Freescale Semiconductor
Engineering Bulletin

Document Number: EB689
Rev. 0, 2/2008

Additional SPE Instructions for Devices that Support VLE

by: Robert Moran
MSG Applications Group

Contents

1 Introduction

1.1 Abstract

The signal processing engine (SPE) for the MPC5500 family provides an instruction set for DSP operations that require multiple data operations in a single instruction. This engineering bulletin describes additional SPE instructions that are implemented on devices with an e200z3 or e200z6 core that supports VLE.

These additional SPE instructions allow a single instruction to perform a vector floating point single precision positive or negative multiply operation along with either an add or substract. These instructions allow for simpler coding of multiply/accumulate operations typically used in filter algorithms.

2 Devices Affected by New Instructions

e200z6 and e200z3 cores that support VLE have additional instructions available in the traditional Power Architecture instruction set.

These instructions perform SPE operations and are currently implemented on the following devices:

Table 1.

Device	Core	VLE Support	Additional SPE Instructions
MPC5533	e200z3	Yes	Yes
MPC5534	e200z3	Yes	Yes
MPC5554	e200z6	No	No
MPC5553	e200z6	No	No
MPC5565	e200z6	Yes	Yes
MPC5566	e200z6	Yes	Yes
MPC5567	e200z6	Yes	Yes

For backwards compatibility to devices that do not support VLE, do not use these new SPE instructions. Compilers may provide a switch to allow or to not allow use of these additional instructions.

2.1 Instructions Op Codes

The op codes of the additional instructions are listed in Table 2.

Table 2. Op Codes of Additional SPE Instructions

Instruction Bit Coding	0…..5	6…..10	11…..15	16…..20	21…..31
evfsmadd	4 (0b0001_00)	RD	RA	RB	0b010_1000_0010
evfsmsub	4 (0b0001_00)	RD	RA	RB	0b010_1000_0011
evfsnmadd	4 (0b0001_00)	RD	RA	RB	0b010_1000_1010
evfsnmsub	4 (0b0001_00)	RD	RA	RB	0b010_1000_1011
efsmadd	4 (0b0001_00)	RD	RA	RB	0b010_1100_0010
efsmsub	4 (0b0001_00)	RD	RA	RB	0b010_1100_0011
efsnmadd	4 (0b0001_00)	RD	RA	RB	0b010_1100_1010
efsnmsub	4 (0b0001_00)	RD	RA	RB	0b010_1100_1011

2.2 Instruction Descriptions

evfsmadd evfsmadd

Vector Floating-Point Single-Precision Multiply-Add

evfsmadd **r**D,**r**A,**r**B

0 5	6 10	11 15	16 20	21 31
0 1 0 0	**r**D	**r**A	**r**B	0 1 0 1 0 0 0 0 0 0 0

$$rD_{0:31} \leftarrow ((rA_{0:31} \times_{fp} rB_{0:31}) +_{sp} rD_{0:31})$$
$$rD_{32:63} \leftarrow ((rA_{32:63} \times_{fp} rB_{32:63}) +_{sp} rD_{32:63})$$

Each single precision floating-point element of **r**A is multiplied with the corresponding element of **r**B. The intermediate product is added to the corresponding element of **r**D and the result is stored in **r**D. If an element of **r**A or **r**B is either zero (or a denormalized number optionally transformed to zero by the implementation), the intermediate product is a properly signed zero.

Otherwise, if an element of **r**A or **r**B is either NaN or infinity, the intermediate product is either *pmax* (asign==bsign), or *nmax* (asign!=bsign). This value is used for the result and stored into **r**D. Otherwise, the intermediate product is added to the corresponding element of **r**D. If **r**D is NaN or infinite, the result is either pmax(dsign==0) or nmax (disgn==1). Otherwise, if an overflow occurs, *pmax* or *nmax* (as appropriate) is stored in **r**D. If an underflow occurs, +0 or – 0 (as appropriate) is stored in **r**D.

