Modul 4: Brojni sistemi

Digital Signal Controller
TMS320F2833x

Texas Instruments Incorporated

“Floating-Point”, “Integer”, “Fixed-Point”

- Dve osnovne kategorije procesora:
  - *Floating-Point*
  - *Integer/Fixed-Point*
- U čemu je razlika?
- Prednosti / nedostaci ?
- *Real-Time Control*:
  - Većina MC su tipa *Fixed-Point*!
  - F2833x podržava oba načina obrade!

Tipovi procesora

- *Floating-Point* procesori
  - Ugrađena hardverska jedinica za podršku *Floating - Point* operacija
  - Primeri: Intelova Pentium serija, Texas Instruments C6000 DSP
  - Veliki dinamički opseg za numerička izračunavanja
  - Viša cena
- *Integer / Fixed-Point* procesori
  - *Fixed-Point* aritmetička jedinica
  - Najveći broj MC-a su *fixed point* mašine
  - Primeri: Freescale S12X, Infineon C166, Texas Instruments MSP430, Atmel AVR
  - Niska cena po MIPS-u

Standard IEEE-754 "Single Precision Floating-Point"

Case 1: if e = 255 and f ≠ 0, then v = NaN

Case 2: if e = 255 and f = 0, then v = $[(-1)^s]$*infinity

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

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

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

Prednosti ⇒ Eksponent omogućava veliki dinamički opseg
Nedostaci ⇒ Preciznost broja zavisi od eksponenta

Floating-Point format

Bit Znaka (S):

- **Negativni: bit 31 = 1 / Positivni: Bit 31 = 0**

- **Mantisa (M):**
  - **Mantisa normalizovana na m0 = 1; m0 se ne smešta u memoriju!**

$$M = 1 + m_1 2^{-1} + m_2 2^{-2} + \cdots + m_{23} 2^{-23} = 1 + \sum_{i=1}^{23} m_i 2^{-i}$$

$$1 \le M < 2$$

- **Exponent (E):**
  - **8 Bitni označeni eksponent, sa pomerajem (OFFSET) = +127**

- **Konačno:**

$$Z = (-1)^S M 2^{E-OFFSET}$$

Floating-Point ne rešava sve!

Primer:

	x =	10.0	(0x41200000)
+	y =	0.000000240	(0x3480D959)
	z =	10.000000240	GREŠKA!

Nije moguće predstaviti 10.000000240 u pokretnom zarezu (jednostruka preciznost)

0x412000000	=	10.000000000
		10.000000240 ⇐ nije moguće!
0x412000001	=	10.000001000

z zaokruženo na 10.000000000

Osnove “Integer” brojnog sistema

- **Binarni brojevi**

$0110_2 = (0*8)+(1*4)+(1*2)+(0*1) = 6_{10}$

$11110_2 = (1*16)+(1*8)+(1*4)+(1*2)+(0*1) = 30_{10}$

- **Dvojični komplement**

$0110_2 = (0*-8)+(1*4)+(1*2)+(0*1) = 6_{10}$

$11110_2 = (1*-16)+(1*8)+(1*4)+(1*2)+(0*1) = -2_{10}$

Množenje 4-bitnih brojeva

```
          0100          4
        x 1101        x  -3
      ---------      ------
      00000100
      00000000
      000100
    + 1100
      ---------      ------
      11110100        -12
```

Akumulator	11110100
Memorija	?

Da li postoji drugi (bolji) brojni sistem?

Binarne frakcije

= -1 + 1/4 + 1/8 = -5/8

Frakcije imaju lepu osobinu da

frakcija x frakcija = frakcija

4-bitno IQ – množenje

0.100	1/2
x 1.101	x - 3/8
00000100	
0000000	
000100	
11100	
11110100	-3/16

Akumulator 11.110100

Memorija 1.110 -1/4

Frakcioni brojevi

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

"IQ" – Format

"I" ⇒ INTEGER – Frakcija
"Q" ⇒ QUOTIENT – Frakcija

Prednosti ⇒ Preciznost ista za sve brojeve u IQ formatu
Nedostaci ⇒ Ograničeni dinamički opseg u odnosu na FP

IQ - Primeri

I1Q3 – Format:

Najnegativniji decimalni broj: -1.0 = 1.000 B

Najpozitivniji decimalni broj : + 0.875 = 0.111 B

Najmanji negativni decimalni broj: $-1*2^{-3}$ (-0.125) = 1.111 B

Najmanji pozitivni decimalni broj: 2^{-3} (+0.125) = 0.001 B

Opseg: **-1.0 …. 0.875 (≈ + 1.0)**
Rezolucija: **2^{-3}**

IQ - Primeri

I3Q1 – Format:

Najnegativniji decimalni broj : -4.0 = 100.0 B

Najpozitivniji decimalni broj : + 3.5 = 011.1 B

Najmanji negativni decimalni broj : $-1 * 2^{-1}$ (- 0.5) = 111.1 B

Najmanji pozitivni decimalni broj : 2^{-1} (+0.5) = 000.1 B

Opseg: **-4.0 …. +3.5 (≈ + 4.0)**
Rezolucija: **2^{-1}**

IQ – Opseg I rezolucija

4-bitni broj:

Format	Most Negative	Most Positive	Resolution (step size)
I1Q3	1.000 -1	0.111 +0.875	0.001 0.125
I2Q2	10.00 -2	01.11 +1.75	00.01 0.25
I3Q1	100.0 -4	011.1 +3.5	000.1 0.5
I4Q0	1000 -8	0111 +7	0001 1

- "Trade – Off" izmedju opsega i rezolucije
- Napomena: Integer Format (I4Q0) je podskup IQ-Math

IQ - Primeri

I1Q31 – Format:

Najnegativniji decimalni broj : -1.0
1.000 0000 0000 0000 0000 0000 0000 0000 B

Najpozitivniji decimalni broj : ≈ + 1.0
0.111 1111 1111 1111 1111 1111 1111 1111 B

Najmanji negativni decimalni broj : $-1*2^{-31}$
1.111 1111 1111 1111 1111 1111 1111 1111 B

Najmanji pozitivni decimalni broj : 2^{-31}
0.000 0000 0000 0000 0000 0000 0000 0001 B

Opseg : -1.0 …. (+1.0)
Rezolucija : 2^{-31}

IQ - Primeri

I8Q24 – Format:

31 0

S	III IIII.	ffff ffff ffff ffff ffff

Najnegativniji decimalni broj : -128
1000 0000. 0000 0000 0000 0000 0000 0000 B

Najpozitivniji decimalni broj : ≈ + 128
0111 1111. 1111 1111 1111 1111 1111 1111 B

Najmanji negativni decimalni broj : $-1*2^{-24}$
1111 1111. 1111 1111 1111 1111 1111 1111 B

Najmanji pozitivni decimalni broj : 2^{-24}
0000 0000. 0000 0000 0000 0000 0000 0001 B

Opseg :	**-128 …. (+128)**
Rezolucija :	**2^{-24}**

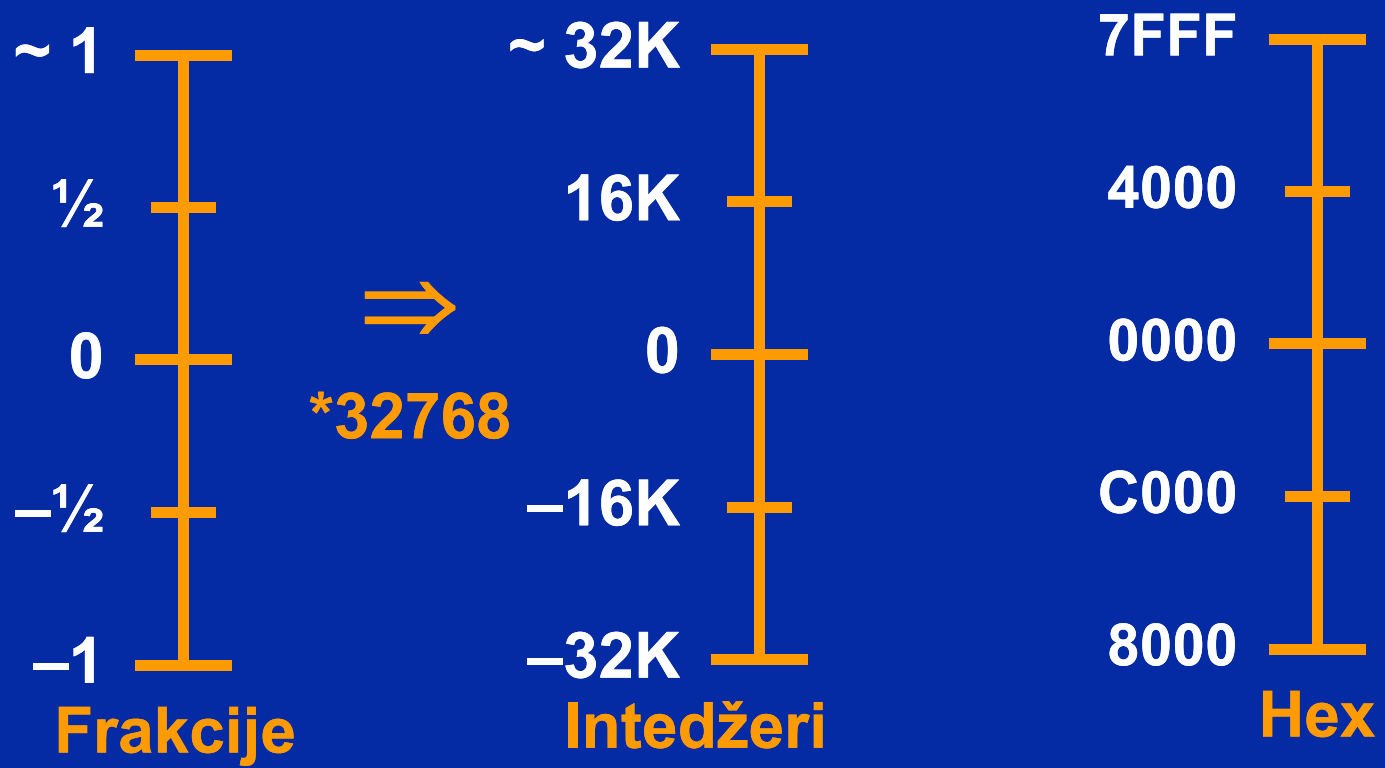
IQ-računica je nekad bolja!

