

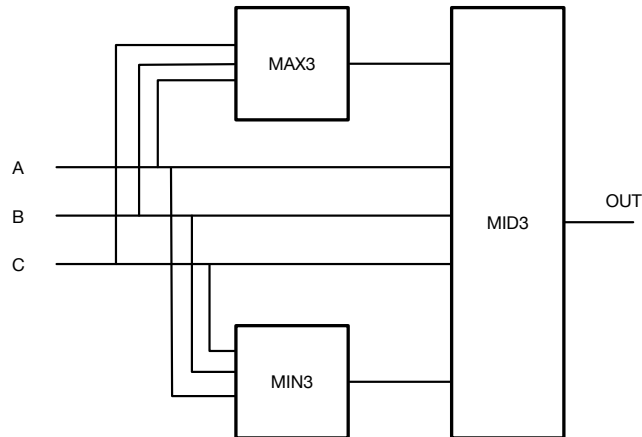
# ICS EXE 1

March 10, 2020

## 1 ICS (Organization)

### 1.1 HCL Programming: MID3

The reference manual of the Hardware Control Language can be found at [here](#). Read the material and design an hardware module MID3, which output the middle number among the 3 inputs ( $A \neq B \neq C$ ). Hint: you can use the help from MIN3 and MAX3, which has been introduced on the class.



```
1 word MIN3 = [  
2     A < B && A < C : A;  
3     B < A && B < C : B;  
4     1               : C;  
5 ];  
6  
7 word MAX3 = [  
8     A > B && A > C : A;  
9     B > A && B > C : B;  
10    1               : C;  
11 ];  
12  
13 word MID3 = [  
14     A != MIN3 && A != MAX3 : A;  
15     B != MIN3 && B != MAX3 : B;  
16     1                       : C;  
17 ];
```

You can use the hcl2c tool to validate your answer.

```
1  quote '#include <stdio.h>'
2  quote '#include <stdlib.h>'
3  quote 'char **data_names;'
4
5  wordsig A 'atoll(data_names[0])'
6  wordsig B 'atoll(data_names[1])'
7  wordsig C 'atoll(data_names[2])'
8  wordsig MIN3 'gen_MIN3()'
9  wordsig MAX3 'gen_MAX3()'
10
11 word MIN3 = [
12     A < B && A < C : A;
13     B < A && B < C : B;
14     1                : C;
15 ];
16
17 word MAX3 = [
18     A > B && A > C : A;
19     B > A && B > C : B;
20     1                : C;
21 ];
22
23 word MID3 = [
24     A != MIN3 && A != MAX3 : A;
25     B != MIN3 && B != MAX3 : B;
26     1                : C;
27 ];
28
29 quote 'int main(int argc, char *argv[]) {'
30 quote '    data_names = argv+1;'
31 quote '    printf("Out = %lld\n", gen_MID3());'
32 quote '    return 0;'
33 quote '}'
```

How to use hcl2c tool:

```
1  $ ./hcl2c < tst.hcl > tst.c
2
3  $ gcc tst.c -o tst
4
5  $ ./tst 4 5 6
6  Out = 5
```

## 1.2 HCL Programming: Adder

Figure 1 is a hardware module called HALF-ADDER, which can add two one-bit value  $A$  and  $B$ .  $S$  is the sum and  $C$  is the carry. Figure 2 is a hardware module named FULL-ADDER,

