Lab2-report

Lab2-1

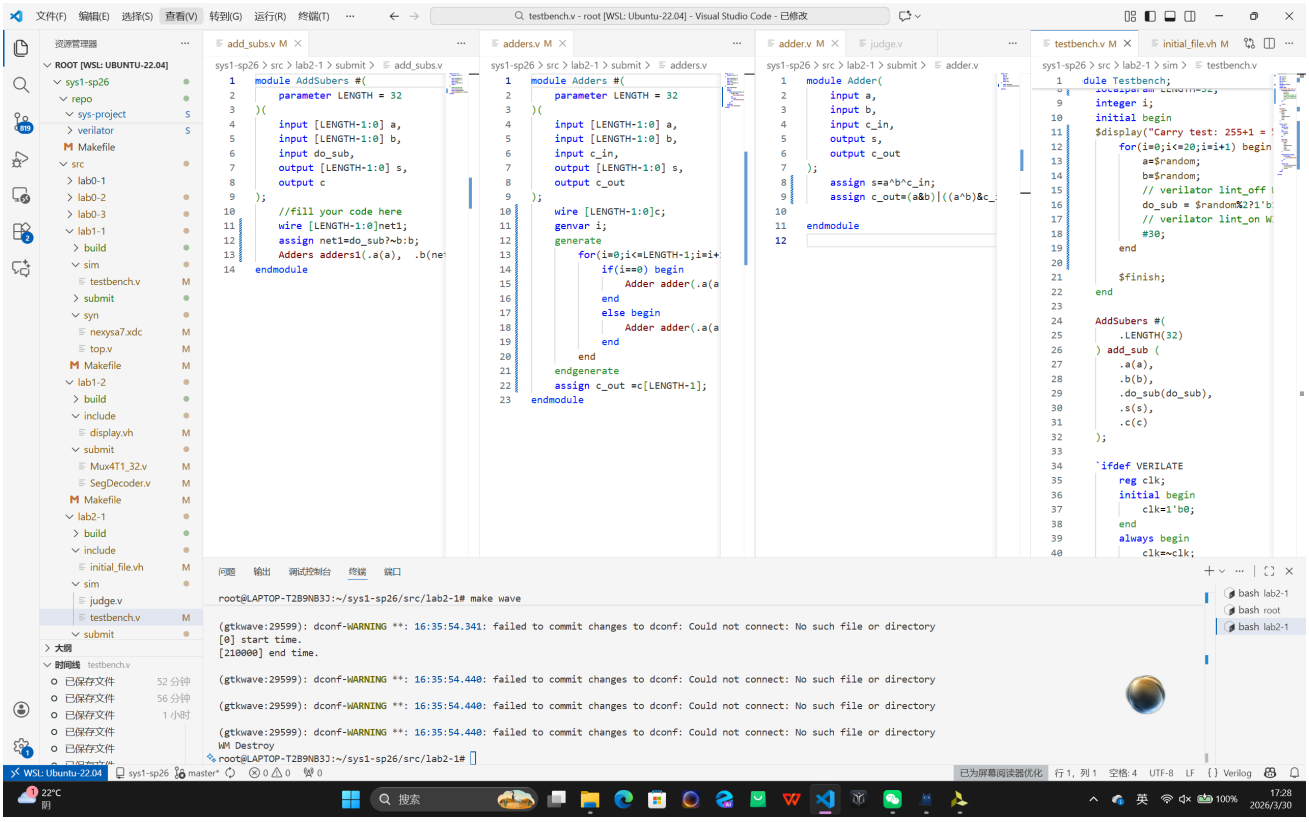
实验目的

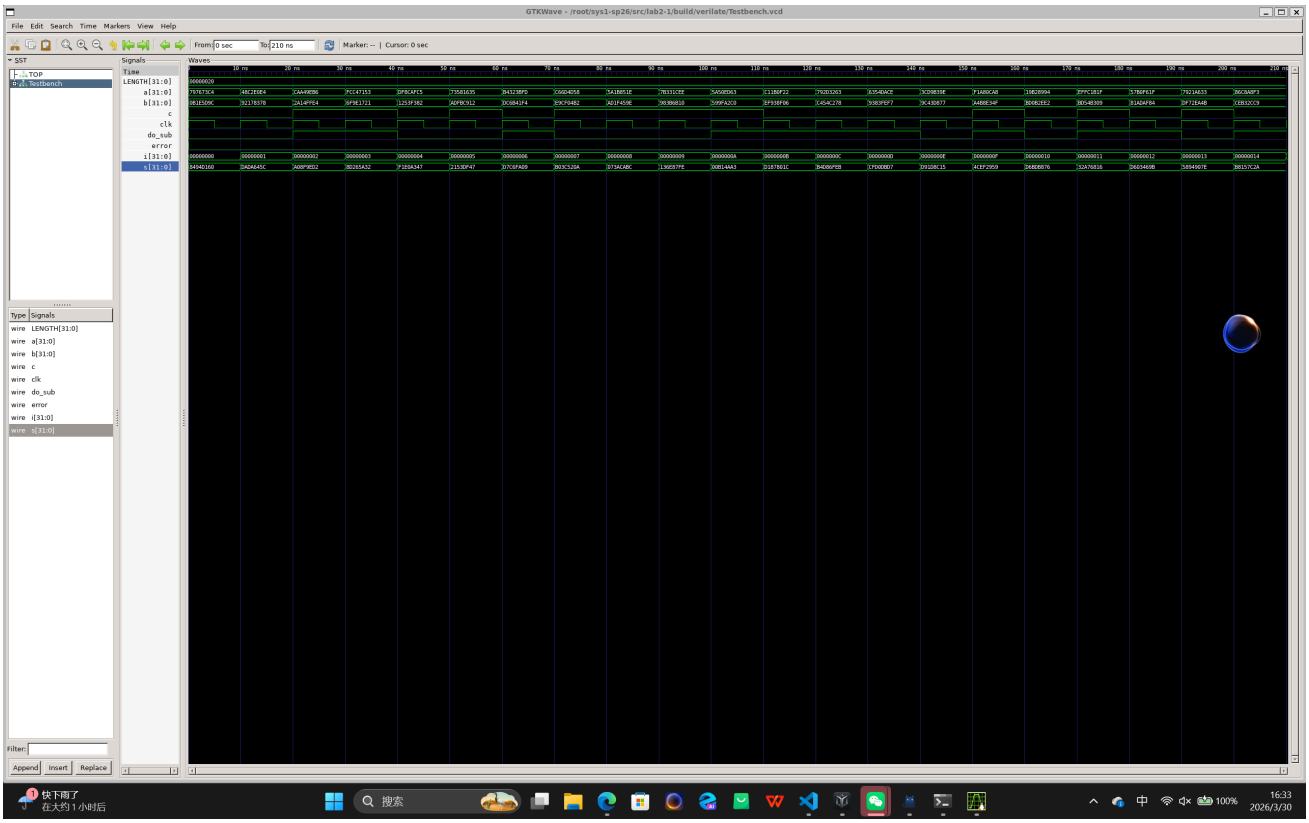
- 强化使用 Verilog 语言进行电路设计的方法
- 掌握行波进位加法器和全加减法器的原理
- 设计实现 64 位超前进位加法器

实验过程

> 在阅读学习了前文的多位加法器之后，本人开始搭建上板了FGBA板。
> 实验步骤如下：

- 进行了对addsub全加减器的实现，并进行了对其进行 `$random()` 的测试。




GTKWave - /root/sys1-sp26/src/lab2-1/build/verilate/Testbench.vcd
File Edit Search Time Markers View Help
SST
TOP
Testbench
Signals
Time
LENGTH[31:0]
a[31:0]
b[31:0]
c
clk
do_sub
error
i[31:0]
o[31:0]
Waves
Type Signals
wire LENGTH[31:0]
wire a[31:0]
wire b[31:0]
wire c
wire clk
wire do_sub
wire error
wire i[31:0]
wire o[31:0]
Filter:
Append Insert Replace
快下雨了
在大约 1 小时后
搜索
16:33
2026/3/30