I8Q24 Primer:

	x = 10.0	(0x0A000000)
+	y = 0.000000240	(0x00000004)
	z = 10.000000240	(0x0A000004)

Tačan rezulat

Kako se frakcioni broj kodira u C-u?



- Primer: predstavljanje frakcionog broja 0.707

```
void main(void) {
    int coef = 32768 * 707 / 1000;
}
```

Frakcioni ili Intedžeri

- Opseg
  - Intedžeri imaju maksimalni opseg koji je određen brojem bitova
  - Frakcioni imaju opseg ±1
- Preciznost
  - Intedžeri imaju maksimalnu preciznost 1
  - Preciznost frakcionih određena brojem bitova

Lab4: „Fixed-point" i „Floating-point"

- *Benchmark* operacije množenja
- k = i * i
- Testne postavke:

1. Množenje intedžera
2. Floating-Point primenom FP biblioteke
3. Floating-Point primenom FP hardverske jedinice

Benchmark result:

	Fixed-point	Floating-Point-Library	Floating-Point-Hardware
code size (words)	3	89	9
clock cycles (6.67 ns)	3	112	5

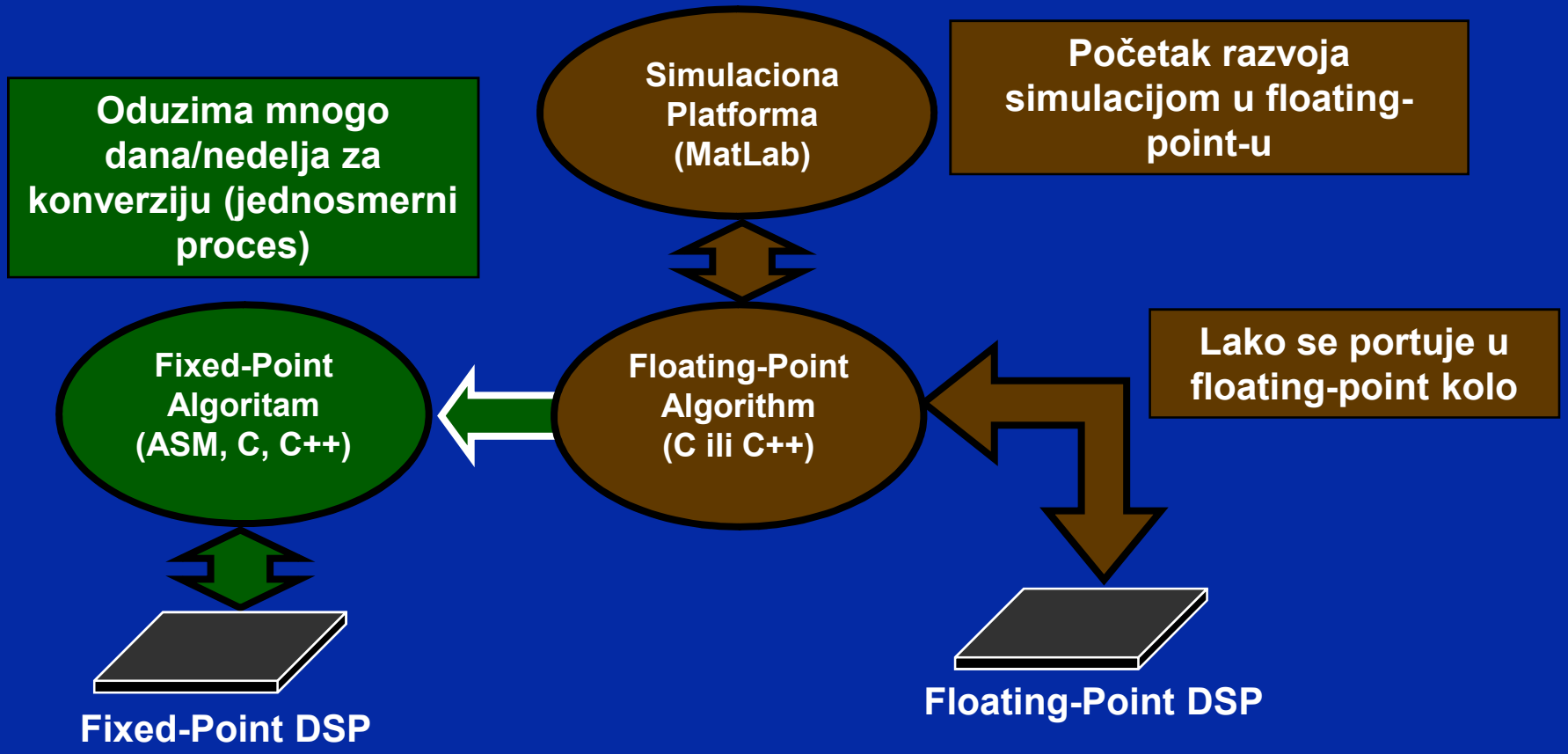
F28x DSP Biblioteka

Signal Processing Libraries & Applications Software	Literature #
ACI3-1: Control with Constant V/Hz	SPRC194
ACI3-3: Sensored Indirect Flux Vector Control	SPRC207
ACI3-3: Sensored Indirect Flux Vector Control (simulation)	SPRC208
ACI3-4: Sensorless Direct Flux Vector Control	SPRC195
ACI3-4: Sensorless Direct Flux Vector Control (simulation)	SPRC209
PMSM3-1: Sensored Field Oriented Control using QEP	SPRC210
PMSM3-2: Sensorless Field Oriented Control	SPRC197
PMSM3-3: Sensored Field Oriented Control using Resolver	SPRC211
PMSM3-4: Sensored Position Control using QEP	SPRC212
BLDC3-1: Sensored Trapezoidal Control using Hall Sensors	SPRC213
BLDC3-2: Sensorless Trapezoidal Drive	SPRC196
DCMOTOR: Speed & Position Control using QEP without Index	SPRC214
Digital Motor Control Library (F/C280x)	SPRC215
Communications Driver Library	SPRC183
DSP Fast Fourier Transform (FFT) Library	SPRC081
DSP Filter Library	SPRC082
DSP Fixed-Point Math Library	SPRC085
DSP IQ Math Library	SPRC087
DSP Signal Generator Library	SPRC083
DSP Software Test Bench (STB) Library	SPRC084
C28x FPU Fast RTS Library	SPRC664
C2833x C/C++ Header Files and Peripheral Examples	SPRC530

