

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 $\sin(\theta)$, $\cos(\theta)$, and $\tan(\theta)$; however I was only able to calculate the CPI and execution time for $\sin(\theta)$ and $\cos(\theta)$. This program stores the values into the memory spots x and y for the integer value found for $\cos(\theta)$ and $\sin(\theta)$ respectively when multiplied by the factor 2^{20} . This integer value divided by 2^{20} gives the actual decimal value for $\cos(\theta)$ and $\sin(\theta)$. 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, $\cos(\theta)$ and $\sin(\theta)$ 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 $\sin(\theta)$ and $\cos(\theta)$) and then divides $\sin(\theta) / \cos(\theta)$ to get $\tan(\theta)$. 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 \rightarrow 246

data op + shift \rightarrow 54

branch \rightarrow 68

load \rightarrow 35

store \rightarrow 2

Total \rightarrow 405 operations

$CPI = \text{summation of } op_i * cycle_i / \text{total operations}$

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

Storing the sin and cos values as their actual decimal values and finding tan. Assume that vmov takes 5 cycles, vldr takes 3, vdiv takes 100, and vstr takes 5 cycles. These are estimations since I wasn't able to find any actual values for the ARM7 TDMI or on ARMSim's site.

vmov \rightarrow 2

ldr \rightarrow + 4

vldr \rightarrow 1

vdiv \rightarrow 3

vstr \rightarrow 3

$405 + 2 + 4 + 1 + 3 + 3 = 418$

$$\frac{246 * 1 + 54 * 2 + 68 * 3 + 39 * 3 + 2 * 2 + 2 * 5 + 1 * 3 + 3 * 100 + 3 * 5}{418}$$

Which comes to 2.409090909 CPI to find actual decimal values of sin/cos and tan

Finding execution time for various clock speeds of 35 KHz, 1 MHz, and 1 GHz.

Execution time = instructions/program * CPI * seconds/clock

Execution time of 35 kHz $\Rightarrow 405 * 1.64691 * (1/(35*10^3)) = .02084375$ seconds

Execution time of 1 MHZ $\Rightarrow 405 * 1.64691 * (1/(1*10^6)) = .000667$
seconds

Execution time of 1 GHZ $\Rightarrow 405 * 1.64691 * (1/(1*10^9)) = .000000667$
seconds

With 2.4090909 CPI

Execution time of 35 kHz $\Rightarrow 418 * 2.4090909 * (1/(35*10^3)) =$
.02877142857 seconds

Execution time of 1 MHZ $\Rightarrow 418 * 2.4090909 * (1/(1*10^6)) = .001007$
seconds

Execution time of 1 GHZ $\Rightarrow 418 * 2.4090909 * (1/(1*10^9)) = .000001007$
seconds

Main Resource used for this project:

http://bsvi.ru/uploads/CORDIC--_10EBA/cordic.pdf