Exceptions:

If the contents of either element of **r**A, **r**B or **r**D are Infinity, Denorm, or NaN; SPEFSCR[FINV,FINVH] are set appropriately and SPEFSCR[FGH,FXH,FG,FX] are cleared appropriately. If SPEFSCR[FINVE] is set, an interrupt is taken and the destination register is not updated. Otherwise if an overflow occurs SPEFSCR[FOVF,FOVFH] are set appropriately.

If an underflow occurs, SPEFSCR[FUNF, FUNFH] are set appropriately. If either underflow or overflow exceptions are enabled and a corresponding status bit is set, an interrupt is taken. If any of these interrupts are taken, the destination register is not updated.

If either result element of this instruction is inexact, or overflows but overflow exceptions are disabled, and no other interrupt is taken, IPEFSCR[FINXS] is set. If the floating-point inexact exception is enabled, an interrupt is taken using the floating-point round interrupt vector. In this case, the destination register is updated with the truncated result(s). The FG and FX bits are properly updated to allow rounding to be performed in the interrupt handler.

FG and FX (FGH and FXH) are cleared if an overflow or underflow exception is taken, or if an invalid operation/input error is signaled for the low (high) element (regardless of FINVE).

evfsmsub evfsmsub

Vector Floating-Point Single-Precision Multiply-Substract

evfsmsub **r**D,**r**A,**r**B

0 5	6 10	11 15	16 20	21 31
0 1 0 0	**r**D	**r**A	**r**B	0 1 0 1 0 0 0 0 0 0 0

$$rD_{0:31} \leftarrow ((rA_{0:31} \times_{fp} rB_{0:31} -_{sp} rD_{0:31})$$

$$rD_{32:63} \leftarrow ((rA_{32:63} \times_{fp} rB_{32:63} -_{sp} rD_{32:63})$$

Each single-precision floating-point element of **r**A is multiplied with the corresponding element of **r**B. The corresponding element of **r**D is subtracted from the intermediate product, and the result is stored in **r**D. If rA or rB are either zero or denormalized, the intermediate product is a properly signed zero.

Otherwise, if rA or rB are either NaN or infinity, the intermediate product is either *pmax* (asign==bsign) or *nmax* (asign!=bsign), and this value is used for the result and stored into rD. Otherwise, the corresponding element of **r**D is subtracted from the intermediate product.

If rD isNaN or infinity, the result is either *nmax* (dsign==0), or *pmax* (dsign==1). Otherwise, if an overflowoccurs, then *pmax* or *nmax* (as appropriate) is stored in rD. If an underflow occurs, then +0 (for rounding modes RN, RZ, RP) or -0 (for rounding mode RM) is stored in rD.

Exceptions:

If the contents of either element of rA, rB, or rD are Infinity, Denorm, or NaN, the SPEFSCR[FINV, FINVH] bits are set appropriately, and the SPEFSCR[FGH, FXH, FG, FX] bits are cleared appropriately.

If SPEFSCR[FINVE] is set, an exception is taken and the destination register is not updated. Otherwise, if an overflow occurs, then the SPEFSCR[FOVF, FOVFH] bits are set appropriately, or if an underflow occurs, then the SPEFSCR[FUNF, FUNFH] bits are set appropriately. If either underflow or overflow exceptions are enabled and a corresponding status bit is set, an exception is taken. If any of these exceptions are taken, the destination register is not updated.

The SPEFSCR[FINXS] bit will be set if either result element of this instruction is inexact, or overflows but overflow exceptions are disabled, and no other exception is taken, or underflows but underflow exceptions are disabled, and no other exception is taken.

If the Floating-point Inexact exception is enabled, an exception is taken using the Floating-point Round exception vector. In this case, the destination register is updated with the truncated result(s). The FG and FX bits are properly updated to allow rounding to be performed in the exception handler.

