



COMPLEX MULTIPLIER

DESCRIPTION

The C_MUL.VHD ipcore described here is a multiplier for signed complex integer operands for implementation in an FPGA, ASIC or CPLD. The core returns a solution for an equation of the form:- $E+Fi=(A+Bi)*(C+Di)$. The complex multiplier is an important constituent of DSP components and systems, such as DFTs, DCTs, video/image processing, RF and land-line communications etc.

FEATURES

- Signed operands
- Internal multiplier can be configured for:-
 - a> Block memory multiplier
 - b> Parallel Array multiplier
 - c> Sequential multiplier
- The internal multiplier selected can be instantiated:-
 - a> Four in parallel with two adders, for parallel operation.
 - b> One multiplier and one adder, for sequential, Time Domain Multiplexed operation.
- Adjustable iteration time for internal combinatorial logic
- Scaled outputs

VHDL Component Declaration:

```
COMPONENT C_MUL
  GENERIC (
    WCND    : INTEGER;
    WPLR    : INTEGER;
    WPRD    : INTEGER;
    WSCL    : INTEGER;
    WRES    : INTEGER;
    SMLT    : INTEGER;
    BMLT    : INTEGER;
    NWD     : INTEGER;
    LWD     : INTEGER;
    CLTI    : INTEGER;
    SER     : INTEGER;
    EXTR    : INTEGER;
    CLTA    : INTEGER;
    NREG    : INTEGER;
    LREG    : INTEGER;
    PLSI    : INTEGER := 1;
    CLTM    : INTEGER;
    DVC     : INTEGER;
    CLTW    : INTEGER;
    FNAM    : STRING := "NONE"
  );
  PORT (
    MCNR    : IN  BUS1D(WCND-1 DOWNT0 0);
    MCNI    : IN  BUS1D(WCND-1 DOWNT0 0);
    MPLR    : IN  BUS1D(WPLR-1 DOWNT0 0);
    MPLI    : IN  BUS1D(WPLR-1 DOWNT0 0);
    CLKI    : IN  NODE := '1';
    RST     : IN  NODE := '1';
    COM     : IN  BUS1D(14 DOWNT0 0) := (OTHERS=>'0');
    ECN     : IN  BUS1D(WCND-1 DOWNT0 0) := (OTHERS=>'0');
    ECL     : IN  BUS1D(WPLR-1 DOWNT0 0) := (OTHERS=>'0');
    BEG     : IN  NODE := '0';
    OUTR    : BUFFER BUS1D(WRES-1 DOWNT0 0);
    OUTI    : BUFFER BUS1D(WRES-1 DOWNT0 0);
    ENBR    : BUFFER NODE;
    ENBI    : BUFFER NODE
  );
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) CMUL.DOC	-	This document
MYLIB.VHD	-	PACKAGE
C_MUL.VHD	-	TOP HIERARCHY DESIGN FILE
P_LSE.VHD	-	DESIGN FILE BELOW TOP HIERARCHY
A_MUL.VHD	-	-do-
B_MUL.VHD	-	-do-
S_MUL.VHD	-	-do-
S_DFF.VHD	-	-do-
F_DIV.VHD	-	-do-
U_DCNT.VHD	-	-do-
M_DFF.VHD	-	-do-
I_NCDEC.VHD	-	-do-
A_DSB.VHD	-	-do-
S_JKF.VHD	-	-do-
S_TFF.VHD	-	-do-
P_AD.VHD	-	-do-
U_PCNT.VHD	-	-do-
B_SHIFT.VHD	-	-do-
R_STK.VHD	-	-do-
D_ECOT.VHD	-	-do-
C_SHIFT.VHD	-	-do-
M_YMUX.VHD	-	-do-
I_P2IP.VHD	-	-do-
A_DSB.VHD	-	-do-
B_SHIFT.VHD	-	-do-
S_TATE.VHD	-	-do-

BLOCK MEMORY MULTIPLIER OPERATION

A multiplier implemented mostly in memory cells. A few gates and flip-flops may be required, depending on parameter settings. See BMUL.DOC for more information

ARRAY MULTIPLIER OPERATION

A multiplier implemented purely with combinational logic, no flip-flops and no memory are used.

