MATH

MATH Co-Processor

XMC™ microcontrollers
September 2016

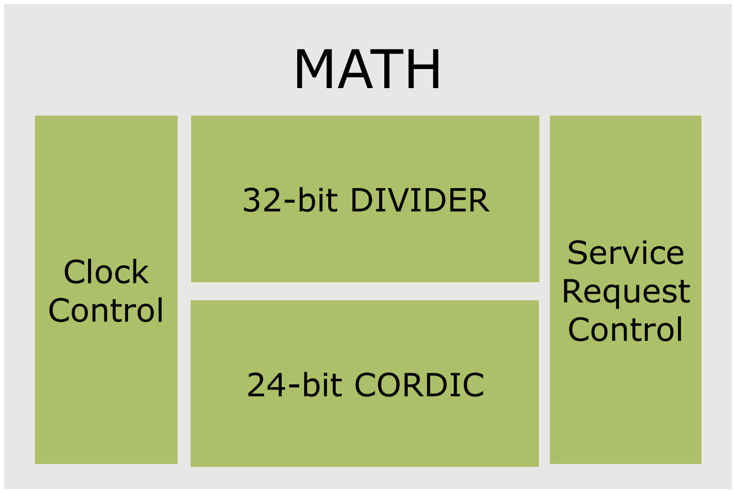
Agenda

1. Overview
2. Key feature: 32-bit divide
3. Key feature: Trigonometric functions
4. Key feature: Vector rotation (Park transform)
5. System integration
6. Result chaining between Divider & CORDIC
7. Benchmarking results

Agenda

1. Overview
2. Key feature: 32-bit divide
3. Key feature: Trigonometric functions
4. Key feature: Vector rotation (Park transform)
5. System integration
6. Result chaining between Divider & CORDIC
7. Benchmarking results

MATH
MATH Co-Processor



Highlights

The MATH Co-Processor provides a 32-bit signed or unsigned divider as well as a 24-bit CORDIC for trigonometric calculations. Both DIVIDER and CORDIC can operate in parallel next to the CORTEX®-M0 CPU core.

Key features

› 32-bit hardware divide for signed and unsigned long integer numbers
› Trigonometric functions executed in parallel to CPU operation
› Vector rotation (PARK transform) executed in 24-bit resolution

Customer benefits

› The calculation time of a divide operation is reduced to 50%
› Increase of computational power for real time critical tasks
› Field oriented motor control algorithms are implemented with high resolution

Agenda

MATH
32-bit divide

- Signed/unsigned 32-bit division in 35 kernel clock cycles
- Operands pre-processing with configurable number of:
  - Left shifts for dividend
  - Right shifts for divisor
- Result post-processing with configurable number of shifts and shift direction

Agenda

MATH
Trigonometric functions (1/2)

Function	Rotation Mode	Vectoring Mode
	d_i = sign (z_i), $z_i \rightarrow 0$	d_i = -sign (y_i), $y_i \rightarrow 0$
Circular $m = 1$ e_i = atan(2^{-i})	X_{final} = K[X cos(Z) - Y sin(Z)] / MPS Y_{final} = K[Y cos(Z) + X sin(Z)] / MPS Z_{final} = 0 where K ≈ 1.646760258121	X_{final} = K sqrt(X^2+Y^2) / MPS Y_{final} = 0 Z_{final} = Z + atan(Y / X) where K ≈ 1.646760258121
Linear $m = 0$ $e_i = 2^{-i}$	X_{final} = X / MPS Y_{final} = [Y + X Z] / MPS Z_{final} = 0	X_{final} = X / MPS Y_{final} = 0 Z_{final} = Z + Y / X
Hyperbolic $m = -1$ e_i = atanh(2^{-i})	X_{final} = k[X cosh(Z) + Y sinh(Z)] / MPS Y_{final} = k[Y cosh(Z) + X sinh(Z)] / MPS Z_{final} = 0 where k ≈ 0.828159360960	X_{final} = k sqrt(X^2-Y^2) / MPS Y_{final} = 0 Z_{final} = Z + atanh(Y / X) where k ≈ 0.828159360960

To calculate sin(angle) and cos(angle)

- Setup function to “Circular”, “Rotation Mode”
- X = 1/K, Y = 0, Z = “angle”

- Result_X = cos(angle)
- Result_Y = sin(angle)

MATH
Trigonometric functions (2/2)