FG and FX (FGH and FXH) will be cleared if an overflow or underflow exception is taken, or if an invalid operation/input error is signaled for the low (high) element (regardless of FINVE).

evfsnmadd evfsnmadd

Vector Floating-Point Single-Precision Negative Multiply-Substract

evfsnmadd **r**D,**r**A,**r**B

0 5	6 10	11 15	16 20	21 31
0 1 0 0	**r**D	**r**A	**r**B	0 1 0 1 0 0 0 0 0 0 0

$$rD_{0:31} \leftarrow ((\ rA_{0:31} \times \ rB_{0:31} -_{sp} rD_{0:31})$$

$$rD_{32:63} \leftarrow ((\ rA_{32:63} \times \ rB_{32:63} -_{sp} rD_{32:63})$$

Each single-precision floating-point element of **r**A is multiplied with the corresponding element of **r**B. The intermediate product is added to the corresponding element of **r**D, and the negated result is stored in **r**D. If rA or rB are either zero or denormalized, the intermediate product is a properly signed zero.

Otherwise, if rA or rB are either NaN or infinity, the intermediate product is either *pmax* (asign==bsign), or *nmax* , (asign!=bsign) and this value is used for the result and stored into rD. Otherwise, the intermediate product is added to the corresponding element of rD, and the final result is negated.

If rD is NaN or infinity, the result is either *nmax* (dsign==0), or *pmax* (dsign==1). Otherwise, if an overflow occurs, then *pmax* or *nmax* (as appropriate) is stored in rD. If an underflow occurs, then -0 (for rounding modes RN, RZ, RP) or +0 (for rounding mode RM) is stored in rD.

Exceptions:

If the contents of either element of rA, rB, or rD are Infinity, Denorm, or NaN, the SPEFSCR[FINV, FINVH] bits are set appropriately, and the SPEFSCR[FGH, FXH, FG, FX] bits are cleared appropriately. If SPEFSCR[FINVE] is set, an exception is taken and the destination register is not updated.

Otherwise, if an overflow occurs, then the SPEFSCR[FOVF, FOVFH] bits are set appropriately, or if an underflow occurs, then the SPEFSCR[FUNF, FUNFH] bits are set appropriately. If either underflow or overflow exceptions are enabled and a corresponding status bit is set, an exception is taken. If any of these exceptions are taken, the destination register is not updated.

The SPEFSCR[FINXS] bit will be set if either result element of this instruction is inexact, or overflows but overflow exceptions are disabled, and no other exception is taken, or underflows but underflow exceptions are disabled, and no other exception is taken.

If the Floating-point Inexact exception is enabled, an exception is taken using the Floating-point Round exception vector. In this case, the destination register is updated with the truncated result(s). The FG and FX bits are properly updated to allow rounding to be performed in the exception handler.

FG and FX (FGH and FXH) will be cleared if an overflow or underflow exception is taken, or if an invalid operation/input error is signaled for the low (high) element (regardless of FINVE).

evfsnmsub evfsnmsub

Vector Floating-Point Single-Precision Negative Multiply-Substract

evfsnmsub **r**D,**r**A,**r**B

0–5	6–10	11–15	16–20	21–31
0 1 0 0	**r**D	**r**A	**r**B	0 1 0 1 0 0 0 0 0 0 0

$$rD_{0:31} \leftarrow ((\ rA_{0:31} \times \ rB_{0:31}\ -_{sp} rD_{0:31})$$
$$rD_{32:63} \leftarrow ((\ rA_{32:63} \times \ rB_{32:63}\ -_{sp}\ rD_{32:63})$$

Each single-precision floating-point element of **r**A is multiplied with the corresponding element of **r**B. The corresponding element of **r**D is subtracted from the intermediate product, and the negated result is stored in **r**D. If an element of **r**A or **r**B are either zero (or a denormalized number optionally transformed to zero by the implementation), the intermediate product is a properly signed zero.

