



# SOC DESIGN IN VHDL

## DESCRIPTION – Memory and Port Address Decoder

The main constituents of an embedded system are CPU or microcontroller, internal and/or external memory and peripheral devices. A critical element required to glue all of this together is a memory and port address decoder. The M\_PAD.VHD core, described here, meets the requirement of embedded systems of any size. It may be used with any microprocessor or microcontroller core, systems with multiple bus masters, such as DMA controllers and multiple CPUs. The core provides for flexible base address selection, required for add-on cards, such as display and disk adapters, sound cards etc. TRI-STATE outputs are provided for embedded systems with multi master operation with discrete devices. Its Segment Lock feature makes design changes in dynamically changing SOPC systems, easier to manage.

## FEATURES

- **Any number of segments** - memory and I/O
- **Unlimited segment depth** – from zero up
- **Segment Lock** – maintains positional association between input and output even when a segment is unused.
- Decodes **sub-segments** within each main segment.
- May be used as part of an **SOPC** or as a **discrete device**.
- May be used with **multiple bus masters** such as with **DMA** and **multiple CPUs**.
- Provides **registered** or **combinatorial** outputs.
- Interface with any kind of CPU or **microcontroller**.
- Makes it easy to use every bit of **FPGA memory** – normally a difficult task
- User configurable **base address** – Flexible width, position and value, for use in systems with **add-on cards**.
- Interface with internal and/or external peripherals
- **Active Hi** and **Active Lo** outputs.
- Supports **I/O mapped memory**