SEQUENTIAL MULTIPLIER OPERATION

A multiplier implemented mostly with flip-flops, a few gates for control logic and no memory. See SMUL.DOC for more information

OUTPUT PORTS

NAME	DESC	WIDTH	COMMENTS
OUTR	Real Result	WRES	The OUTR output is registered internally in a buffer, enabled with the ENBR output signal and remains steady after the end of operation, indicated by the ENBI output See "SCALING and PADDING the RESULT"
OUTI	Imaginary/Real Result	WRES	When the parameter SER=1, this is a serial output. It brings out the Real result first and then the imaginary. The Real and Imaginary results may be registered externally with the ENBR and ENBI outputs respectively. The Imaginary result remains steady on this port, after the ENBI output When SER=0 this port exclusively contains the Imaginary result See " SCALING and PADDING the RESULT "
ENBR	Enable Real Buffer	1	Clock enable to register the Real result on the OUTI(SER=1)/OUTR(SER=0) port.
ENBI	Enable Imaginary Buffer	1	Clock enable to register Imaginary result on the OUTI port. Also indicates end of Operation.
CON	CMUL On	1	Hi from falling edge of BEG i/p to falling edge of ENBI o/p. indicates operation in progress.



INPUT PORTS

NAME	DESC	WIDTH	COMMENTS
MCNR	Real Multiplicand	WCND	Signed integer
MCNI	Imaginary Multiplicand	WCND	Signed integer
MPLR	Real Multiplier	WPLR	Signed integer
MPLI	Imaginary Multiplier	WPLR	Signed integer
BEG	Operand load	1	Active hi, 1CLKI wide (or more), loads operands in first CLKI pulse
CLKI	clock	1	Synchronizes all internal operations
RST	reset	1	Resets all internal registers
COM	-	16	Reserved, leave unused
ECN	-	WCND	Reserved, leave unused
ECL	-	WPLR	Reserved, leave unused

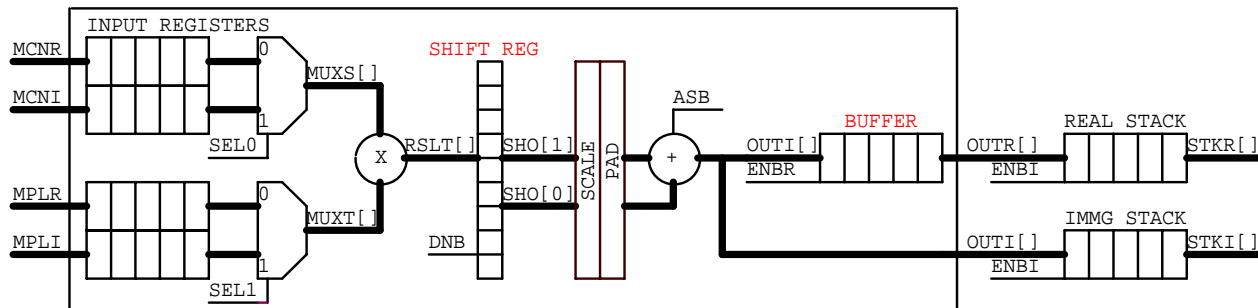
PARAMETERS

NAME	TYPE	MIN	DESCRIPTION
WCND	Integer	1	Width of multiplicand, input port
WPLR	Integer	1	Width of multiplier, input port
WPRD	Integer	1	Width of internal, intermediate products. See " SCALING and PADDING the RESULT "
WSCL	Integer	1	Width of scaled intermediate products. See " SCALING and PADDING the RESULT "
WRES	Integer	1	Width of result, output port See " SCALING and PADDING the RESULT "
SMLT	Integer	0	Sequential multiplier for the intermediate products. When BMLT+SMLT=0 Array is used.
