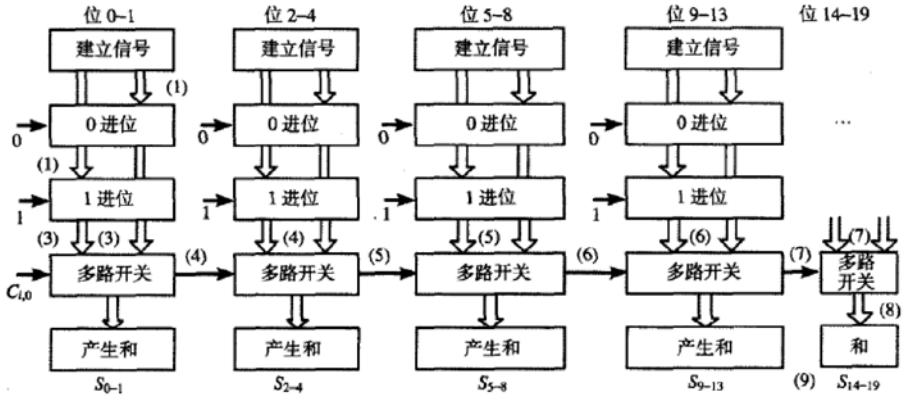
数字集成电路大作业报告

樊子辰 无 54 2015011065

一 结构设计

在 32 位加法器设计的过程中，我们采用了多种结构并进行比较。对于进位的处理，我们采用了平方根选择进位加法器。平方根选择进位加法器与行波进位和线性选择进位的结构相比速度较快，而与超前进位结构相比功耗较小，于是平方根选择进位结构便成为了我们的不二之选。



平方根选择进位结构

对于 1bit 的全加器电路结构，我们尝试了两种结构，一种是曼切斯特加法器电路，一种是镜像加法器电路。其中曼切斯特加法器的进位链又有动态和静态两种实现方式。经过比较，排除与仿真，我们最终确定了自己最终的加法器结构——基于镜像全加器的平方根选择进位加法器。