## APPLICATION

FPGA or ASIC based embedded systems



## VHDL Component Declaration:

```
COMPONENT M_PAD IS
  GENERIC
    (
      IOP   : IARY := (0,0);
      IIP   : IARY := (0,0);
      REG   : INTEGER := 0;
      OPN   : INTEGER := 0;
      SCP   : BUS1D := "00";
      ABW   : INTEGER := 0;
      MPD   : INTEGER := 0;
      EBUS  : INTEGER := 0;

      PBA   : INTEGER := 0;
      PBU   : INTEGER := 0;
      PBL   : INTEGER := 0
    );
  PORT(
    AEN   : IN   NODE := '1';
    CLKI  : IN   NODE := '0';
    ENB   : IN   NODE := '1';
    CLR   : IN   NODE := '1';
    RD    : IN   NODE := '0';
    WR    : IN   NODE := '0';
    IBR   : IN   NODE := '0';
    AD    : IN   BUS1D(NTI(ABW) DOWNT0 0) := (OTHERS => '0');

    ISR   : BUFFER NODE;
    EBR   : BUFFER NODE;
    EBD   : BUFFER NODE;
    SRH   : BUFFER BUS1D(NIZ(IIP,IOP) DOWNT0 0);
    RRH   : BUFFER BUS1D(NIZ(IIP,IOP) DOWNT0 0);
    WRH   : BUFFER BUS1D(NIZ(IIP,IOP) DOWNT0 0);
    S_RL  : BUFFER BUS1D(NIZ(IIP,IOP) DOWNT0 0);
    R_RL  : BUFFER BUS1D(NIZ(IIP,IOP) DOWNT0 0);
    W_RL  : BUFFER BUS1D(NIZ(IIP,IOP) DOWNT0 0);
    CRS   : BUFFER BUS3D(OPN*NIZ(IIP,IOP) DOWNT0 0,
                        OPN*(MPD-1) DOWNT0 0,
                        OPN*(MPD-1) DOWNT0 0);
    CWS   : BUFFER BUS3D(OPN*NIZ(IIP,IOP) DOWNT0 0,
                        OPN*(MPD-1) DOWNT0 0,
                        OPN*(MPD-1) DOWNT0 0);
    CSS   : BUFFER BUS3D(OPN*NIZ(IIP,IOP) DOWNT0 0,
                        OPN*(MPD-1) DOWNT0 0,
                        OPN*(MPD-1) 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) MPAD.DOC	-	This document
a) MYLIB.VHD	-	PACKAGE
b) M_PAD.VHD	-	TOP HIERARCHY DESIGN FILE
c) M_DFF.VHD	-	DESIGN FILE BELOW TOP HIERARCHY
d) S_DFF.VHD	-	-DO-
e) P_AD.VHD	-	-DO-
o) D_ECOD.VHD	-	-DO-
r) M_TRI.VHD	-	-DO-
s) S_TRI.VHD	-	-DO-



## **FEATURES IN DETAIL**

### **SUB-SEGMENTS**

A number of commercially available ASICs and PLD ipcores (such as the RTCC core used in the sample design below) contain more than one status, control and data register. These registers can be accessed in one of two ways:-

- Daisy chained access
- Random access

In the second case access is normally given by decoding a single select input along with one or more address lines. If the i/o space required, can be accommodated within an already defined i/o segment, then a dedicated segment need not be defined. In this way the number of address lines required and the associated logic to generate them can be reduced.

### **SEGMENT LOCK**

Engineers can design a common SOPC with a CPU, memory and peripherals for all applications. This SOPC can then be customized for specific applications by changing generics in a Package. Memory sizes can be changed or made zero, peripheral cores may be enabled or disabled. Consequently the memory and i/o map will constantly change. M\_PAD will still generate the port and memory signals for zero size memories and disabled cores but it will drive them with an inactive logic level. Thus the connections to these disabled peripherals and memories need not be changed. When these peripherals and memories are later enabled M\_PAD will drive them again, with active logic levels.

The CWS[][][], CRS[][][] outputs are the decoded output, input port signals. The 2nd index, [][x] of these array follows the index of the IOP[] and IIP[] arrays resp. In other words, the CWS[][0][] array contains the output port signals corresponding to the output segment with depth specified in IOP[0] and so on. In a given situation if the value of IOP[0], IOP[1] and IOP[2] were specified and in another situation if the value of IOP[1] were to become zero, the position of the IOP[0] and IOP[2] segments in the CWS[][][] output array would remain unchanged viz CWS[][0][] and CWS[][2][] and the position of IOP[1] viz CWS[][1][] would remain locked for later use