BMLT	Integer	0	Block Memory multiplier for the intermediate products. When BMLT+SMLT=0 Array is used.
EXTR	Integer	0	Reserved, leave unused
NREG LREG	Integer	0	Multiplicand(NREG)/Multiplier(LREG) operand to be registered internally-Yes/No-1/0. If 'No', it must be steady after for CCL number of CLKI pulses after start of computation as shown below. If 'YES' operand must be stable in the 1 st CLKI of the BEG input
NWD	Integer	1	Multiplicand nibble size. Used only when BMLT=1, else ignored
LWD	Integer	1	Multiplier nibble size. Used only when BMLT=1, else ignored
SER	Integer	0	When SER=1, the internal intermediate products are calculated sequentially, otherwise in parallel.
CLTI	Integer	0	Clk pulses for one iteration of internal multiplier. When SML+SER>0, CLTI must be one or more else may be 0.
CLTA	Integer	0	Clk pulses for addition operations. When SML+SER>0, CLTA must be one or more else may be 0.
CLTW	Integer	0	Clk pulses for write operations. CLTW>=0
PLSI	Integer	0	Width of BEG input in terms of CLKI. BEG may be one CLKI wide(PLSI=1) or more(PLSI=0).
CLTM	Integer	0	Number of CLKI pulses for the "Mux Settling(MXS)" state. When SER=1 State used for operand load and/or mux settling for intermediate products after the first. For the 1 st intermediate product, operands are stable during computation (see LREG,NREG). If SML=0: Intermediate product computation doesn't require operands to be loaded. The MXS state if present (CLTM>0) is used for mux settling before start of computation. If