Otherwise, if an element of **r**A or **r**B are either NaN or infinity, the intermediate product is either *pmax* (asign==bsign), or *nmax* (asign!=bsign), and this value is negated to obtain the result and stored into **r**D. Otherwise, the corresponding element of **r**D is subtracted from the intermediate product, and the final result is negated. If **r**D is NaN or infinit, the result is either *pmax*(dsign==0)or nmax (dsign==1). Otherwise, if an overflow occurs, *pmax* or *nmax* (as appropriate) is stored in rD. If an underflow occurs, +0 or -0 (as appropriate) is stored in rD.

Exceptions:

If the contents of either element of **r**A, **r**B or **r**D are Infinity, Denorm, or NaN, SPEFSCR[FINV,FINVH] are set appropriately, and SPEFSCR[FGH,FXH,FG,FX] are cleared appropriately. If SPEFSCR[FINVE] is set, an interrupt is taken and the destination register is not updated. Otherwise, if an overflow occurs, SPEFSCR[FOVF,FOVFH] are set appropriately, or if an underflow occurs, SPEFSCR[FUNF,FUNFH] are set appropriately. If either underflow or overflow exceptions are enabled and a corresponding status bit is set, an interrupt is taken. If any of these interrupts are taken, the destination register is not updated.

If either result element of this instruction is inexact, or overflows but overflow exceptions are disabled, and no other interrupt is taken, or underflows but underflow exceptions are disabled, and no other interrupt is taken, SPEFSCR[FINXS] is set. If the floating-point inexact exception is enabled, an interrupt is taken using the floating-point round interrupt vector. In this case, the destination register is updated with the truncated result(s). The FG and FX bits are properly updated to allow rounding to be performed in the interrupt handler.

FG and FX (FGH and FXH) are cleared if an overflow or underflow exception is taken, or if an invalid operation/input error is signaled for the low (high) element (regardless of FINVE).

efsmadd efsmadd

Floating-Point Single-Precision Multiply-Add

efsmadd rD,**r**A,**r**B

0 5	6 10	11 15	16 20	21 31
0 1 0 0	**r**D	**r**A	**r**B	0 1 0 1 0 0 0 0 0 0 0

$$rD_{0:31} \leftarrow ((\ rA_{0:31} \times\ rB_{0:31} -_{sp} rD_{0:31})$$
$$rD_{32:63} \leftarrow ((\ rA_{32:63} \times\ rB_{32:63} -_{sp} rD_{32:63})$$

The low element of **r**A is multiplied by the low element of **r**B, the intermediate product is added to the low element of **r**D, and the result is stored in the low element of **r**D. If **r**A or **r**B are either zero or denormalized, the intermediate product is a properly signed zero.

Otherwise, if **r**A or **r**B are either NaN or infinity, the intermediate product is either *pmax* (asign==bsign), or *nmax* (asign!=bsign), and this value is used for the result and stored into **r**D. Otherwise, the intermediate product is added to the corresponding element of **r**D. If **r**D is NaN or infinity, the result is either *pmax* (dsign==0), or nmax (dsign==1). Otherwise, if an overflow occurs, then *pmax* or *nmax* (as appropriate) is stored in **r**D. If an underflow occurs, then +0 (for rounding modes RN, RZ, RP) or -0 (for rounding mode RM) is stored in **r**D.

Exceptions:

If the contents of **r**A or **r**B are Infinity, Denorm, or NaN, the SPEFSCR[FINV] bit is set. If SPEFSCR[FINVE] is set, an exception is taken, and the destination register is not updated. Otherwise, if an overflow occurs, then the SPEFSCR[FOVF] bit is set, or if an underflow occurs, then the SPEFSCR[FUNF] bit is set. If either underflow or overflow exceptions are enabled and the corresponding bit is set, an exception is taken. If any of these exceptions are taken, the destination register is not updated.

The SPEFSCR[FINXS] bit will be set if the result of this instruction is inexact, or if an overflow occurs on the add, but overflow exceptions are disabled, and no other exception is taken. If the Floating-point Inexact exception is enabled, an exception is taken using the Floating-point Round exception vector. In this case, the destination register is updated with the truncated result, the FG and FX bits are properly updated to allow rounding to be performed in the exception handler, and the FGH and FXH bits are cleared.