### **OUTPUT PADDING**

The 3rd index of the CWS[][][x] and CRS[][][x] arrays, has a maximum value of MPD-1 however elements of all indexes beyond the depth of the respective segments are padded with zero.

### **I/O MAPPED MEMORY**

I/O mapped memory is one that uses the port transfer instructions of the CPU, to transfer data, rather than the memory transfer instructions, used for traditional memory. It is indicated in situations where the entire memory map of the CPU is occupied by traditional memory and additional memory is required. Even when memory space is available, this additional memory will normally not be contiguous with the program and may therefore require a compiler and CPU that supports data segmentation, or a compiler that supports data offsetting. The designer then may have to design source code that can switch between data segments. Notwithstanding this, the I/O space normally requires fewer bits of the address bus than the memory space, adding a segment in the I/O space therefore does not require a larger address bus. Adding a segment in the memory space may require a larger address bus and therefore more silicon usage in a PLD. I/O mapped memory is preferred in such situations.



## PORTS AND PARAMETERS

**INPUT PORTS – Table 1**

NAME	DESCRIPTION	WIDTH	COMMENTS
AEN	Address Enable	1	Active Hi, used in multi bus master systems. In single master systems, leave unused or Hi. In multi master systems when EBUS=1, AEN=0 will TRI-STATE all outputs (except EBD). When EBUS=0, AEN=0 will drive all outputs Lo.
CLKI	Clock	1	Positive edge triggered. Synchronizes all internal operations. Used only when REG=1
ENB	Clock enable	1	Active Hi, ALE (Address Latch Enable) input, used only when REG=1
CLR	Reset	1	Asynchronous, active lo, resets all internal logic. Used only when REG=1
RD	Read	1	Active hi, memory (OPN=0) or port (OPN=0) read signal, when REG=0 this signal is synchronous to the bus command state and comes from the bus controller or microcontroller. When REG=1 it is an asynchronous signal decoded from the bus status signals of the CPU
WR	Write	1	Active hi, memory (OPN=0) or port (OPN=0), when REG=0 this signal is synchronous to the bus command state and comes from the bus controller or microcontroller. When REG=1 it is an asynchronous signal decoded from the bus status signals of the CPU
IBR	Data In	1	EBD output of another M_PAD. Is ORed with the EBD output of this core
AD[]	Address bus	ABW	Latched address bus when REG=0, unlatched address bus when REG=1

**OUTPUT PORTS – Table 2**

NAME	DESCRIPTION	WIDTH	COMMENTS
ISR	Internal read	1	When Hi, indicates that memory (OPN=0) or port(OPN=1) read data is present on the internal data bus. Is used as a mux select line to pass this data to the CPU
EBR	Ext Data Bus Sel	1	When REG=1 it is a registered version of EBD. When REG=0 is the same as EBD. Is used as a mux select line to pass the ext data bus to the CPU
EBD	Ext Data Bus Sel	1	When Hi, indicates that read (RD input=Hi) data is present on the external data bus. Used as IBR i/p of another instantiation of M_PAD. If another instantiation of M_PAD is not present, it is unused. It is the only o/p that is not fed by a TRI-STATE buffer and is a combinatorial version of the EBR output.
SRH[]	Segment select	NSG <sup>1</sup>	Active Hi vector with index 0 for the left-most element in IOP[]/IIP[]
RRH[]	Segment read	NSG <sup>1</sup>	- do -
WRH[]	Segment write	NSG <sup>1</sup>	- do -
S_RL[]	Segment select	NSG <sup>1</sup>	Active Lo vector with index 0 for the left-most element in IOP[]/IIP[]
R_RL[]	Segment read	NSG <sup>1</sup>	- do -
W_RL[]	Segment write	NSG <sup>1</sup>	- do -