- 在过程中，我出现了如下的问题，我对其进行了debug。

```
root@LAPTOP-T2B9NB3J:~# cd /root/sys1-sp26/src/lab2-1
root@LAPTOP-T2B9NB3J:~/sys1-sp26/src/lab2-1# make verilate
mkdir -p /root/sys1-sp26/src/lab2-1/build/verilate
cp /root/sys1-sp26/src/lab2-1/../../repo/sys-project/lab2-1/syn/initial.hex /root/sys1-sp26/src/lab2-1/build/verilate
make -C /root/sys1-sp26/src/lab2-1/build/verilate -f VTestbench.mk Testbench
make[1]: Entering directory '/root/sys1-sp26/src/lab2-1/build/verilate'
make[1]: 'Testbench' is up to date.
make[1]: Leaving directory '/root/sys1-sp26/src/lab2-1/build/verilate'
cd /root/sys1-sp26/src/lab2-1/build/verilate; ./Testbench
a=1220731108, b=2451014520, do_sub=0, s=3671417756, c=0, error
a=3399786166, b= 706019300, do_sub=1, s= 525311698, c=0, error
a=4240732499, b=1872631585, do_sub=1, s=1822792114, c=0, error
a=3750539205, b= 307491714, do_sub=0, s=3453967431, c=0, error
a=1935152693, b=2918959378, do_sub=0, s=3735281447, c=0, error
a=3023191037, b=3698016756, do_sub=1, s=2544302601, c=0, error
a=3329051992, b=3922658482, do_sub=0, s= 799164906, c=0, error
a=1511752990, b=2904507806, do_sub=0, s=4144283776, c=0, error
a=2066947310, b=2554030864, do_sub=0, s=3808983038, c=0, error
a=1515253091, b=1503634112, do_sub=1, s=4231049379, c=0, error
a=3239776034, b=4019425030, do_sub=1, s=3514269660, c=0, error
a=2033005155, b=3293889144, do_sub=1, s=1116082155, c=0, error
a=1666505422, b=2474901239, do_sub=1, s= 254335943, c=0, error
a=1020900254, b=2621692023, do_sub=0, s=2694474729, c=0, error
a=4054322344, b=2763580239, do_sub=1, s=2867793945, c=0, error
a= 431131028, b=3171626722, do_sub=0, s=2763630454, c=0, error
a=4026276639, b=3176444681, do_sub=1, s=2908186646, c=0, error
a=1471215135, b=2175643524, do_sub=1, s= 702719643, c=0, error
a=2032248371, b=3748850251, do_sub=0, s=2790476920, c=0, error
a=2261297395, b=3467848905, do_sub=1, s=3078912970, c=0, error
- /root/sys1-sp26/src/lab2-1/sim/testbench.v:18: Verilog $finish
a=2261297395, b=3467848905, do_sub=1, s=3078912970, c=0, error
root@LAPTOP-T2B9NB3J:~/sys1-sp26/src/lab2-1# make verilate
mkdir -p /root/sys1-sp26/src/lab2-1/build/verilate
cp /root/sys1-sp26/src/lab2-1/../../repo/sys-project/lab2-1/syn/initial.hex /root/sys1-sp26/src/lab2-1/build/verilate
make -C /root/sys1-sp26/src/lab2-1/build/verilate -f VTestbench.mk Testbench
make[1]: Entering directory '/root/sys1-sp26/src/lab2-1/build/verilate'
make[1]: 'Testbench' is up to date.
make[1]: Leaving directory '/root/sys1-sp26/src/lab2-1/build/verilate'
cd /root/sys1-sp26/src/lab2-1/build/verilate; ./Testbench
a=1220731108, b=2451014520, do_sub=0, s=3671417756, c=0, error
a=3399786166, b= 706019300, do_sub=1, s= 525311698, c=0, error
a=4240732499, b=1872631585, do_sub=1, s=1822792114, c=0, error
a=3750539205, b= 307491714, do_sub=0, s=3453967431, c=0, error
a=1935152693, b=2918959378, do_sub=0, s=3735281447, c=0, error
a=3023191037, b=3698016756, do_sub=1, s=2544302601, c=0, error
a=3329051992, b=3922658482, do_sub=0, s= 799164906, c=0, error
a=1511752990, b=2904507806, do_sub=0, s=4144283776, c=0, error
a=2066947310, b=2554030864, do_sub=0, s=3808983038, c=0, error
a=1515253091, b=1503634112, do_sub=1, s=4231049379, c=0, error
a=3239776034, b=4019425030, do_sub=1, s=3514269660, c=0, error
a=2033005155, b=3293889144, do_sub=1, s=1116082155, c=0, error
a=1666505422, b=2474901239, do_sub=1, s= 254335943, c=0, error
a=1020900254, b=2621692023, do_sub=0, s=2694474729, c=0, error
aster* ⟲  ⊗ 0 ⚠ 0  (( )) 0
```

