Armv8-M Mainline DSP Extension

Course 4 Advanced Armv8-M Features

Objectives

At the end of this topic, you will be able to:

- Summarize the DSP Extension and its related features in the Armv8-M architecture
- Understand different instruction mnemonics and how they operate
- Describe the tools and library support for the DSP Extension including CMSIS-DSP

Agenda

- **Overview of DSP in the Armv8-M Architecture**

- DSP Extension Instruction Set

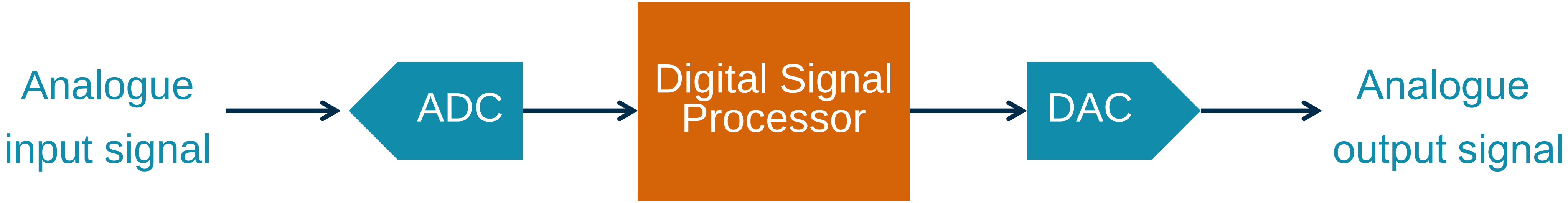
- Tools and Library Support

What is DSP?

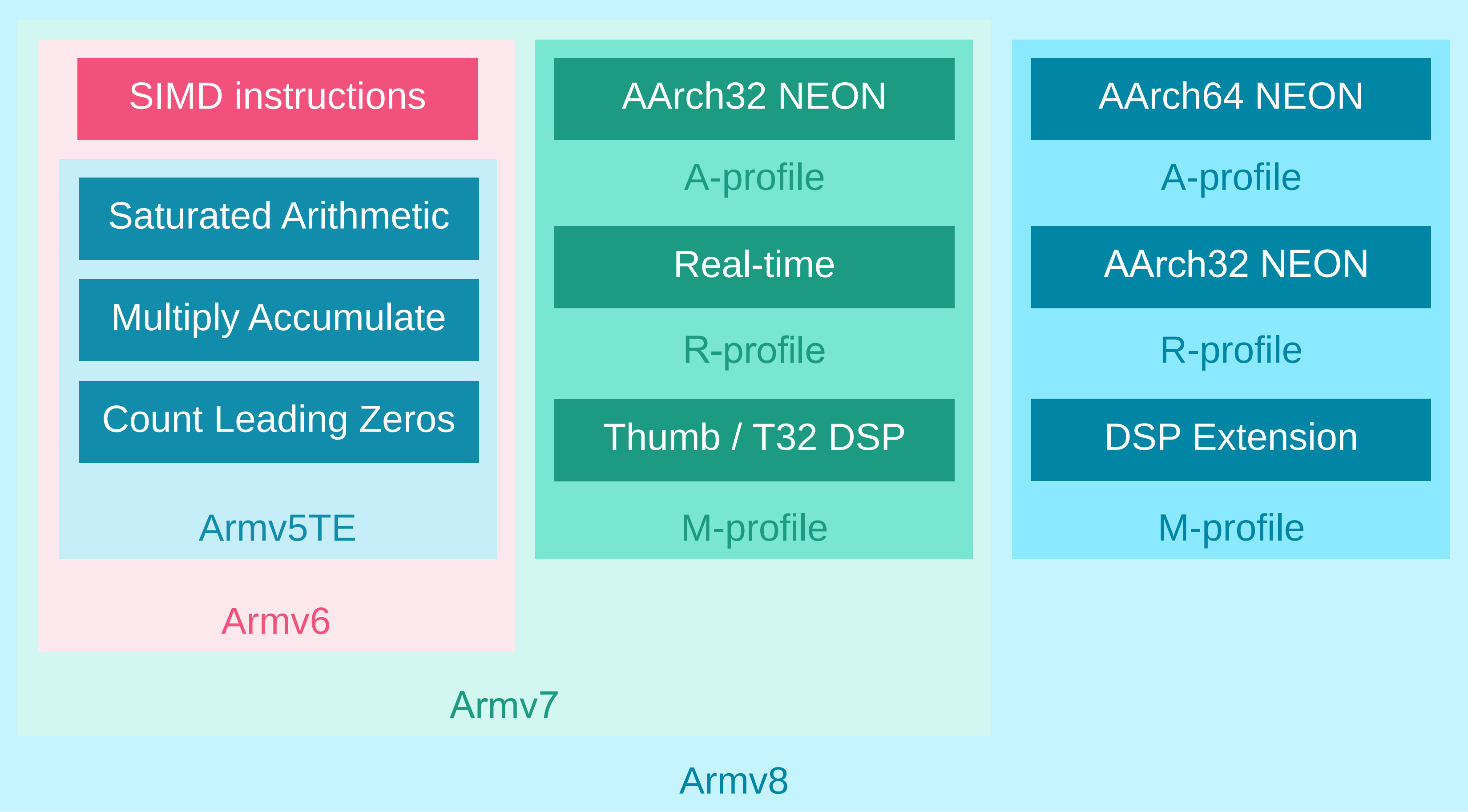
An important and fundamental element of many electrical engineering programmes

