



# **BLOCK MEMORY MULTIPLIER**

## **DESCRIPTION**

The B\_MUL.VHD ipcore described here is a multiplier for signed or unsigned integer operands implemented in Block memory of an FPGA or ASIC.

## **FEATURES**

- Used in a host of DSP applications such as Digital filters, DFTs and video processing
- Signed or unsigned operands
- Configured for one input constant (Constant multiplier) or both inputs variable
- Internal dual port RAM for Constant multiplier mode.
- CPU interface with configurable Address and Data bus width for Constant multiplier mode.
- Supports Symetric(inputs are of equal size) or Asymetric(inputs of unequal size) operands
- Supports equal or unequal operand slicing
- Instantiate as Parallel or Time Domain Multiplexed
- Adjustable iteration time for internal combintional logic

## **VHDL Component Declaration:**

```
COMPONENT B_MUL
  GENERIC ( WCND      : INTEGER ;
           WPLR      : INTEGER ;
           WRES      : INTEGER ;
           EXTR      : INTEGER ;
           NRG       : INTEGER ;
           LRG       : INTEGER ;
           SIG       : INTEGER ;
           NWD       : INTEGER ;
           LWD       : INTEGER ;
           CLTI      : INTEGER ;
           AWD       : INTEGER ;
           DWD       : INTEGER ;
           CPLR      : INTEGER ;
           PLSI      : INTEGER:=1 ;
           WDL       : INTEGER:=1 ;
           REPC      : INTEGER ;
           CLTA      : INTEGER:=0 ;
           CLTM      : INTEGER:=1 ;
           CLTS      : INTEGER:=1 ;
           DVC       : INTEGER:=2 ;
           FNAM      : STRING:="NONE" ) ;
  PORT ( MCNX      : IN   BUS1D(WCND-1 DOWNT0 0) ;
        MPLX      : IN   BUS1D(WPLR-1 DOWNT0 0) ;
        BEG       : IN   NODE:= '0' ;
        CLKI      : IN   NODE:= '1' ;
        RST       : IN   NODE:= '1' ;
        COM       : IN   BUS1D(0 DOWNT0 0):=(OTHERS=>'0') ;
        ECN       : IN   BUS1D(0 DOWNT0 0):=(OTHERS=>'0') ;
        ECL       : IN   BUS1D(0 DOWNT0 0):=(OTHERS=>'0') ;
        ICS       : IN   NODE:= '0' ;
        MAB       : IN   BUS1D(AWD-1 DOWNT0 0) ;
        MDB       : IN   BUS1D(DWD-1 DOWNT0 0) ;
        RSLT      : BUFFER BUS1D(WRES-1 DOWNT0 0) ;
        RSLD      : BUFFER BUS1D(WRES-1 DOWNT0 0) ;
        DNL       : BUFFER BUS1D(WDL-1 DOWNT0 0) ;
        IOM       : BUFFER BUS1D(14 DOWNT0 0) ;
END COMPONENT ;
```