二 电路设计

2.0 全加器逻辑

为了化简电路结构，我们定义某些中间信号

$$G = AB$$

$$D = \bar{A}\bar{B}$$

$$P = A \oplus B$$

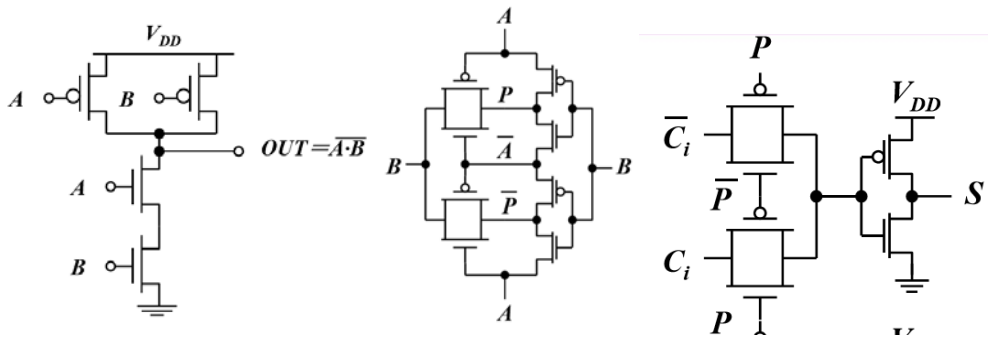
于是可以将 S 和 Co 重新写为 P 和 G（或 D）的函数

$$Co(G, P) = G + PCi$$

$$S(G, P) = P \oplus Ci$$

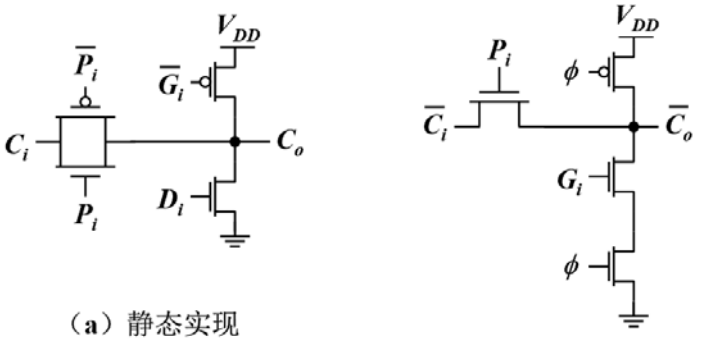
2.1 曼切斯特全加器

和产生电路与传输门型加法器类似：



用互补静态 CMOS　　产生 P 和 P‘　　采用传输门的方式来
产生 G/G’,D/D’　　产生和位 S 信号

进位链采用曼切斯特进位门的形式：

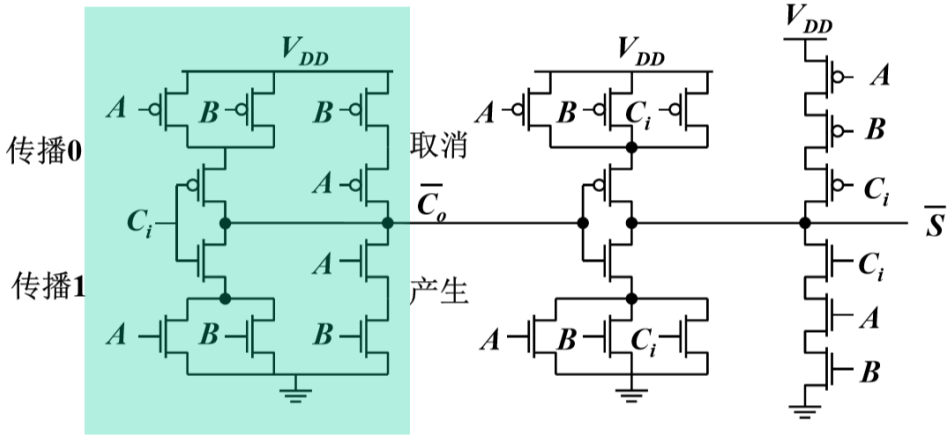


（a）静态实现　　（b）动态实现

对于动态实现，我们经过尝试后发现，因为需要设定一个高频 clock 信号，大量验证加法器正确性会比较困难，于是我们采用了静态实现的方式。

2.2 镜像全加器

另外一种全加器结构是镜像全加器，其结构如下：



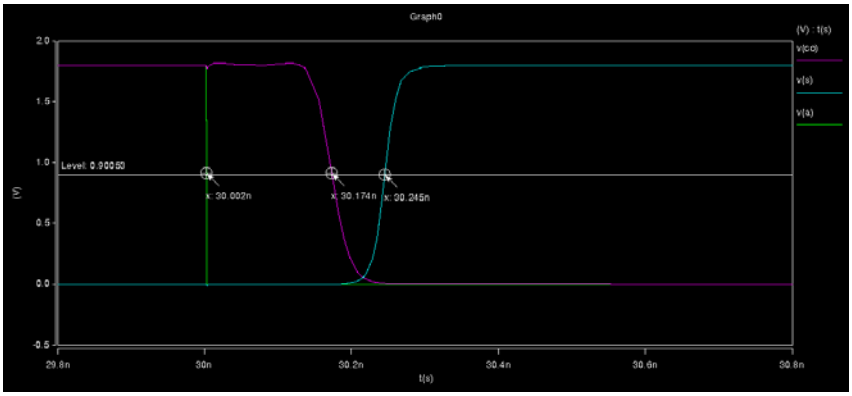
镜像全加器结构

镜像全加器的优点是晶体管数目少、不需要单独的单元产生 GDP 信号、结构对称且在求和电路中的所有晶体管都可以采用最小尺寸。

