IQ Math on the Texas Instruments TMS320C28x DSP

"Virtual" Floating-Point Programming On A 32-Bit Fixed-Point Machine

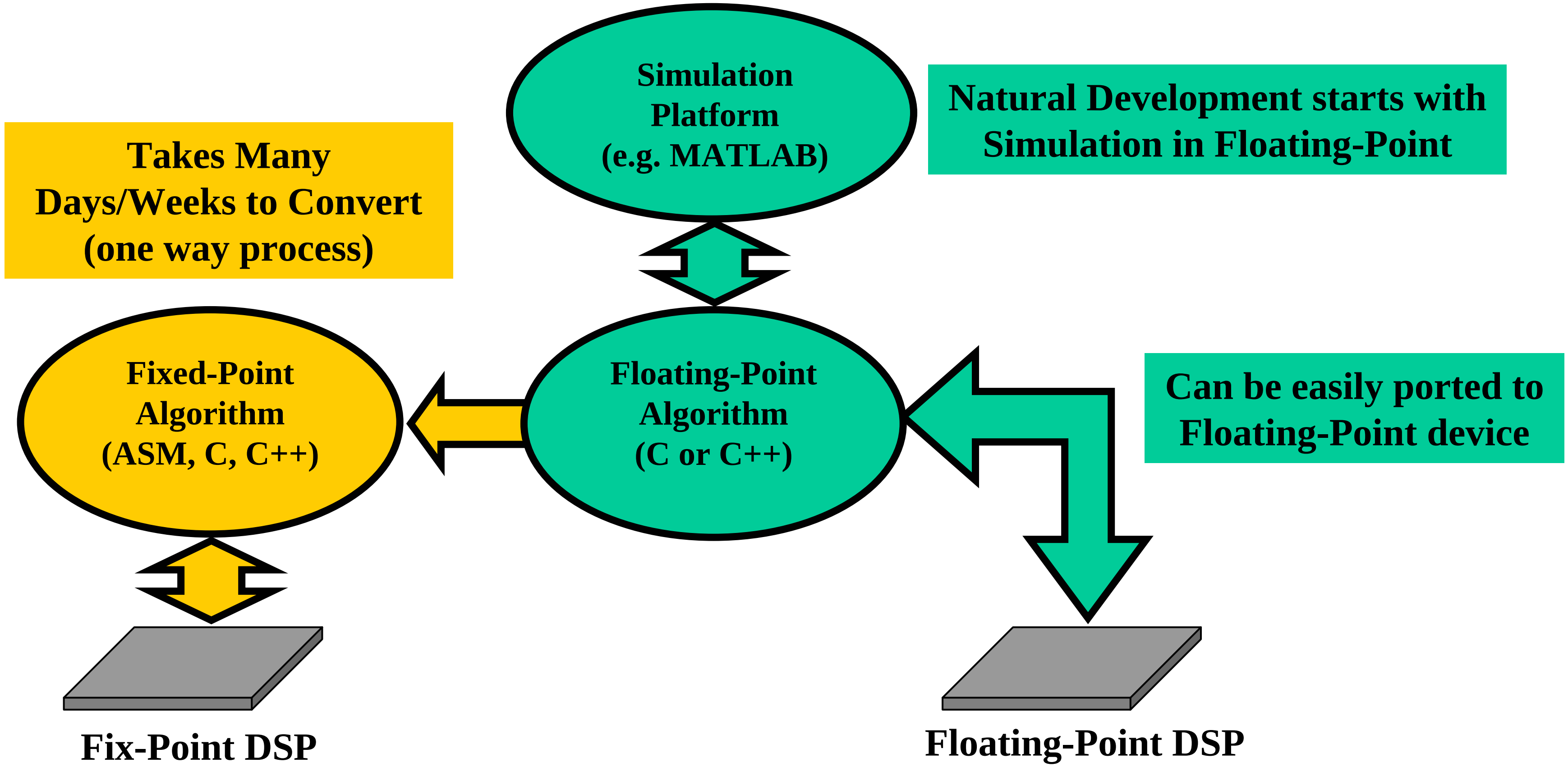
David M. Alter

Senior Member Technical Staff
Texas Instruments Inc.

IQmath Agenda

- The Fixed-Point DSP Development Dilemma
- Floating-Point vs. IQmath Representation
- The IQmath Approach and How it Addresses the Problem
- AC Induction Motor Application Example
- Additional Tidbits and Summary

The Fix-Point Development Dilemma



IQ math can help solve this problem!

IQmath Agenda

- The Fixed-Point DSP Development Dilemma
- Floating-Point vs. IQmath Representation
- The IQmath Approach and How it Addresses the Problem
- AC Induction Motor Application Example
- Additional Tidbits and Summary

IEEE-754 Single Precision Floating Pt.

31	30 … 23	22 … 0
s	eeeeeeee	fffffffffffffffffffffff
sign bit	8 bit exponent	23 bit mantissa (fraction)

Case 1: if $e = 255$ and $f \neq 0$, then v = NaN (not-a-number)

Case 2: if $e = 255$ and $f = 0$, then $v = (-1^s)*\text{infinity}$

Case 3: if $0 < e < 255$, then $v = (-1^s)*[2^{(e-127)}]*(1.f)$

Case 4: if $e = 0$ and $f \neq 0$, then $v = (-1^s)*[2^{-126}]*(0.f)$

Case 5: if $e = 0$ and $f = 0$, then $v = (-1^s)*0$

Example: 0x41200000 = 0 | 100 0001 0 | 010 0000 0000 ... 0000 | b

s | $e = 130$ | $f = 2^{-2} = 0.25$

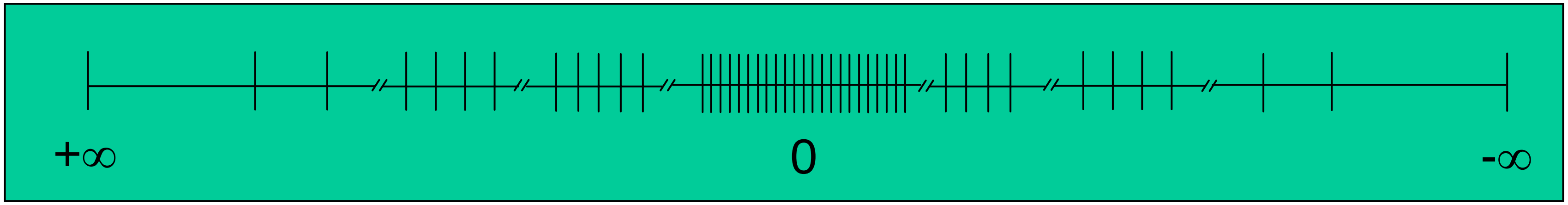
$\Rightarrow$ Case 3 $v = (-1^0)*2^{(130-127)}*1.25 = 10.0$

Advantage: Exponent gives large dynamic range.

Disadvantage: Precision of a number depends on its exponent.