## FILES YOU GET

```

i)FUNC.DOC      - Documentation of functions & data types used in the core.
ii)README.DOC  - Compile and licensing information.
iii)BMUL.DOC   - This document

a)MYLIB.VHD    - PACKAGE
b)B_MUL.VHD    - TOP HIERARCHY DESIGN FILE
c)P_LSE.VHD    - DESIGN FILE BELOW TOP HIERARCHY
d)S_DFF.VHD    - -do-
e)F_DIV.VHD    - -do-
f)U_DCNT.VHD   - -do-
g)M_DFF.VHD    - -do-
h)I_NCDEC.VHD  - -do-
i)A_DSB.VHD    - -do-
j)S_JKF.VHD    - -do-
k)S_TFF.VHD    - -do-
l)P_AD.VHD     - -do-
m)U_PCNT.VHD   - -do-
n)B_SHIFT.VHD  - -do-
o)R_STK.VHD    - -do-
p)D_ECOD.VHD   - -do-
q)C_SHIFT.VHD  - -do-
r)M_YMUX.VHD   - -do-
s)I_P2IP.VHD   - -do-
s)S_TATE.VHD   - -do-
s)M_STK.VHD    - -do-

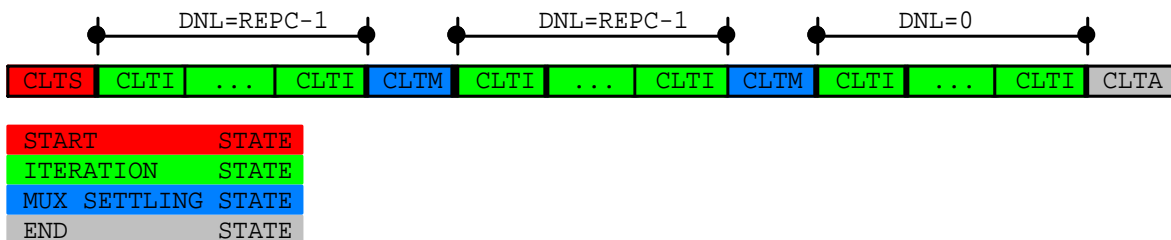
```

## COMPILE INSTRUCTIONS

Create a "metou" folder in your <D:> and save the Partial Product Initialisation files, given at the end of this document, into it. Create the sample design "mytop.vhd", as given below. Replace the Generic statement with Parameter values given in the "Timing Diagram" section of this document. Compile and simulate the design to reproduce the same timing diagrams. Modify the input operands as you please to observe different results.

## OPERATION

The state machine starts at the end of the 1st CLKI pulse of the BEG input and consists of REPC loops of a multiplier. A loop starts with a START state of CLTS clks(1st loop) or MUX SETTTLING state of CLTM clks (2nd loop and onwards) allowing the operands for the multiplier to settle. This is followed by one or more iteration states of CLTI clks. A loop counter increments at the end of the last iteration state and indicates the end of a loop and the beginning of a new one. The last loop is followed by an END state of CLTA clks and then the END of LOOP pulse. The START, MUX-SETTLING and END states are inserted only if their duration is more than 0.





## COMPUTATION CLOCKS (CCL)

```

IF REM(WCND/NWD)>0                --Multiplicand Nibbles
  NNN=INT(WCND/NWD)+1
ELSE
  NNN= INT(WCND/NWD)
ENDI

IF CPLR=0                          --Multiplier Nibbles
  IF REM(WPLR/LWD)>0
    NLN=INT(WPLR/LWD)+1
  ELSE
    NLN= INT(WPLR/LWD)
  ENDI
ELSE
  NLN=0
ENDI

NITR=NNN*NLN                       --Number of Iterations
CCL=NITR*CLTI                    --Computation clocks

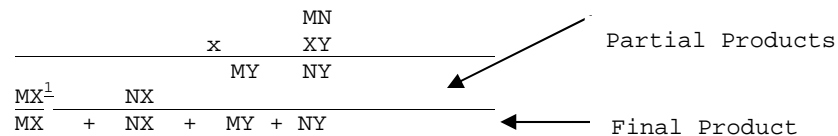
```

### WHERE

CLTI=Clocks for one iteration

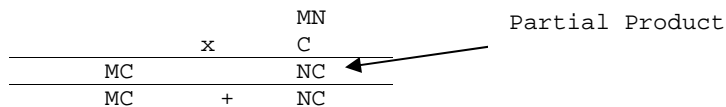
## VARIABLE MULTIPLIER OPERATION

Each letter in the operands represents a nibble of width NWC(multiplicand) and LWC(multiplier). The leftmost letter represents the MSB nibble and the rightmost the LSB. Partial products are formed by multiplying each nibble of the Multiplier with that of the Multiplicand. Partial products thus formed are shifted to the left, to account for their relative significance and then added, as in standard decimal multiplication. Multiplication logic is not synthesized, but precalculated and resident in the LUT.