CLTM=0, MXS state is not present. If SML=1: Intermediate product computation requires operands to be loaded. The MXS state must be present (CLTM>0) and is used for mux settling and operand loading before start of next intermediate product computation. When SER=0 Intermediate products are computed in parallel, the CLTM is ignored and MXS is not present.
FNAM	String	0	Path of Partial product LUT(unused when CPLR=1)
DVC	Integer	0	Device Family - CYCLONE(0),FLEX10K(1),ACEX(2)
DIFF	integer	0	Is the number of clocks by which the duration of the MXS state, of the 3rd intermediate multiplication is extended, thus extending the duration of the multiplication cycle. This gives the designer additional time to load the Real component of the result, appearing on the OUTI port. Unused when SER=0

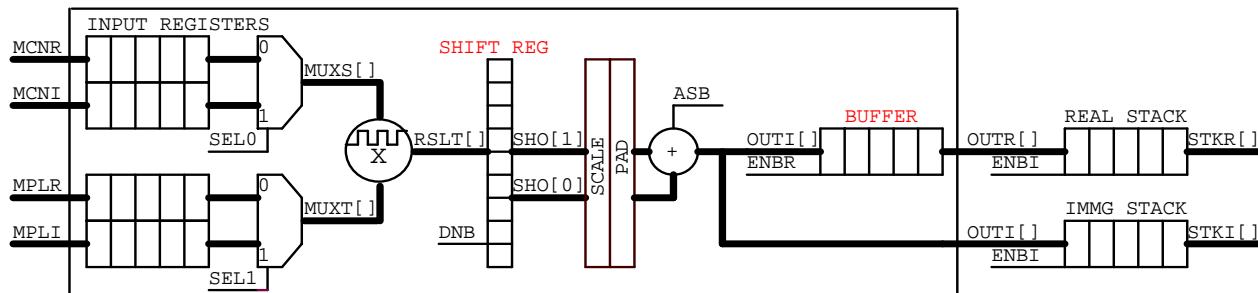


TIME DOMAIN MULTIPLEXED

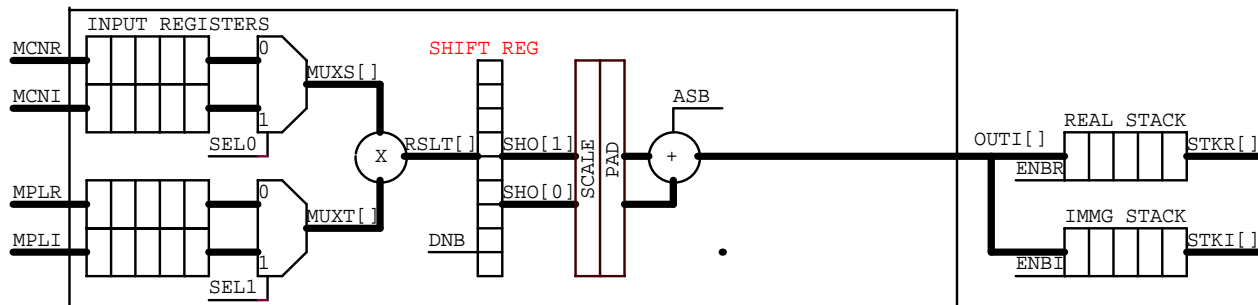
SEQUENTIAL CONFIGURATION, INTERNAL BUFFER USED, COMBINATIONAL MULTIPLIER



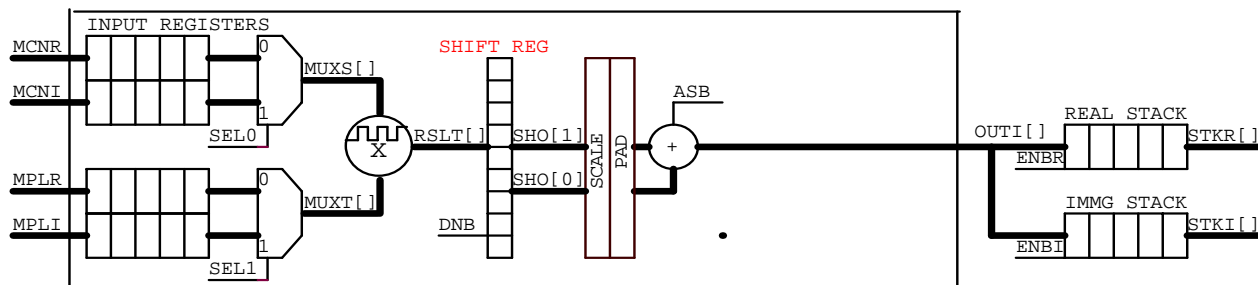
SEQUENTIAL CONFIGURATION, INTERNAL BUFFER USED, SEQUENTIAL MULTIPLIER



SEQUENTIAL CONFIGURATION, INTERNAL BUFFER BYPASSED, COMBINATIONAL MULTIPLIER

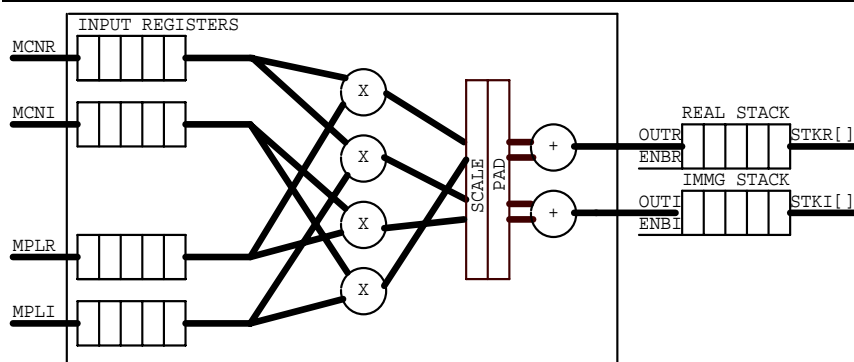


SEQUENTIAL CONFIGURATION, INTERNAL BUFFER BYPASSED, SEQUENTIAL MULTIPLIER

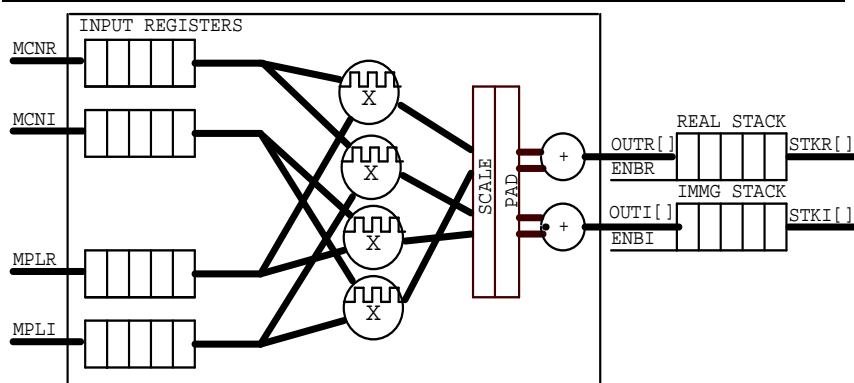




PARALLEL CONFIGURATION, COMBINATIONAL MULTIPLIERS



PARALLEL CONFIGURATION, SEQUENTIAL MULTIPLIERS



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.