1. 在写到 `do_sub = $random%2?1'b1:1'b0;` 的时候，系统一直报错。原因是将一个32位的数写进了一个一位的数里面。（虽然我也不知道为什么 ）经过了上网搜索发现需要加上此代码来绕过宽度的检测：

```
    // verilator lint_off WIDTHTRUNC
do_sub = $random%2?1'b1:1'b0;
// verilator lint_on WIDTHTRUNC
```

2. 图片中的bug原因是我没有将各个模块的接口接好导致数据不能完整地传输到`Addsub.v`的文件中

实验结果

结果展示

- 使用 `for` 进行仿真。
  testbench.v（仅仅展示for循环的部分）

```
……

localparam LENGTH=32;
    integer i;
    initial begin
        for(i=0;i<=20;i=i+1) begin
            a=$random;
            b=$random;
            // verilator lint_off WIDTHTRUNC
            do_sub = $random%2?1'b1:1'b0;
            // verilator lint_on WIDTHTRUNC
            #30;
        end

        $finish;
    end
……
```

```
module Adders #(
    parameter LENGTH = 64
)(
    input [LENGTH-1:0] a,
    input [LENGTH-1:0] b,
    input c_in,
    output [LENGTH-1:0] s,
    output c_out
);
    wire [LENGTH-1:0]c;
    genvar i;
    generate
        for(i=0;i<=LENGTH-1;i=i+1) begin: adder_gen
            if(i==0) begin
                Adder
adder(.a(a[i]),.b(b[i]),.c_in(c_in),.s(s[i]),.c_out(c[i]));
            end
            else begin
                Adder adder(.a(a[i]),.b(b[i]),.c_in(c[i-
1]),.s(s[i]),.c_out(c[i]));
```

```
            end
        end
    endgenerate
    assign c_out =c[LENGTH-1];
endmodule
```

总结：在仿真里面的循环参量是使用 `integer` ， `short` ， `longint` 。而在模块中的循环变量是使用 `genvar` 的变量

- 使用 `$random` 进行仿真样例生成：

```
            a=$random;
            b=$random;
            // verilator lint_off WIDTHTRUNC
            do_sub = $random%2?1'b1:1'b0;
            // verilator lint_on WIDTHTRUNC
```