Function	Rotation Mode	Vectoring Mode
	d_i = sign (z_i), $z_i \rightarrow 0$	d_i = -sign (y_i), $y_i \rightarrow 0$
Circular $m = 1$ e_i = atan(2^{-i})	X_{final} = K[X cos(Z) - Y sin(Z)] / MPS Y_{final} = K[Y cos(Z) + X sin(Z)] / MPS Z_{final} = 0 where K ≈ 1.646760258121	X_{final} = K sqrt(X^2+Y^2) / MPS Y_{final} = 0 Z_{final} = Z + atan(Y / X) where K ≈ 1.646760258121
Linear $m = 0$ $e_i = 2^{-i}$	X_{final} = X / MPS Y_{final} = [Y + X Z] / MPS Z_{final} = 0	X_{final} = X / MPS Y_{final} = 0 Z_{final} = Z + Y / X
Hyperbolic $m = -1$ e_i = atanh(2^{-i})	X_{final} = k[X cosh(Z) + Y sinh(Z)] / MPS Y_{final} = k[Y cosh(Z) + X sinh(Z)] / MPS Z_{final} = 0 where k ≈ 0.828159360960	X_{final} = k sqrt(X^2-Y^2) / MPS Y_{final} = 0 Z_{final} = Z + atanh(Y / X) where k ≈ 0.828159360960

To calculate arctan(Y/X)

- Setup function to “Circular”, “Vectoring Mode”
- Z = 0

- Result_X = K sqrt(X^2 + Y^2)
- Result_Z = arctan(Y/X)

Agenda

1. Overview
2. Key feature: 32-bit divide
3. Key feature: Trigonometric functions
4. Key feature: Vector rotation (Park transform)
5. System integration
6. Result chaining between Divider & CORDIC
7. Benchmarking results

MATH
Vector rotation (Park transform)

Function	Rotation Mode	Vectoring Mode
	d_i = sign (z_i), $z_i \to 0$	d_i = -sign (y_i), $y_i \to 0$
Circular $m = 1$ $e_i = \text{atan}(2^{-i})$	X_{final} = K[X cos(Z) - Y sin(Z)] / MPS Y_{final} = K[Y cos(Z) + X sin(Z)] / MPS Z_{final} = 0 where K ≈ 1.646760258121	X_{final} = K sqrt(X^2+Y^2) / MPS Y_{final} = 0 Z_{final} = Z + atan(Y / X) where K ≈ 1.646760258121
Linear $m = 0$ $e_i = 2^{-i}$	X_{final} = X / MPS Y_{final} = [Y + X Z] / MPS Z_{final} = 0	X_{final} = X / MPS Y_{final} = 0 Z_{final} = Z + Y / X
Hyperbolic $m = -1$ $e_i = \text{atanh}(2^{-i})$	X_{final} = k[X cosh(Z) + Y sinh(Z)] / MPS Y_{final} = k[Y cosh(Z) + X sinh(Z)] / MPS Z_{final} = 0 where k ≈ 0.828159360960	X_{final} = k sqrt(X^2-Y^2) / MPS Y_{final} = 0 Z_{final} = Z + atanh(Y / X) where k ≈ 0.828159360960

Park transform

$$i_d = i_\alpha \cos\varphi + i_\beta \sin\varphi$$

$$i_q = -i_\alpha \sin\varphi + i_\beta \cos\varphi$$

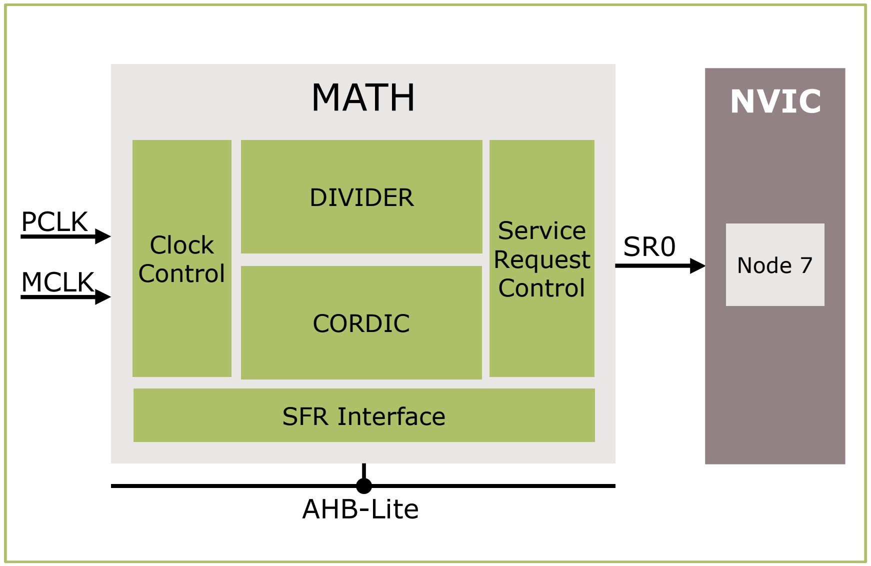
To calculate Park transform

- Setup function to "Circular", "Rotation Mode"
- X = i_β, Y = i_α, Z = φ

- Result_X = iq
- Result_Y = id

Agenda

MATH
System integration



- Target applications
  - Motor control
  - Intelligent lighting
  - Power conversion

XMC1100	XMC1200	XMC1300
		●

The math co-processor can be clocked with a frequency of up to 64 MHz and is accessible via the SFR interface. The sub-blocks, a 32-bit divider and a 24-bit CORDIC can be used next to the CPU independently. The execution of the math unit can be configured to be twice the MCU clock. Hence a divide is executed in 18 CPU clocks and a CORDIC function takes up to 31 CPU clocks.