FGH, FXH, FG and FX will be cleared if an overflow, underflow, or invalid operation/input error is signaled, regardless of enabled exceptions.

efsmsub efsmsub

Floating-Point Single-Precision Multiply-Substract

efsmsub **r**D,**r**A,**r**B

0–5	6–10	11–15	16–20	21–31
0 1 0 0	**r**D	**r**A	**r**B	0 1 0 1 0 0 0 0 0 0 0

$$rD_{0:31} \leftarrow ((rA_{0:31} \times rB_{0:31}\ -_{sp}\ rD_{0:31})$$

$$rD_{32:63} \leftarrow ((rA_{32:63} \times rB_{32:63}\ -_{sp}\ rD_{32:63})$$

The low element of **r**A is multiplied by the low element of **r**B, the low element of **r**D is subtracted from the intermediate product, and the result is stored in the low element of **r**D. If **r**A or **r**B are either zero or denormalized, the intermediate product is a properly signed zero. Otherwise, if **r**A or **r**B are either NaN or infinity, the intermediate product is either *pmax* (asign==bsign), or *nmax*(asign!=bsign), and this value is used for the result and stored into **r**D.

Otherwise, the low element ofrD is subtracted from the intermediate product. If rD is NaN or infinity, the result is either *nmax*(dsign==0), or pmax (dsign==1). Otherwise, if an overflow occurs, then *pmax* or *nmax* (as appropriate) is stored in **r**D. If an underflow occurs, then +0 (for rounding modes RN, RZ, RP) or -0 (for rounding mode RM) is stored in **r**D.

Exceptions:

If the contents of **r**A or **r**B are Infinity, Denorm, or NaN, the SPEFSCR[FINV] bit is set. If SPEFSCR[FINVE] is set, an exception is taken, and the destination register is not updated. Otherwise, if an overflow occurs, then the SPEFSCR[FOVF] bit is set, or if an underflow occurs, then the SPEFSCR[FUNF] bit is set.

If either underflow or overflow exceptions are enabled and the corresponding bit is set, an exception is taken. If any of these exceptions are taken, the destination register is not updated. If the result of this instruction is inexact or if an overflow occurs but overflow exceptions are disabled, and no other exception is taken, the SPEFSCR[FINXS] bit will be set.

If the Floating-point Inexact exception is enabled, an exception is taken using the Floating-point Round exception vector. In this case, the destination register is updated with the truncated result, the FGand FX bits are properly updated to allow rounding to be performed in the exception handler, and the FGH and FXH bits are cleared.

FGH, FXH, FG and FX will be cleared if an overflow, underflow, or invalid operation/input error is signaled, regardless of enabled exceptions.

efsnmadd efsnmadd

Floating-Point Single-Precision Negative Multiply-Add

efsnmadd **r**D,**r**A,**r**B

0 5	6 10	11 15	16 20	21 31
0 1 0 0	**r**D	**r**A	**r**B	0 1 0 1 0 0 0 0 0 0 0

$$rD_{0:31} \leftarrow ((\ rA_{0:31} \times\ rB_{0:31}\ -_{sp} rD_{0:31})$$
$$rD_{32:63} \leftarrow\ ((\ rA_{32:63} \times\ \ rB_{32:63}\ -_{sp}\ rD_{32:63})$$

The low element of **r**A is multiplied by the low element of **r**B, the intermediate product is added to the low element of **r**D, and the negated result is stored in the low element of **r**D. If **r**A or **r**B are either zero or denormalized, the intermediate product is a properly signed zero.

Otherwise, if **r**A or **r**B are either NaN or infinity, the intermediate product is either *pmax* (asign==bsign), or *nmax* (asign!=bsign), and this value is used for the result and stored into rD. Otherwise, the intermediate product is added to the corresponding element of **r**D, and the final result is negated.

