sf20

16-bit microprocessors

Quick Reference Guide

V1.0
December 2014

Author: Martin Raubuch

info@racors.com

Introduction

The sf20 is a 16-bit microprocessor architecture for embedded control & computing applications with limited code size requirements. Main focus of the ISA (Instruction Set Architecture) definition is on high clock rates and small core implementations. Two ISA versions are available:
sf20b: base (b) ISA for general purpose control & computing
sf20d: dsp (d) extension: base ISA with extensions for DSP applications

The sf20 is a load/store architecture. All operands of computation instructions are either constants or contained in registers. Load/store instructions are used to transfer operands between registers and memory. The sf20d ISA slightly deviates from this concept. Some of the additional instructions have one memory source operand for improved DSP performance.

The sf20 base ISA defines a generic and complete instruction set for efficient high level language compiler implementations.

sf20b (base ISA) features

- Harvard architecture with separate instruction and data interfaces
- 64k x 20-bit instruction address space
- 64kBytes data address space
- Fixed length 20-bit instruction coding
- 16 interrupts with programmable start addresses
- 16 x 16-bit general purpose registers and 8 special registers
- Native support for 8-bit and 16-bit signed and unsigned integer data types
- Higher precision integer and float data types supported by multi-instruction sequences
- Rich set of load/store addressing modes, including indirect with index and update addressing
- Little endian byte ordering
- Load/store multiple instructions for code efficient copying and function prologue/epilogue
- Bit manipulation & test instructions: set, clear, toggle & test
- 16*16 multiply with either 16-bit high word or 16-bit low word results
- Flexible debug concept with application specific debug modules

sf20d (DSP extension ISA) additional features

- multiply high instructions with optional left-shift and with one source operand in memory
- multiply and accumulate instruction with optional left-shift and with one source operand in memory
- multiply and subtract instruction with optional left-shift and with one source operand in memory
- clip, clip with left-shift and clip to unsigned byte instructions
- Eight additional special registers
- Three indirect addressing modes for memory source operands, with offset and direct/indirect update
- Four address update registers for fast and code efficient navigation through coefficient/sample tables
- Four registers with accumulation extension for sum-of-products calculations with 32-bit precision

General Purpose Registers

Register Bits				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rn	An	Name																	
0		R0	16 x 16-bit general purpose registers	R0															
1		R1		R1															
2		R2		R2															
3		R3		R3															
4		R4		R4															
5		R5		R5															
6		R6		R6															
7		R7		R7															
8	0	R8		R8															
9	1	R9		R9															
10	2	RA		RA															
11	3	RB		RB															
12	4	RC		RC															
13	5	RD		RD															
14	6	RE		RE															
15	7	RF		RF															

Special Registers

Register Bits				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRn		Name	16 special registers																
0		CC	Condition Codes													N	Z	O	C
1		CS	Control & Status	IVPT													IS	IE	IR
2		LC	10-bit Loop Counter							LC									
4		U0	Signed 10-bit (address) Update 0							U0									
12		SA	Subroutine (return) Address	SA															
13		IA	Interrupt (return) Address	IA															
14		TA	Target Address (indirect jump)	TA															
15		ID	Core ID	REV				IMA				ISA				FML = 7			

Register Fields		
CC	C	Carry flag
	O	Overflow flag
	Z	Zero flag
	N	Negative flag
CS	IR	Interrupt, 0: not in an interrupt, 1: interrupt processing
	IE	Interrupt Enable, 0: interrupts disabled, 1: interrupts enabled
	IS	Interrupt (enable) Shadow, used to save/restore IE when scie/rsie instructions are executed
	IVTP	Interrupt Vectors Table Pointer, defines the 11 MSBs [15:5] of the interrupt vector table start address
ID	FML	Core Family, 1: eco16, 2: eco32, 4: sf32, 5: sf16, 6: sf18, 7: sf20,
	ISA	ISA = 1 = b (base), 2 = d (dsp)
	IMA	Implementation Architeture, 1: l = light, 4: u = ultralight
	REV	Revision, starting with 1