Available from TI Website ⇒ http://www.ti.com/c2000

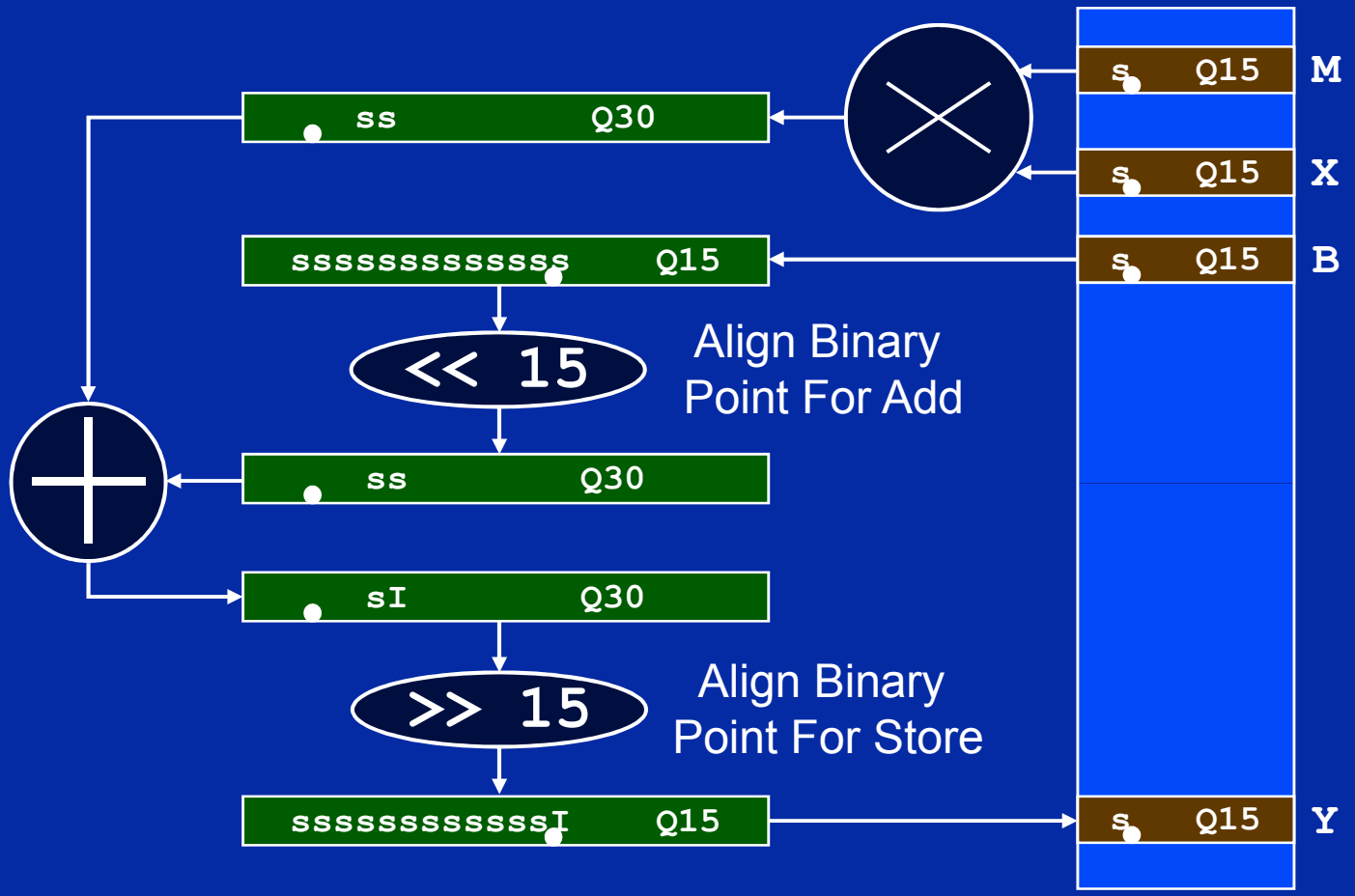
Kako se koristi frakciona računica?

Dilema Fixed-Point razvoja



Klasično 16-bitno “Q” izračunavanje

y = mx + b

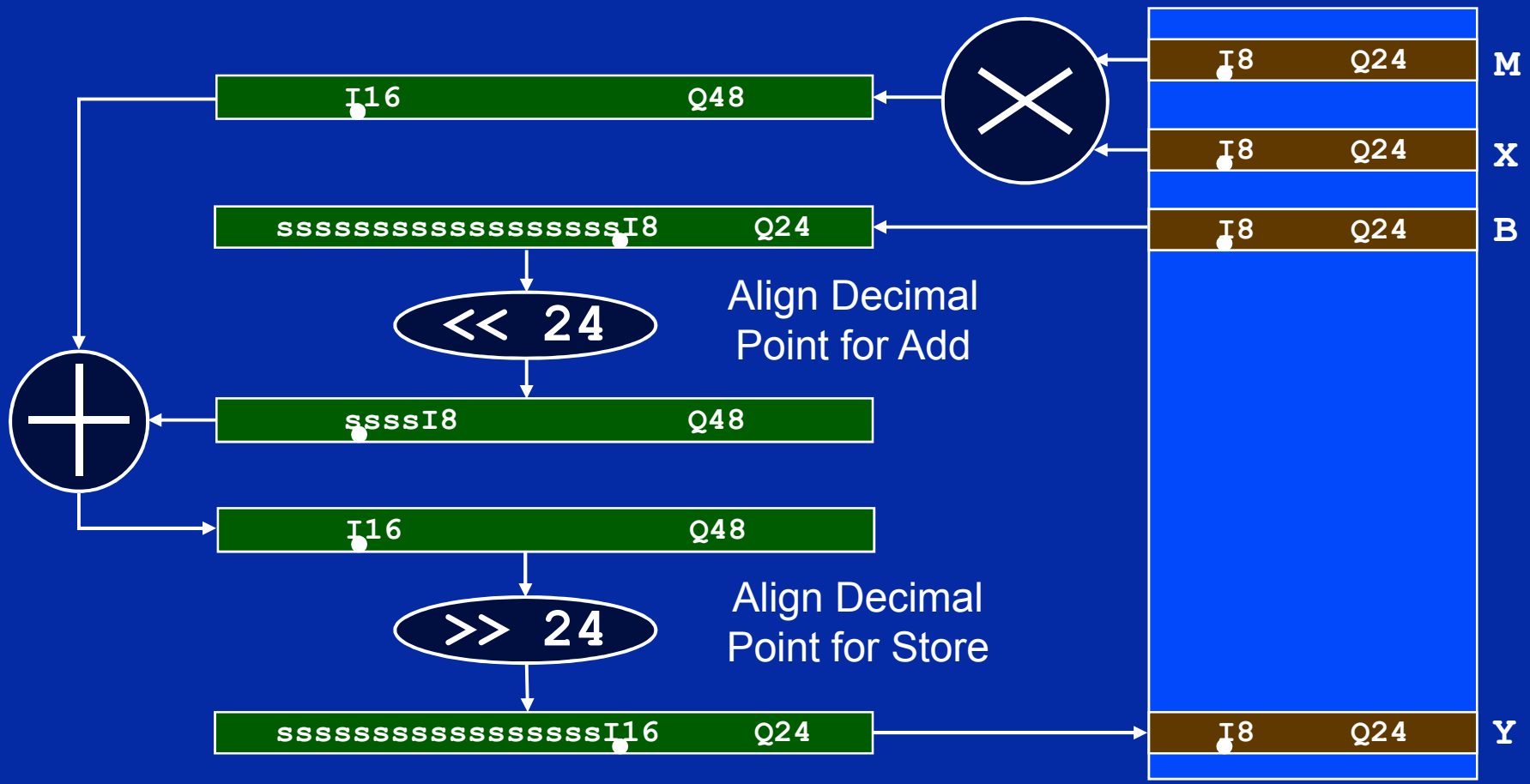


C:

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

Klasično 32-bitno “Q” izračunavanje

y = mx + b



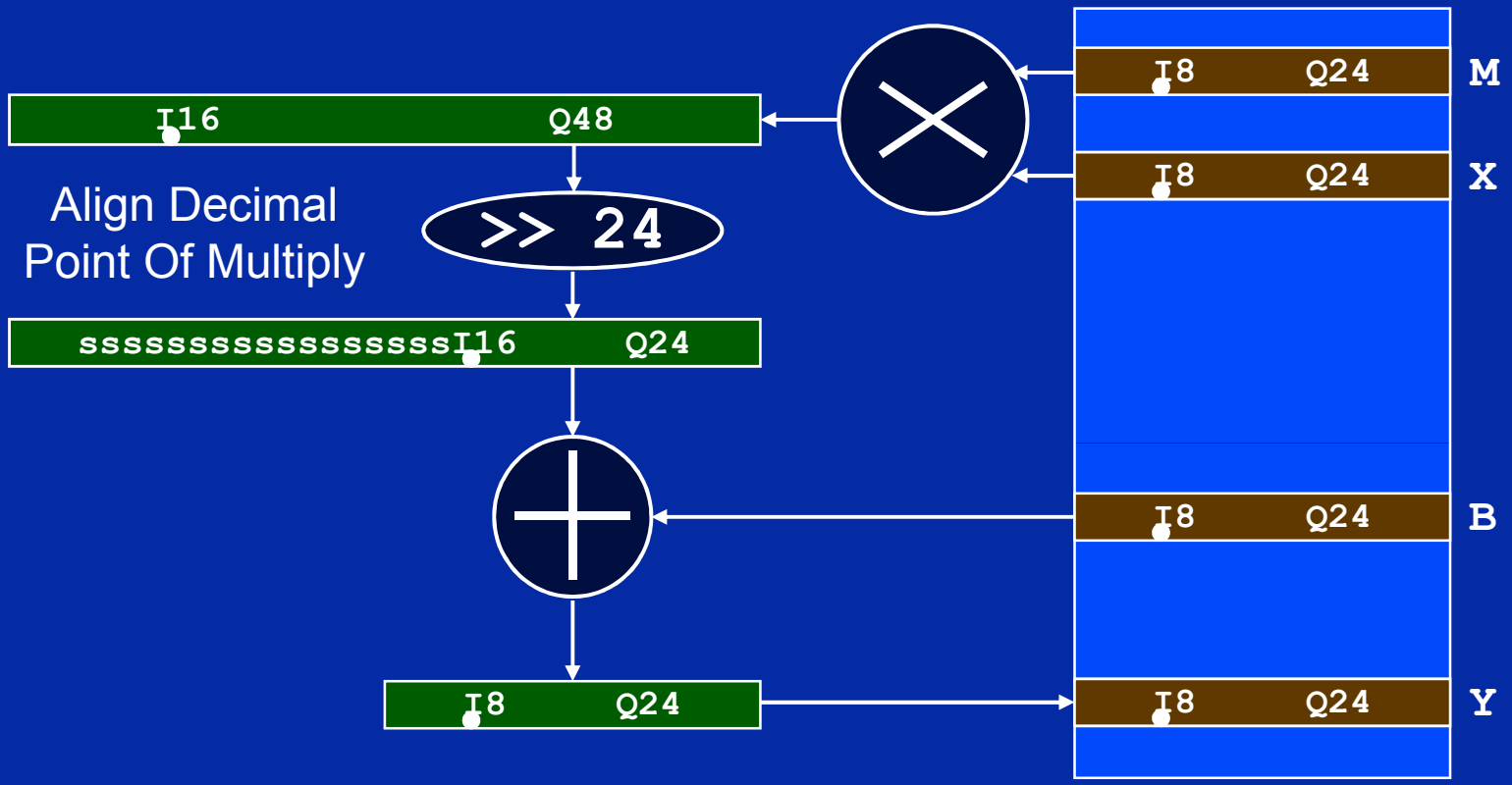
C:

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

Napomena: Zahteva se podrška 64-bitnih intedžera u kompajleru (klasičan C ovo nema)

32-bitno IQ izračunavanje

y = mx + b



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

IQ izračunavanje
Operacija množenja

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

Redefinisanje množenja:

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

Tako da se prethodna relacija svodi na:

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

C28x compajler podržava "_IQmpy" kao "intrinsic"; asemblerski kod:

```
MOVL    XT,@M
IMPYL   P,XT,@X         ; P   = low  32-bits of M*X
QMPYL   ACC,XT,@X       ; ACC = high 32-bits of M*X
LSL64   ACC:P,#(32-Q)   ; ACC = ACC:P << 32-Q
                        ; (same as P = ACC:P >> Q)
ADDL    ACC,@B          ; Add B
MOVL    @Y,ACC          ; Result = Y = _IQmpy(M*X) + B
; 7 Cycles
```

IQ izračunavanje - IQmath

liči na floating-point!

Floating-Point

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

Y = M * X + B;
```

Klasična Fix-Point Q

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

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

"IQmath" u C

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

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

"IQmath" u C++

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

Y = M * X + B;
```

"IQmath" kod se lako čita!

Primena IQmath
GLOBAL_Q

Korisnik bira "Global Q" vrednost za celu aplikaciju

- **GLOBAL_Q**

na osnovu potrebnog dinamičkog opsega i rezolucije, na primer:

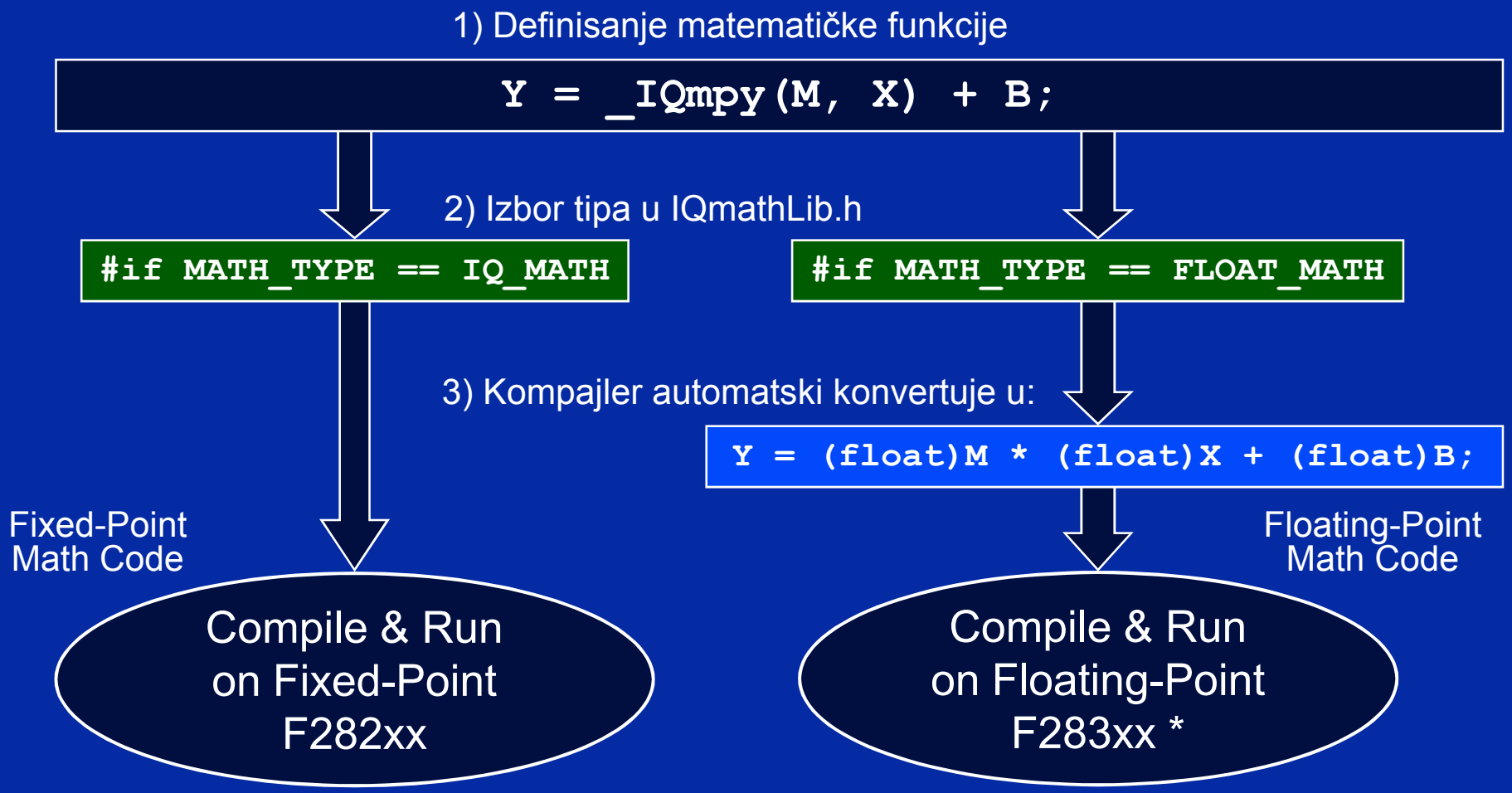
GLOBAL_Q	Max Val	Min Val	Resolution
28	7.999 999 996	-8.000 000 000	0.000 000 004
24	127.999 999 94	-128.000 000 00	0.000 000 06
20	2047.999 999	-2048.000 000	0.000 001

```
#define  GLOBAL_Q  18     // set in "IQmathLib.h" file
_iq  Y, M, X, B;
Y = _IQmpy(M,X) + B;      // all values are in Q = 18
```

Korisnik može eksplicitno da specificira vrednost Q:

```
_iq20  Y, M, X, B;
Y = _IQ20mpy(M,X) + B;    // all values are in Q = 20
```