Number Line Insight

Floating Point:



Non-uniform distribution

- **Precision greatest near zero**
- **Less precision the further you get from zero**

IQ Fractional Representation

31 0

`S IIIIIII.ffffffffffffffffffffffff`

← 32 bit mantissa →

$$-2^{I} + 2^{I-1} + \ldots + 2^{1} + 2^{0} \bullet 2^{-1} + 2^{-2} + \ldots + 2^{-Q}$$

I8Q24 Example: 0x41200000

= 0100 0001 . 0010 0000 0000 0000 0000 0000 b

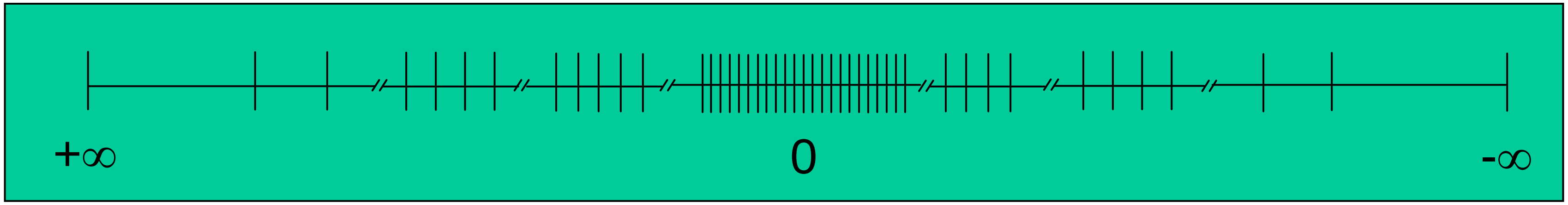
= $2^{6} + 2^{0} + 2^{-3}$ = 65.125

Advantage: Same precision for all numbers in an IQ format.

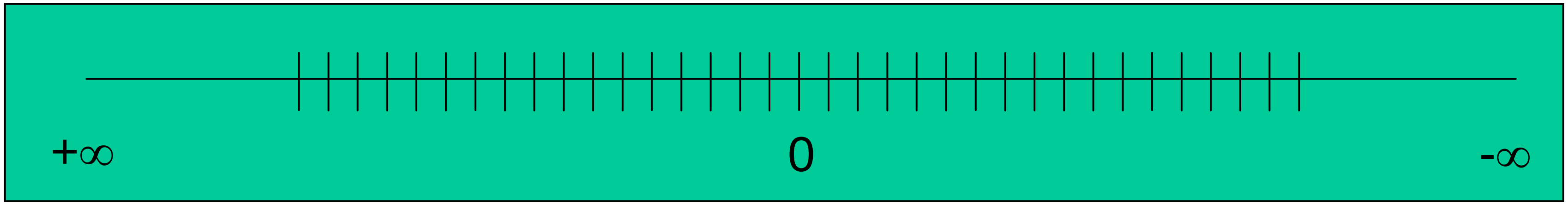
Disadvantage: Limited dynamic range versus to floating point.

Number Line Insight
Distributions

Floating Point: non-uniform distribution (variable precision)



IQ Fractions: uniform distribution (same precision everywhere)



Both floating-point and IQ formats have 2^{32} possible values on the number line. It's how each distributes these values that differs.

Floating Point doesn't Solve Everything!

Floating Point Example

	x =	10.0	`(0x41200000)`
+	y =	0.000000238	`(0x347F8CF1)`
	z =	10.000000238	WRONG!

Single-precision floating point cannot represent 10.000000238

0x412000000	=	10.000000000
		10.000000238 – doesn't exist!
0x412000001	=	10.000000950

So z gets rounded down to 10.000000000.

IQ Math Can Handle It!

I8Q24
Example

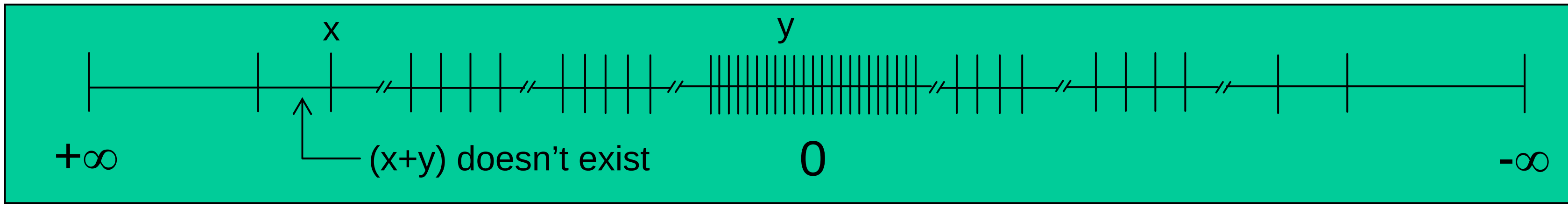
	x =	10.0	(0x0C000000)
+	y =	0.000000238	(0x00000004)
	z =	10.000000238	(0x0C000004)

Exact Result!

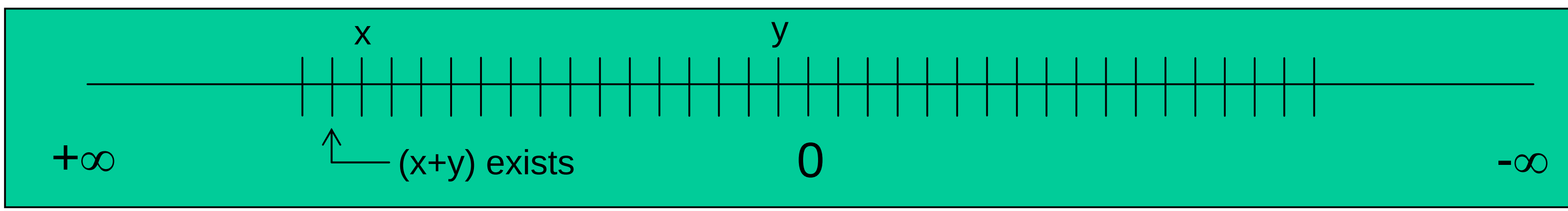
What’s Happening Here?

Number Line Insight: Closed Sets under Addition

Floating Point: Not Closed under Addition



IQ Fractions: Closed under Addition



<u>Definition:</u>

Given a set S with x$\in$S, y$\in$S, and z := (x + y), S is called *closed under addition* if z$\in$S $\forall$ x, y where min(S) < z < max(S).

IQmath Agenda

- The Fix-Point DSP Development Dilemma
- Floating Point vs. IQmath Representation
- The IQmath Approach and How it Addresses the Problem
- AC Induction Motor Application Example
- Additional Tidbits and Summary

IQ vs. Q Fractional Formats

- IQ and Q both refer to fractional interpretations of a binary number.
- The term Q generally refers to a fraction that has no integer bits (other than the sign bit), such that its range is between -1 and +1.
- The term IQ refers to a fraction that has one or more integer bits. These integer bits increase the range of the format beyond ±1 at the expense of an equal number of bits of precision.

IQ format: (I8Q24 shown)