A comparatively mathematical topic, traditionally taught in lectures

{real-time} processing of sampled and quantized signals {representing physical quantities}



Evolution of DSP instructions in the Arm architecture



SIMD instructions

Saturated Arithmetic

Multiply Accumulate

Count Leading Zeros

Armv5TE

Armv6

AArch32 NEON

A-profile

Real-time

R-profile

Thumb / T32 DSP

M-profile

Armv7

AArch64 NEON

A-profile

AArch32 NEON

R-profile

DSP Extension

M-profile

Armv8

Cortex-M portfolio DSP extension overview

DSP Extension optimized for fast fixed-point and complex number arithmetic

- Saturating operations – increased use of sticky Q bit in APSR
- Asymmetric multiplies and multiply-accumulates

Armv6 SIMD instructions

- Use core registers
- Powerful byte and half-word operations

Additional APSR field

- SIMD comparison field GE

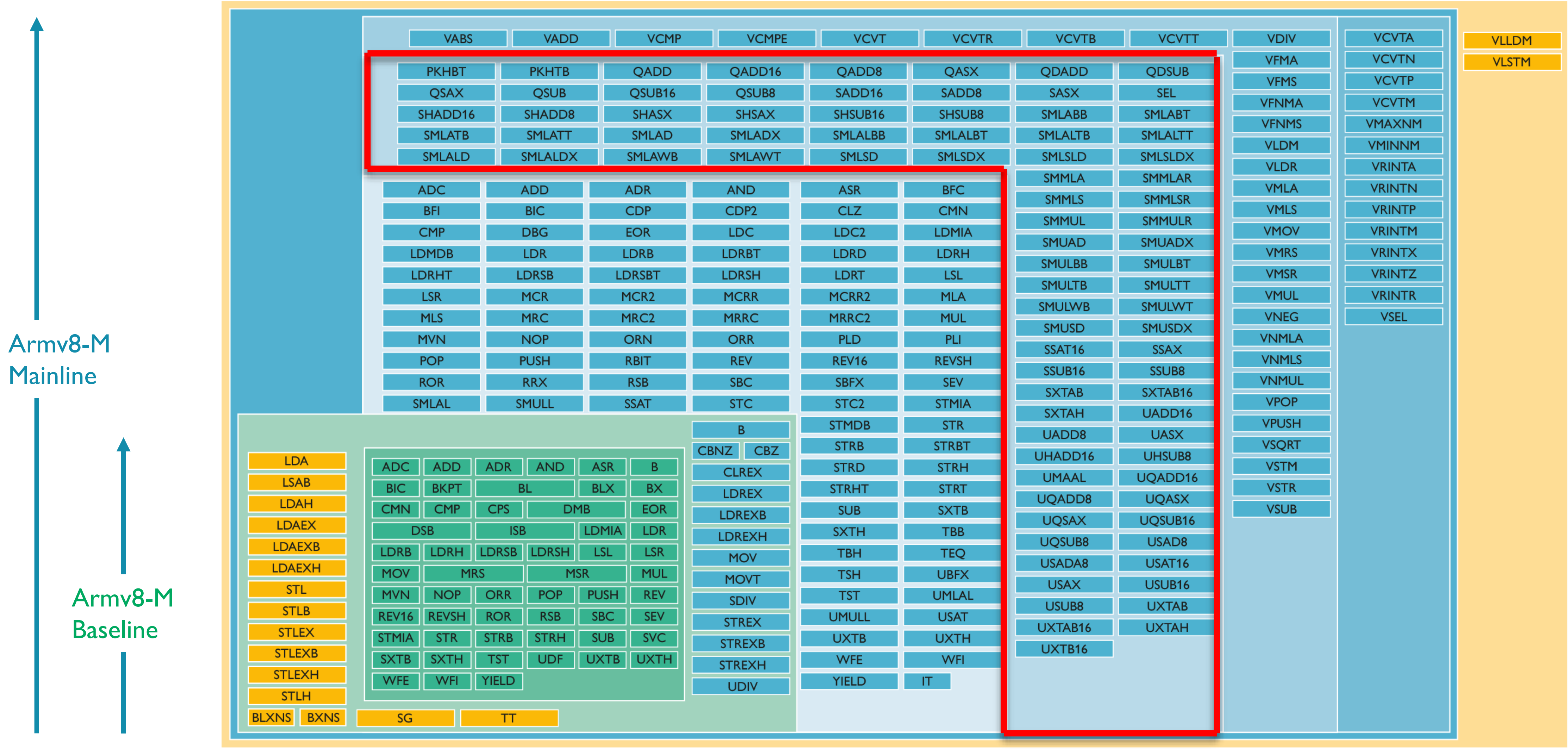
Armv7-AR upwards compatible

Agenda

- Overview of DSP in the Armv8-M architecture
- **DSP Extension Instruction Set**
- Tools and Library support

Armv8-M instruction set

New Armv8-M instructions



Armv8-M Mainline

Armv8-M Baseline

VABS VADD VCMP VCMPE VCVT VCVTR VCVTB VCVTT

PKHBT PKHTB QADD QADD16 QADD8 QASX QDADD QDSUB
QSAX QSUB QSUB16 QSUB8 SADD16 SADD8 SASX SEL
SHADD16 SHADD8 SHASX SHSAX SHSUB16 SHSUB8 SMLABB SMLABT