IQmath obezbeđuje kompatibilnost između Floating-Point i Fixed-Point



Sve "IQmath" operacije imaju ekvivalet u floating-point operaciji

IQmath Library: Math & Trig Funkcije

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
substract	A - B	A - B	A – B
boolean	>, >=, <, <=, ==, |=, &&, ||	>, >=, <, <=, ==, |=, &&, ||	>, >=, <, <=, ==, |=, &&, ||
trig and power functions	sin(A),cos(A) sin(A*2pi),cos(A*2pi) asin(A),acos(A) atan(A),atan2(A,B) atan2(A,B)/2pi sqrt(A),1/sqrt(A) sqrt(A*A + B*B) exp(A)	_IQsin(A), _IQcos(A) _IQsinPU(A), _IQcosPU(A) _IQasin(A),_IQacos(A) _IQatan(A), _IQatan2(A,B) _IQatan2PU(A,B) _IQsqrt(A), _IQisqrt(A) _IQmag(A,B) _IQexp(A)	IQsin(A),IQcos(A) IQsinPU(A),IQcosPU(A) IQasin(A),IQacos(A) IQatan(A),IQatan2(A,B) IQatan2PU(A,B) IQsqrt(A),IQisqrt(A) IQmag(A,B) IQexp(A)
saturation	if(A > Pos) A = Pos if(A < Neg) A = Neg	_IQsat(A,Pos,Neg)	IQsat(A,Pos,Neg)

Accuracy of functions/operations approx ~28 to ~31 bits

IQmath Library: Funkcije konverzije

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)
IQ to ASCII	sprintf(A,B,C)	_IQtoA(A,B,C)	IQtoA(A,B,C)

IQmath.lib > contains library of math functions
IQmathLib.h > C header file
IQmathCPP.h > C++ header file

Primer AC Indukcionog motora

One of the more complex motor control algorithms

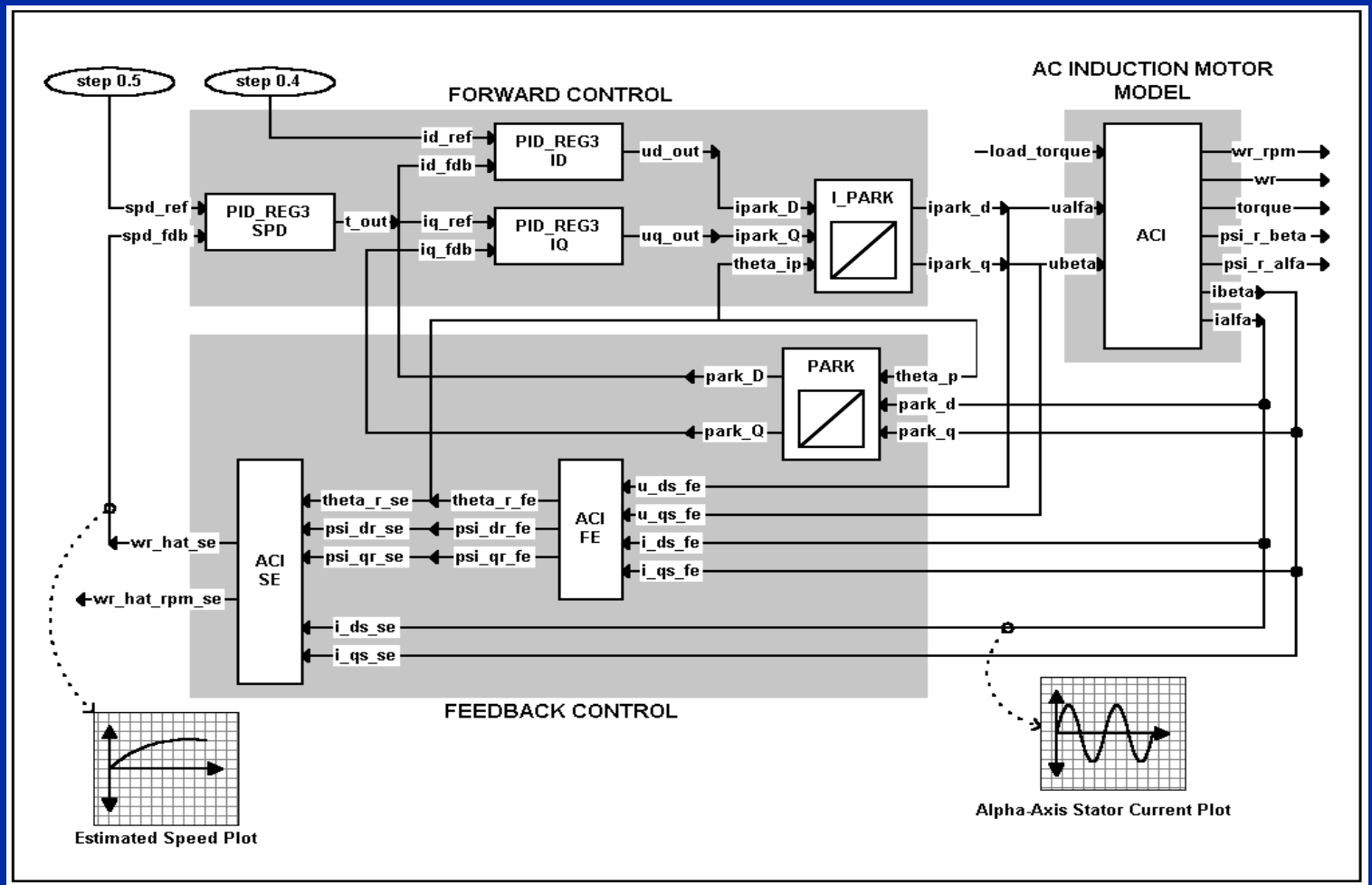


Figure 5

- Sensorless, ACI induction machine direct rotor flux control
- Goal: motor speed estimation & alpha-axis stator current estimation

Primer AC Indukcionog motora

Park Transform – floating-point C code

```
#include “math.h”

#define  TWO_PI   6.28318530717959
void park_calc(PARK *v)
{
     float cos_ang , sin_ang;
     sin_ang = sin(TWO_PI * v->ang);
     cos_ang = cos(TWO_PI * v->ang);

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

Primer AC Indukcionog motora

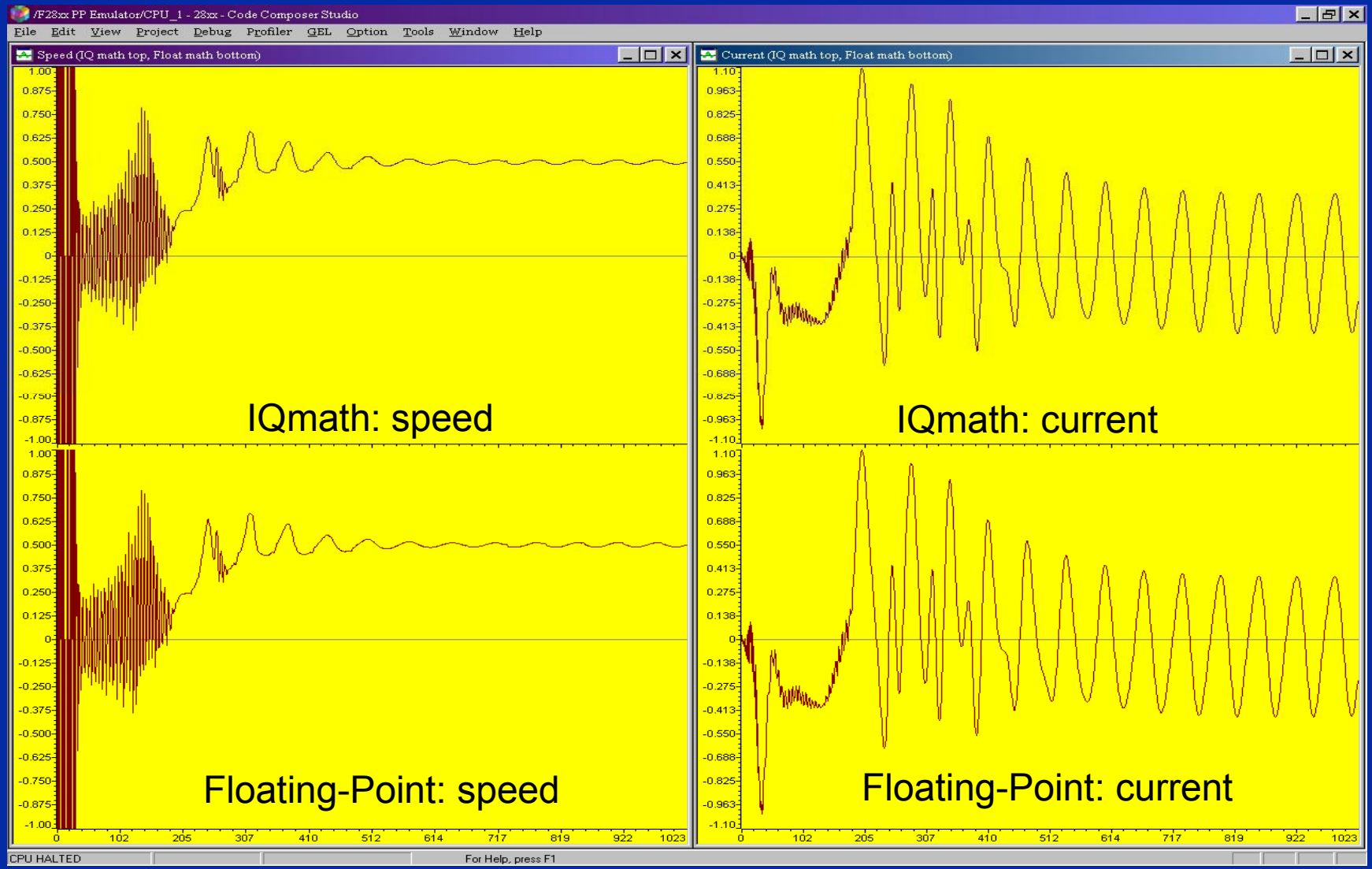
Park Transform - converting to “IQmath” C code