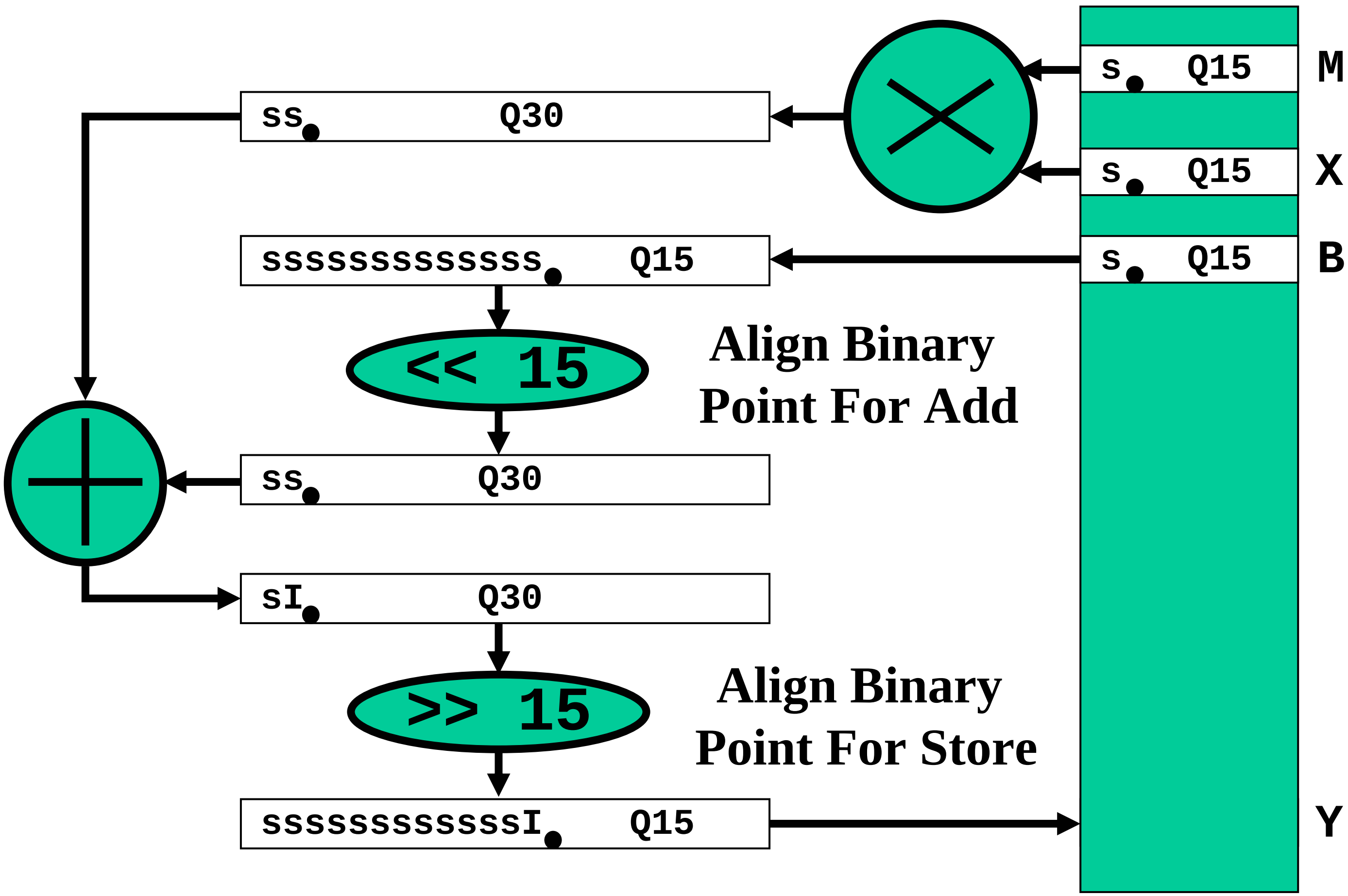
31	0
S IIIIIII.ffffffffffffffffffffffff	

Traditional 16-bit Q format:

15	0
S . fffffffffffffff	

Traditional 16-bit "Q" Math Approach

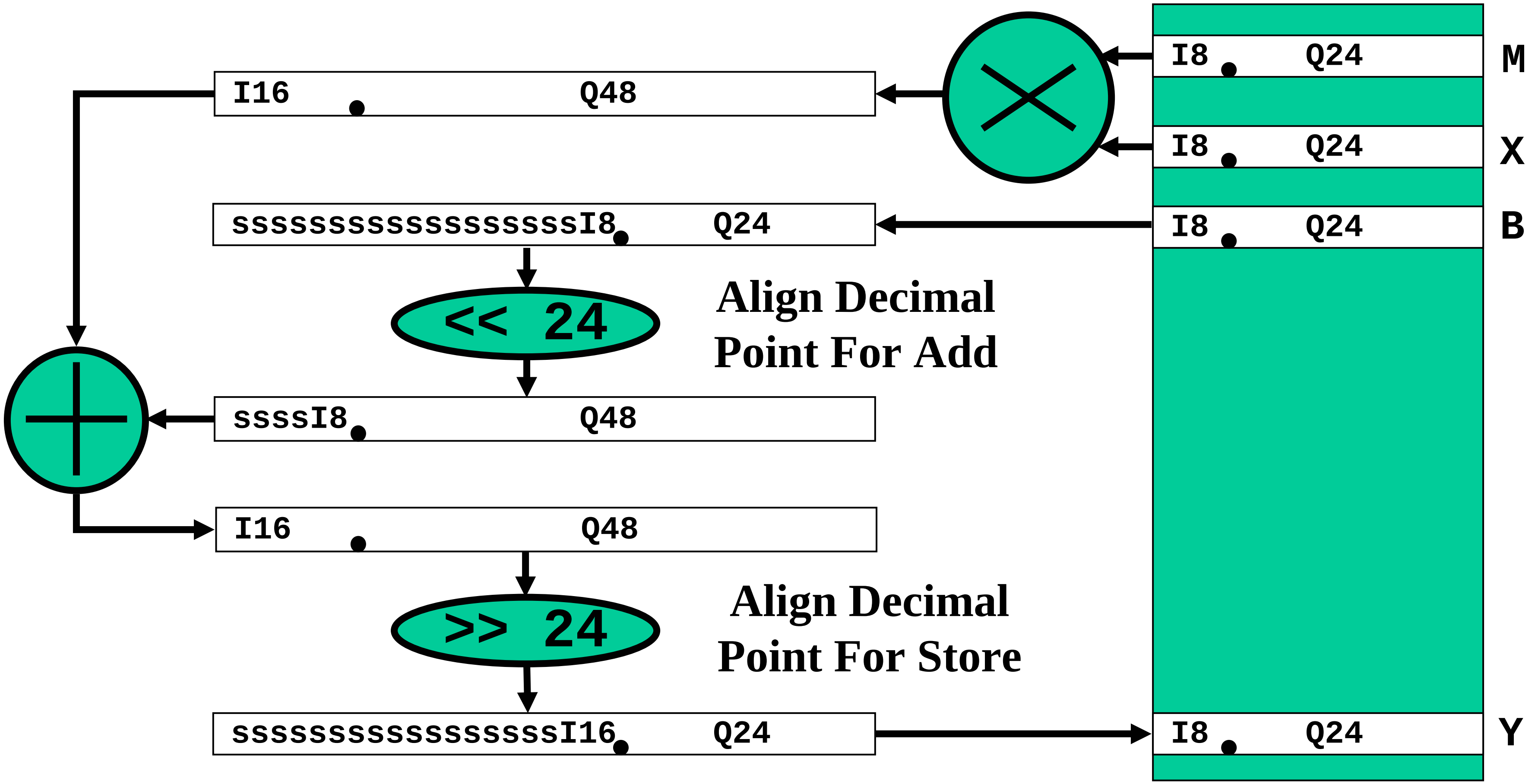
Example: y = mx + b



in C: `Y = ((i32) M * (i32) X + (i32) B << Q) >> Q;`

"Q" Math Approach Applied to 32-bit Numbers

Example: y = mx + b

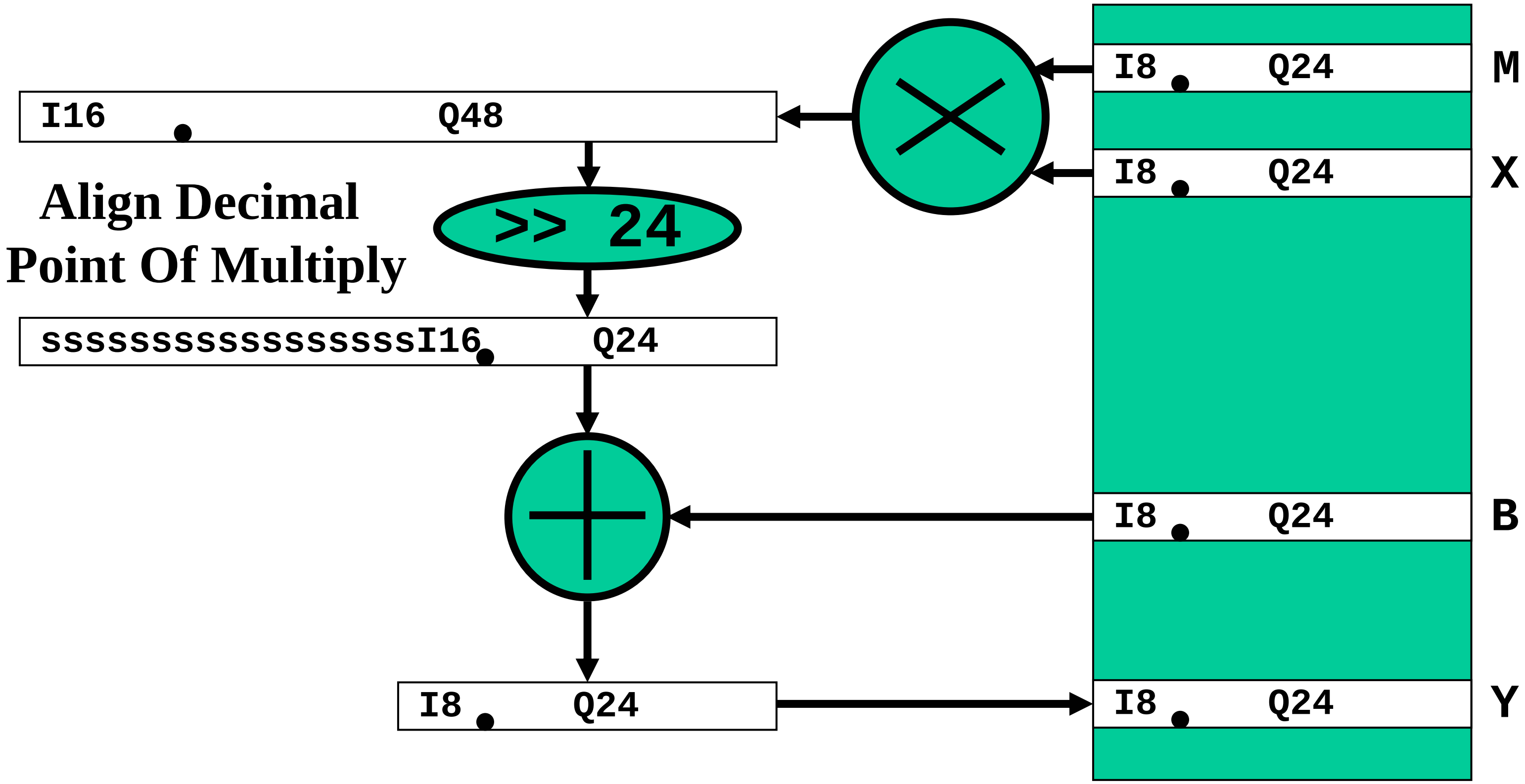


in C: `Y = ((i64) M * (i64) X + (i64) B << Q) >> Q;`

Note: Requires Support For 64-bit Integer Data Type In Compiler

32-bit IQmath Approach

Example: y = mx + b



in C:
```
Y = ((i64) M * (i64) X) >> Q + B;
```

IQmath: The Multiply

```
Y = ((i64) M * (i64) X) >> Q + B;
```

Redefine The Multiply Operation As Follows:

```
_IQmpy(M,X) == ((i64) M * (i64) X) >> Q
```

This Simplifies The Equation As Follows:

```
Y = _IQmpy(M,X) + B;
```

C28x Compiler Supports "_IQmpy()" Intrinsic. Assembly Code Generated:

5 cycle basic IQ multiply operation using C-code.

```
1  MOVL    XT,@M            ; XT  = M
1  IMPYL   P,XT,@X          ; P   = lo 32-bits of M*X
2  QMPYL   ACC,XT,@X        ; ACC = hi 32-bits of M*X
1  LSL64   ACC:P,#(32-Q)    ; ACC = ACC:P << 32-Q

1  ADDL    ACC,@B           ; Add B
1  MOVL    @Y,ACC           ; Store result
```

IQmath: It looks like floating-point!

Floating-Point

```
float  Y, M, X, B;

Y = M * X + B;
```

Traditional Fix-Point Q

```
long Y, M, X, B;

Y = ((i64) M * (i64) X + (i64) B << Q)) >> Q;
```

"IQmath" In C

```
_iq  Y, M, X, B;

Y = _IQmpy(M, X) + B;
```

"IQmath" In C++

```
iq  Y, M, X, B;

Y = M * X + B;
```

IQmath can take advantage of the operator overloading feature of C++

IQmath Library v1.4: Math & Trig Functions

Operation	Floating-Point	“IQmath” in C	“IQmath” in C++
type	float A, B;	_iq A, B;	iq A, B;
constant	A = 1.2345	A = _IQ(1.2345)	A = IQ(1.2345)
multiply	A * B	_IQmpy(A , B)	A * B
divide	A / B	_IQdiv (A , B)	A / B
add	A + B	A + B	A + B
subtract	A - B	A - B	A – B
boolean	>, >=, <, <=, ==, |=, &&, ||	>, >=, <, <=, ==, |=, &&, ||	>, >=, <, <=, ==, |=, &&, ||
trig functions	sin(A),cos(A) sin(A*2pi),cos(A*2pi) atan(A),atan2(A,B) atan2(A,B)/2pi sqrt(A),1/sqrt(A) sqrt(A*A + B*B)	_IQsin(A), _IQcos(A) _IQsinPU(A), _IQcosPU(A) _IQatan(A), _IQatan2(A,B) _IQatan2PU(A,B) _IQsqrt(A), _IQisqrt(A) _IQmag(A,B)	IQsin(A),IQcos(A) IQsinPU(A),IQcosPU(A) IQatan(A),IQatan2(A,B) IQatan2PU(A,B) IQsqrt(A),IQisqrt(A) IQmag(A,B)
saturation	if(A > Pos) A = Pos if(A < Neg) A = Neg	_IQsat(A,Pos,Neg)	IQsat(A,Pos,Neg)

Accuracy Of Functions/Operations Approximately 28 to 31 bits

IQmath Library v1.4: Conversion Functions

Operation	Floating-Point	"IQmath" in C	"IQmath" in C++
iq to iqN	A	_IQtoIQN(A)	IQtoIQN(A)
iqN to iq	A	_IQNtoIQ(A)	IQNtoIQ(A)
integer(iq)	(long) A	_IQint(A)	IQint(A)
fraction(iq)	A – (long) A	_IQfrac(A)	IQfrac(A)
iq = iq*long	A * (float) B	_IQmpyI32(A,B)	IQmpyI32(A,B)
integer(iq*long)	(long) (A * (float) B)	_IQmpyI32int(A,B)	IQmpyI32int(A,B)
fraction(iq*long)	A - (long) (A * (float) B)	_IQmpyI32frac(A,B)	IQmpyI32frac(A,B)
qN to iq	A	_QNtoIQ(A)	QNtoIQ(A)
iq to qN	A	_IQtoQN(A)	IQtoQN(A)
string to iq	atof(char)	_atoIQ(char)	atoIQ(char)
IQ to float	A	_IQtoF(A)	IQtoF(A)

Most of the above "Functions" are C macros, and compile very efficiently.