Load/Store instructions

Operand Symbols		Operand Addressing															
$DA11_S$	11-bit absolut data address, signed, 0x0000-0x03FF and 0xFC00-0xFFFF	$(DA11_S)$,Rd	$(DO8_S$,An),Rd	(Rx,An),Rd	(An)+,Rd	-(An),Rd	(An)*,Rd	(An)+,RGS	-(An),RGS	Rs,($DA11_S$)	Rs,($DO8_S$,An)	Rs,(Rx,An)	Rs,(An)+	Rs,-(An)	Rs,(An)*	RGS,(An)+	RGS,-(An)
$DO8_S$	8-bit data address offset, signed, -128-127, including zero																
Rs	register Rn (R0-RF) used as source operand																
Rd	register Rn (R0-RF), used as destination operand																
Rx	register Rn (R0-RF), used as index operand																
An	register An (R8-RF) used as indirect address																
RGS	register selection, may include R0-R7, R9-RD, SA																
Mnemo	Description	Addressing Modes															
ldbt	load byte (8-bit) and zero-extend to 16 bits	*	*	*	*	*	*	*	*								
stbt	store byte (8-bit)									*	*	*	*	*	*	*	*
ldsh	load short (16-bit)	*	*	*	*	*	*	*	*								
stsh	store short (16-bit)									*	*	*	*	*	*	*	*

eda (16-bit effective data address) generation (Load/Store Addressing Modes)		
$DA11_S$	11-bit direct address, signed	eda = $DA11_S$
($DO8_S$,An)	indirect with 8-bit signed offset	eda = An + $DO8_S$
(Rx,An)	indirect with scaled index	eda = An + size*Rx, size = 1 (byte), 2 (short)
(An)+	indirect with post-increment	eda = An, An += size, size = 1 (byte), 2 (short)
-(An)	indirect with pre-decrement	eda = An - size, size = 1 (byte), 2 (short), An = eda
(An)*	indirect with post-update	eda = An, An += U0 (special register)

For ldbt and stbt instructions register selections RGS can include registers R0-R7, RC and RD. For ldsh and stsh instructions register selections RGS can include registers SA (subroutine address), R2-R7 and R9-RB.

Flow Instructions

Operand Options			Op.-Addr.				
Implied	no operands or operands are implicitly defined		Implied	$IA16_U$	$IO14_S$	$IO10_S$	$IO10_S$,S
$IO10_S$	10-bit instruction address offset (20-bit word granularity), -512 to 511 instructions						
$IO14_S$	14-bit instruction address offset (20-bit word granularity), -8192 to 8191 instructions						
$IA16_U$	16-bit absolut instruction address in 20-bit word granularity, 0x0000 to 0xFFFF						
S	Speculation, False (S=0) or True (S=1)						
Mnemo	Description		Ad.Mode				
jump, jump-to-subroutine, return							
jump	jump		*				
jpsr	jump to subroutine		*	*			
rtsr	return from subroutine		*				
rtir	return from interrupt		*				
unconditional branch							
bral	branch always				*		
conditional & unconditional branches							
mnemo	condition CND specified as logical equation of C = CC.C, O = CC.O, Z = CC.Z and N = CC.N						
brnc	branch if no carry	CND = ~C					*
brcr	branch if carry	CND = C					*
brno	branch if no overflow	CND = ~O					*
brof	branch if overflow	CND = O					*
brnz	branch if non zero	CND = ~Z					*
brzr	branch if zero	CND = Z					*
brps	branch if positive	CND = ~N					*
brng	branch if negative	CND = N					*
brls	branch if lower or same	CND = C | Z					*
brhi	branch if higher	CND = ~C & ~Z					*
brlo	branch if lower	CND = (N & ~O) | (~N & O)					*
brge	branch if greater or equal	CND = (N & O) | (~N & ~O)					*
brle	branch if lower or equal	CND = Z | (N & ~O) | (~N & O)					*
brgt	branch if greater	CND = ~Z & ((N & O) | (~N & ~O))					*
brlc	branch if loop counter is unequal zero	LC -= 1, CND = LC != 0				*	
other							
stie	set interrupt enable, sets IE bit in CS		*				
clie	clear interrupt enable, clears IE bit in CS		*				
rsie	restore interrupt enable, transfers IS bit of CS to IE bit of CS		*				
scie	save and clear interrupt enable, transfers IE bit of CS to IS bit of CS, then clears IE		*				
stop	stop, enter stopped (debug) state		*				
svpc	save program counter (write to debug port)		*				
rspc	restore program counter (read from debug port)		*				