```
#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);
                                  }
```

Primer AC Indukcionog motora

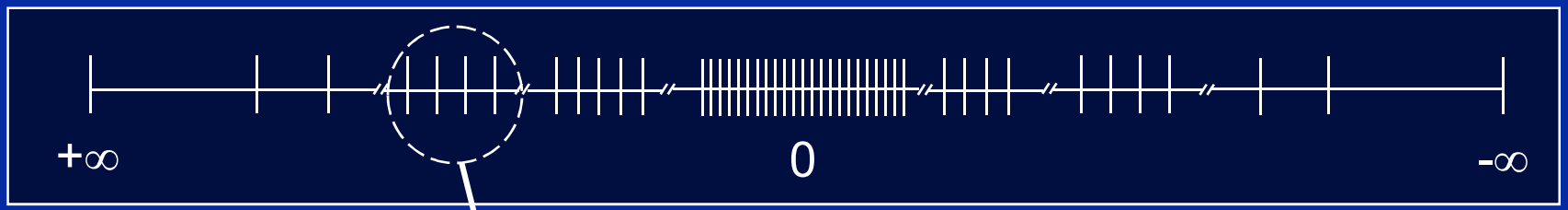
GLOBAL_Q = 24, sistem stabilan



Zašto je ovo stabilno?

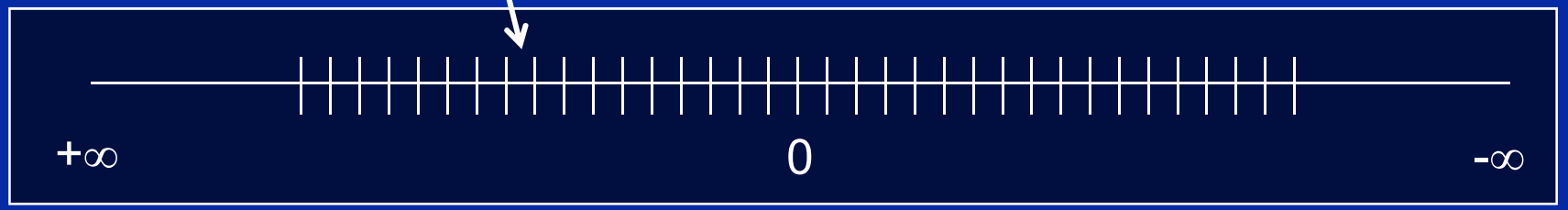
Jednaka preciznost u oblasti izračunavanja

Floating-Point:



Ista preciznost kao I8Q24

I8Q24 Fractions:

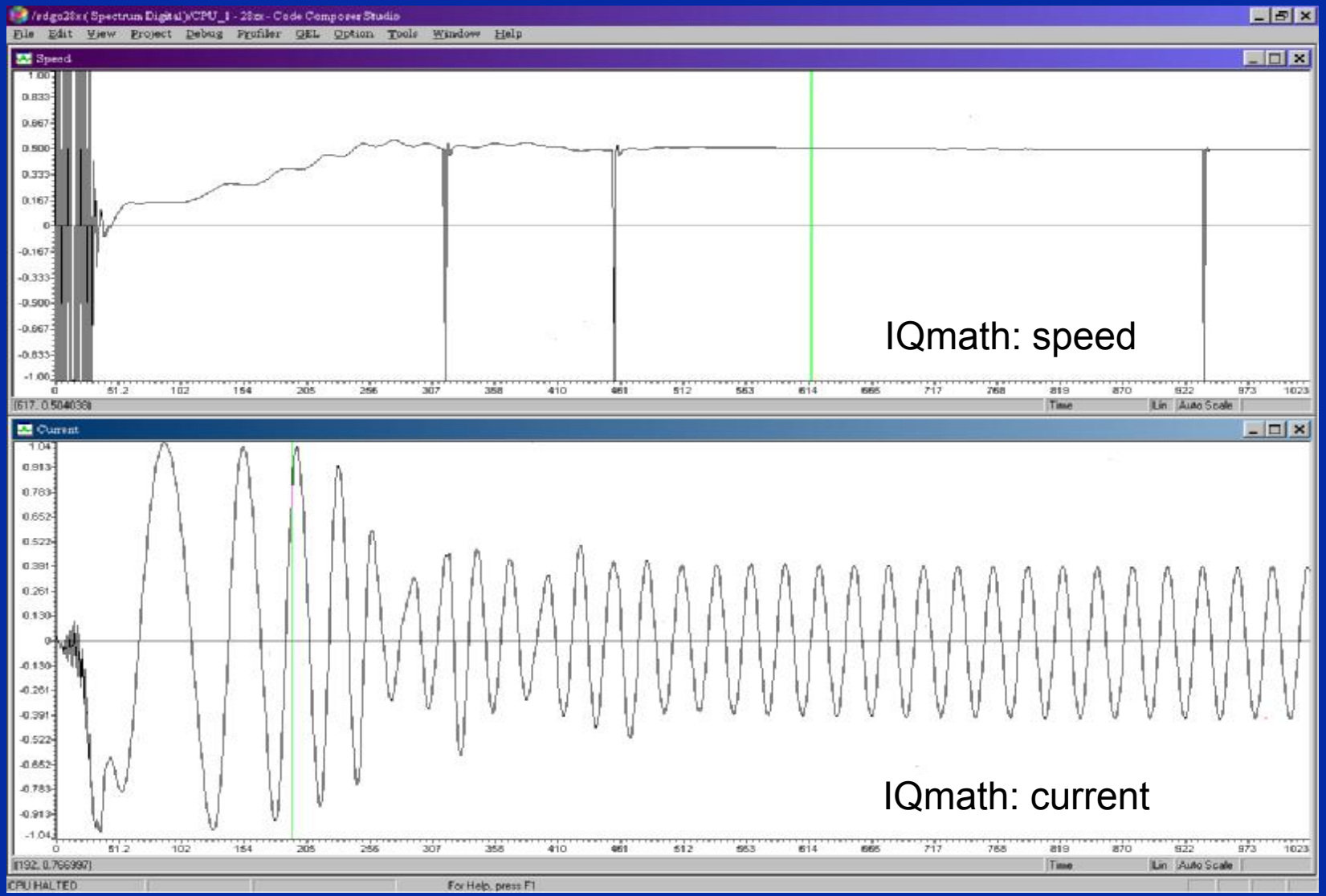


U oblasti gde se izračunavanja izvršavaju, preciznost *single-precision floating-point* je jednaka preciznosti I8Q24 formata.

Sledi da su rezultati slični!

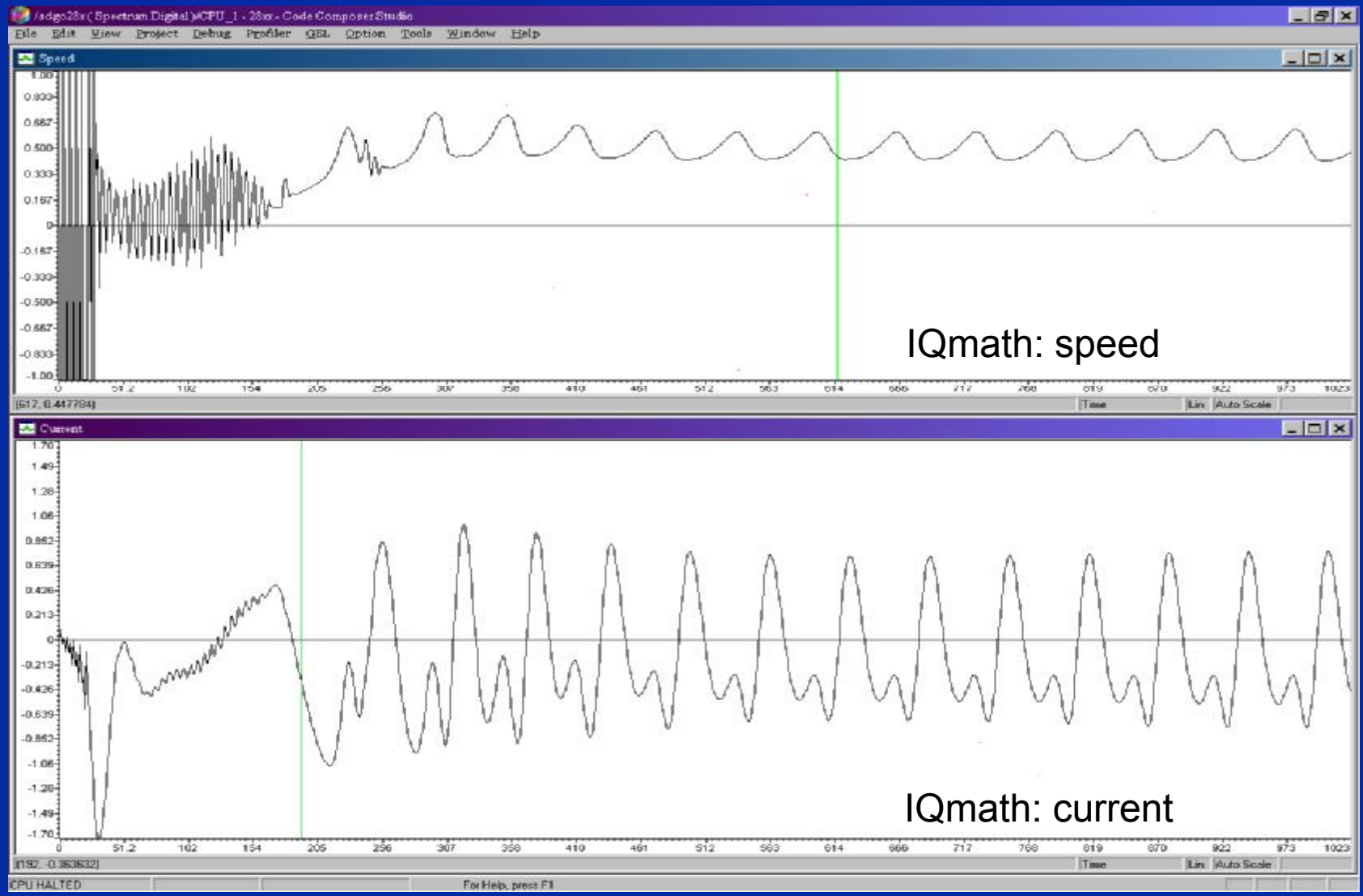
Primer AC Indukcionog motora

GLOBAL_Q = 27, sistem nestabilan



Primer AC Indukcionog motora

GLOBAL_Q = 16, sistem nestabilan



Primer AC Indukcionog motora
Q oblast stabilnosti

Oblast Q	Opseg stabilnosti
Q31 to Q27	Nestabilan (mali dinamički opseg)
Q26 to Q19	Stabilan
Q18 to Q0	Nestabilan (mala rezolucija, problemi kvantizacije)

Projektant mora odabrati pravu vrednost za GLOBAL_Q!

Primer AC Indukcionog motora
Upoređenje performansi