三 仿真验证结果

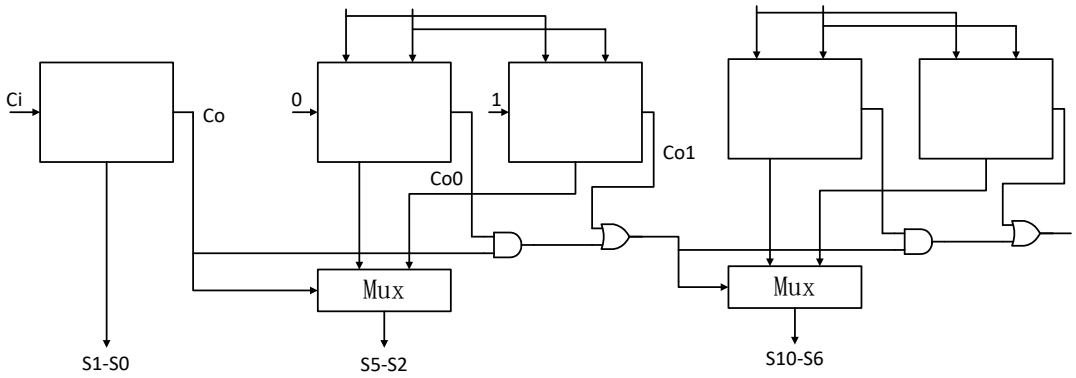
3.1 基于曼切斯特进位链的平方根选择进位加法器

我们先仿真了 1bit 的基于曼切斯特进位的加法器电路，初始仿真结果如下：

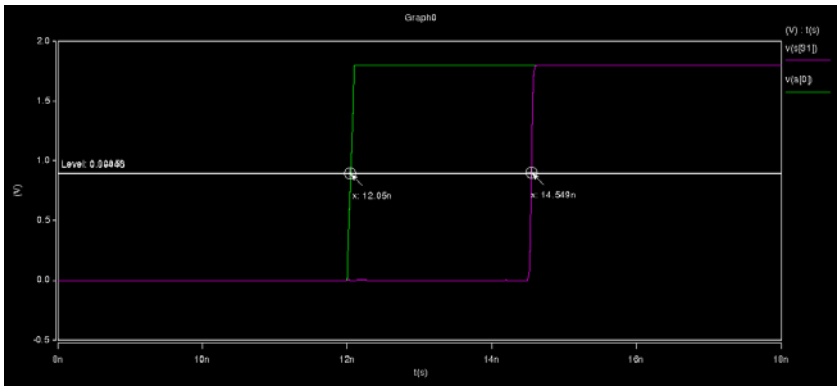


1bit 传输门+曼切斯特加法器延时

得到延迟为 0.243ns，接下来我们将 1bit 全加器组合成为 32bit 加法器，其中平方根选择进位结构选择 2+4+5+7+8 的结构。具体结构图如下：

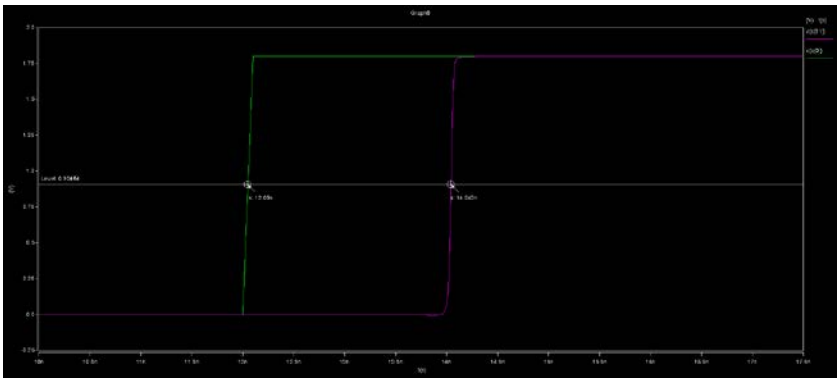


经过仿真我们得到 32bit 加法器总最长延时为 2.499ns：



基于曼切斯特进位的 32bit 加法器延时

我们测试了 600 组随机数据，得到结果均正确，测试功耗为$1.3219 \times 10^{-2}A \times 1.8V = 23.8mW$! 这个功耗实在是太大了，无法接受的结果。我们找到原因，发现是因为我们并没有调整晶体管的尺寸大小，初始选择的尺寸都较大，导致延时较大，功耗也较大。于是我们更改了尺寸大小，并且在更改尺寸的过程中还发现最小尺寸可能会使一些模块无法正常驱动，最终，我们调试了较小的能正确工作的尺寸后，得到延时为 1.993ns：