SMLATB SMLATT SMLAD SMLADX SMLALBB SMLALBT SMLALTB SMLALTT
SMLALD SMLALDX SMLAWB SMLAWT SMLSD SMLSDX SMLSLD SMLSLDX
SMMLA SMMLAR
SMMLS SMMLSR
SMMUL SMMULR
SMUAD SMUADX
SMULBB SMULBT
SMULTB SMULTT
SMULWB SMULWT
SMUSD SMUSDX
SSAT16 SSAX
SSUB16 SSUB8
SXTAB SXTAB16
SXTAH UADD16
UADD8 UASX
UHADD16 UHSUB8
UMAAL UQADD16
UQADD8 UQASX
UQSAX UQSUB16
UQSUB8 USAD8
USADA8 USAT16
USAX USUB16
USUB8 UXTAB
UXTAB16 UXTAH
UXTB16

ADC ADD ADR AND ASR BFC
BFI BIC CDP CDP2 CLZ CMN
CMP DBG EOR LDC LDC2 LDMIA
LDMDB LDR LDRB LDRBT LDRD LDRH
LDRHT LDRSB LDRSBT LDRSH LDRT LSL
LSR MCR MCR2 MCRR MCRR2 MLA
MLS MRC MRC2 MRRC MRRC2 MUL
MVN NOP ORN ORR PLD PLI
POP PUSH RBIT REV REV16 REVSH
ROR RRX RSB SBC SBFX SEV
SMLAL SMULL SSAT STC STC2 STMIA
STMDB STR
STRB STRBT
STRD STRH
STRHT STRT
SUB SXTB
SXTH TBB
TBH TEQ
TSH UBFX
TST UMLAL
UMULL USAT
UXTB UXTH
WFE WFI
YIELD IT

B
CBNZ CBZ
CLREX
LDREX
LDREXB
LDREXH
MOV
MOVT
SDIV
STREX
STREXB
STREXH
UDIV

ADC ADD ADR AND ASR B
BIC BKPT BL BLX BX
CMN CMP CPS DMB EOR
DSB ISB LDMIA LDR
LDRB LDRH LDRSB LDRSH LSL LSR
MOV MRS MSR MUL
MVN NOP ORR POP PUSH REV
REV16 REVSH ROR RSB SBC SEV
STMIA STR STRB STRH SUB SVC
SXTB SXTH TST UDF UXTB UXTH
WFE WFI YIELD

LDA
LSAB
LDAH
LDAEX
LDAEXB
LDAEXH
STL
STLB
STLEX
STLEXB
STLEXH
STLH
BLXNS BXNS SG TT

VDIV
VFMA
VFMS
VFNMA
VFNMS
VLDM
VLDR
VMLA
VMLS
VMOV
VMRS
VMSR
VMUL
VNEG
VNMLA
VNMLS
VNMUL
VPOP
VPUSH
VSQRT
VSTM
VSTR
VSUB