IQmath: The GLOBAL_Q Simplification

Based On The Required Dynamic Range Or Resolution

GLOBAL_Q	Max Val	Min Val	Resolution
28	7.999 999 996	-8.000 000 000	0.000 000 004
24	127.999 999 94	-128.000 000 00	0.000 000 06
20	2047.999 999	-2048.000 000	0.000 001

The user selects a "Global Q" value for the entire application:

```
#define  GLOBAL_Q  24    // set in "IQmathLib.h" file
_iq  Y, M, X, B;
Y = _IQmpy(M,X) + B;     // all values are in I8Q24
```

The user can also explicitly specify the IQ value to use:

```
_iq20  Y, M, X, B;
Y = _IQ20mpy(M,X) + B;   // all values are in I12Q20
```

IQmath: The MATH_TYPE Constant

One Source Code Set Targets Fixed or Floating Point Devices

```
Y = _IQmpy(M, X) + B;
```

User Selects Target Math Type (in "IQmathLib.h" file)

```
#if MATH_TYPE == IQ_MATH
```

Compile & Run Using "IQmath" On C28x

```
#if MATH_TYPE == FLOAT_MATH
```

```
Y = (float)M * (float)X + (float)B;
```

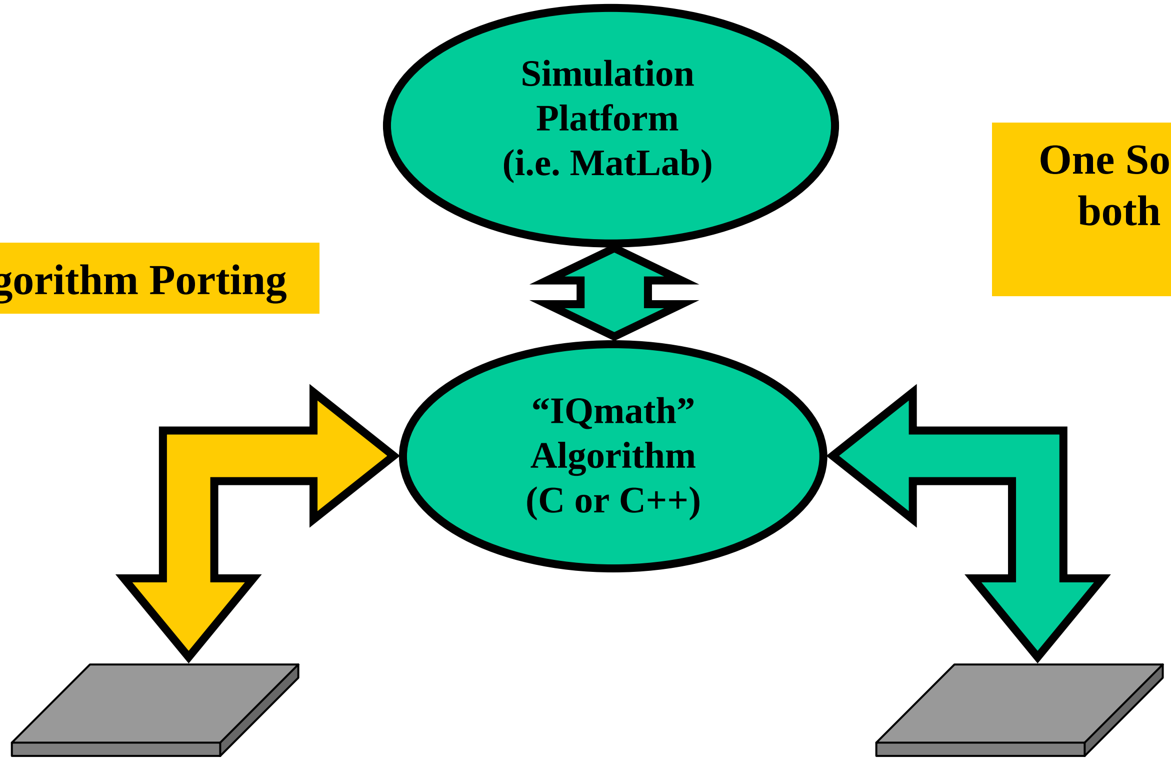
Compile & Run Using Floating-Point Math on any floating point compiler

All "IQmath" Operations Have An Equivalent Floating-Point Operation

The Fixed Point Development Dilemma Solved with IQmath!

Speedy Algorithm Porting

One Source Code Set for both Simulation and Hardware



Fix-Point DSP

(i.e. C28x)

Floating-Point DSP

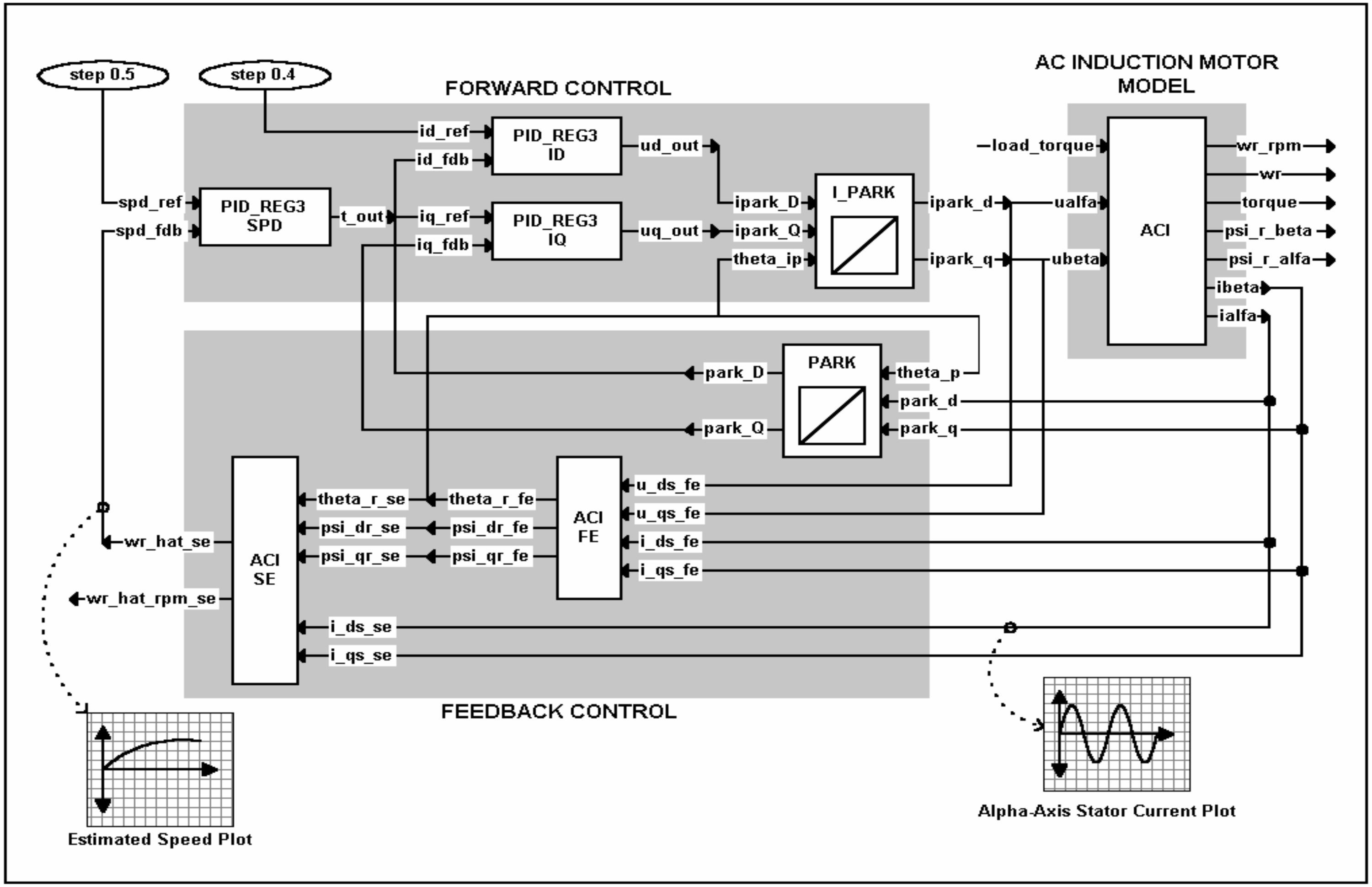
(i.e. C3x, C67x)