Arithmetic Computation Instructions

Operand Field Elements		Operand Addressing							
$C8_U$	8-bit constant (Unigned) 0 to 255	$C8_U$,Rb	$C16_U$,Rb	$C10_S$,Rs1	Rs0,Rs1	Rs	Rs,Rd	Rs0,Rs1,Rd	Cond. Code Updated
$C10_S$	10-bit constant (Signed) -512 to 511								
$C16_U$	16-bit constant (Unsigned), 8 LSBs are not coded and are all zeros 0x0000 to 0xFF00								
Rs,Rs0,Rs1	register Rn, used as source (Rs), source 0 (Rs0) or source 1 (Rs1) operand								
Rd,Rb	register Rn, used as destination (Rd) or as both source and destination (Rb) operand								
Mnemo	Description	Addressing Modes							
addt	add to	*						*	*
addc	add with carry, destination = source0 + source1 + CC.C							*	*
adcf	add carry flag, destination = source + CC.C						*		*
addh	add to high word		*						
subf	subtract from, source0 from source1	*						*	*
subc	subtract with carry, destination = source1 - source0 - CC.C							*	*
sbcf	subtract carry flag, destination = source - CC.C						*		*
comp	compare (subtract source 0 from source 1)			*	*				*
cmpc	compare with carry				*				*
cpcf	compare carry flag (subtract CC.C from source)					*			*
negt	negate						*		
absl	absolute value						*		
clzr	count leading zeros						*		
sxbt	sign extend byte						*		
sxsh	sign extend short						*		
mlcu	multiply constant unsigned, 8*16 -> 24, store lower 16-bit of result in destination	*							
mult	multiply unsigned, 16*16 -> 32, store low word of result in destination							*	
mlhu	multiply high unsigned, 16*16 -> 32, store high word of result in destination							*	
mlhs	multiply high signed, 16*16 -> 32, store high word of result in destination							*	

Miscellaneous Instructions

Operand Options		Operand Addressing													
$BTI4_U$, $SHC4_U$	4-bit bit index (Unsigned), 0 to 15, 4-bit shift count (Unsigned), 0 to 15	$BTI4_U$,Rs1	$BTI4_U$,Rs1,Rd	$SHC4_U$,Rs1,Rd	$C10_U$,SRd	$C10_S$,Rd	$C8_U$,RLb	Rs0,Rs1	Rs,Rd	Rs,SRd	SRs,Rd	Rs0,Rs1,Rd	Rd	Rs	Condition Code Update
$C8_U$	8-bit constant (Unsigned) 0 to 255														
$C10_U$	10-bit constant (Unsigned), 0 to 1023														
$C10_S$	10-bit constant (Signed), -512 to 511, used with `mtsr` $C10_S$,Un (n=0,1,2,3)														
Rs,Rs0,Rs1,Rd	register Rn, used as source (Rs), source 0 (Rs0), source 1 (Rs1) or destination (Rd) operand														
SRs, SRd	special register SRn, used as source (SRs) or destination (SRd) operand														
Mnemo	Description	Addressing Modes													
Logic															
andb	logical and bit wise, also calculates parity of the result						*					*			*
iorb	logical inclusive or bit wise						*					*			
xorb	logical exclusive or bit wise						*					*			
invt	invert								*						
Move															
move	move					*			*						
mvsr	move stack reference, adds constant $C10_S$ to R8 (stack pointer) and stores the result in Rd					*									
mfsr	move from special register										*				
mtsr	move to special register, with constant source and Un (U0-U3) destination the constant is signed				*					*					
mfdp	move from debug port, transfers the debug port input value to the destination operand												*		
mtdp	move to debug port, transfers the source operand to the core's debug port													*	
Shift															
shlz	shift left with zero fill			*								*			
shlf	shift left with feedback (from MSB)			*								*			
shru	shift right unsigned			*								*			
shrs	shift right signed			*								*			
Bit Manipulation															
btst	bit set		*									*			
btcl	bit clear		*									*			
bttg	bit toggle		*									*			
btts	bit test	*						*							*

General Purpose Registers

Register Bits					15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Rn	Ra	An	Name																	
0			R0	16 x 16-bit general purpose registers	R0															
1			R1		R1															
2			R2		R2															
3			R3		R3															
4	0		R4		R4															
5	1		R5		R5															
6	2		R6		R6															
7	3		R7		R7															
8		0	R8		R8															
9		1	R9		R9															
10		2	RA		RA															
11		3	RB		RB															
12		4	RC		RC															
13		5	RD		RD															
14		6	RE		RE															
15		7	RF		RF															