VCVTA
VCVTN
VCVTP
VCVTM
VMAXNM
VMINNM
VRINTA
VRINTN
VRINTP
VRINTM
VRINTX
VRINTZ
VRINTR
VSEL

VLLDM
VLSTM

Armv8-M DSP Extension instructions

SIMD instructions

	S Signed	Q Signed Saturating	SH Signed Halving	U Unsigned	UQ Unsigned Saturating	UH Unsigned Halving
ADD8	SADD8	QADD8	SHADD8	UADD8	UQADD8	UHADD8
SUB8	SSUB8	QSUB8	SHSUB8	USUB8	UQSUB8	UHSUB8
ADD16	SADD16	QADD16	SHADD16	UADD16	UQADD16	UHADD16
SUB16	SSUB16	QSUB16	SHSUB16	USUB16	UQSUB16	UHSUB16
ASX	SASX	QASX	SHASX	UASX	UQASX	UHASX
SAX	SSAX	QSAX	SHSAX	USAX	UQSAX	UHSAX
USAD8	Unsigned Sum of Absolute Difference (8 bits)					
USADA8	Unsigned Sum of Absolute Difference and Accumulate (8 bits)					

ASX

1. Exchanges halfwords of the second operand register
2. Adds top halfwords and subtracts bottom halfwords

SAX

1. Exchanges halfwords of the second operand register
2. Subtracts top halfwords and adds bottom halfwords

All of these instructions set GE bits

SIMD Example: UADD8

```
UADD8  r0, r1, r2
```

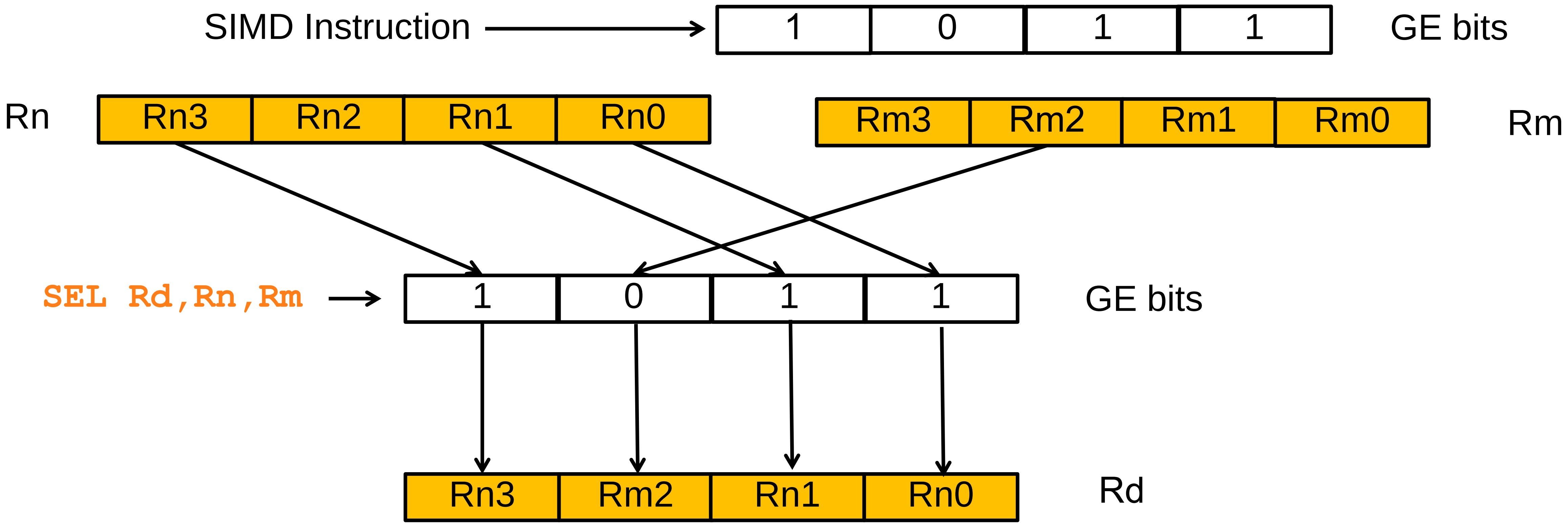
- Register split into equal size and type elements
- Operation performed on same element of each register
- Sets APSR.GE bits according to the results of the additions (if there’s a carry)

31	23	15	7 0	
0x10	0x34	0x09	0xAB	R2
+	+	+	+	
0xF0	0x78	0xF8	0xCD	R1
=	=	=	=	
0x00	0xAC	0xFF	0x78	R0

