# Question 1

input [7:0] x, y;

wire [6:0] t0 = x[7:1]; // or equivalently t0 = x>>1 t0 = x div 2

wire [8:0] t1 = t0 + y; // (x div 2 mod 128 + y)

output [9:0] z = {t1, x[0]}; // bit-concatenation (x div 2 mod 128 + y)

**Question:** What is the word-level intent of the netlist output z (what is the polynomial representation for z)?

z=(x div 2+y)\*2 + (x mod 2) = (x

+2y)

# Question 2

input [7:0] x, y, z;

wire [7:0] t1 = x+y;

output [7:0] t2 = t1+z; // t2 = (x + y) + z

wire [7:0] t3 = x+z;

output [7:0] t4 = t3+y; // t4 = (x + z) + y

output [8:0] t5 = t1+z; // t5 = (x + y) + z

output [8:0] t6 = t3+y; // t6 = (x + z) + y

**Question:** Are t2 and t4 equivalent? Justify your answer.

T1 = (x+y) mod 256; T2 = ((x+y) mod 256 + z)mod256 = (x+y+z)mod256

T2 == T4

T5 = ((x+y) mod256 + z)mod512

T6 = ((x+z) mod256 + y)mod512

**Question:** Are t5 and t6 equivalent? Justify your answer.

x=1, y=3, z=2

T5 = (1+3)mod4 + 2 mod 8 = 2

T6 = (1+2)mod4 + 3 mod 8 = 6

# Question 3

Can you write a C/C++ function that takes in 10 numbers, and returns the sum and the mean?

//Int num = [a1,....,a10];

Int\* func(int num[10])

{

Int sum=0;

for(int i=0;i<10;i++)

Sum += num[i];

Int ss[2] = [sum,sum/10];

Return ss;

}

<bunch of other stuff>

\*(ss+0)

\*(ss+1)

Now imagine a hardware circuit implementing the above function. Can you think of some optimizations that a C synthesis tool would perform to generate optimal hardware from the code? Can you prove the correctness of those optimizations?

void func(int num[10], int &sum, int &mean)

{

sum=0;

for(sc\_int<4,unsigned> i=0;i<10;i++)

{Sum += num[i]; assert(i <= 10 );

}

assert(i == 10);

Mean = sum/10;

}