IQmath Agenda

- The Fixed-Point DSP Development Dilemma
- Floating-Point vs. IQmath Representation
- The IQmath Approach and How it Addresses the Problem
- AC Induction Motor Application Example
- Additional Tidbits and Summary

AC Induction Motor Example:

- **Sensorless, ACI induction machine direct rotor flux control**
  (one of the more complex motor control techniques)
- **Goal: Motor speed estimation & Alpha-Axis Stator current estimation**



AC Induction Motor Example

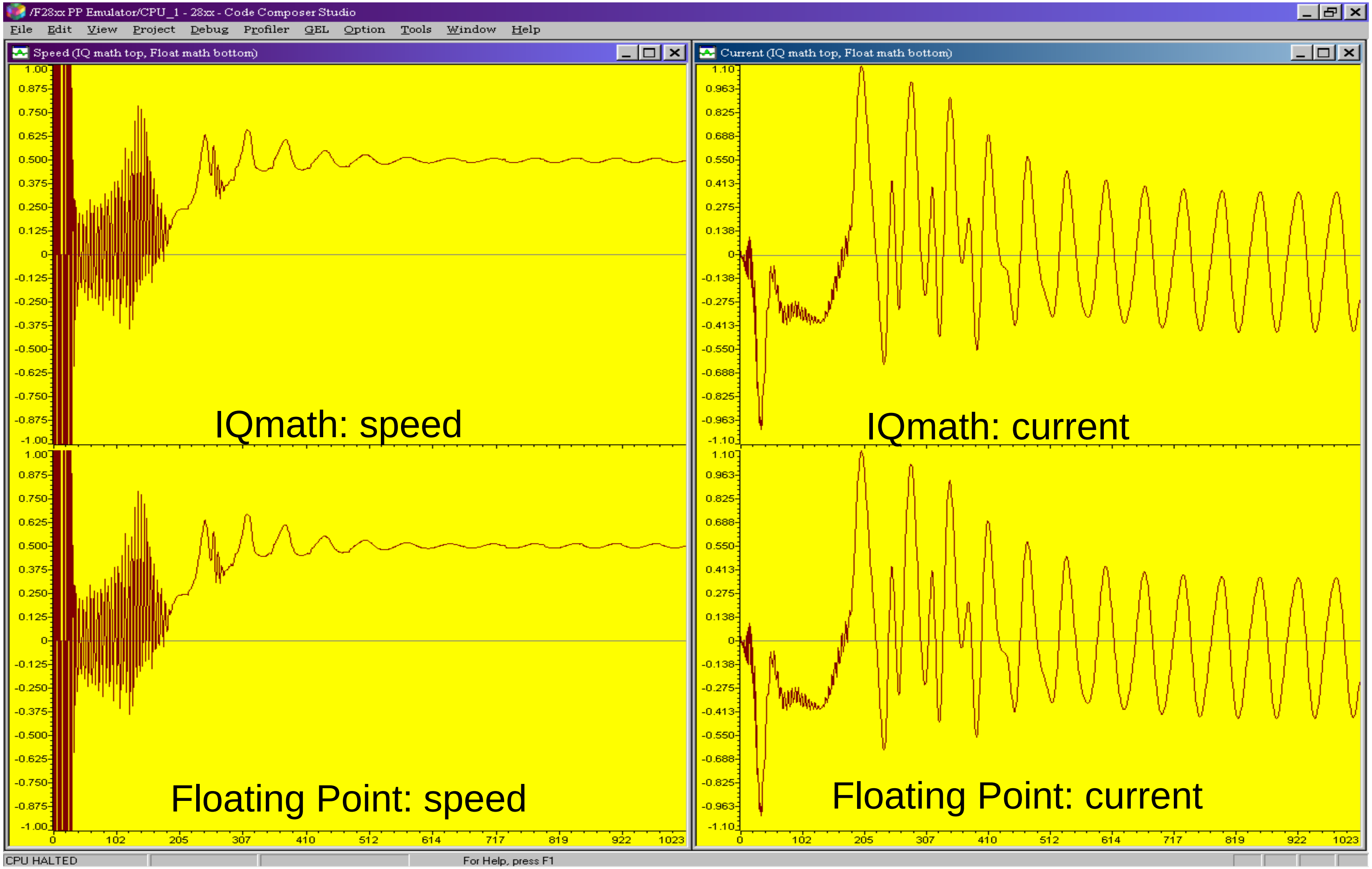
Converting to “IQmath” C Code

```
#include “math.h”
#include  “IQmathLib.h”
#define  TWO_PI   _IQ(6.28318530717959)
void park_calc(PARK *v)
{
     _iq   cos_ang , sin_ang;
     sin_ang = _IQsin(_IQmpy(TWO_PI , v->ang));
     cos_ang = _IQcos(_IQmpy(TWO_PI , v->ang));

     v->de = _IQmpy(v->ds , cos_ang) + _IQmpy(v->qs , sin_ang);
     v->qe = _IQmpy(v->qs , cos_ang) - _IQmpy(v->ds , sin_ang);
}
```

