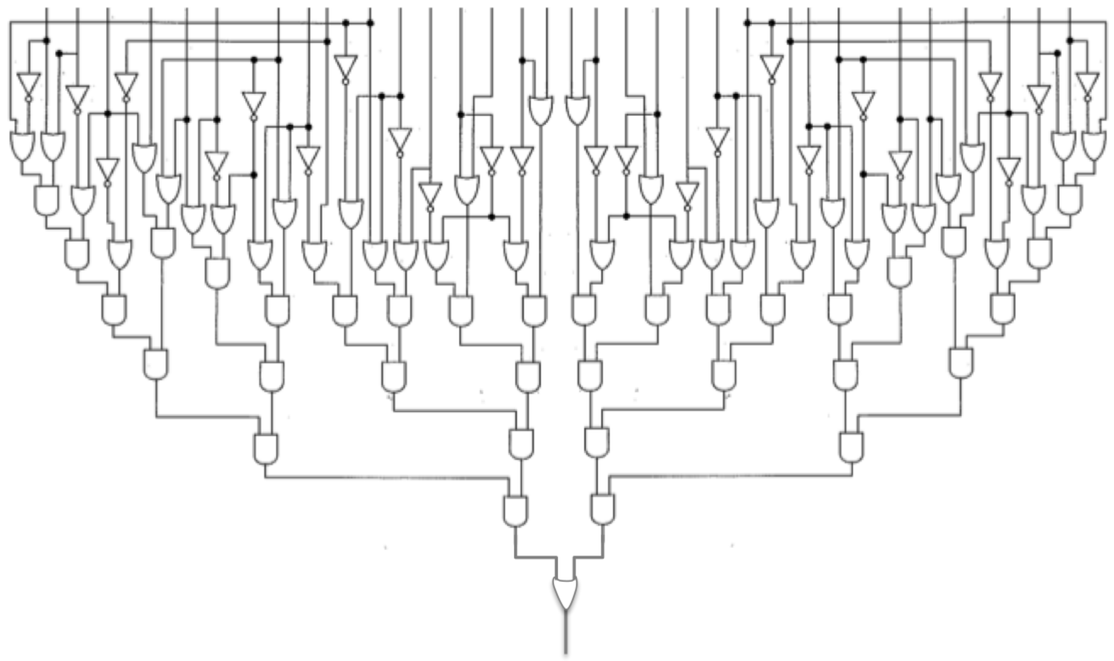
1. **(Using tree-structured communication 50 points or using serial communication 40 points)** The Circuit Satisfiability Problem for a given binary circuit is to find the set of inputs that cause that circuit to produce '1' as its output. For example, here is a 32-bit circuit diagram:



Download the program circuitSatisfiability.c.

i. Using your favorite text editor, modify circuitSatisfiability.c so that it uses the MPI_Wtime() function to time the computation:

```
...
double startTime = 0.0, totalTime = 0.0;
startTime = MPI_Wtime();
for (i = 0; i < UINT_MAX; i++) {
    count += checkCircuit(id, i);
}
totalTime = MPI_Wtime() - startTime;
printf("Process %d finished in time %f secs.\n", id, totalTime);
...
```

With these modifications, the program will self-report how long it took to check the circuit.

ii. Use MPI's version of the reduction pattern to sum the distributed processes' count-values into a global count, and have process 0 output this global count.

iii. When your program compiles correctly and produces the correct results, time its execution with 1, 2, 4, 8, and 16 processes. Create a line-chart of your execution times, with the number of processes on the X-axis and time on the Y-axis. Label the scale of your X-axis with the number of processes (1, 2, 4, 8, and 16) so that this axis indicates the scale at which measurements were taken.

2. **(Using tree-structured communication 50 points or using serial communication 40 points)**

   Suppose we toss darts randomly at a square dartboard, whose bullseye is at the origin, and whose sides are 2 feet in length. Suppose also that there's a circle inscribed in the square dartboard. The radius of the circle is 1 foot, and it's area is π square feet. If the points that are hit by the darts are uniformly distributed (and we always hit the square), then the number of darts that hit inside the circle should approximately satisfy the equation

   $$\frac{\text{number in circle}}{\text{total number of tosses}} = \frac{\pi}{4},$$

   since the ratio of the area of the circle to the area of the square is $\pi/4$.

   We can use this formula to estimate the value of π with a random number generator:

```
number_in_circle = 0;
for (toss = 0; toss < number_of_tosses; toss++) {
   x = random double between −1 and 1;
   y = random double between −1 and 1;
   distance_squared = x*x + y*y;
   if (distance_squared <= 1) number_in_circle++;
}
pi_estimate = 4*number_in_circle/((double) number_of_tosses);
```

   This is called a "Monte Carlo" method, since it uses randomness (the dart tosses).

   Write an MPI program that uses a Monte Carlo method to estimate π. Process 0 should read in the total number of tosses and broadcast it to the other processes. Use `MPI_Reduce` to find the global sum of the local variable `number_in_circle`, and have process 0 print the result. You may want to use **long long int**s for the number of hits in the circle and the number of tosses, since both may have to be very large to get a reasonable estimate of π.

Please upload your report and source code to Moodle.