CRS[][][]	Port read signals	Note 2	Calling the three dimensions x,y,z. Dimension 'x' represents the segment number with zero for the left most segment in the IOP[]/IIP[] arrays. The 'y' dimension indicates the number of address bits decoded. When y=0 all the port address bits are decoded, when y=1, all except the LSB(bit 0) are decoded, when y=2, all except bit 1 are decoded and so on. The third dimension 'z' is the decoded value of the address bits. Signals in this port are active Hi. Generated only for OPN=1. When OPN=0 it reduces to a single bit with a value of 0 i.e CRS(0,0,0)=0'
CWS[][][]	Port write signals	Note 2	See CRS above
CSS[][][]	Port RD or WR	Note 2	See CRS above

Notes:-

1> NSG= #elements in IOP[] and IIP[] input arrays. If the number of elements in them differ, NSG is the larger one.

2> 3d array - [NSG-1..0], [MPD-1..0], [MPD-1..0]. Where MPD>=largest element in IOP[] and IIP[] input arrays



**PARAMETERS** – Table 3 (all INTEGER type, except where specifically mentioned)

NAME	MIN	DESCRIPTION
IOP[]	---	Unconstrained integer array of output port segment depths (when OPN=1) or memory segment depths (when OPN=0). It may have any number of elements and each element may have any value greater than or equal to zero. Each element of IOP[] shares address space with an element in IIP[] present at the same index position. A segment at a given index position in IOP[] will have its port signals at the same index in the first dimension of the three dimensional output arrays CSS[][][],CRS[][][] and CWS[][][] ie there is a one-to-one positional association between IOP[] and IIP[] and between IOP[] and (CSS,CRS,CWS) ,eg IOP[1] will share address space with IIP[1] and will have its port signals at CSS[1][][] , CRS[1][][] , CWS[1][][] . The left-most element in IOP[] has an index of 0
IIP[]	---	Unconstrained integer array of input port segment depths (when OPN=1) , unused when OPN=0. It may have any number of elements and each element may have any value greater than or equal to zero. Each element of IIP[] shares address space with an element in IOP[] present at the same index position. A segment at a given index position in IIP[] will have its port signals at the same index in the first dimension of the three dimensional output arrays CSS[][][],CRS[][][] and CWS[][][] ie there is a one-to-one positional association between IIP[] and IOP[] and between IIP[] and (CSS,CRS,CWS) ,eg IIP[1] will share address space with IOP[1] and will have its port signals at CSS[1][][] , CRS[1][][] , CWS[1][][] . The left-most element in IIP[] has an index of 0. When OPN=0 this port must be left unused or all elements must be 0
REG	1	When REG=1 all outputs (except EBD) are registered. REG can be made 1 only when the WR and RD inputs are combinatorial signals decoded from the pre-command state bus status signals coming from a CPU and arriving at the M_PAD core 1 CLKI before start of the bus command state. The CLR input must be a registered signal derived from the said status signals. If these conditions cannot be met REG must be 0
OPN	0	Memory decode(0) / port decode (1)
SCP[]	---	Unconstrained vector with a one-to-one positional association with the elements in IIP[] and IOP[] , indicates whether the elements in these arrays are external(1) or internal(0) port or memory devices. This if SCP[0]='0', IIP[0] (element 0) and IOP[0] are external and so on.
ABW <sup>2</sup>	1	Width of the AD[] input port
MPD <sup>3</sup>	1	Width of third dimension of CRS[][][],CWS[][][] and CSS[][][] output arrays.
EBUS	0	When this core is used as a discrete device(all pins connected to PCB) in a multi-master system, EBUS=1. When this core is used as a discrete device in a single master system, EBUS may be 0 or 1. When some or all pins are used internally, EBUS must be unused or 0. EBUS determines how the AEN i/p is used.