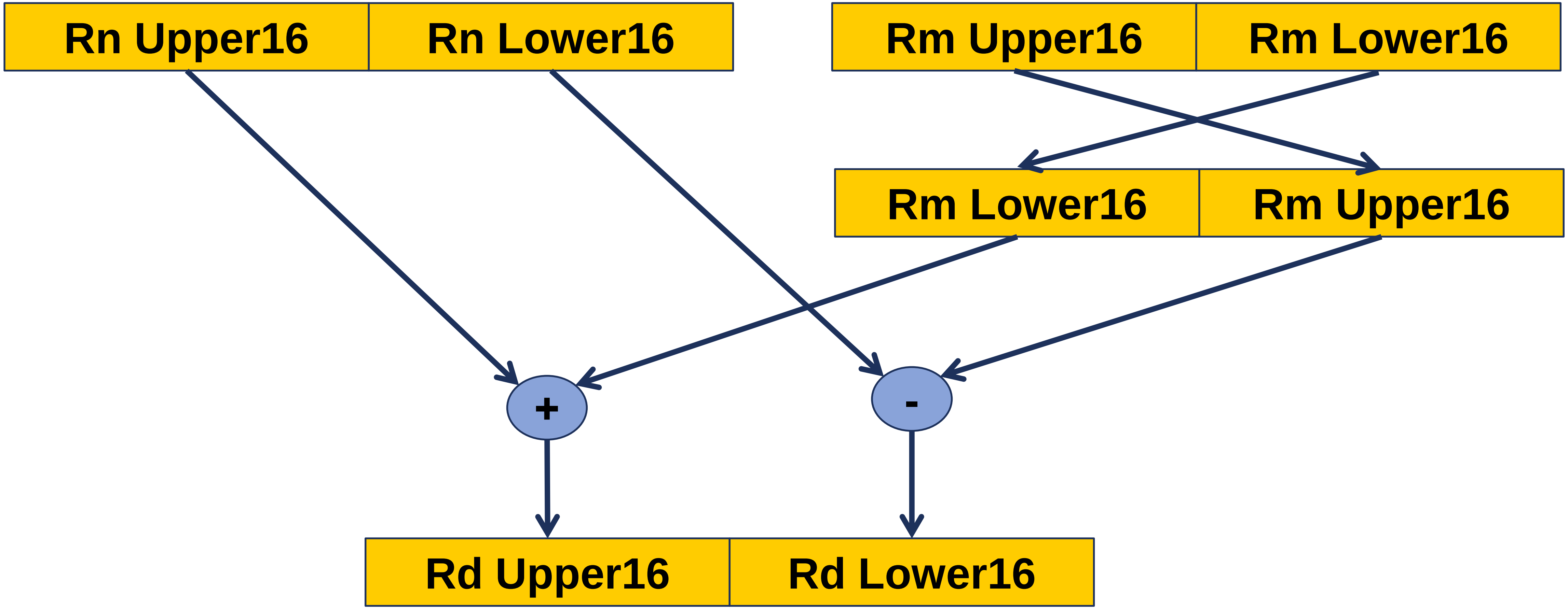
1	0	1	1	APSR.GE

SIMD comparisons

- GE bits are set by SIMD instructions
- SEL instruction select bytes from Rn or Rm, based on GE.
- Halfword SIMD instructions set two bits per field
- Byte-wise SIMD instructions set 1 bit per field



ASX instruction



Rn Upper16 | Rn Lower16

Rm Upper16 | Rm Lower16

Rm Lower16 | Rm Upper16

+

-

Rd Upper16 | Rd Lower16

Add **S**ubtract e**X**change

SAX swaps + and -

Saturation arithmetic

31 28	27	26 25	24	23	19 16	15 0
N Z C V	Q				GE[3:0]	