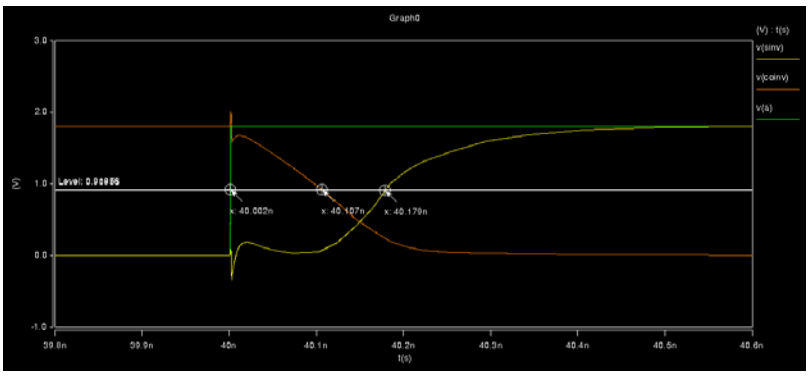


减小尺寸后的 32bit 加法器延时

其功耗也减小了许多，测试得到功耗为$5.779 \times 10^{-3}A \times 1.8V = 10.4mW$,然而功耗依然较大。我们分析原因可能是求和电路运用传输门加法器导致其延时较长，且中途需要产生许多如 G，P，D 等信号导致电路较为复杂，且光是进位的优化采用曼切斯特进位链反而加大了延时，可能进位直接利用传输门加法器的设计会更好，只是我们还没有尝试。

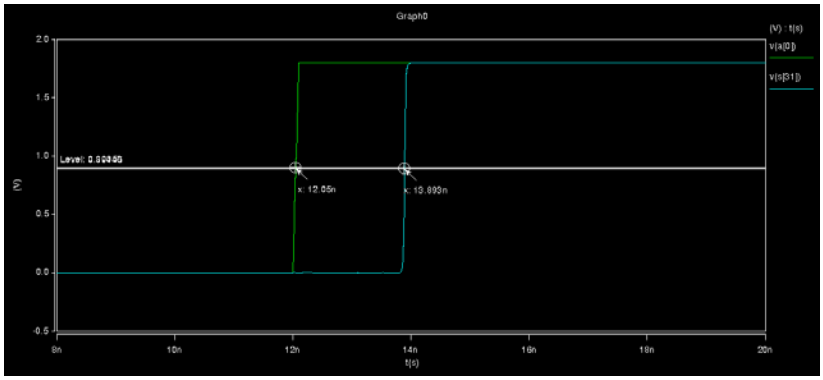
3.2 基于镜像全加器的平方根选择进位加法器

选择进位结构依然如前一部分图所示，我们仿真了 1bit 的镜像全加器，得到延时是 0.177ns:



1bit 镜像加法器延时

连接后仿真 32bit 全加器，得到最长延时为 1.843ns:



32bit 基于镜像全加器的加法器延时

同样仿真 600 组随机数，得到功耗为$1.8049 \times 10^{-4}A \times 1.8V = 324.8\mu W$。发现这种电路功耗很低，延时较小。我们最终选定基于镜像全加器的平方根选择进位加法器作为我们的最终设计。

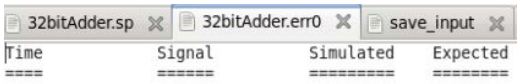
我们计算此设计的总面积如下：运用 1bit 镜像全加器 32 个，反相器 34 个，级间连接模块 5 个，二选一选择器 30 个，其中 1bit 镜像全加器面积（117*Lmin）um，反相器（3*Lmin）um，级间连接模块（24*Lmin）um，二选一选择器（18*Lmin）um，总面积为（4506*Lmin）um（答辩 ppt 中有点算错了）。由 Lmin=0.18um，可以计算得到总面积为 811.08um。