Benchmark	C28x C floating-point std. RTS lib (150 MHz)	C28x C floating-point fast RTS lib (150 MHz)	C28x C IQmath v1.4d (150 MHz)
B1: ACI module cycles	401	401	625
B2: Feedforward control cycles	421	371	403
B3: Feedback control cycles	2336	792	1011
Total control cycles (B2+B3)	2757	1163	1414
% of available MHz used (20 kHz control loop)	36.8%	15.5%	18.9%

Notes: C28x compiled on codegen tools v5.0.0, -g (debug enabled), -o3 (max. optimization)
fast RTS lib v1.0beta1
IQmath lib v1.4d

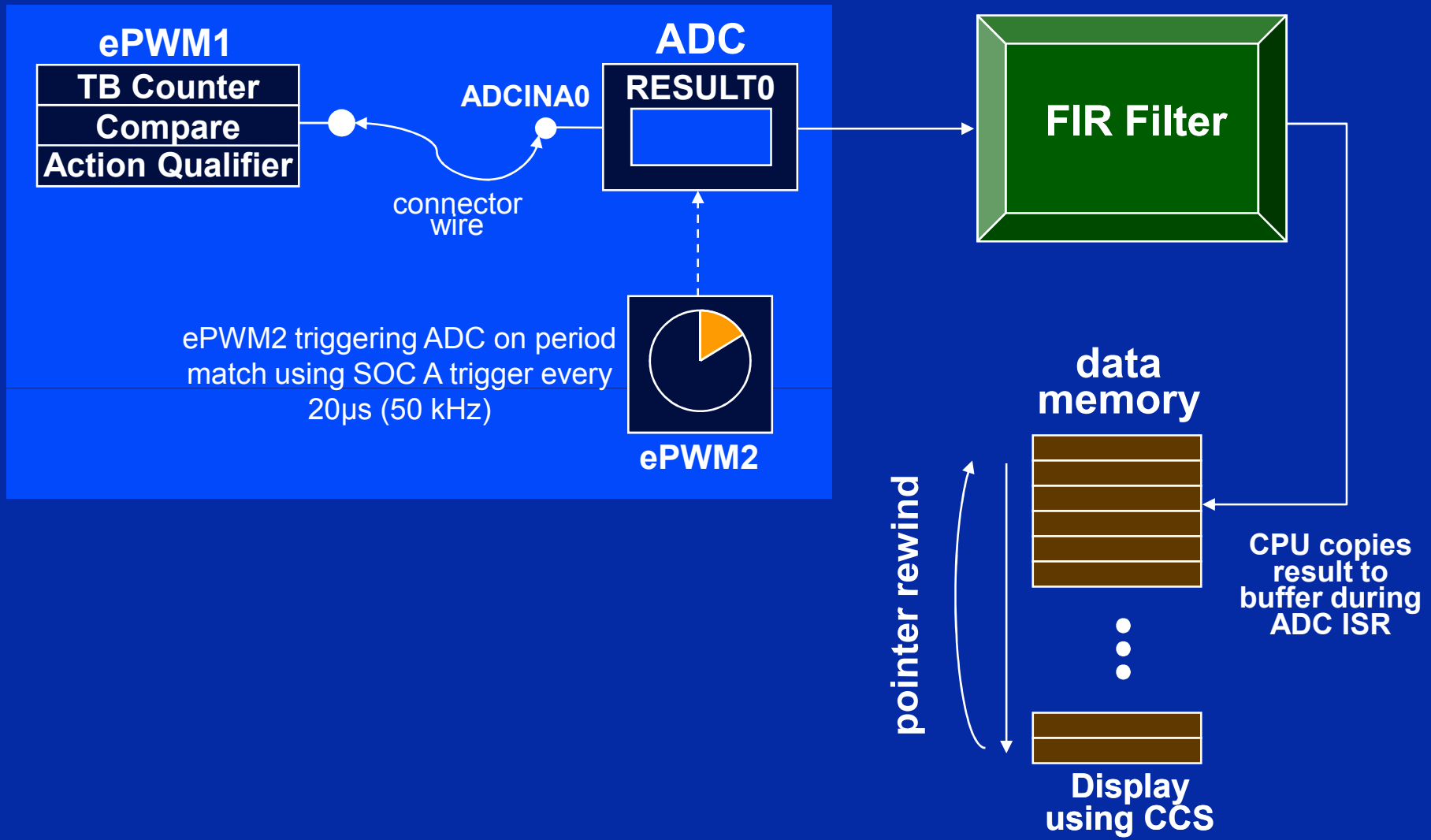
IQmath Approach Summary

"IQmath" + fixed-point processor with 32-bit capabilities =

- Seamless portability of code between fixed and floating-point devices
  - User selects target math type in "IQmathLib.h" file
    - #if MATH_TYPE == IQ_MATH
    - #if MATH_TYPE == FLOAT_MATH
- One source code set for simulation vs. target device
- Numerical resolution adjustability based on application requirement
  - Set in "IQmathLib.h" file
    - #define GLOBAL_Q 18
  - Explicitly specify Q value
    - _iq20 X, Y, Z;
- Numerical accuracy without sacrificing time and cycles