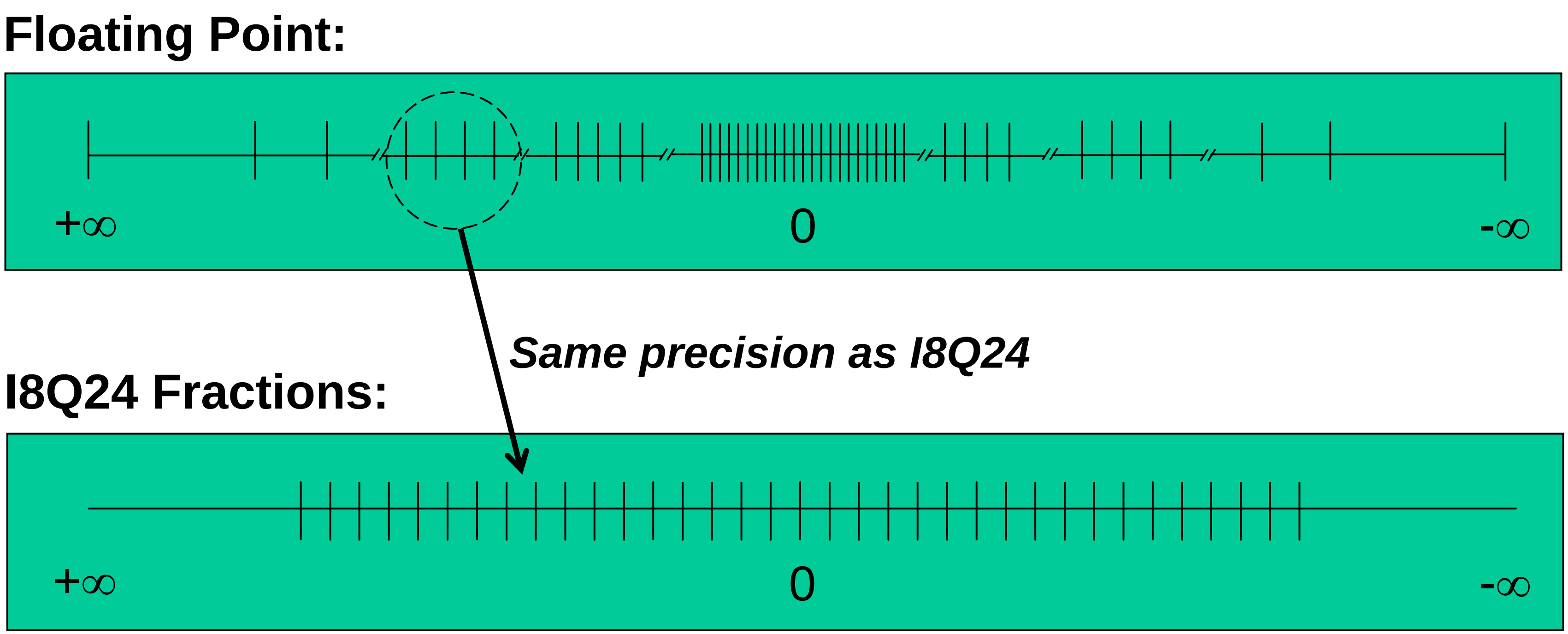
AC Induction Motor Example:
Converting to “IQmath” C++ Code

```
#include “math.h”
extern “C” { #include “IQmathLib.h” }
#include “IQmathCPP.h”
#define  TWO_PI  IQ(6.28318530717959)
void park_calc(PARK *v)
{
     iq    cos_ang , sin_ang;
     sin_ang = IQsin(TWO_PI * v->ang);
     cos_ang = IQcos(TWO_PI * v->ang);

     v->de = (v->ds * cos_ang) + (v->qs * sin_ang);
     v->qe = (v->qs * cos_ang) - (v->ds * sin_ang);
}
```

AC Induction Motor Example:
GLOBAL_Q = 24, system stable



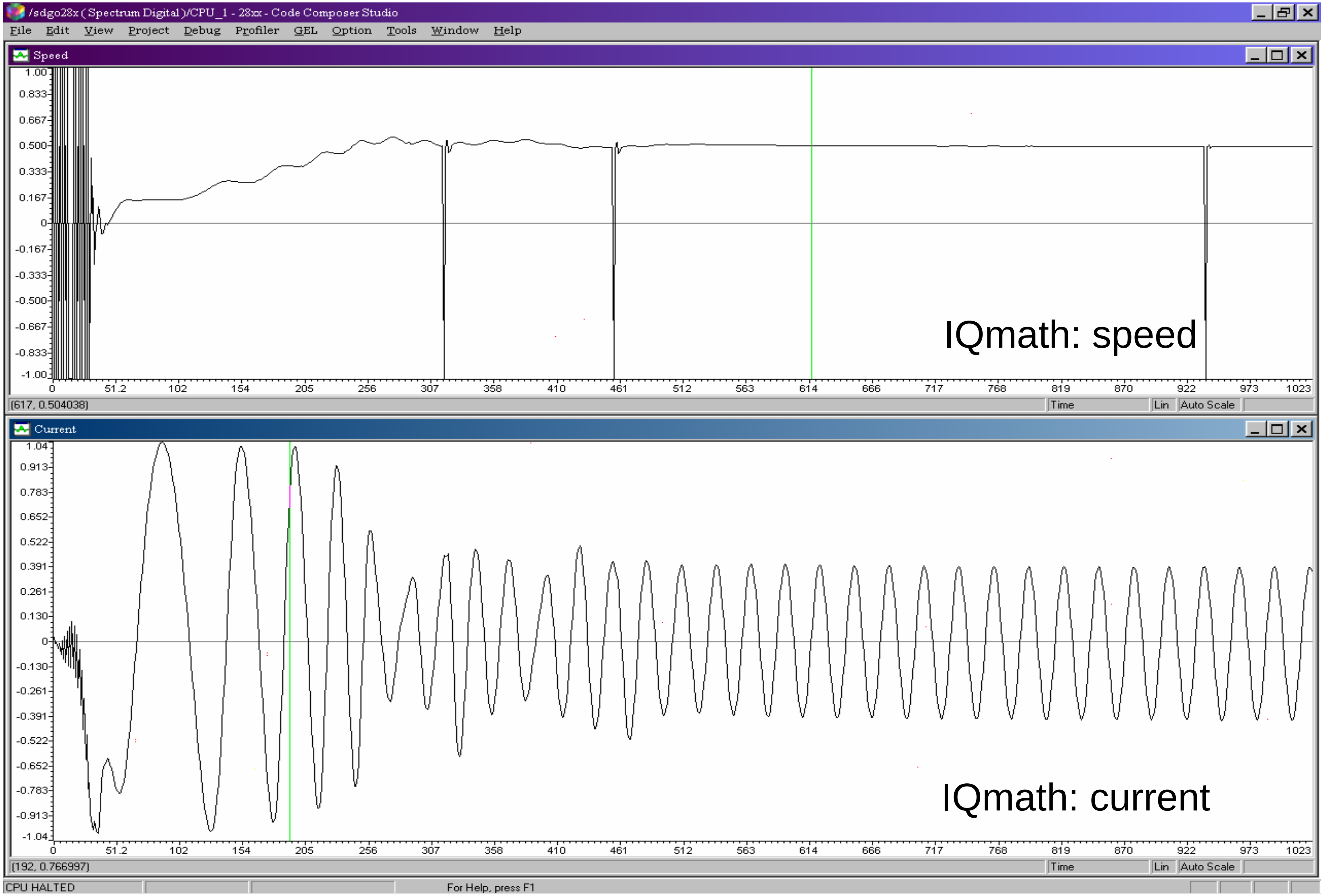
What's Happening Here?
Equal Precision in the Computation Region



In the region where these particular computations occur, the precision of single-precision floating point just happens to equal the precision of the I8Q24 format.