PBA <sup>1</sup>	0	Base address. Is the value of a slice, bound by PBU and PBL, of the address bus AD[] for which this instantiation of M_PAD will generate output signals. In other words, the base address demarcates a portion of the I/O address map (when OPN=1) and a part of the memory address map (when OPN=0) for the devices under the control of M_PAD.
PBU <sup>1</sup>	0	Upper index of base address value (PBA) in the AD[] input
PBL <sup>1</sup>	0	Lower index of base address value (PBA) in the AD[] input

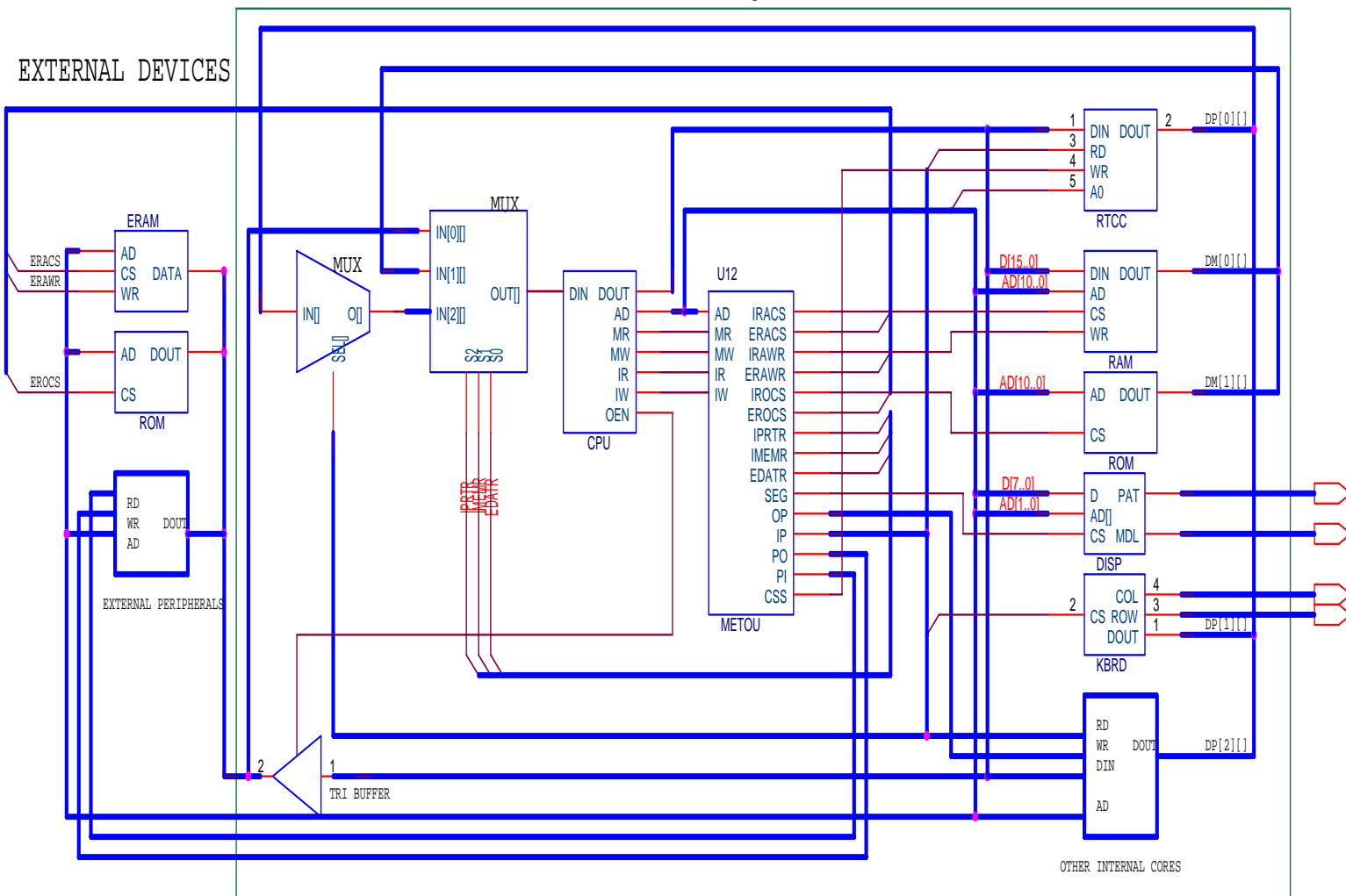
Notes:-

- 1> When PBA=PBU=PBL=0, the entire address map belongs to M\_PAD i.e. Base Address logic is disabled.
- 2> ABW>=PSU+1 (see 'SAMPLE DESIGN OUTPUTS' below, for calculation of PSU).
- 3> MPD>=largest element in IOP[] and IIP[] input arrays



## SAMPLE DESIGN - SCHEMATIC

FPGA



### NOTES

- 1>The external data bus is shorted to all external peripherals and will contain data of only the selected device, this is because of the tri-state buffers present at the o/p of these devices. This configuration functions as a multiplexer. FPGAs on the other hand do not support internal shorting and also do support tri-state buffers for exclusively internal use, hence the use of multiplexers.
- 2>Internal memory has one data in port (DIN) and one data out port (DOUT). External devices on the other hand have only one bi-directional data port. This is because FPGAs do not support internal bi-directional signals. Bi-directionality requires the use of tri-state buffers the outputs of which are shorted, this requirement cannot be met inside the FPGA.
- 3>The tri-state buffer shown above is used to implement a bi-directional pin for the external data bus.



## SAMPLE DESIGN-REQUIREMENTS

**PERIPHERAL LIST** – Table 4

NAME	DESCRIPTION	LOCATION
RTCC	REAL-TIME CLOCK & CALENDER	INSIDE DEVICE
KBRD	MATRIX KEYBOARD CONTROLLER	-do-
DISP	LED DISPLAY CONTROLLER (4 modules X 16 seg)	-do-
IRAM	INTERNAL RAM	-do-
IROM	INTERNAL ROM	-do-
CPU	ANY CPU OR MICRO-CONTROLLER SUCH AS - NIOS,ARM,8051,SAM, ETC.	-do-
METOU	SAMPLE DESIGN	-do-
ERAM	EXTERNAL RAM	OUTSIDE DEVICE
EROM	EXTRNAL ROM	-do-

### NOTES

- 1> Since input and output ports are differentiated by the RD and WR inputs they can share the same address space and therefore the same segments. See IOP and IIP in TABLE 3 above.
- 2> **IOPN,IIPN,LEDS,EOPN,EIPN,IRAD,IROD,ERAD** and **EROD** used in tables 5,6 and 7 are constants used in the VHDL coding of the sample design.
- 3> The **RTCC** core requires one WR port to write to the date and time registers. It requires the A0 line to indicate which of the two registers are being written to. We can use the MSB bits of the address bus to decode the WR input and the LSB bit of the address bus (AD0) for the A0 input. We thus require two output ports
- 4> For this sample design the **DISP** core has been configured for 4 modules and 16 segments. It therefore requires 4 address locations for its Video RAM. For the sample design we use I/O mapped memory for the reasons mentioned earlier.

**OUTPUT PORT REQUIREMENT** – Table 5

NAME	# PORTS	SEGMENT #	LOCATION
RTCC	2	1	INTERNAL
OTHERS	6	1	INTERNAL
<b>total</b>	<b>IOPN=8</b>		
DISP	4	2	INTERNAL
<b>total</b>	<b>LEDS=4</b>		
OTHERS	6	3	EXTERNAL
<b>total</b>	<b>EOPN=6</b>		

**INPUT PORT REQUIREMENT** –Table 6

NAME	# PORTS	SEGMENT #	LOCATION
RTCC	1	1	INTERNAL
KBRD	1	1	INTERNAL
OTHERS	1	1	INTERNAL
<b>total</b>	<b>IIPN=3</b>		
OTHERS	3	3	EXTERNAL
<b>total</b>	<b>EIPN=3</b>		

**MEMORY REQUIREMENT** –Table 7

NAME	# LOCATIONS	SEGMENT #	LOCATION
IRAM	2048	1	INTERNAL
<b>total</b>	<b>IRAD=2048</b>		
IROM	2048	2	INTERNAL
<b>total</b>	<b>IROD=2048</b>		
ERAM	32767	3	EXTERNAL
<b>total</b>	<b>ERAD=32767</b>		
EROM	32767	4	EXTERNAL
<b>total</b>	<b>EROD=32767</b>		



## SAMPLE DESIGN-CODING

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

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