综上，我们得到性能如下：

最长延时	1.843ns
平均功耗	324.8uW
面积	811.8um

3.3 随机数的产生与验证

因为 32 位的加法较为复杂，我们不可能人工验证，于是我们采用了数字矢量文件的方式，先用 matlab 产生随机数和他们的和，然后 hspice 可以将仿真得到的结果和理论结果进行比较，最后输出一个.err0 文件。如果没有错误，.err0 文件的输出是这样的：



如果有错误，则会输出错误的位数，其仿真值和理论值。这样大大减小了我们验证电路正确性的时间。

matlab 产生随机数及其和的代码如下：

```
N=600;
A=round((2^32-1)*rand(1,N));
B=round((2^32-1)*rand(1,N));
S=A+B;
A_hex = dec2hex(A);
B_hex = dec2hex(B);
S_hex_full = dec2hex(S);
S_hex = S_hex_full(:,2:9);
for i=1:N
    In_hex(i,:)=[A_hex(i,:),B_hex(i,:),S_hex(i,:)];
end
fid_IN=fopen('IN.txt','wt');
for i=1:N
    fprintf(fid_IN,'%s\n',In_hex(i,:));
end
fclose(fid_IN);
```

数字矢量文件如下：

```
;==Vector file header====
radix 44444444444444444444444
vname A[[31:28]] A[[27:24]] A[[23:20]] A[[19:16]] A[[15:12]] A[[11:8]] A[[7:4]] A[[3:0]] B[[31:28]] B[[27:24]] B[[23:20]] B[[19:16]] B[[15:12]] B[[11:8]] B
[[7:4]] B[[3:0]] S[[31:28]] S[[27:24]] S[[23:20]] S[[19:16]] S[[15:12]] S[[11:8]] S[[7:4]] S[[3:0]]
io i i i i i i i i i i i i i i i i o o o o o o o o
tunit ns
odelay 10.0
tdelay 2.0
slope 0.1
vih 1.8
vil 0.0
period 10.0
;=============
040857DBF1372C1DF53F83F8
AF139EA5D97463DC88880281
E0BE295E2324FBC403E32522
6E8AD7AAA9254D0C17B024B6
A19FB07509DA75D4AB7A2649
9606C41F25EE03A6BBF4C7C5
3A379F822F4CB341698452C3
C8DF210DA53670A76E1591B4
49A28844DB47137224E99BB6
ECA3E126DA76D484C71AB5AA
4C68DDDC7E794772CAE2254E
894B2A65C7E795995132BFFE
5569EBE4464FB6609BB9A244
3CBF4C0FF7A632E234657EF1
8B94CC46FD935498892820DE
1B9148F6159E73ED312FBCE3
231C9C5B546D57647789F3BF
182C663444AF3A595CDBA08D
4F9611993746B7BE86DCC957
D008F02CC45B89B5946479E1
```

四 设计心得

通过这次大作业的练习让我们充分理解了加法器的运行原理，并在主动设计的过程中发现问题，并为减小延时和功耗进行电路，晶体管尺寸上的改进。这次大作业让我认识到设计一个具体的系统还是不容易的，我和我的组员经常 debug 到凌晨，而中途我的一场重病也严重拖延了我们小组的设计进程，以至于没有太多的时间再进行进一步的优化设计。我分析我们还可能优化的点如下：

1 对于传输门+曼切斯特加法器，我们可以全部采用传输门的形式。

2 对于镜像加法器，我们并没有像其他组一样查阅相关的文献，可能可以通过文献查阅获得更好的设计。

最后，我要特别感谢吴老师在我们做大作业的过程中对我们的细心指导和不辞辛劳的为我们组答疑解惑，在老师的帮助下我们组的设计得以优化和完善。

附：

文件夹中文件：

32bitAdder_Mirror.sp　　32bit 镜像加法器

32bitAdder_Manchester.sp　32bit 曼切斯特加法器

digital_file_IN.vec　　数字矢量输入文件