## CONSTANT MULTIPLIER OPERATION

Each letter in the multiplicand represents an NWC bit wide nibble. M represents the MSB nibble and N represents the LSB. 'C' is the constant multiplier. Partial products are formed by multiplying 'C' w/ each nibble of the multiplicand. Partial products thus formed are shifted to the left, to account for their relative significance and then added, as in standard decimal multiplication. Multiplication logic is not synthesized, but precalculated and resident in the LUT.



## COMBINATIONAL vs SEQUENTIAL DESIGN

When both operands have only one nibble each a fully parallel component is instantiated, using only Gates and Memory, otherwise a Time Domain Multiplexed component is instantiated. For parallel instantiation NWD must be equal to WCND and NLD to WPLR.

```

MPP=0                               --MPP=0 Indicates, design is combinational
IF (NNN-1)>0 OR (NLN-1)>0
  MPP=1                               --MPP=1 Indicates, design is part combinational, part
sequential
ENDI

```



## THE RESULT REGISTER

When MPP=0 a register for the result is not required and the result follows the input after a propagation delay. If the inputs are required to be registered(LRG+NRG>0) they will be visible to the internal logic only after the first CLKI pulse of the BEG input and thus so will the result. In this situation the RSLD[] and RSLT[] are identical.

When MPP=1 a register for the result is required and will store the final result (RSLD[]) with the IOM(8) signal, where it will remain steady until one CLKI after the rising edge of the next BEG input.

## MEMORY BITS USED-(MBITS)

```
IF (WCND-NWD)>0
    NWC=NWD+SIG
ELSE
    NWC=NWD
ENDI
IF (WPLR-LWD)>0
    LWC=LWD+SIG
ELSE
    LWC=LWD
ENDI
```

**MBITS=[NWC+LWC(1-CPLR)]\*[ NWC+LWC(1-CPLR)-SIG+(WPLR\*CPLR)]**

## INPUT PORTS

NAME	DESC	WIDTH	COMMENTS
MCNX	Multiplicand	WCND	Signed/Unsigned integer
MPLX	Multiplier	WPLR	Signed/Unsigned integer . Not used when CPLR parameter=1
BEG	Operand load	1	Active hi, 1CLKI wide (or more), loads operands in START state (CLTS clks)
CLKI	clock	1	Synchronizes all internal operations
RST	reset	1	Resets all internal registers
COM	Control	16	For external control. IOM o/p from an external IP2IP component. See IP2IP.DOC
ECN	Control	WDN	For external control. See IP2IP.DOC
ECL	Control	WDL	For external control. See IP2IP.DOC
ICS	MEM select	1	Chip select to write constant partial products in internal dual port ram (used if CPLR=1)
MAB	MEM address	AWD	Address bus for internal dual port ram (unused when CPLR=0)
MDB	MEM data	DWD	Data bus for internal dual port ram (unused when CPLR=0)

## PARAMETERS(All integer type, except FNAM)

NAME	DESCRIPTION
WCND	Width of multiplicand, input port. WCND>=1
WPLR	Width of multiplier, input port. WPLR>=1
WRES	Width of result, output port. WRES>=1
EXTR	Make 1 for external control by an IP2IP component.
NRG	Multiplicand(NRG)/Multiplier(LRG) operand to be registered -Yes/No-1/0. If 'No', must be steady for CCL CLKI pulses after start of computation as shown below. If 'YES' must be stable in the 1 <sup>st</sup> CLKI of the BEG i/p
LRG	
SIG	Inputs and outputs are in signed 2's compliment format (1) or unsigned (0)
NWD	Multiplicand nibble size. NWD>=1
LWD	Multiplier nibble size. Unused when CPLR=1 and may be set 0
AWD	Width of address bus for access to internal dual port ram. AWD>=1
DWD	Width of data bus for internal dual port ram. DWD>=1
CPLR	Multiplier is constant Yes/No-1/0
PLSI,WDL,REPC,CLTI,CLTA,CLTM,CLTS	
Replace phrase "Sequential Process" with "Multiplier" and read parameters with same name in "IP2IP.DOC"	
DVC	Device Family - CYCLONE(0),FLEX10K(1),ACEX(2)
FNAM	Path of Partial product LUT(used when CPLR=1)



