TI MSP430: Hardware Multiplier

by Alex Milenkovich, milenkovic@computer.org

The MSP430 instruction set does not include instructions for multiplication of integers. The reasons for this design decisions are twofold. First, some typical applications of low-end embedded systems do not require multiply operations. On the other hand, implementing a multiplier in the datapath of the MSP430 processor would increase the hardware complexity (and hence the cost), as well as lengthen the latency on the data path, slowing down all operations. Therefore, the MSP430 designers opted for a compromise by implementing a hardware multiplier as an optional peripheral – some MSP430 devices include the hardware muliplier, while some other MSP430 devices do not. System designers thus can select an MSP430 device that suite their needs the best.

The hardware multiplier is physically outside the processor core and we interface it as any other peripheral. The multiplier consists of a number of registers that are visible in the address space and a circuit that carries out multiplication operations. These registers are loaded or read using regular CPU instructions. A block diagram of the hardware multiplier is shown in Figure 1. The shaded blocks are registers visible to software developers and they can be written to or read from (read/write - rw), or just read only (r).

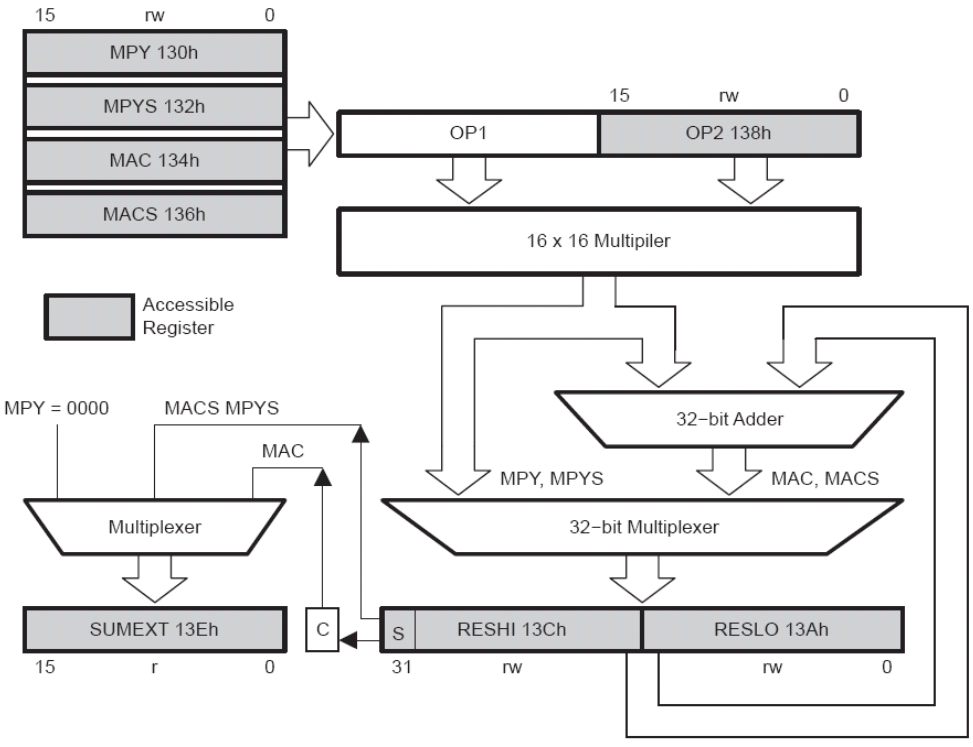

Figure 1. Hardware Multiplier Block Diagram.

The hardware multiplier supports several operations as follows:

- Unsigned multiply
- Signed multiply
- Unsigned multiply accumulate
- Signed multiply accumulate
- 16×16 bits, 16×8 bits, 8×16 bits, 8×8 bits.

The type of operation is selected by the address the first operand is written to. The hardware multiplier has two 16-bit operand registers, OP1 and OP2, and three result registers, RESLO, RESHI, and SUMEXT. RESLO stores the low word of the result, RESHI stores the high word of the result, and SUMEXT stores information about the result. The result is ready in three MCLK cycles and can be read with the next instruction after writing to OP2, except when using an indirect addressing mode to access the result. When using indirect addressing for the result, a NOP is required before the result is ready.

The operand one register OP1 has four addresses (Figure 2), used to select the multiply mode. Writing the first operand to the desired address selects the type of multiply operation but does not start any operation. Writing the second operand to the operand two register OP2 initiates the multiply operation. Writing OP2 starts the selected operation with the values stored in OP1 and OP2. The result is written into the three result registers RESLO,
RESHI, and SUMEXT.
Repeated multiply operations may be performed without reloading OP1 if the OP1 value is used for successive operations. It is not necessary to re-write the OP1 value to perform the operations.

OP1 Address	Register Name	Operation
0130h	MPY	Unsigned multiply
0132h	MPYS	Signed multiply
0134h	MAC	Unsigned multiply accumulate
0136h	MACS	Signed multiply accumulate

Figure 2. Hardware Multiplier OP1 Addresses.

The result low register RESLO holds the lower 16-bits of the calculation result. The result high register RESHI contents depend on the multiply operation and are listed in Figure 3.

