Introduction to Hardware Design
Numbers: Figures and Code Snippets

Profs. Sundeep Rangan, Siddharth Garg

0.1 Truncation

```
    logic signed [W-1:0] a = A;


    logic signed [W1-1:0] a;
    logic signed [W2-1:0] b;

    // Assign with truncation
    b = a; // lower W2 bits of a assigned to b

    // Assign with saturation
    if (a > (1 << (W2 - 1)) - 1) begin
        b = (1 << (W2 - 1)) - 1;
    end else if (a < -(1 << (W2 - 1))) begin
        b = -(1 << (W2 - 1));
    end else begin
        b = a;
    end
```

Addition overflow

```
logic signed [7:0] a;
logic signed [10:0] b;
logic signed [8:0] c;

c = a + b; // potential overflow

// SV implments with signed extension
// then truncation
logic signed [10:0] cfull;
cfull = $signed(a) + $signed(b);
c = cfull[8:0]; // Truncation
```

Simulating truncation in python:

```
def truncate(x, n, signed=True):
    mask = (1 << n) - 1
    x = x & mask
    if signed:
        I = (x >= (1 << (n - 1)))
        x -= (1 << n)*I
    return x
```

0.2 Multipication

```
logic signed [8:0] a;   // 9-bit signed
logic signed [12:0] b;  // 13-bit signed
logic signed [21:0] c1;  // 22-bit signed
logic signed [20:0] c2;  // 21-bit signed

c1 = a * b; // No overflow
c2 = a * b; // Potential overflow
```