## OUTPUT PORTS

NAME	DESC	WIDTH	COMMENTS
RSLT	Result (Q node)	WRES	'Q' node of Result register. Registered internally. Remains steady after IOM(8). <u>When SIG=0</u> WRES is normally = WPLR + WCND. When WRES < WPLR + WCND, result is truncated When WRES > WPLR + WCND, result is padded with 0's. <u>When SIG=1</u> WRES is normally = WPLR + WCND -1. When WRES < WPLR + WCND -1, result is truncated When WRES > WPLR + WCND -1, result is sign extended to WRES bits.
RSLD	Result (D' node)	WRES	Same as 'RSLT' o/p node, except it is the 'D' input of the Result register. May be registered externally with the IOM(8) output. Holds valid data only in the last iteration upto the end of IOM(8).
DNL	Loop Count	WDL	Replace phrase "Sequential Process" with "Multiplier" and read o/p ports with same name in "IP2IP.DOC"
IOM()	Control	1	

## SAMPLE DESIGN

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
LIBRARY MYLIB;
USE MYLIB.MYLIB.ALL;
```

```
ENTITY MYTOP IS
    PORT (CLKI      : IN  NODE;
          RST       : IN  NODE;
          MCNX      : IN  BUS1D(8 DOWNTO 0);
          MPLX      : IN  BUS1D(2 DOWNTO 0);
          BEG       : IN  NODE;
          RSLT      : BUFFER BUS1D(11 DOWNTO 0);
          DNA       : BUFFER NODE
    );
```

```
END MYTOP;
```

```
ARCHITECTURE MYTOP OF MYTOP IS
BEGIN
```

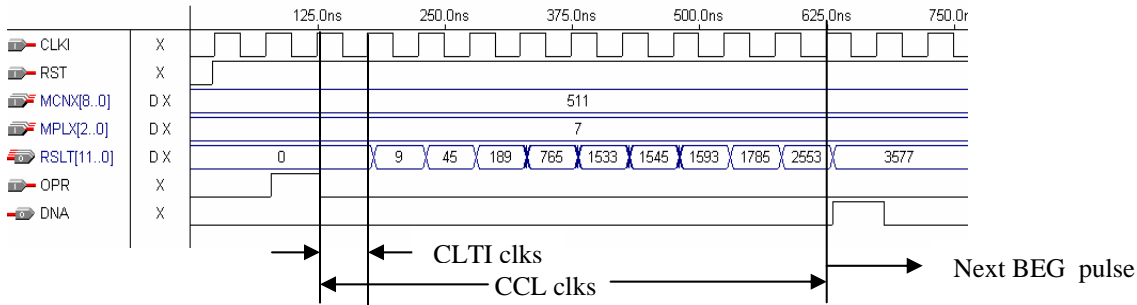
```
A1: B_MUL GENERIC MAP (WCND=>9, WPLR=>3, WRES=>12, NRG=>1, LRG=>1, SIG=>0,
    NWD=>2, LWD=>2, CLTI=>1, WD=>1, DWD=>1, CPLR=>0, PLSI=>1,
    DVC=>2, FNAM=>"D:\MAX2WORK\MYLIB\PKGM\BMULU.MIF")
```