Special Registers

Register Bits				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRn		Name	16 special registers																
0		CC	Condition Codes													N	Z	O	C
1		CS	Control & Status	IVPT													IS	IE	IR
2		LC	10-bit Loop Counter							LC									
3		EP	Extension Parameters												S	RND			
4		U0	Signed, 10-bit (address) Update 0							U0									
5		U1	Signed, 10-bit (address) Update 1							U1									
6		U2	Signed, 10-bit (address) Update 2							U2									
7		U3	Signed, 10-bit (address) Update 3							U3									
8		AE0	Accumulation Extension 0	AE0															
9		AE1	Accumulation Extension 1	AE1															
10		AE2	Accumulation Extension 2	AE2															
11		AE3	Accumulation Extension 3	AE3															
12		SA	Subroutine (return) Address	SA															
13		IA	Interrupt (return) Address	IA															
14		TA	Target Address (indirect jump)	TA															
15		ID	Core ID	REV				IMA				ISA				FML = 7			

Register Fields		
CC	C	Carry flag
	O	Overflow flag
	Z	Zero flag
	N	Negative flag
CS	IVTP	Interrupt Vectors Table Pointer, defines the 11 MSBs [15:5] of the interrupt vector table start address
	IR	Interrupt, 0: not in an interrupt, 1: interrupt processing
	IE	Interrupt Enable, 0: interrupts disabled, 1: interrupts enabled
	IS	Interrupt (enable) Shadow, used to save/restore IE when scie/rsie instructions are executed
EP	RND	Round control, 0: no round, 1-10 adds (1 << (RND+13)) to 32-bit multiply products
	S	Shift, left-shift control for the msnsd, mshsd, msasd and msssd instr., S=0: 1-bit, S=1: 2-bits
ID	FML	Core Family, 1: eco16, 2: eco32, 4: sf32, 5: sf16, 6: sf18, 7: sf20,
	ISA	ISA = 1 = b (base), 2 = d (dsp)
	IMA	Implementation Architeture, 1: l = light, 4: u = ultralight
	REV	Revision, starting with 1

DSP ISA Extension Instructions

Operand Options		Op. Adr.					
$DO5_U$	5-bit unsigned & scaled data address offset, 0, 2, 4, …, 30	Rs0,Rs1,Rd	Rs,Rd	($DO5_U$,An),Rs1,Ra	(An,$AU5_S$)*,Rs1,Ra	$DA8_U$,Rs1,Ra	(An,Un)*,Rs1,Ra
$AU5_S$	5-bit signed & scaled data address update -16, -14, … -2, 2, 4, …, 14, 16						
$DA8_U$	8-bit unsigned & scaled direct address, 0x0000 - 0x00FE						
Rs,Rs0,Rs1,Rd	register Rn, used as source (Rs), source 0 (Rs0), source 1 (Rs1) or destination (Rd) operand						
Ra	accumulator register R4-R7 used as destination or both source and destination						
An	register An (R8-RF) used as indirect address						
Un	update register U0-U3, sign-extended and added to An after the operand read from memory						
Mnemo	Description	Addressing Modes					
addsd	left shift source operand 1 by one bit and add	*					
subsd	left shift source operand 1 by one bit and subtract	*					
clipd	clip signed to 0xC000 / 0x3FFF boundaries		*				
clshd	clip and shift, same as clip but with a 1-bit left shift after the clipping		*				
clubd	clip to unsigned byte, if < 0 clips to 0, if > 0xFF clips to 0xFF		*				
mlhsd	multiply signed, 16*16 -> 32, store high word of result in destination			*	*	*	*
mlnsd	multiply & negate signed, 16*16 -> 32, store high word of result in destination			*	*	*	*
macsd	multiply & accumulate signed, 16*16 -> 32, add high word of result to destination			*	*	*	*
massd	multiply & subtract signed, 16*16 -> 32, subtract high word of result from destination			*	*	*	*
mshsd	multiply & left-shift signed, 16*16 -> 32, store high word of result in destination			*	*	*	*
msnsd	multiply, left-shift & negate signed, 16*16 -> 32, store high word of result in destination			*	*	*	*
msasd	multiply, left-shift signed & accumulate, 16*16 -> 32, add high word of result to destination			*	*	*	*
msssd	multiply, left-shift signed & subtract, 16*16 -> 32, subtract high word of result from destination			*	*	*	*