```
PACKAGE M_ETOU IS  
  CONSTANT  IRAD  : INTEGER:=2048;           --INT RAM DEPTH  
  CONSTANT  IROD  : INTEGER:=2048;           --INT ROM DEPTH  
  CONSTANT  ERAD  : INTEGER:=32767;          --EXT RAM DEPTH  
  CONSTANT  EROD  : INTEGER:=32767;          --EXT ROM DEPTH  
  CONSTANT  IOPN  : INTEGER:=8;             --# INT O/P PORTS  
  CONSTANT  IIPN  : INTEGER:=3;             --# INT I/P PORTS  
  CONSTANT  LEDS  : INTEGER:=4;             --# INT LED MODULES  
  CONSTANT  EOPN  : INTEGER:=6;             --# EXT O/P PORTS  
  CONSTANT  EIPN  : INTEGER:=3;             --# EXT I/P PORTS  
  
  CONSTANT  MPD   : INTEGER:=8;             --MAX I/O PORT DEPTH  
  CONSTANT  MMD   : INTEGER:=32767;         --MAX MEMORY DEPTH  
  CONSTANT  ABW   : INTEGER:=20;           --ADDR BUS WIDTH  
END M_ETOU;
```

----- PACKAGE ENDS -----

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

```
LIBRARY MYLIB;  
USE MYLIB.MYLIB.ALL;  
USE WORK.M_ETOU.ALL;
```

```
ENTITY METOU IS  
  PORT( AD      : IN      BUS1D(ABW-1 DOWNT0 0);  
        MR      : IN      NODE;  
        MW      : IN      NODE;  
        IR      : IN      NODE;  
        IW      : IN      NODE;  
        IRACS   : BUFFER  NODE;           --IRAM CS-ACTIVE LO  
        ERACS   : BUFFER  NODE;           --ERAM CS-ACTIVE LO  
        IRAWR   : BUFFER  NODE;           --IRAM WR-ACTIVE LO  
        ERAWR   : BUFFER  NODE;           --ERAM WR-ACTIVE LO  
        IROCS   : BUFFER  NODE;           --IROM CS-ACTIVE LO  
        EROCS   : BUFFER  NODE;           --EROM CS-ACTIVE LO  
        IPRTR   : BUFFER  NODE;           --INT PORT RD  
        IMEMR   : BUFFER  NODE;           --INT MEM RD  
        EDATR   : BUFFER  NODE;           --INT DATA BUS  
        SEG     : BUFFER  NODE;           --LED DISPLAY VRAM SEGMENT  
        OP      : BUFFER  BUS1D(IOPN-1 DOWNT0 0);  
        IP      : BUFFER  BUS1D(IIPN-1 DOWNT0 0);  
        PO      : BUFFER  BUS1D(EOPN-1 DOWNT0 0);  
        PI      : BUFFER  BUS1D(EIPN-1 DOWNT0 0);  
        CSS     : BUFFER  BUS1D(MPD-1 DOWNT0 0));  
END METOU;
```



## ARCHITECTURE METOU OF METOU IS

```
SIGNAL   EBR           :NODE;
SIGNAL   OPX           :BUS3D(2 DOWNT0 0,MPD-1 DOWNT0 0,MPD-1 DOWNT0 0);
SIGNAL   IPX           :BUS3D(2 DOWNT0 0,MPD-1 DOWNT0 0,MPD-1 DOWNT0 0);

BEGIN

                                --PORT DECODER--
OP <=PAD(IOPN-1,EXTR_3D(OPX,0,0));      --INTERNAL O/P PORTS
IP <=PAD(IIPN-1,EXTR_3D(IPX,0,0));      --INTERNAL I/P PORTS
PO <=PAD(EOPN-1,EXTR_3D(OPX,2,0));      --EXTERNAL O/P PORTS
PI <=PAD(EIPN-1,EXTR_3D(IPX,2,0));      --EXTERNAL I/P PORTS
CSS<=PAD(MPD-1 ,EXTR_3D(OPX,0,1));      --1st SUB-SEG OF 1st MAIN OUT PORT SEG

A9: M_PAD GENERIC MAP(
    IOP    =>(IOPN,LEDS,EOPN),          --PORT O/P SEG DEPTHS
    IIP    =>(IIPN,0 ,EIPN),            --PORT I/P SEG DEPTHS
    OPN    =>1,                          --PORT DECODE
    SCP    =>"001",                      --INTERNAL/EXTERNAL-0/1
    ABW    =>ABW,                        --ADDRESS BUS WIDTH
    MPD    =>MPD,                        --MAX I/O PORT SEGMENT DEPTH
    PBU    =>5,                          --BASE ADDR UPPER INDEX
    PBL    =>5,                          --BASE ADDRESS LOWER INDEX
    PBA    =>0)                          --BASE ADDRESS VALUE
PORT MAP(
    RD=>IR,WR=>IW,AD=>AD,                --I/O READ,I/O WRITE,ADDRESS BUS

    CWS    =>OPX,
    CRS    =>IPX,
    ISR    =>IPRTR,                      --INT I/O RD-ACTIVE HI
    EBD    =>EBR,                        --EXT,!BASE ADDR I/O RD-ACTIVE HI
    SRH(1) =>SEG);                      --LED DISPLAY VIDEO RAM WRITE