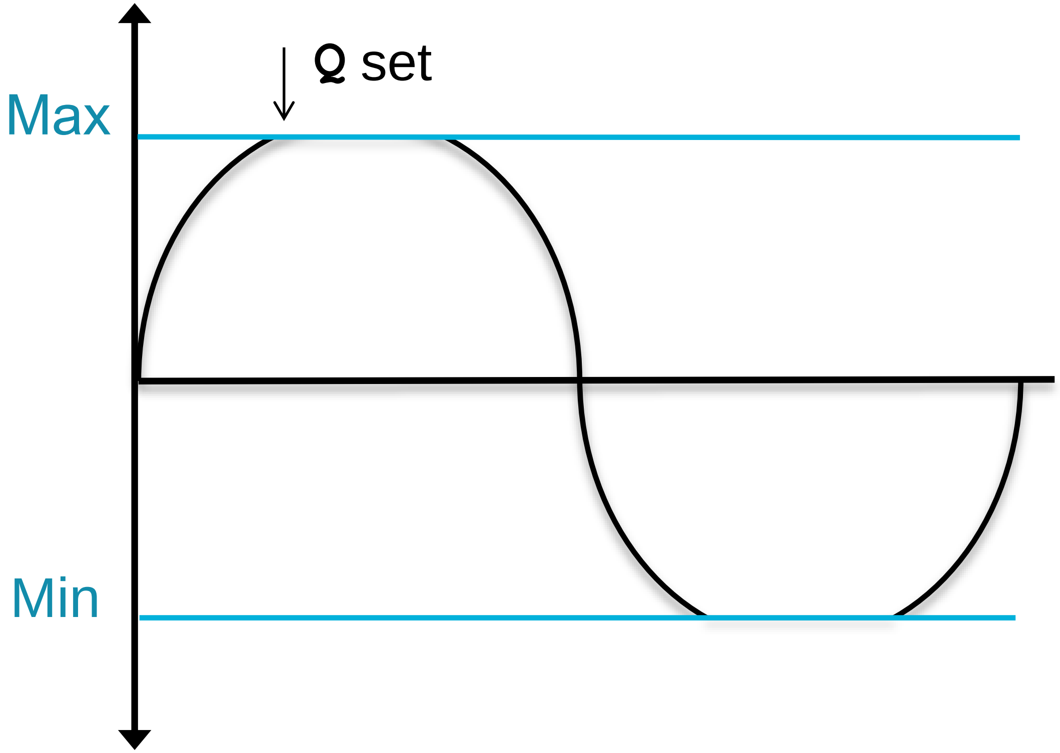
APSR

In normal integer arithmetic, incrementing beyond maximum value or decrementing below 0 makes the result wrap

- Sign is irrelevant

In saturating arithmetic, incrementing beyond maximum value or decrementing below minimum value makes the result clamp

- Can be signed or unsigned
- The output of the operation is “clipped”
- Q bit is set, and only cleared explicitly
- 8-, 16- and 32-bit saturating instructions available



SIMD Example: QADD8

```
QADD8  r0, r1, r2
```

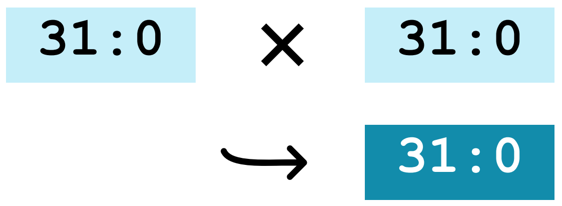
- Register split into equal size and type elements
- Operation performed on same element of each register
- Sets APSR.Q bits according to the results of the additions

31	23	15	7 0	
0xFF	0xFF	0xFF	0xFF	R2
+	+	+	+	
0xF0	0x78	0xF8	0xCD	R1
=	=	=	=	
0xEF	0x77	0xF7	0xCC	R0

Multiply operations

Available with DSP features	SMULBB SMULBT SMULTB SMULTT SMLABB SMLABT SMLATB SMLATT SMLALBB SMLALBT SMLALTB SMLALTT SMULWB SMULWT SMLAWB SMLAWT SMUAD SMUADX SMUSD SMUSDX SMLAD SMLADX SMLSD SMLSDX SMLALD SMLALDX SMLSLD SMLSLDX SMMLA SMMLAR SMMLS SMMLSR SMMUL SMMULR
Standard on Armv8-M	MUL MLA MLS SMULL UMULL SMLAL UMLAL UMAAL