In some use cases, the result of one sub-block is needed as data input for the other sub-block. A hardware mechanism is provided for autonomous execution of both calculation with result chaining.

The result that is read from the SFR-interface is always provided as the latest result after processing the math unit’s command. In case the read instruction is executed while the math is still busy, the bus-interface will add wait states until the latest result is available.

Agenda

MATH
Result chaining between Divider & CORDIC (1/6)

CORDIC’s result to DIV’s input

- Result of the CORDIC operation can be “forward” directly to the Divider operand register, DVD and DVS

DIV’s result to CORDIC’s input

- QUOT and RMD result can be “forward” directly to the CORDIC operand register, CORD[Z:X]

Result chaining between Divider & CORDIC (2/6)

Global Control Register (GLBCON)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		CORDZRC		0	CORDYRC		0	CORDXRC		DVSRC			DVDRC		
r		rw		r	rw		r	rw		rw			rw		

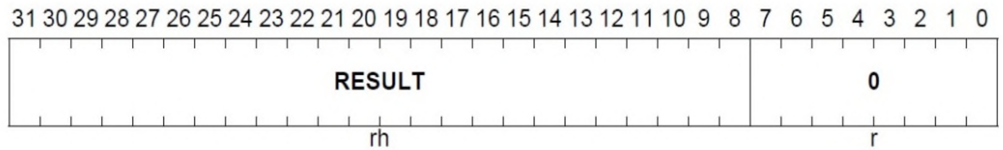
Field	Description
DVDRC DVSRC	**Dividend Register Result Chaining** The DVD register in DIV will be updated with the selected result register value when the result chaining trigger event occurs. 000_B No result chaining is selected 001_B QUOT register is the selected source 010_B RMD register is the selected source 011_B CORRX is the selected source 100_B CORRY is the selected source 101_B CORRZ is the selected source
CORDXRC CORDYRC CORDZRC	**CORDX Register Result Chaining** The CORDX register in CORDIC will be updated with the selected result register value when the result chaining trigger event occurs. 00_B No result chaining is selected 01_B QUOT register is the selected source 10_B RMD register is the selected source

MATH
Result chaining between Divider & CORDIC (3/6)

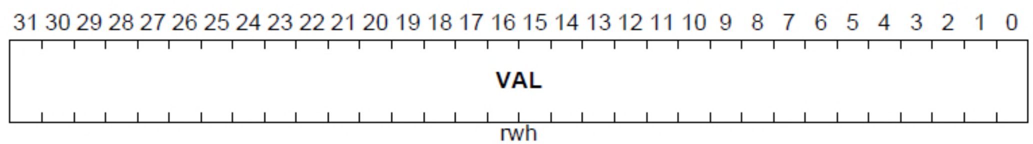
The next few slides illustrate a simple example for result chaining

End of CORDIC computation

CORRX (X Result Register)



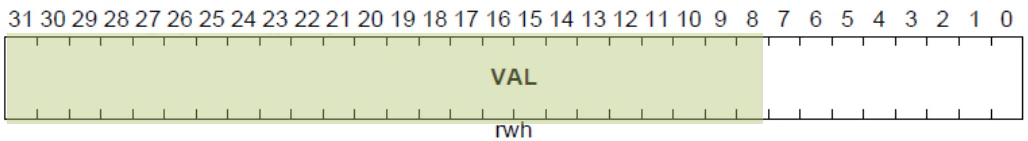
DVS (Divisor Register)



After the computation of the CORDIC operation, the result will be written to CORRX. This result will also be written to DIV’s DVS.