SCALING AND PADDING THE RESULT

The max width of internal, intermediate products is given by $WNRM = WPLR + WCND - 1$. When $WPRD < WNRM$, MSB of the intermediate products are truncated else padded to $WPRD$ bits and are represented internally as $RSLT[WPRD-1..0]$. When $WSCL < WPRD$, the internal products ($RSLT[]$) are divided by $2^{WPRD-WSCL}$ and are then represented as $IPRD[WSCL-1..0]$, these are then padded or truncated to $WRES$ bits and added to form the output, $OUTR[WRES-1..0]$ and $OUTI[WRES-1..0]$.

MEMORY UTILISATION

When $BMLT=1$ see *BMUL.doc*, else 0 memory bits are used

COMPUTATION START

Computation starts one $CLKI$ pulse after the rising edge of the BEG input (shown in timing diagrams as 'OPR')

NUMBER OF ITERATIONS (NITR)

The number of iterations per multiplier is given by $NITR$



SMLT=1 (SEQUENTIAL MULTIPLIER USED)

```
WDX=SIG*(WPLR-1)+(1-SIG)*WPLR
IF WCND>WDX
  NITR=WCND
ELSE
  NITR=WDX
ENDIF
```

--Width of Multiplier w/o sign
 --iterations/intermediate product
 --iterations/intermediate product

BMLT=1 (BLOCK MEMORY MULTIPLIER USED)

```
IF REM(WCND/NWD)>0
  NNN=INT(WCND/NWD)+1
ELSE
  NNN= INT(WCND/NWD)
ENDI
IF REM(WPLR/LWD)>0
  NLN=INT(WPLR/LWD)+1
ELSE
  NLN= INT(WPLR/LWD)
ENDI
NITR=NNN*NLN
```

--iterations/intermediate product

BMLT=0, SMLT=0 (ARRAY MULTIPLIER USED)

```
NITR=1
```

--iterations/intermediate product

TYPE OF INTERMEDIATE MULTIPLIER

```
IF NITR=<1
  SML=0
ELSE
  SML=1
ENDI
```

--Multiplier type
 --Combinational Multiplier
 -- Sequential Multiplier

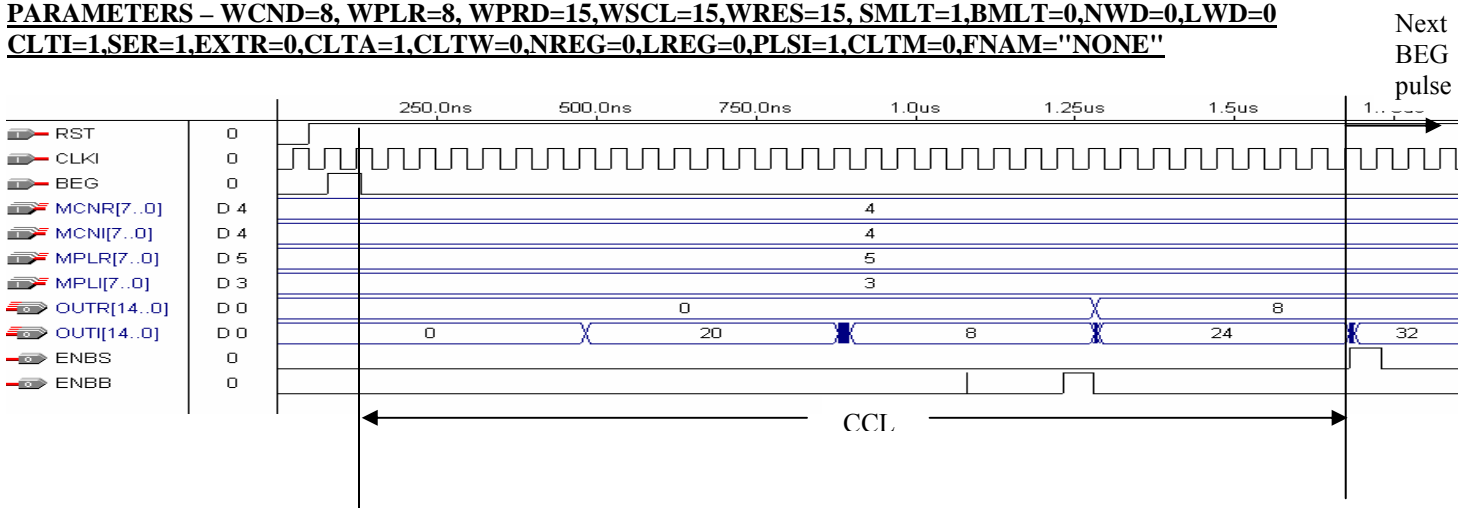
COMPUTATION CLOCKS (CCL)

$$CCL = [(4 * SER) + (1 - SER)] * (CLTI * NITR) + SER * [(3 * CLTM) + DIFF] + [(1 - SML) * SER + SML] * (CLTA + CLTW)$$

Where DIFF=MAX(0,CLTA+CLTW-CLTM- NITR*CLTI)

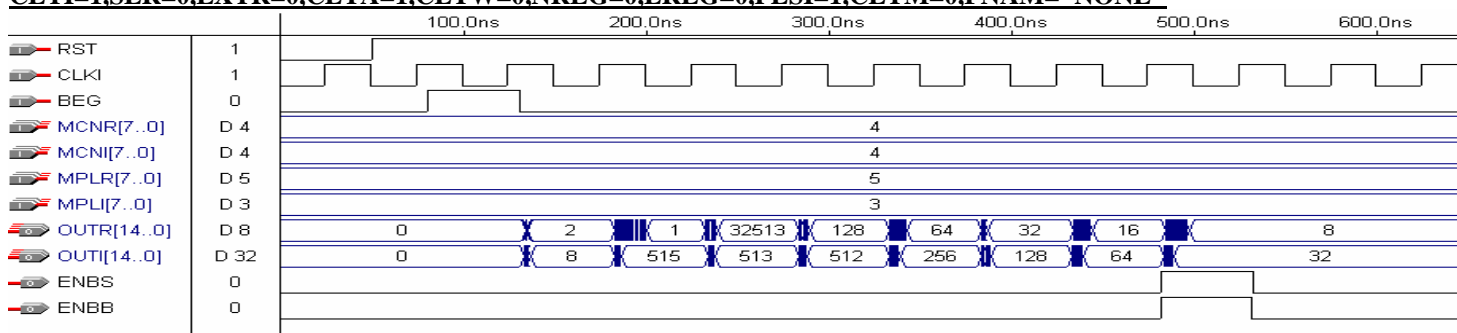
SAMPLE DESIGN TIMING DIAGRAMS

NOTE: The ENBB and ENBS signals shown here correspond to the ENBR and ENBI outputs resp.