- 综合实验全加减法器：
  adder.v:

```
module Adder(
    input a,
    input b,
    input c_in,
    output s,
    output c_out
);
    assign s=a^b^c_in;
    assign c_out=(a&b)|((a^b)&c_in);

endmodule
```

c_out指的是是否给下一位进行进位，c_in指的是上一位是否发生了进位，s是本位的结果
adders.v:

```
module Adders #(
    parameter LENGTH = 64
)(
    input [LENGTH-1:0] a,
    input [LENGTH-1:0] b,
    input c_in,
    output [LENGTH-1:0] s,
```

```
    output c_out
);
    wire [LENGTH-1:0]c;
    genvar i;
    generate
        for(i=0;i<=LENGTH-1;i=i+1) begin: adder_gen
            if(i==0) begin
                Adder
adder(.a(a[i]),.b(b[i]),.c_in(c_in),.s(s[i]),.c_out(c[i]));
            end
            else begin
                Adder adder(.a(a[i]),.b(b[i]),.c_in(c[i-
1]),.s(s[i]),.c_out(c[i]));
            end
        end
    endgenerate
    assign c_out =c[LENGTH-1];
endmodule
```

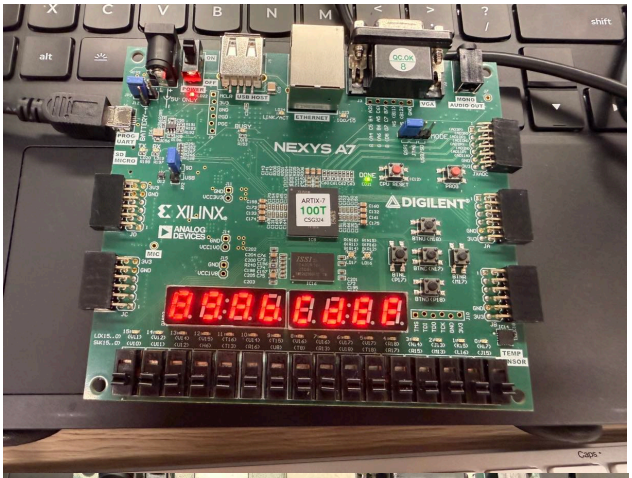
其中`if`的模块是分支了时候是个位数的加减法。`generate`模块则是遍历了所有位数的加减法。

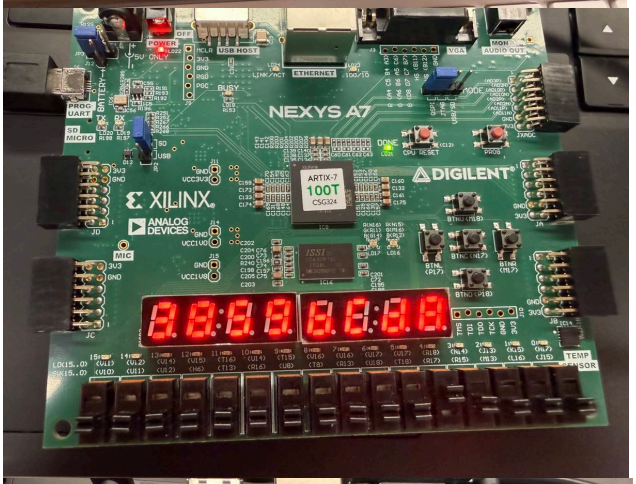
add_subs.v:

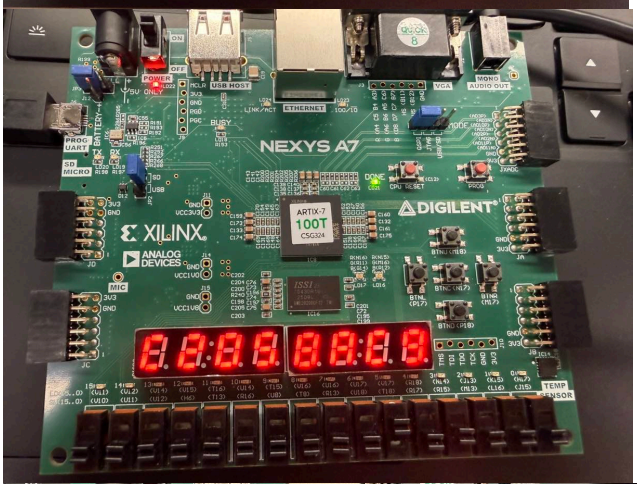
```
module AddSubers #(
    parameter LENGTH = 64
)(
    input [LENGTH-1:0] a,
    input [LENGTH-1:0] b,
    input do_sub,
    output [LENGTH-1:0] s,
    output c
);
    //fill your code here
    wire [LENGTH-1:0]net1;
    assign net1=do_sub?~b:b;
    Adders adders1(.a(a),  .b(net1), .c_in(do_sub),  .s(s), .c_out(c));
endmodule
```

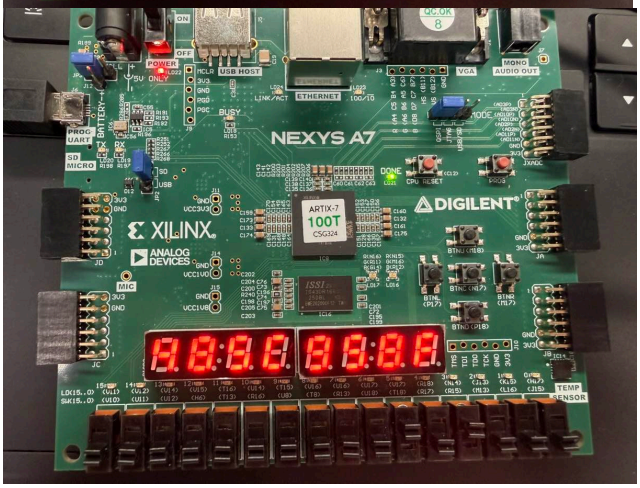
其中net1接受的是b的反码，而经过`c_in`的处理之后，最终的b取得是他的补码，符合2^m意义下的加减法群。

- 上板的结果：


NEXYS A7
DIGILENT
XILINX
ANALOG DEVICES
ARTIX-7
100T
CSG324


NEXYS A7
DIGILENT
XILINX
ANALOG DEVICES
ARTIX-7
100T
CSG324


NEXYS A7
DIGILENT
XILINX
ANALOG DEVICES
ARTIX-7
100T
CSG324


NEXYS A7
DIGILENT
XILINX
ANALOG DEVICES
ARTIX-7
100T
CSG324

思考题

对于 `repo/sys-project/lab2-1/syn/top.v` 中的 `reg [63:0] adder [0:7]`，通过仿真或者综合下板观察 adder 的每个 bit 会被初始化为何值？请尝试对对 `readmemh` 如何初始化一维向量数组的每个 bit 给出结论。

1. adder会被一一赋为 `initial.hex` 里面的16进制数转二进制的结果 `[63:0]` 存的是行，`[0:7]` 是列的标号。
2. 经过查阅发现 `readmemh` 是以字节为单位来对adder里面的数组元素进行存储的，而不是通过一个一个bit来进行存储。

请总结超前进位加法器和行波进位加法器的优缺点?

行波加法器优点是设计简单，缺点在于必须在前一个单加法器运算完成之后才能进行运算。而超前进位加法器的有点在于可以同时进行加法的运算，使得运算的效率更高，但是随之而言他的缺点就是电路设计复杂，耗能高。

如何表示运算溢出？请给出溢出表示（用 `AddSubers` 的输入与输出以及各位进位 ci计算）。

无符号

- 溢出会在 `c_out` 里面表示为1，而结果的运算就是$a+b-(1111111111111111.....1111)_2$

有符号

- 溢出发现的规则：1. 正数加正数等于负数 2. 负数加负数等于正数。
- 运算方式：是在2^m的数群意义之下的运算，像一个转盘一样轮回运算。

心得体会

我真的棒棒哒