If **r**D is NaN or infinity, the result is either *nmax* (dsign==0), or *pmax* (dsign==1). Otherwise, if an overflow occurs, then *pmax* or *nmax* (as appropriate) is stored in **r**D. If an underflow occurs, then -0 (for rounding modes RN, RZ, RP) or +0 (for rounding mode RM) is stored in **r**D.

Exceptions:

If the contents of **r**A or **r**B are Infinity, Denorm, or NaN, the SPEFSCR[FINV] bit is set. If SPEFSCR[FINVE] is set, an exception is taken, and the destination register is not updated. Otherwise, if an overflow occurs, then the SPEFSCR[FOVF] bit is set, or if an underflow occurs, then the SPEFSCR[FUNF] bit is set. If either underflow or overflow exceptions are enabled and the corresponding bit is set, an exception is taken. If any of these exceptions are taken, the destination register is not updated.

If the result of this instruction is inexact or if an overflow occurs but overflow exceptions are disabled, and no other exception is taken, the SPEFSCR[FINXS] bit will be set. If the Floating-point Inexact exception is enabled, an exception is taken using the Floating-point Round exception vector. In this case, the destination register is updated with the truncated result, the FG and FX bits are properly updated to allow rounding to be performed in the exception handler, and the FGH and FXH bits are cleared.

FGH, FXH, FG and FX will be cleared if an overflow, underflow, or invalid operation/input error is signaled, regardless of enabled exceptions.

efsnmsub efsnmsub

Floating-Point Single-Precision Negative Multiply-Substract

efsnmsub **r**D,**r**A,**r**B

0 5	6 10	11 15	16 20	21 31
0 1 0 0	**r**D	**r**A	**r**B	0 1 0 1 0 0 0 0 0 0 0

$$rD_{0:31} \leftarrow ((\ rA_{0:31} \times \ rB_{0:31}\ -_{sp} rD_{0:31})$$

$$rD_{32:63} \leftarrow ((\ rA_{32:63} \times \ rB_{32:63}\ -_{sp}\ rD_{32:63})$$

The low element of element of **r**A is multiplied by the low element of **r**B. The low element of **r**D is subtracted from the intermediate product, and the negated result is stored in the low element of **r**D. If **r**A or **r**B are either zero or denormalized, the intermediate product is a properly signed zero.

Otherwise, if **r**A or **r**B are either NaN or infinity, the intermediate product is either *pmax* (asign==bsign), or *nmax* (asign!=bsign), and this value is negated to obtain the result and is stored into **r**D. Otherwise, the low element of **r**D is subtracted from the intermediate product, and the final result is negated. If **r**D is NaN or infinity, the final result is either *pmax* (dsign==0), or *nmax* (dsign==1). Otherwise, if an overflow occurs, then *pmax* or *nmax* (as appropriate) is stored in **r**D. If an underflow occurs, then -0 (for rounding modes RN, RZ, RP) or +0 (for rounding mode RM) is stored in **r**D.

Exceptions:

If the contents of **r**A or **r**B are Infinity, Denorm, or NaN, the SPEFSCR[FINV] bit is set. If SPEFSCR[FINVE] is set, an exception is taken, and the destination register is not updated. Otherwise, if an overflow occurs, then the SPEFSCR[FOVF] bit is set, or if an underflow occurs, then the SPEFSCR[FUNF] bit is set. If either underflow or overflow exceptions are enabled and the corresponding bit is set, an exception is taken. If any of these exceptions are taken, the destination register is not updated.

If the result of this instruction is inexact or if an overflow occurs but overflow exceptions are disabled, and no other exception is taken, the SPEFSCR[FINXS] bit will be set. If the Floating-point Inexact exception is enabled, an exception is taken using the Floating-point Round exception vector. In this case, the destination register is updated with the truncated result, the FG and FX bits are properly updated to allow rounding to be performed in the exception handler, and the FGH and FXH bits are cleared.

FGH, FXH, FG and FX will be cleared if an overflow, underflow, or invalid operation/input error is signaled, regardless of enabled exceptions.

THIS PAGE IS INTENTIONALLY BLANK

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: EB689
Rev. 0
2/2008