PARAMETERS - WCND=8, WPLR=8, WPRD=15, WSCL=15, WRES=15, SMLT=1, BMLT=0, NWD=0, LWD=0
CLTI=1, SER=1, EXTR=0, CLTA=1, CLTW=0, NREG=0, LREG=0, PLSI=1, CLTM=0, FNAM="NONE"

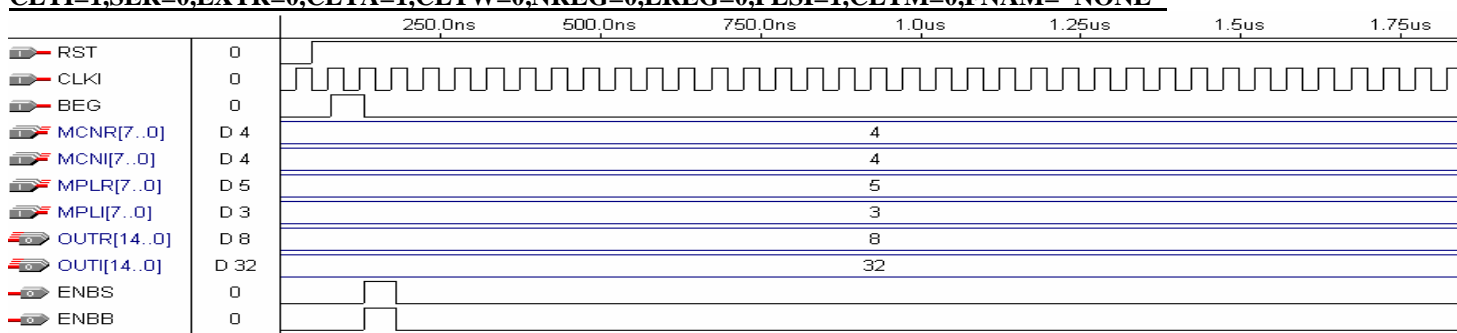




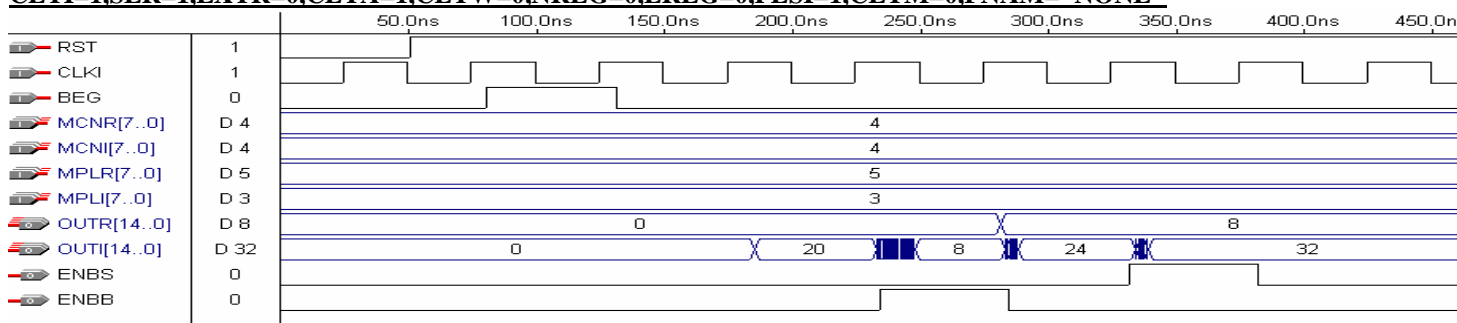
**PARAMETERS – WCND=8, WPLR=8, WPRD=15, WSCL=15, WRES=15, SMLT=1, BMLT=0, NWD=0, LWD=0
CLTI=1, SER=0, EXTR=0, CLTA=1, CLTW=0, NREG=0, LREG=0, PLSI=1, CLTM=0, FNAME="NONE"**



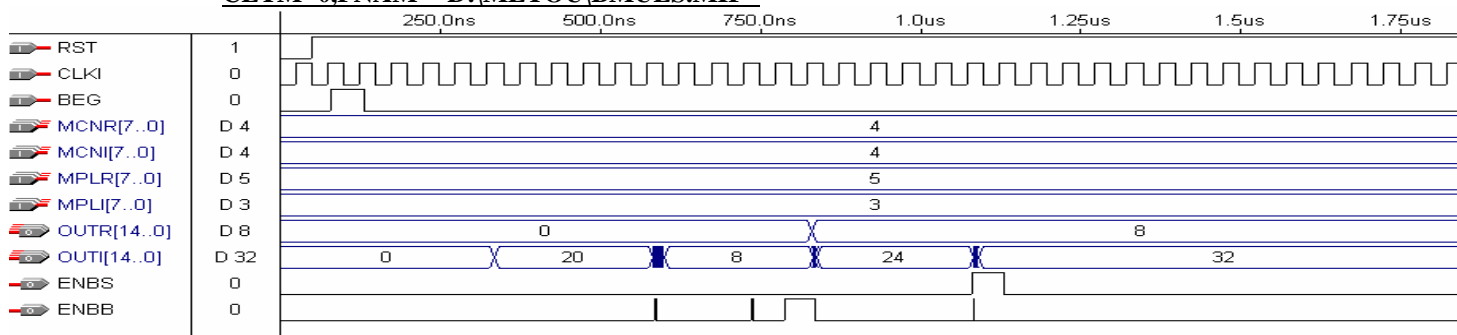
**PARAMETERS – WCND=8, WPLR=8, WPRD=15, WSCL=15, WRES=15, SMLT=0, BMLT=0, NWD=0, LWD=0
CLTI=1, SER=0, EXTR=0, CLTA=1, CLTW=0, NREG=0, LREG=0, PLSI=1, CLTM=0, FNAME="NONE"**



**PARAMETERS – WCND=8, WPLR=8, WPRD=15, WSCL=15, WRES=15, SMLT=0, BMLT=0, NWD=0, LWD=0
CLTI=1, SER=1, EXTR=0, CLTA=1, CLTW=0, NREG=0, LREG=0, PLSI=1, CLTM=0, FNAME="NONE"**



**PARAMETERS – WCND=8, WPLR=8, WPRD=15, WSCL=15, WRES=15, SMLT=0, BMLT=1,
NWD=4, LWD=4, CLTI=1, SER=1, EXTR=0, CLTA=1, CLTW=0, NREG=0, LREG=0, PLSI=1,