```
    PORT
MAP (MCNX, MPLX, BEG, CLKI, RST, OPEN, OPEN, OPEN, OPEN, OPEN, OPEN, RSLT, DNA);
END MYTOP;
```

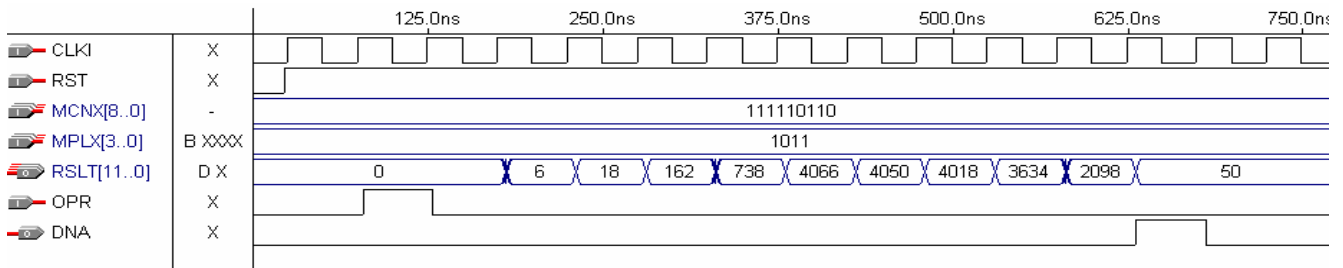


## SAMPLE DESIGN TIMING DIAGRAMS

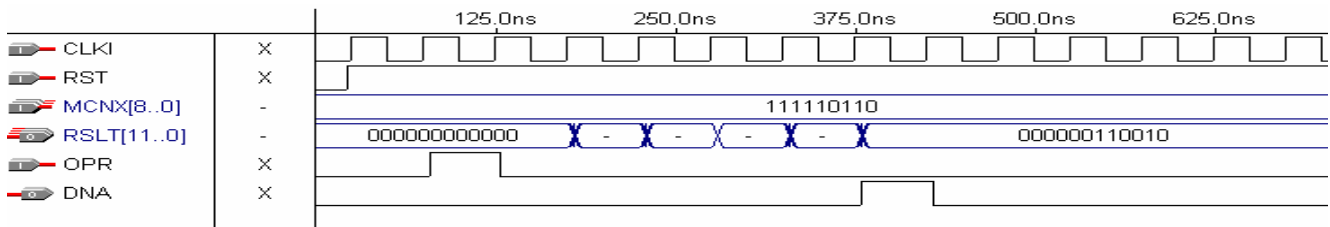
**PARAMETERS – WCND=9, WPLR=3, WRES=12, , EXTR=0, SIG=0, NWD=2, LWD=2, CLTI=1, AWD=1, DWD=1,CPLR=0, FNAM="D:\METOU\BMULU.MIF"**



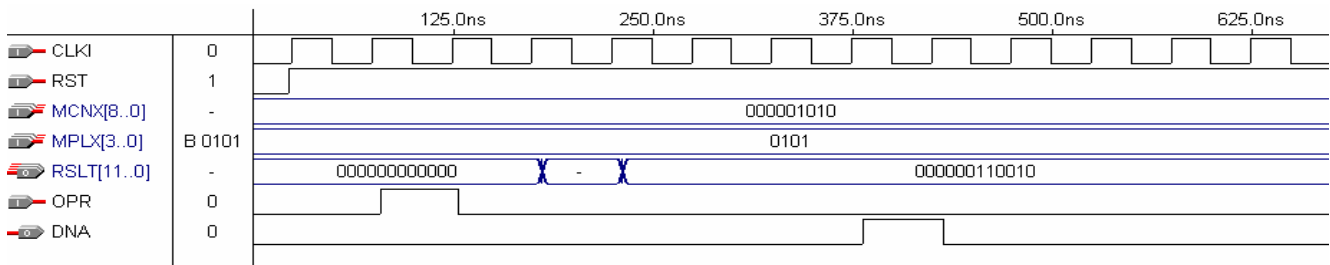
**PARAMETERS – WCND=9, WPLR=4, WRES=12, , EXTR=0, SIG=1, NWD=2, LWD=2, CLTI=1,, AWD=1, DWD=1,CPLR=0, FNAM="D:\METOU\BMULS.MIF"**



**TIMING DIAGRAM – WCND=9, WPLR=4, WRES=12, EXTR=0,SIG=1, NWD=2, LWD=0, CLTI=1,, AWD=19, DWD=16,CPLR=1, FNAM="D:\METOU\BMULCS.MIF"**



**TIMING DIAGRAM – WCND=9, WPLR=3, WRES=12, EXTR=0,SIG=1, NWD=2, LWD=0, CLTI=1,, AWD=19, DWD=16,CPLR=1, FNAM="D:\METOU\BMULCU.MIF"**





## PARTIAL PRODUCT LUT INITIALISATION FILES

### BMULU.MIF