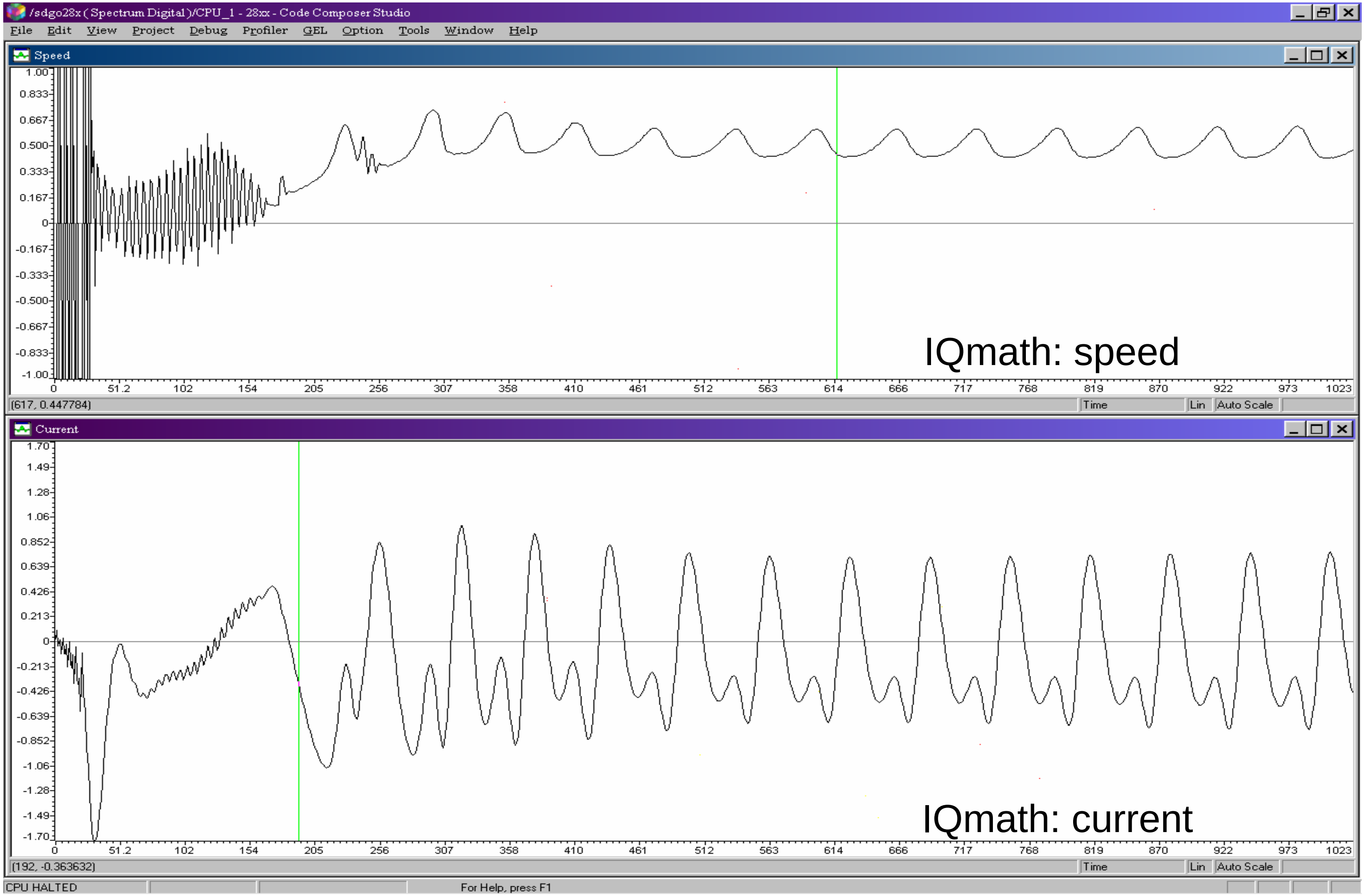
So, both produce similar results!

AC Induction Motor Example:
GLOBAL_Q = 27, system unstable



AC Induction Motor Example:

GLOBAL_Q = 16, system unstable



AC Induction Motor Example:
Q stability range

Q range	Stability Range
Q31 to Q27	**Unstable** (not enough dynamic range)
Q26 to Q19	**Stable**
Q18 to Q0	**Unstable** (not enough resolution, quantization problems)

The developer must pick the right GLOBAL_Q value.

TI has done the rest of the work for you!

AC Induction Motor Example:
Performance Comparisons

Benchmark	C28x C IQmath (-g) (150 MHz)	C3x C float (-g) (75 MHz)	C67x C float (-g) (167 MHz)	C67x C float (no –g) (167 MHz)
Feedforward control cycles	482	308	627	432
Feedback control cycles	1081	829	1295	973
Total Control Law cycles	1563	1137	1922	1405
Total Mips used (20 kHz loop)	31.3 Mips	22.7 Mips	38.4 Mips	28.1 Mips
% of available Mips used (20 kHz loop)	20.9%	30.3%	23.0%	16.8%

```
Notes: C28x Compiled On CGT V3.03, V1.4c IQmath Lib (debug enabled (-g), max opt).
       C3x  Compiled On CGT V5.12, RTS30R Lib (debug enabled (-g), max opt).
       C67x Compiled On CGT V4.20, RTSFast Lib (debug enabled (-g)/disabled, max opt).
       On C67x, when debug is enabled (-g), parallel execution of operations are limited.
       On C28x & C3x, turning off debug (no –g) has minimal impact on performance.
```

Operation	Feedforward Control	Feedback Control
Add/Sub	23	27
Multiply	18	46
Divide	0	1
Compare	2	5
Sine/Cosine	6	6
Atan	0	1

IQmath Agenda

- The Fixed-Point DSP Development Dilemma
- Floating-Point vs. IQmath Representation
- The IQmath Approach and How it Addresses the Problem
- AC Induction Motor Application Example
- Additional Tidbits and Summary

IQmath: The Analog to Digital Link

Putting ADC Results into IQ Format on the TMS320C28x DSP

x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	**RESULTx**

Do not sign extend

31   15   0

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	x	x	x	x	x	x	x	x	x	x	x	0	0	0	0	**32-bit long**

Notice that the 32-bit long is already in I16Q16 format

```
#define AdcFsVoltage   IQ(3.0)   // ADC full scale voltage
_iq Result, temp;                // ADC result
void main(void) {
// convert the unsigned 16-bit result to unsigned 32-bit
    // temp = AdcRegs.ADCRESULT0;
// convert resulting IQ16 to IQ format
    // temp = _IQ16toIQ(temp);
// scale by ADC full-scale range (optional)
    // Result = _IQmpy(AdcFsVoltage, temp);
    Result = _IQmpy(AdcFsVoltage, _IQ16toIQ( (_iq)AdcRegs.ADCRESULT0));
}
```

IQmath: What's in the works at TI?

Operation	Floating-Point Equivalent	"IQmath" in C	"IQmath" in C++
Math and Trig Functions	exp(A) ln(A) log10(A) pwr(A,B) acos(A) asin(A) tan(A)	_IQexp(A) _IQln(A) _IQlog10(A) _IQpwr(A,B) _IQacos(A) _IQasin(A) _IQtan(A)	IQexp(A) IQln(A) IQlog10(A) IQpwr(A,B) IQacos(A) IQasin(A) IQtan(A)

DSP Functions	IQ Real FFT IQ Complex FFT IQ IIR Filters IQ FIR Filters IQ LMS Filters

Application Libraries	Motor Control (e.g. BLDC, ACPM)

IQmath Summary

- **Applicable anywhere large dynamic range is not required**
  - Motor and Servo Control (PID, State Estimator, Kalman,...)
  - Modems
  - Audio (MP3, etc.)
  - Imaging (JPEG, etc.)
  - Any application using 16-bit or 32-bit fixed-point Q math
- **One source code set for simulation and target device**
  - Seamless portability of code between fixed and floating-point devices
  - User selects target math type in “IQmathLib.h” file
    - #define MATH_TYPE == IQ_MATH
    - #define MATH_TYPE == FLOAT_MATH
- **Numerical precision selectable based on application requirement**
  - Set in “IQmathLib.h” file: #define GLOBAL_Q 18
  - Explicitly specify Q value: _iq20 X, Y, Z;
- **Rapid conversion/porting and implementation of algorithms**

Overall, you may just think it’s floating point!

IQmath Summary

The C28x IQmath library, literature #SPRC087, is freeware
http://www.dspvillage.ti.com (follow C2000 DSP links)

Technical Support Available from the TI Product Information Center
phone: 972-644-5580 email: support@ti.com

Thank you for Watching!