CLTM=0, FNAME="D:\METOU\BMULS.MIF"**





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;
          BEG    :IN  NODE;
          MCNR   :IN  BUS1D(7 DOWNT0 0);
          MCNI   :IN  BUS1D(7 DOWNT0 0);
          MPLR   :IN  BUS1D(7 DOWNT0 0);
          MPLI   :IN  BUS1D(7 DOWNT0 0);
          OUTR   :BUFFER BUS1D(11 DOWNT0 0);
          OUTI   :BUFFER BUS1D(11 DOWNT0 0);
          ENBR   :BUFFER NODE;
          ENBI   :BUFFER NODE
    );
END MYTOP;

ARCHITECTURE MYTOP OF MYTOP IS
BEGIN
A1: C_MUL GENERIC MAP(WCND=>8,WPLR=>8,WPRD=>15,WRES=>15,SMLT=>1,BMLT=>0,NWD=>4,LWD=>4,
                     CLTI=>1,SER=>1,EXTR=>0,CLTA=>1,NREG=>0,LREG=>0,PLSI=>1,CLTM=>0
                     FNAM=>"D:\METOU\BMULS.MIF")
    PORT MAP(MCNR,MCNI,MPLR,MPLI,CLKI,RST,OPEN,OPEN,OPEN,OPEN,BEG,
            OUTR,OUTI,ENBR,ENBI);
END MYTOP;
```