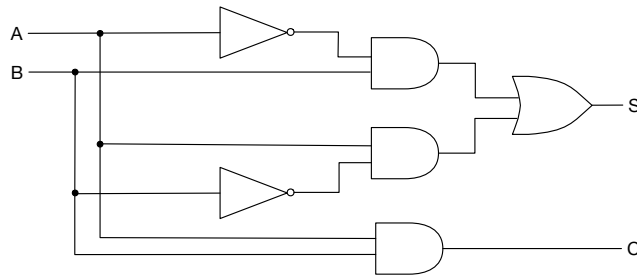


Figure 1: Half adder

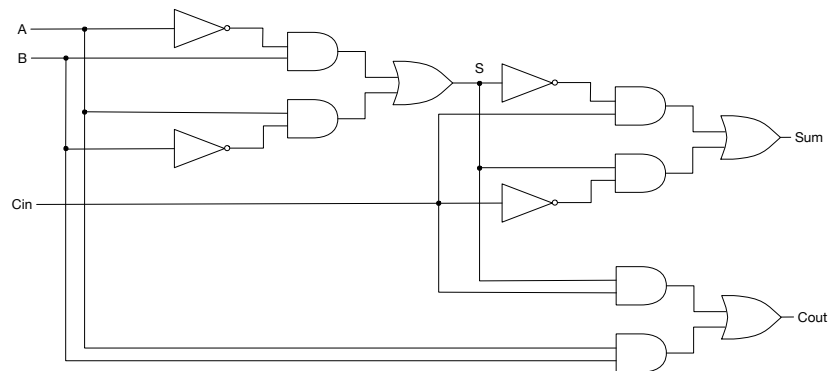


Figure 2: Full adder

You can write the HCL expression of  $C$  like this:

```
1 bool C = A && B;
```

### 1.2.1

Please write the HCL expression of  $S$ :

```
1 bool S = (!A && B) || (A && !B);
```

### 1.2.2

Please write the HCL expression of *Sum* (You can use A, B, and S now):

```
1 bool Sum = (!S && Cin) || (S && !Cin);
```

### 1.2.3

Please write the HCL expression of *Cout* (You can use A, B, and S now):

```
1 bool Cout = (S && Cin) || (A && B);
```

### 1.2.4

Now we have a 1-bit full adder as shown in figure 3. How to use this hardware module to construct a 4-bit full adder? Please draw it in figure 4.

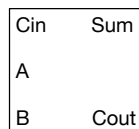


Figure 3: 1-bit full adder

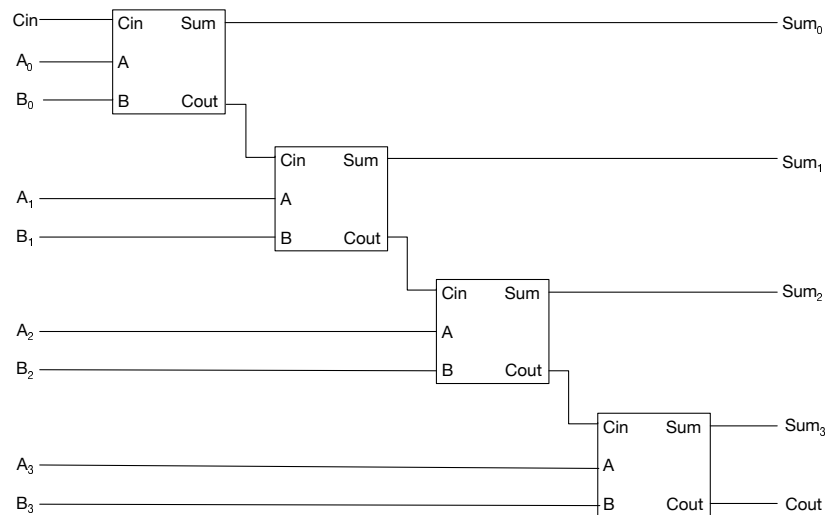


Figure 4: 4-bit full adder

## 2 ICS (System Software)

### 2.1 File

List 5 commonly used Unix system calls for file I/O. What's the relationship between them and those functions in the c standard I/O library?

ANS:

```
1 open
2 lseek
3 write
4 read
5 close
```

These are syscalls that directly operate on disk without buffering. Those functions in c library, like `fscanf`, `fprintf`, are buffered. They call these syscalls when flushing the buffer to corresponding file.

### 2.2 Virtual Memory

Morden computer's physical memory could reach 128GB or higher. Why do we still need virtual memory?

ANS: Virtual memory is not just about making memory larger. Virtual memory provides each process with a large, uniform, and private address space, thus simplifies memory management. It also protects the address space of each process from corruption by other processes.

### 2.3 Process

#### 2.3.1

What is the relationship between virtual memory and process?

Virtual memory provides each process with a private address space to isolate from other processes.

#### 2.3.2

What is the relationship between application and process?

An application is composed of one or more processes.

### 2.4 User&Kernel Mode

#### 2.4.1

What are the differences between user mode and kernel mode?

Privileged instructions can be executed only in kernel mode. Besides, kernel memory can be accessed only in kernel mode.

### 2.4.2

```
1  int array[10];  
2  void func1(void) {  
3      array[5000] = 40;  
4      return 0;  
5  }  
6  
7  void func2(void) {  
8      array[5] = 40;  
9      return 0;  
10 }
```

During the execution of the two functions given above, will the control flow changes from user mode to kernel mode?

Executing func1 will definitely trap to kernel mode, as a page fault occurs in an invalid address.

Executing func2 may trap to kernel mode. If the array hasn't been touched before, a page fault will occur. Interrupts may also happen during its execution.