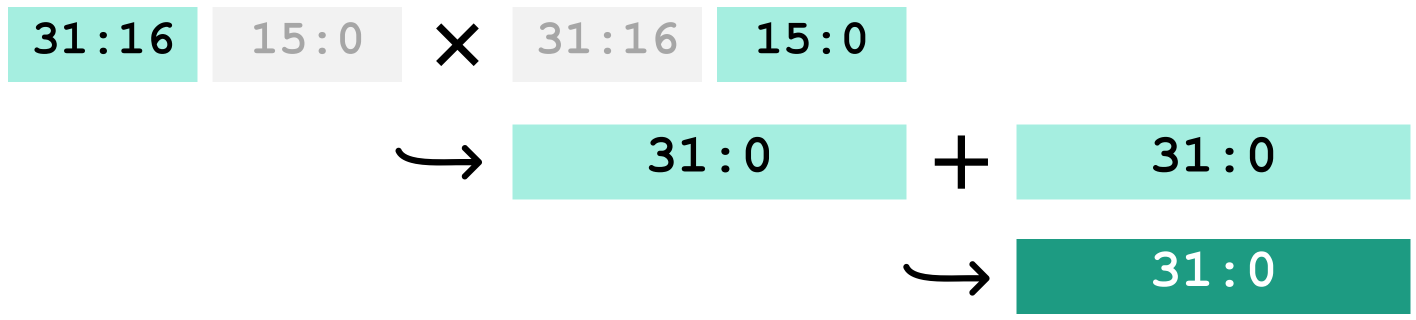
`MUL Rdm, Rn, Rdm`



31:0 × 31:0

→ 31:0

`SMLATB Rd, Rn, Rm, Ra`



31:16 15:0 × 31:16 15:0

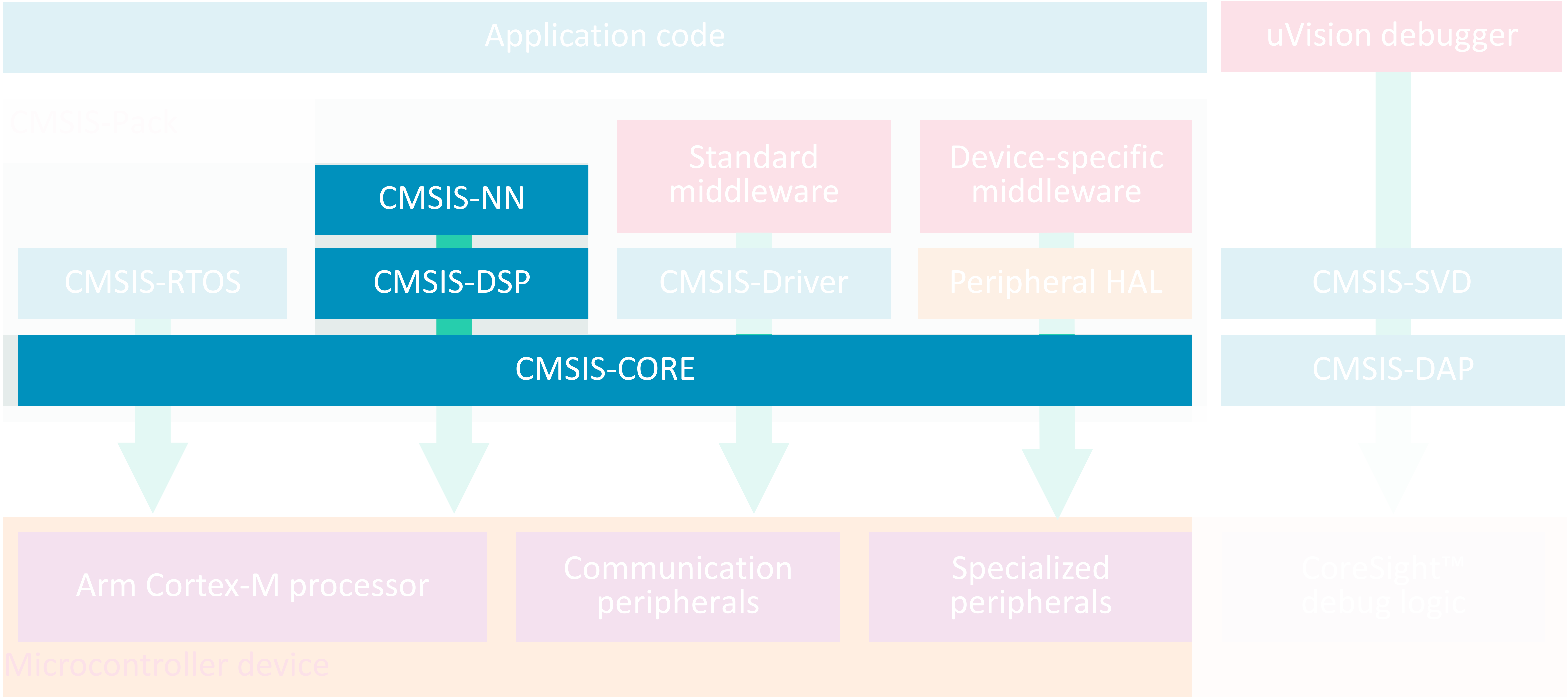
→ 31:0 + 31:0

→ 31:0

Agenda

- Overview of DSP in the Armv8-M architecture
- DSP Extension Instruction Set
- **Tools and Library support**

