CHAPTER 1 Basics

1.1 Solve using C++ rules and show binary numbers rules:

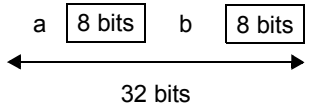
Show binary number answers:

1. evaluate
```
int a = 1;
int b = 2;
bool x = a && b;
bool y = a & b;
```

2. evaluate
```
char a = 0xF5;
char b = 0x0F;
char c = a & b;
char d = a | b;
char e = a ^ b;
char f = a && b;
char g = a || b;
```

assign a variable the data in Figure 1.1

FIGURE 1.1




```
char h = ~a;
```

3. pointers
```
int a = 1;
int **a = &a;
*aa = 3;
int **aaa = aa;
```

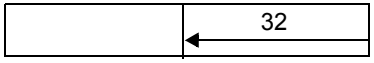
```
int b = 2;
int *b = b;

aa = b;
**aa = 4;
*aa = 32;
```

eval C
1. shift and mask
int x = 0x05F1;

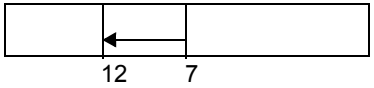
put x in a 64 bit var

FIGURE 1.2



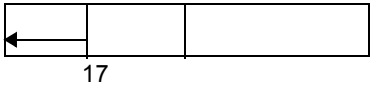
2. extract the 5 bits from bit 7 onward

FIGURE 1.3



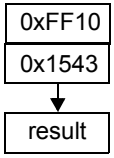
and insert in another var at bit 17

FIGURE 1.4



Use the following numbers: 0xFF10 and 0x1543 for the next tests (3,4,5):

FIGURE 1.5



3. AND the 2 numbers
4. OR the 2 numbers
5. XOR the 2 numbers

1.2 Verilog evaluation:

```
wire [31:0] x = 32'bFF00FF00
wire [31:0] y = 32'b0035F0F0
wire [31:0] y0 = y & x;
wire [31:0] y1 = x | y;
wire [31:0] y2 = x ^ y;
wire [31:0] y3 = x ~ y; legal ?
wire [31:0] y4 = x |~ y;
wire [31:0] y5 = x &~ y;
wire [31:0] y6 = x ^~ y;
wire [31:0] y7 = x || y;
wire [31:0] y8 = x && y;
wire [31:0] y9 = !x;
wire [31:0] y10 = x << 4;
wire [31:0] y10 = 1 << x;
wire [31:0] y11 = (x & 0xFF00) << 10;
wire [31:0] y12 = (x & 0xF00) << 7;
wire [31:0] y13 = (x | 0xF00) >> 3;
wire [31:0] y14 = (x | 0x0F) << 9;
```

put x in a reg:

FIGURE 1.6