```
WIDTH = 4;  
DEPTH = 16;  
ADDRESS_RADIX = DEC;  
DATA_RADIX = BIN;
```

#### CONTENT BEGIN

```
0 : 0000;  
1 : 0000;  
2 : 0000;  
3 : 0000;  
4 : 0000;  
5 : 0001;  
6 : 0010;  
7 : 0011;  
8 : 0000;  
9 : 0010;  
10 : 0100;  
11 : 0110;  
12 : 0000;  
13 : 0011;  
14 : 0110;  
15 : 1001;  
END;;
```

### BMULCS.MIF

```
WIDTH = 6;  
DEPTH = 8;  
ADDRESS_RADIX = DEC;  
DATA_RADIX = BIN;
```

#### CONTENT BEGIN

```
0 : 000000;  
1 : 111011;  
2 : 110110;  
3 : 110001;  
4 : 000000;  
5 : 001111;  
6 : 001010;  
7 : 000101;  
END;
```

### BMULCU.MIF

```
WIDTH = 5;  
DEPTH = 4;  
ADDRESS_RADIX = DEC;  
DATA_RADIX = BIN;
```

#### CONTENT BEGIN

```
0 : 00000;  
1 : 00101;  
2 : 01010;  
3 : 01111;  
END;
```

### BMULS.MIF

```
WIDTH = 5;  
DEPTH = 64;  
ADDRESS_RADIX = DEC;  
DATA_RADIX = BIN;
```

#### CONTENT BEGIN

```
0 : 00000;  
1 : 00000;  
2 : 00000;  
3 : 00000;  
4 : 00000;  
5 : 00000;  
6 : 00000;  
7 : 00000;  
8 : 00000;  
9 : 00001;  
10 : 00010;  
11 : 00011;  
12 : 00000;  
13 : 11101;  
14 : 11110;  
15 : 11111;  
16 : 00000;  
17 : 00010;  
18 : 00100;  
19 : 00110;  
20 : 00000;  
21 : 11010;  
22 : 11100;  
23 : 11110;  
24 : 00000;  
25 : 00011;  
26 : 00110;  
27 : 01001;  
28 : 00000;  
29 : 10111;  
30 : 11010;  
31 : 11101;  
32 : 00000;  
33 : 11100;  
34 : 11000;  
35 : 10100;  
36 : 00000;  
37 : 01100;  
38 : 01000;  
39 : 00100;  
40 : 00000;  
41 : 11101;  
42 : 11010;  
43 : 10111;  
44 : 00000;  
45 : 01001;  
46 : 00110;  
47 : 00011;  
48 : 00000;  
49 : 11110;  
50 : 11100;  
51 : 11010;  
52 : 00000;  
53 : 00110;  
54 : 00100;  
55 : 00010;  
56 : 00000;  
57 : 11111;  
58 : 11110;  
59 : 11101;  
60 : 00000;  
61 : 00011;  
62 : 00010;  
63 : 00001;  
END;
```