Elia Deppe
CMSC 411
Prof Russ Cain
Group Members:
Elia Deppe
Dylan Demchuk
Brad Harmening
Daniel Ackerson

The cordic program I designed can take values from 0 to 90 degrees, and find sin(θ), cos(θ), and tan(θ); however I was only able to calculate the CPI and execution time for sin(θ) and cos(θ). This program stores the values into the memory spots x and y for the integer value found for cos(θ) and sin(θ) respectively when multiplied by the factor 2^{20}. This integer value divided by 2^{20} gives the actual decimal value for for cos(θ) and sin(θ). I demonstrate this later in the program by using floating point division and storing those values in memory locations sin, cos, and tan, but I was unable to find CPI for this latter part of the program.

Using this algorithm, cos(θ) and sin(θ) are calculated at the same time and thus share the same execution time. The initial start of the program loads the values needed: x is the cordic gain for the 27 arctan values 2^20 (scaled up), y is zero, i (the loop counter) is set to zero, and the angle that will change in the loop and the desired angle are both set to registers. The first value from the arctan table is also loaded.

The loop of the program executes as such:

```
start:
        if desired angle > current angle
                call greater than
        else
                call less than

greater than:
        set x'' and y'' to zero
        x'' = x - (y >> i)
        y'' = y + (x >> i)
        current angle = current angle + arctan angle
        increment i
        set arctan angle to next value in memory
        x = x''
        y = y''
        jump to check

less than:
        set x'' and y'' to zero
        x'' = x + (y >> i)
```

```
        y'' = y - (x >> i)
        current angle = current angle - arctan angle
        increment i
        set arctan angle to next value in memory
        x = x''
        y = y''
        jump to check

check:
        if i < 27
                jump to start
        else
                store x to memory location x
                store y to memory location y
```

The rest of the program utilizing floating point simply transfers the values into sflop registers, divides them by 2^{20} (getting the actual values for sin(θ) and cos(θ)) and then divides sin(θ) / cos(θ) to get tan(θ). These values are then stored.

CPI

Finding CPI, I used the information found in the ARM7TDMI (which the ARMSIM is based off of) documentation describing clock cycles found here
http://infocenter.arm.com/help/topic/com.arm.doc.ddi0210c/DDI0210B.pdf

The start of the program has 6 load operations and 3 data operations. The loop has 9 data operations, 2 data operations + shift, 1 load operation, and 2.5 branches (worst case; there is 1 branch in greater than and not in less than). The final part of storing is two load operations and 2 store operations. Important thing to note is the loop executes 27 times, so the number of operations are all multiplied by 27. This leads to:
data op → 246
data op + shift → 54
branch → 68
load → 35
store → 2
Total → 405 operations

CPI = summation of op_i * $cycle_i$ / total operations

$$\frac{246 * 1 + 54 * 2 + 68 * 3 + 35 * 3 + 2 * 2}{405} = 1.64691\ CPI$$

Finding execution time for various clock speeds of 35kHz, 1 MHz, and 1 GHz.
Execution time = instructions/program * CPI * seconds/clock

Execution time of 35 kHz ⇒ 405 * 1.64691 * (1/(35*10^3)) = .02084375 seconds
Execution time of 1 MHZ ⇒ 405 * 1.64691 * (1/(1*10^6)) = .000667seconds
Execution time of 1 GHZ ⇒ 405 * 1.64691 * (1/(1*10^9)) = .000000667 seconds

Main Resource used for this project: http://bsvi.ru/uploads/CORDIC--_10EBA/cordic.pdf