                                ---MEMORY DECODER---
A10:M_PAD GENERIC MAP(
    IOP    =>(IRAD,IROD,ERAD,EROD),      --MEM SEG DEPTHS
    OPN    =>0,                          --MEMORY DECODE
    SCP    =>"0011",                    --INTERNAL/EXTERNAL-0/1
    ABW    =>ABW,
    MPD    =>MMD)
PORT MAP(
    RD=>MR,WR=>MW,AD=>AD,IBR=>EBR,

    S_RL(0) =>IRACS,                    --IRAM CS-ACTIVE LO
    S_RL(2) =>ERACS,                    --ERAM CS-ACTIVE LO
    W_RL(0) =>IRAWR,                    --IRAM WR-ACTIVE LO
    W_RL(2) =>ERAWR,                    --ERAM WR-ACTIVE LO
    R_RL(1) =>IROCS,                    --IROM CS-ACTIVE LO
    R_RL(3) =>EROCs,                    --EROM CS-ACTIVE LO
    ISR    =>IMEMR,                      --INT MEM RD-ACTIVE HI
    EBR    =>EDATR);                    --EXT,!BASE ADDR RD-ACTIVE HI

END METOU;
```



## SAMPLE DESIGN-OUTPUTS

### PORT DECODER

IOP[ ]=O/P SEGMENT DEPTHS	= [ IOPN, LEDS, EOPN ]	= [ 8, 4, 6 ]
IIP[ ]=I/P SEGMENT DEPTHS	= [ IIPN, 0, EIPN ]	= [ 3, 0, 3 ]
NMS =# OF NON 0 SEGMENTS		= 3
NSG =# OF SEGMENTS		= 3
MNP =DEPTH OF DEEPEST SEGMENT		= 8
NIS =# OF PORTS PER SEGMENT-1	= MNP-1	= 7
PSU =MSB OF ADDRESS BUS	= LOG2 (MNP-1) + LOG2 (NMS-1) + 1	= 4
SGW =WIDTH OF SEGMENT ADDR-1	= LOG2 (NMS-1)	= 1
SGI =LOWER INDEX OF SEGMENT ADDR=PSU-SGW		= 3
	=UPPER INDEX+1 OF PORT ADDR	
OM[ ] =SEGMENT ADDRESS	= [ AD <sub>PSU</sub> .. AD <sub>SGI</sub> ]	= [ AD4 .. AD3 ]

### Address Decoder PA[SGI-1..0][NIS..0] –Table 8

PA	NIS	6	5	4	3	2	1	0	i	Decode [AD <sub>SGI-1</sub> ..AD <sub>SGI-i</sub> ]
0	AD2*AD1*AD0	AD2*AD1*AD0	AD2*AD1*AD0	AD2*AD1*AD0	AD2*AD1*AD0	AD2*AD1*AD0	AD2*AD1*AD0	AD2*AD1*AD0	SGI	Decode [AD2..AD0]
1	0	0	0	0	AD2*AD1	AD2*AD1	AD2*AD1	AD2*AD1	2	Decode [AD2..AD1]
SGI-1	0	0	0	0	0	0	AD2	AD2	1	Decode [AD2..AD2]

- The index 'i' below is the segment count.
- The index 'j' is the non-zero segment count ie it increments only if the segment depth is non-zero.
- The rows in which the 'k' index is zero are the port select signals for the segment.
- The rows in which the 'k' index is not zero are the select signals for the sub-segments within the segment
- SRH[i] are the segment select signals. They are the result of comparing the segment address OM[] with index 'j' and ANDING with (RD or WR)
- RRH[i] are the segment read signals. They are the result of comparing the segment address OM[] with index 'j' and ANDING with (RD)
- WRH[i] are the segment write signals. They are the result of comparing the segment address OM[] with index 'j' and ANDING with (WR)

### Port Decoder

#### Port Select Decoder CSS[NSG-1..0][MPD-1..0][MPD-1..0]-Table 9

MPD-1	m								i,	k	j
	7	6	5	4	3	2	1	0			
0	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	0	0	0
0	0	0	0	0	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	0	1	0
