THE IDT R4640 "MAD" INSTRUCTION YIELDS SIGNIFICANT PERFORMANCE IMPROVEMENT OF OVER 30% VERSUS USING "MULT"

APPLICATION NOTE AN-172

by Sunil Kakkar

The IDT R4640 is a true 64 bit high performance, low cost processor. Operating at 180MHz, it achieves more than 235 dhrystone 2.1 MIPS performance and 90 million multiply/accumulate (MAC) operations per second through the MAD instruction. This MAC performance is a big plus for embedded DSP applications and achieves improvements of over 30% over applications which only use the multiply instruction.

The R4640 has a MAD (Multiply and Accumulate) instruction which uses HI-LO as a 64 bit accumulator. The MAD instruction is defined as :

MAD rs, rt

and it has the following effect :

HI,LO <-- HI,LO + rs*rt

For 16 bit operands, the latency and the repeat rates for MAD are 3 cycles and 2 cycles respectively. For 32 bit operands, they are 4 cycles and 3 cycles respectively.

The MULT instruction is defined as :

MULT rs, rt

and it has the following effect :

HI,LO <-- rs * rt

MULT has the same latency and repeat rates as MAD, but we cannot take advantage of accumulating the result, a function very commonly used in DSP programs. So, for accumulation, the HI-LO registers will have to be saved in general registers and later on added together. This will lead to additional "MFHI", "MFLO" and "ADD" instructions thereby causing a major performance hit. Back to Back "MULT"s are hence not possible when accumulation of results is required. This leads to an additional performance hit because the lower repeat rates of the "MULT" instruction cannot be exploited in these cases.

Let us look at a simple example of calculating the following series in a DSP C program:

```
        /**
         series = 1 + (2^i  * 3^i) + xy + x^2 + x^3 + xyz + xz +
yz
        **/

    sum=1;

    for (i = 0; i < 2; i++)  {

        sum += power(i,2) * power(i,3);
        sum += x*y;
        sum += (x*x);
        sum += x*x*x;
        sum += x*y*z;
        sum += x*z;
        sum += y*z;
    }
```

The function "power (int j,int l)" returns "l to the power of j"
.

Now, let us look at the assembly language code for the portion of the program used to calculate the above series, without assuming the availability of the MAD instruction:

```
    ##################################################
        #
        # Assembly Language Code for the series
        # 1 + (2^i * 3^i) + xy + x^2 + x^3 + xyz + xz + yz
        #
        # DOES NOT USE "MAD"s. USES "MULT"s
        #
        # Atleast 30% Performance Hit by using "MULT"
        # instead of "MAD"
        #
    ##################################################

    $L47:
        jal     power
        li      $5,0x00000002

        move    $16,$2
        move    $4,$18
        jal     power
        li      $5,0x00000003
```

```
##############################################
  #
  #     Total Number of Instructions
  #     in Critical Code =          23
  #
  #     Total Number of Cycles
  #     in Critical Code =          37
  #
##############################################


mult    $16,$2          # 3 cycles
mflo    $8              # 1 cycle
lw      $5,16($sp)      # 1 cycle
mult    $5,$5           # 3 cycles
mflo    $4              # 1 cycle
lw      $2,20($sp)      # 1 cycle
mult    $5,$2           # 3 cycles
mflo    $3              # 1 cycle
mult    $4,$5           # 3 cycles
mflo    $10             # 1 cycle
mult    $3,$19          # 3 cycles
mflo    $6              # 1 cycle
mult    $5,$19          # 3 cycles
mflo    $7              # 1 cycle
mult    $2,$19          # 3 cycles
addu    $18,$18,1       # 1 cycle
addu    $17,$17,$8      # 1 cycle
addu    $17,$17,$3      # 1 cycle
addu    $17,$17,$4      # 1 cycle
addu    $17,$17,$10     # 1 cycle
addu    $17,$17,$6      # 1 cycle
addu    $17,$17,$7      # 1 cycle
mflo    $2              # 1 cycle


addu    $17,$17,$2
slt     $2,$18,2
bne     $2,$0,$L47
nop
```

As you can see, each MULT takes 3 cycles. In addition, each MULT has to be followed by a MFLO to save the LO register and later on ADDU instructions are required to add these values that are saved from the LO register. For 64 bit results, MFHI would also be required.

Now let us look at the assembly language version of the code used to calculate the above series by using the MAD instruction.

```
##############################################
  #
  # Assembly Language Code for the series
  # 1 + (2^i * 3^i) + xy + x^2 + x^3 + xyz + xz + yz
  #
  # USES SINGLE AND BACK TO BACK "MAD"s
  #
   # Atleast 30% performance Improvement by using
"MAD"s
  #
  #
 ##############################################

$L61:
    jal    power
    li     $5,0x00000002

    move   $16,$2
    move   $4,$18
    jal    power
    li     $5,0x00000003

 ##############################################
  #
  #     Total Number of Instructions
  #     in Critical Code =          15
  #
  #     Total Number of Cycles
  #     in Critical Code =          26
  #
 ##############################################
```

```
mtlo    $17             # 1 cycle
mad     $16,$2          # 3 cycles
mflo    $17             # 1 cycles
lw      $3,16($sp)      # 1 cycle
lw      $5,20($sp)      # 1 cycles
mul     $4,$3,$5        # 3 cycles
addu    $17,$17,$4      # 1 cycle
mul     $2,$3,$3        # 3 cycles
addu    $17,$17,$2      # 1 cycle
mtlo    $17             # 1 cycle
mad     $2,$3           # 2 cycles   Repeat Rate
mad     $4,$19          # 2 cycles
mad     $19,$3          # 2 cycles
mad     $19,$5          # 3 cycles
mflo    $17             # 1 cycle

   # 90 MILLION MADS PER SECOND AT 180 MHz
   # Latency for one MAD is 3 cycles
   # Repeat rate for MAD is only 2 cycles

addu    $18,$18,1
slt     $2,$18,2
bnel    $2,$0,$L61
move    $4,$18
```

As you can see, the most critical loop uses only 26 cycles versus 37 cycles without MADs which is a 30% improvement. Further, the same number of cycles would be required in the second case using MADs for 64 bit results also which would yield a performance improvement of close to 40% versus 64 bit results for MULTs.

Conclusion

Hence, the MAD instruction in R4640 can be utilized to get a distinct performance advantage in DSP programs. This performance advantage could easily reach 40% versus processors which do not support a Multiply/Accumulate instruction.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers skilled in the art designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only for development of an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising out of your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Rev.1.0 Mar 2020)

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:
www.renesas.com/contact/

Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.