddrR<=a5+b5;
        AddrW<=a5+b5-1;
        end if;
    when 6=>
        b6:=(b6+1)mod 86;
        if(b6=0) then
        AddrW<=c6;
        AddrR<=a6;
        else
        AddrR<=a6+b6;
        AddrW<=a6+b6-1;
        end if;
    when 7=>
        b7:=(b7+1)mod 69;
        if(b7=0) then
        AddrW<=c7;
        AddrR<=a7;
        else
        AddrR<=a7+b7;
        AddrW<=a7+b7-1;
        end if;
    when 8=>
        b8:=(b8+1)mod 52;
        if(b8=0) then
        AddrW<=c8;
        AddrR<=a8;
        else
```

```
        AddrR<=a8+b8;
        AddrW<=a8+b8-1;
        end if;
    when 9=>
        b9:=(b9+1)mod 35;
        if(b9=0) then
        AddrW<=c9;
        AddrR<=a9;
        else
        AddrR<=a9+b9;
        AddrW<=a9+b9-1;
        end if;
    when 10=>
        b10:=(b10+1)mod 18;
        if(b10=0) then
        AddrW<=c10;
        AddrR<=a10;
        else
        AddrR<=a10+b10;
        AddrW<=a10+b10-1;
        end if;
    when 11=>
        AddrW<=a11;AddrR<=a11;
    end case;
end if;
end process;
end behav;
-----------------------------------------------------------------
```

解交织器模块（DeInterLeaver.vhd）

```
-----------------------------------------------------------------
-- DeInterLeaver module
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity DeInterLeaver is
port(clk:in std_logic;
    DataIn:in std_logic_vector(7 downto 0);
    DataOut:out std_logic_vector(7 downto 0));
end entity;
architecture stru of DeInterLeaver is
    --read & write address of deinterleaver module
    component DeAddrGen is
    port(clk:in std_logic;
```

```
        AddrR:out integer range 0 to 2047;
        AddrW:out integer range 0 to 2047
        );
    end component;
    --ram module
    component RAM is
    port(AddrR:in integer range 0 to 2047;
        AddrW:in integer range 0 to 2047;
        DataIn:in std_logic_vector(7 downto 0);
        DataOut:out std_logic_vector(7 downto 0);
        CS,RE,WE:in std_logic);
    end component;
    --read & write control module
    component WRcon is
        port(clk:in std_logic;
        WE,RE:out std_logic);
    end component;
    --data lock module
    component DataLock is
        port(Data:in std_logic_vector(7 downto 0);
        DataLock:out std_logic_vector(7 downto 0);
        RE:in std_logic);
    end component;
signal adr,adw:integer range 0 to 2047;
signal we,re:std_logic;
signal d:std_logic_vector(7 downto 0);
begin
  u0:DeAddrGen port map(clk,adr,adw);
  u1:WRcon port map(clk,we,re);
  u2:RAM port map(adr,adw,DataIn,d,'0',re,we);
  u3:DataLock port map(d,DataOut,re);
end stru;
-----------------------------------------------------------------
```

时钟延时模块(ClockDelay.vhd)

```
-----------------------------------------------------------------
--ClockDelay module
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity ClockDelay is
port(clk:in std_logic;
    clk_de:out std_logic);
end ClockDelay;
```

```
architecture behav of ClockDelay is
signal c,d:std_logic:='0';
begin
clk_de<=clk and d;
process(clk)
begin
if(clk'event and clk='1') then
    if(c='0') then
        c<='1';
    end if;
elsif(clk'event and clk='0') then
    if(c='1') then
        d<='1';
    end if;
end if;
end process;
end behav;
--------------------------------------------------------------------
```

交织解交织综合模块（In_DeModel.vhd）

```
--------------------------------------------------------------------
-- InterLeaver & DeInterleaver synthetical Model
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity In_DeModel is
port(clk:in std_logic;
    DataIn:in std_logic_vector(7 downto 0);
    DataOut:Out std_logic_vector(7 downto 0));
end entity;
architecture stru of In_DeModel is
    --InterLeaver module
    component InterLeaver is
    port(clk:in std_logic;
        DataIn:in std_logic_vector(7 downto 0);
        DataOut:out std_logic_vector(7 downto 0));
    end component;
    component DeInterLeaver is
    port(clk:in std_logic;
        DataIn:in std_logic_vector(7 downto 0);
        DataOut:out std_logic_vector(7 downto 0));
    end component;
    component ClockDelay is
    port(clk:in std_logic;
```

```
        clk_de:out std_logic);
    end component;
    signal d:std_logic_vector(7 downto 0);
    signal clk_de:std_logic;
begin
    u0:InterLeaver port map(clk,DataIn,d);
    u1:ClockDelay port map(clk,clk_de);
    u2:DeInterLeaver port map(clk_de,d,DataOut);
end stru;
--------------------------------------------------------------------
```

交织解交织综合模块 TestBench（In_DeModel_vhd_tst.vht）

```
--------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

ENTITY In_DeModel_vhd_tst IS
END In_DeModel_vhd_tst;
ARCHITECTURE In_DeModel_arch OF In_DeModel_vhd_tst IS
-- constants
-- signals
SIGNAL clk : STD_LOGIC;
SIGNAL DataIn : STD_LOGIC_VECTOR(7 DOWNTO 0):="00000000";
-- output of deinterleaver
SIGNAL DataOut : STD_LOGIC_VECTOR(7 DOWNTO 0);
-- output of interleaver
SIGNAL DataOut1 : STD_LOGIC_VECTOR(7 DOWNTO 0);
COMPONENT In_DeModel
    PORT (
    clk : IN STD_LOGIC;
    DataIn : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
    DataOut : OUT STD_LOGIC_VECTOR(7 DOWNTO 0)
    );
END COMPONENT;

component InterLeaver
port(clk:in std_logic;
    DataIn:in std_logic_vector(7 downto 0);
    DataOut:out std_logic_vector(7 downto 0));
end component;

BEGIN
    --instance i1 output of deinterleaver
```

```
    i1 : In_DeModel
    PORT MAP (
    clk => clk,
    DataIn => DataIn,
    DataOut => DataOut
);
--instance i1 output of interleaver
    i2 : InterLeaver
    port map(
    clk => clk,
    DataIn => DataIn,
    DataOut => DataOut1
);
--clock signal
clk_gen: PROCESS
constant period:time:=20 ns;
BEGIN
    clk<='0';
    wait for period/2;
    clk<='1';
    wait for period/2;
END PROCESS clk_gen;
--input signal
DataIn_gen:process(clk)
begin
if(rising_edge(clk))then
    DataIn<=DataIn+'1';
end if;
end process DataIn_gen;

END In_DeModel_arch;

--------------------------------THE END-----------------------------
```