0	0	0	0	0	0	0	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	0	log2(SID[0]-1)	0
0	0	0	0	0	0	0	0	0	0	MPD-1	0
0	0	0	0	0	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	1	0	1
0	0	0	0	0	0	0	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	1	log2(SID[1]-1)	1
0	0	0	0	0	0	0	0	0	1	MPD-1	1
0	0	0	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	2	0	2
0	0	0	0	0	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	2	1	2
0	0	0	0	0	0	0	PA[k,m]*SRH[i]	PA[k,m]*SRH[i]	2	log2(SID[2]-1)	2
0	0	0	0	0	0	0	0	0	NSG-1	MPD-1	NMS-1

#### Port Read Decoder CSR[NSG-1..0][MPD-1..0][MPD-1..0]-Table 10

MPD-1	m								i,	k	j
	7	6	5	4	3	2	1	0			
0	0	0	0	0	0	PA[k,m]*RRH[i]	PA[k,m]*RRH[i]	PA[k,m]*RRH[i]	0	0	0
0	0	0	0	0	0	0	PA[k,m]*RRH[i]	PA[k,m]*RRH[i]	0	log2(IIP[0]-1)	0
0	0	0	0	0	0	0	0	0	0	MPD-1	0
0	0	0	0	0	0	0	0	0	1	log2(IIP[1]-1)	1
0	0	0	0	0	0	0	0	0	1	MPD-1	1
0	0	0	0	0	0	PA[k,m]*RRH[i]	PA[k,m]*RRH[i]	PA[k,m]*RRH[i]	2	0	2
0	0	0	0	0	0	0	PA[k,m]*RRH[i]	PA[k,m]*RRH[i]	2	log2(IIP[2]-1)	2
0	0	0	0	0	0	0	0	0	NSG-1	MPD-1	NMS-1



## Port Write Decoder CSW[NSG-1.0][MPD-1.0][MPD-1.0]-Table 11

m										i	k	j
MPD-1	7	6	5	4	3	2	1	0				
0	PA[k,m]*WRH[i]	PA[k,m]*WRH[i]	PA[k,m]*WRH[i]	PA[k,m]*WRH[i]	PA[k,m]*WRH[i]	PA[k,m]*WRH[i]	PA[k,m]*WRH[i]	PA[k,m]*WRH[i]	PA[k,m]*WRH[i]	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	0	log2(IOP[0]-1)	0
0	0	0	0	0	0	0	0	0	0	0	MPD-1	0
0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	log2(IOP[1]-1)	1
0	0	0	0	0	0	0	0	0	0	0	MPD-1	1
0	0	0	0	0	0	0	0	0	0	0	0	2
0	0	0	0	0	0	0	0	0	0	0	1	2
0	0	0	0	0	0	0	0	0	0	0	log2(IOP[2]-1)	2
0	0	0	0	0	0	0	0	0	0	0	NSG-1 MPD-1	NMS-1

### MEMORY DECODER

IOP[ ] = O/P SEGMENT DEPTHS	= [ IRAD, IROD, ERAD, EROD ]	= [ 2048, 2048, 32767, 32767 ]
NMS = # OF NON 0 SEGMENTS		= 4
NSG = # OF SEGMENTS		= 4
MNP = DEPTH OF DEEPEST SEGMENT		= 32767
NIS = # OF PORTS PER SEGMENT -1	= MNP-1	= 32766
PSU = MSB OF ADDRESS BUS	= LOG2(MNP-1) + LOG2(NMS-1) + 1 = 14 + 1 + 1 = 16	
SGW = WIDTH OF SEGMENT ADDR-1	= LOG2(NMS-1)	= 1
SGI = LOWER INDEX OF SEGMENT ADDR	= PSU - SGW	= 15
	= UPPER INDEX + 1 OF PORT ADDR	
OM[ ] = SEGMENT ADDRESS	= [ AD <sub>PSU</sub> .. AD <sub>SGI</sub> ]	= [ AD16 .. AD15 ]

\*PROCEDURE FOR GENERATING THE O/P SIGNALS IS THE SAME AS FOR THE 'PORT DECODER' DONE ABOVE

### MEMORY and PORT ADDRESS MAP

#### PORT ADDRESS MAP

SEGMENT	PORT START	PORT END
AD4..AD3	AD2..AD0	AD2..AD0
00	000	111
01	000	011
10	000	101

#### MEMORY ADDRESS MAP

SEGMENT	MEM START	MEM END
AD16..AD15	AD14..AD0	AD14..AD0
00	000 0000 0000 0000	000 0111 1111 1111
01	000 0000 0000 0000	000 0111 1111 1111
10	000 0000 0000 0000	111 1111 1111 1111
11	000 0000 0000 0000	111 1111 1111 1111



## SAMPLE DESIGN-RESOURCE UTILISATION

Device-Altera EP1K100QC208-3  
Logic cells-38  
Memory-0

## SAMPLE DESIGN-TIMING SIMULATION