Common Microcontroller Software Interface Standard



Application code

uVision debugger

CMSIS-Pack

CMSIS-NN

Standard middleware

Device-specific middleware

CMSIS-RTOS

CMSIS-DSP

CMSIS-Driver

Peripheral HAL

CMSIS-SVD

CMSIS-CORE

CMSIS-DAP

Arm Cortex-M processor

Communication peripherals

Specialized peripherals

CoreSight™ debug logic

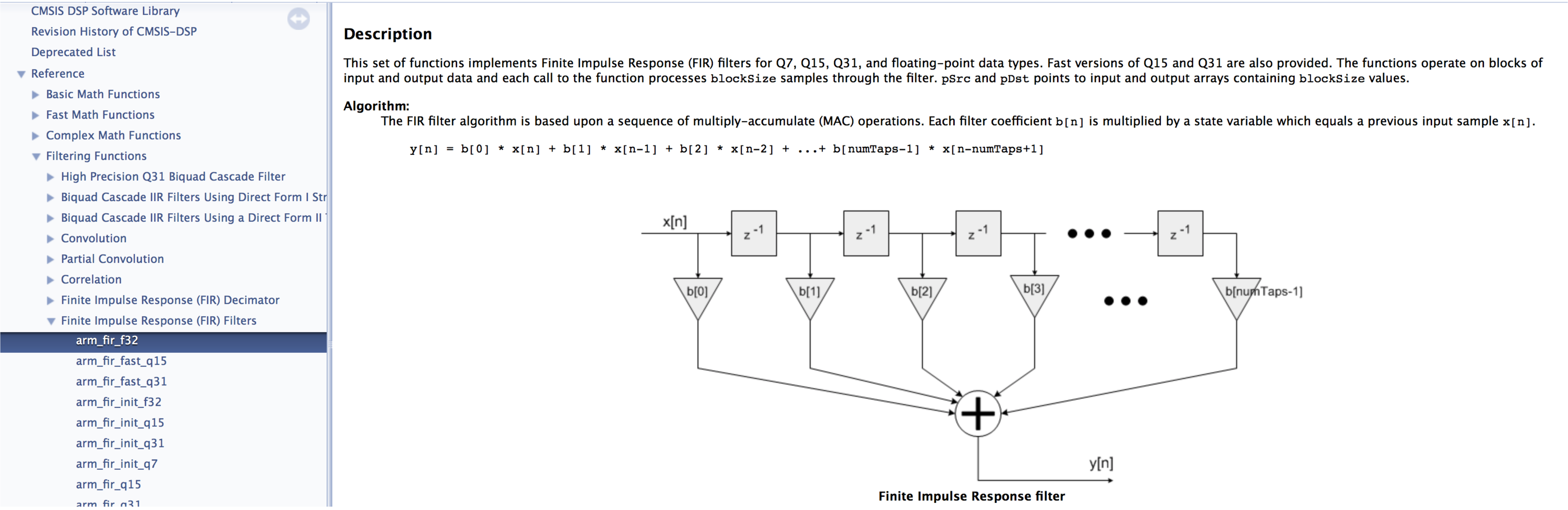
Microcontroller device

CMSIS-DSP

Collection of 61 algorithms, including Complex Maths & FIR Filter functions

- C source code, optimized for Cortex-M processors

Works with CMSIS-Compliant C Compilers: Arm Compiler, IAR, and GCC



Description

This set of functions implements Finite Impulse Response (FIR) filters for Q7, Q15, Q31, and floating-point data types. Fast versions of Q15 and Q31 are also provided. The functions operate on blocks of input and output data and each call to the function processes `blockSize` samples through the filter. `pSrc` and `pDst` points to input and output arrays containing `blockSize` values.

Algorithm:

The FIR filter algorithm is based upon a sequence of multiply-accumulate (MAC) operations. Each filter coefficient `b[n]` is multiplied by a state variable which equals a previous input sample `x[n]`.

```
    y[n] = b[0] * x[n] + b[1] * x[n-1] + b[2] * x[n-2] + ...+ b[numTaps-1] * x[n-numTaps+1]
```

Finite Impulse Response filter

References

Armv8-M Architecture Reference Manual

DSP for Cortex-M : https://developer.arm.com/architectures/instruction-sets/dsp-extensions/dsp-for-cortex-m

CMSIS-DSP (Free DSP library for Cortex-M processors)
https://github.com/ARM-software/CMSIS_5

Thank You

Danke

Gracias

Grazie

谢谢

ありがとう

Asante

Merci

감사합니다

धन्यवाद

Kiitos

شكرًا

ধন্যবাদ

תודה