Mode	RESHI Contents
MPY	Upper 16-bits of the result
MPYS	The MSB is the sign of the result. The remaining bits are the upper 15-bits of the result. Two's complement notation is used for the result.
MAC	Upper 16-bits of the result
MACS	Upper 16-bits of the result. Two's complement notation is used for the result.

Figure 3. RESHI Contents.

The sum extension registers SUMEXT contents depend on the multiply operation and are listed in Figure 4.

Mode	SUMEXT
MPY	SUMEXT is always 0000h
MPYS	SUMEXT contains the extended sign of the result 00000h Result was positive or zero 0FFFFh Result was negative
MAC	SUMEXT contains the carry of the result 0000h No carry for result 0001h Result has a carry
MACS	SUMEXT contains the extended sign of the result 00000h Result was positive or zero 0FFFFh Result was negative

Figure 4. SUMEXT Contents.

The multiplier does not automatically detect underflow or overflow in the MACS mode. The accumulator range for positive numbers is 0 to 7FFF FFFFh and for negative numbers is 0FFFF FFFFh to 8000 0000h. An underflow occurs when the sum of two negative numbers yields a result that is in the range for a positive number. An overflow occurs when the sum of two positive numbers yields a result that is in the range for a negative number. In both of these cases, the SUMEXT register contains the sign of the result, 0FFFFh for overflow and 0000h for underflow. User software must detect and handle these conditions appropriately.

1.1. Software Examples

Let us consider several examples, starting with 16x16 bit unsigned multiply. First, we load the first operand. Because we want the unsigned multiply operation we load the register MPY, using MOV #0x1234, &MPY (we use absolute addressing mode for the MPY register). Next, we load the second operand using MOV #0x5678, &OP2; this will also trigger the hardware multiplier to start multiplying the operands in OP1 and OP2. Finally, the last step is to read the results -- we can simply read the content of registers RESLO, RESHI, and SXT.
Note: All 8x8 modes use the absolute address for the registers because the assembler will not allow .B access to word registers when using the labels from the standard definitions file.

```
; 16x16 Unsigned Multiply
   MOV   #01234h,&MPY ; Load first operand
   MOV   #05678h,&OP2 ; Load second operand
;  ...                ; Process results

; 8x8 Unsigned Multiply. Absolute addressing.
   MOV.B #012h,&0130h ; Load first operand
   MOV.B #034h,&0138h ; Load 2nd operand
;  ...                ; Process results

; 16x16 Signed Multiply
   MOV   #01234h,&MPYS ; Load first operand
   MOV   #05678h,&OP2 ; Load 2nd operand
;  ...                ; Process results

; 8x8 Signed Multiply. Absolute addressing.
   MOV.B #012h,&0132h ; Load first operand
   SXT   &MPYS        ; Sign extend first operand
   MOV.B #034h,&0138h ; Load 2nd operand
   SXT   &OP2         ; Sign extend 2nd operand
                      ; (triggers 2nd multiplication)
;  ...                ; Process results


; 16x16 Unsigned Multiply Accumulate
   MOV   #01234h,&MAC ; Load first operand
   MOV   #05678h,&OP2 ; Load 2nd operand
;  ...                ; Process results

; 8x8 Unsigned Multiply Accumulate. Absolute addressing
   MOV.B #012h,&0134h ; Load first operand
   MOV.B #034h,&0138h ; Load 2nd operand
;  ...                ; Process results

; 16x16 Signed Multiply Accumulate
   MOV   #01234h,&MACS ; Load first operand
   MOV   #05678h,&OP2 ; Load 2nd operand
;  ...                ; Process results

; 8x8 Signed Multiply Accumulate. Absolute addressing
   MOV.B #012h,&0136h ; Load first operand
   SXT   &MACS        ; Sign extend first operand
   MOV.B #034h,R5     ; Temp. location for 2nd operand
   SXT   R5           ; Sign extend 2nd operand
   MOV   R5,&OP2      ; Load 2nd operand
;  ...                ; Process results


; Access multiplier results with indirect addressing
   MOV   #RESLO,R5    ; RESLO address in R5 for indirect
   MOV   &OPER1,&MPY  ; Load 1st operand
   MOV   &OPER2,&OP2  ; Load 2nd operand
   NOP                ; Need one cycle
   MOV   @R5+,&xxx    ; Move RESLO
   MOV   @R5,&xxx     ; Move RESHI
```

1.2. Interrupts and Multiplier

If an interrupt occurs after writing OP1, but before writing OP2, and the multiplier is used in servicing that interrupt, the original multiplier mode selection is lost and the results are unpredictable. To avoid this, disable interrupts before using the hardware multiplier or do not use the multiplier in interrupt service routines.

```
; Disable interrupts before using the hardware multiplier
   DINT            ; Disable interrupts
   NOP             ; Required for DINT
   MOV   #xxh,&MPY ; Load 1st operand
   MOV   #xxh,&OP2 ; Load 2nd operand
   EINT            ; Interrupts may be enable before
                   ; Process results
```