MATH
Result chaining between Divider & CORDIC (4/6)

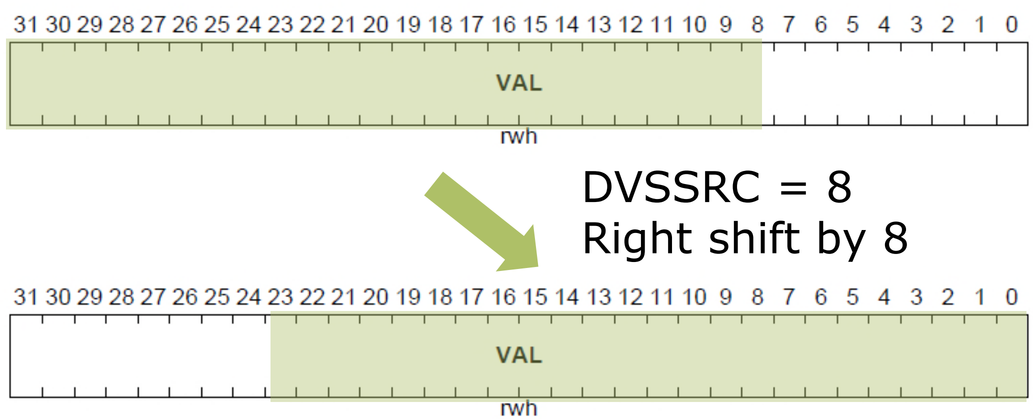
DVS (Divisor Register)



- As the 24-bit CORDIC result is assigned to bit[8 to 31], it might be necessary for some pre-processing of the input value before the DIV operation
- DIVCON.DVSSRC – right shift the input value before the division operation

Value at DVS

Value use for the Division operation



MATH
Result chaining between Divider & CORDIC (5/6)

Function	Vectoring Mode
	d_i = -sign (y_i), $y_i \rightarrow 0$
Circular m = 1 e_i = atan(2^{-i})	X_{final} = K sqrt(X^2+Y^2) / MPS Y_{final} = 0 Z_{final} = Z + atan(Y / X) where K ≈ 1.646760258121

- CORDIC is setup to Circular Vectoring Mode
- CORDIC will start when CORDX is written
- DIV will start when DVS is written
- The result of CORDIC's CORRX will also update DIV's DVS with the same value
- This action will trigger the DIV operation to start
- The DIV's post-processing compensated for the difference in bit length of CORDIC(24-bit) and DIV(32-bit)
- As a result, the writing of CORDIC's CORDX orderly start both CORDIC and DIV

MATH
Result chaining between Divider & CORDIC (6/6)

```
GLBCON = 0x0003;
        // DVSRC = 011b;          // DVS result will be updated when
                                  // CORRX has new result
DIVCON = 0x08000000;
        // ST_MODE = 0;           // Auto-Start when DVS is written
        // DVSSRC = 8;            // DVS value will be shifted right by 8

DVD = 0x12345678;                 // Preload the Dividend value first


CON = 0x0020;
        // MODE = 01b;            // Circular Mode
        // ROTVEC = 0;            // Vectoring Mode
        // ST_MODE = 0;           // Auto-Start when CORDX is written

CORDY = (0x5678<<8);              // Load Y parameter

CORDX = (0x1234<<8);              // Load X parameter and start CORDIC

                                  // Result Chain to DIV's DVS will auto start DIV
```

Agenda

MATH
Benchmarking results (1/2)

- Execution time of a division operation and a cosine operation running on the MATH library is benchmarked against that of a similar operation running on standard C library
- Conditions:
  - Execution time refers to complete function execution, inclusive of co-processor configuration, writing of operands and state checking
  - Ratio of PCLK to MCLK is 2:1
  - Compliers from IFX, Keil and IAR were used

MATH
Benchmarking results (2/2)

› The benchmarking results are shown in the table below:

Compiler	Division (MCLK cycles)		Cosine (MCLK cycles)	
	With MATH LIB	With Std C LIB	With MATH LIB	With Std C LIB
IAR EWARM v7.10	99	712	234	4574
Keil µVision v5.10	95	230	238	6514
DAVE™ v3.1.10	114	415	258	9832

› Significant performance boosts are seen when using MATH library over standard C library:

– ~ 7x performance for division

– ~ 38x performance for cosine

General information

- For latest updates, please refer to:

  www.infineon.com/xmc1000

- For support:

  http://www.infineonforums.com/forums/8-XMC-Forum

Support material

Collaterals and Brochures	› Product Briefs › Selection Guides › Application Brochures › Presentations › Press Releases, Ads	› www.infineon.com/XMC
Technical Material	› Application Notes › Technical Articles › Simulation Models › Datasheets, MCDS Files › PCB Design Data	› www.infineon.com/XMC › Kits and Boards › DAVE™ › Software and Tool Ecosystem
Videos	› Technical Videos › Product Information Videos	› Infineon Media Center › XMC Mediathek
Contact	› Forums › Product Support	› Infineon Forums › Technical Assistance Center (TAC)

Disclaimer

The information given in this training materials is given as a hint for the implementation of the Infineon Technologies component only and shall not be regarded as any description or warranty of a certain functionality, condition or quality of the Infineon Technologies component.

Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this training material.

Part of your life. Part of tomorrow.