- Rapid conversion/porting and implementation of algorithms

IQmath library is freeware - available from TI DSP website
http://www.ti.com/c2000

Vežba: IQ – Math FIR - Filter

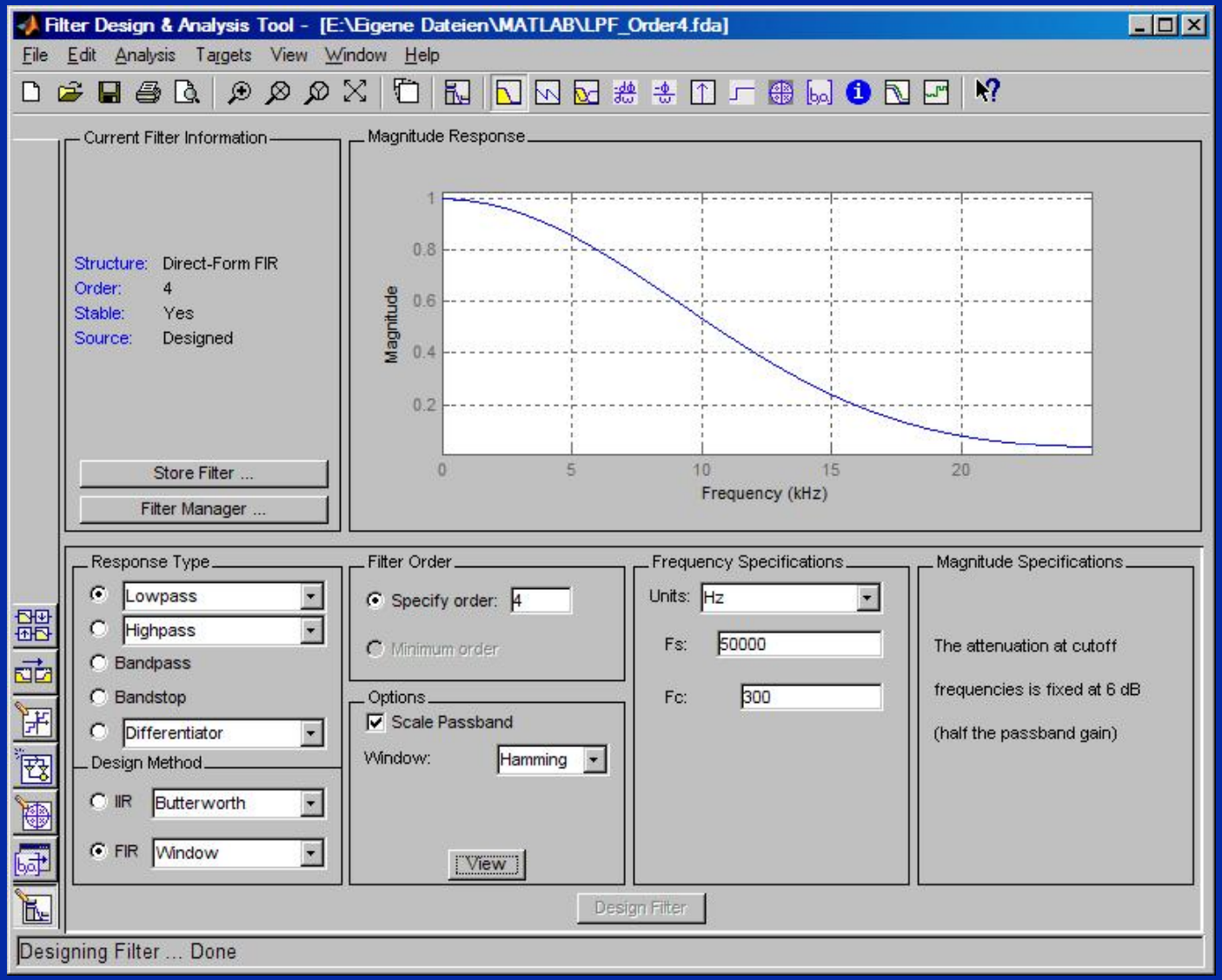


Vežba: IQ – Math FIR - Filter

Cilj:

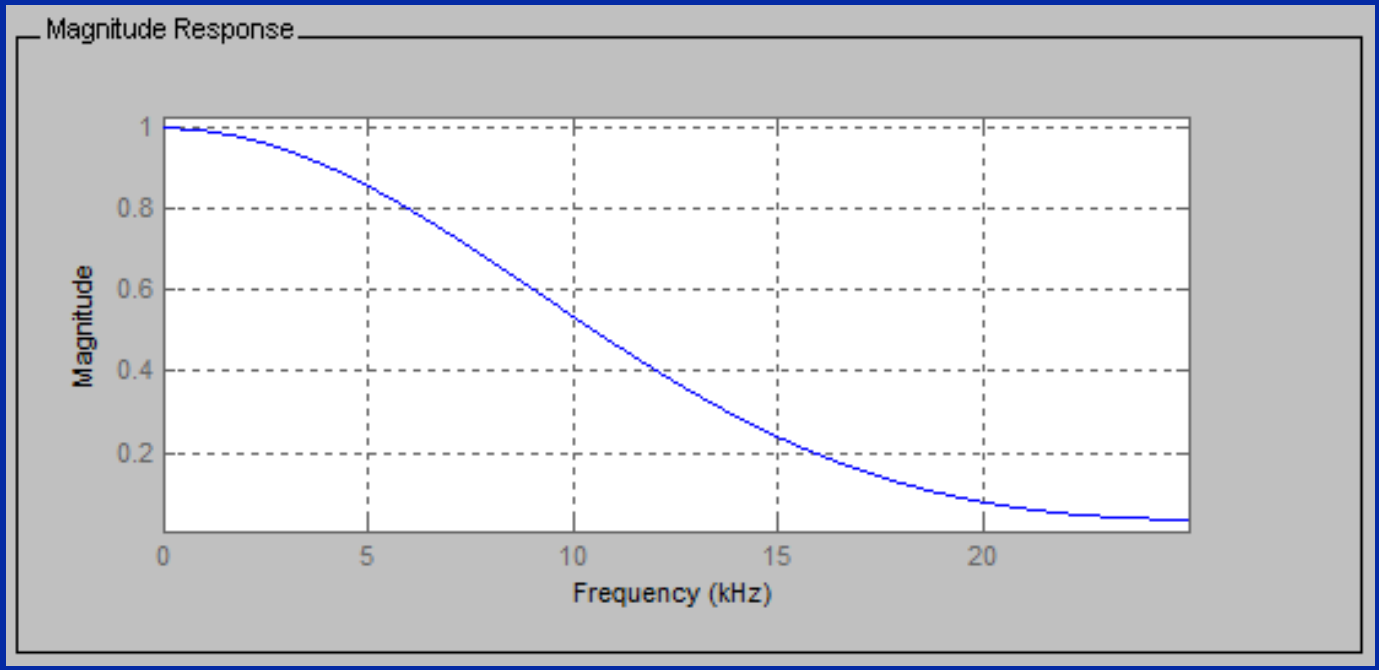
1. Generisati asimetričan PWM signal frekvencije 2 kHz sa faktorom ispune 25% (duty cycle)
2. Odmeravati signal pomoću AD konvertora sa brzinom odmeravanja od 50 kHz
3. Smestiti rezultat u cirkularni bafer
4. Rezultat AD konverzije propustiti kroz NF digitalni filter (FIR-finite impulse response). Funkciju filtriranja izračunavti pomoću IQ-Math funkcija
5. Nacrtati grafik filtriranog i nefiltriranog signala u grafičkom alatu Code Composer Studija
6. Uporediti rezultate

MATLAB FDATOOL

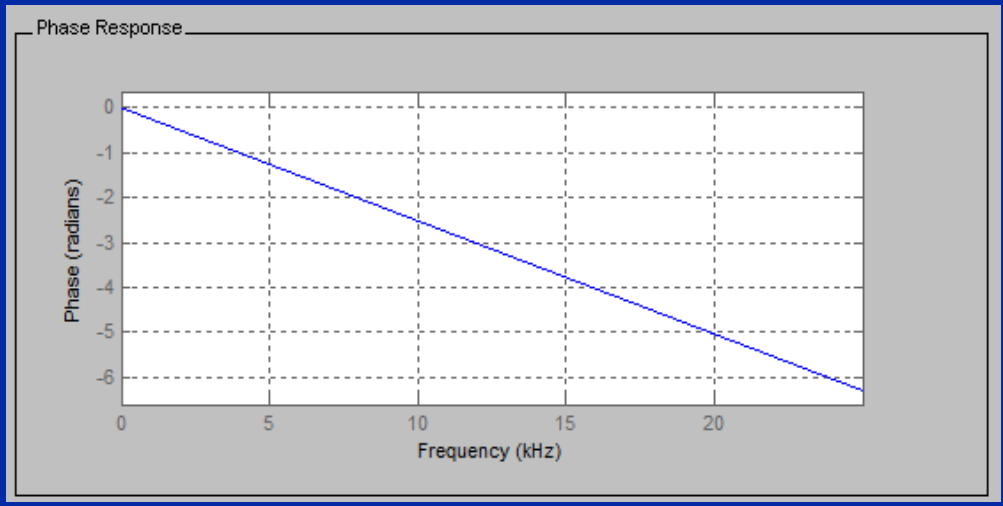


MATLAB FDATOOL

- FIR – Lowpass; 4th –order
- Corner Frequency: 300 Hz
- Sampling Frequency: 50 kHz
- Hamming - Window



MATLAB FDATOOL



Filter Coefficients

Numerator:
0.035686946550290158
0.24105816763777463
0.44650977162387046
0.24105816763777463
0.035686946550290158