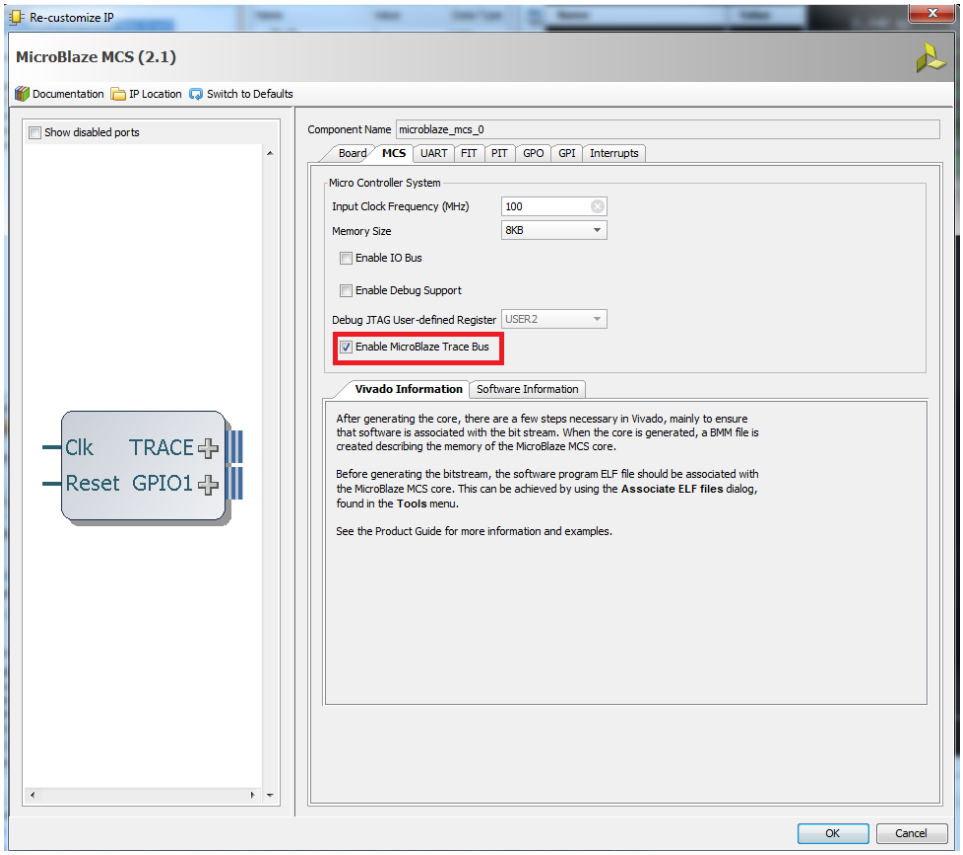
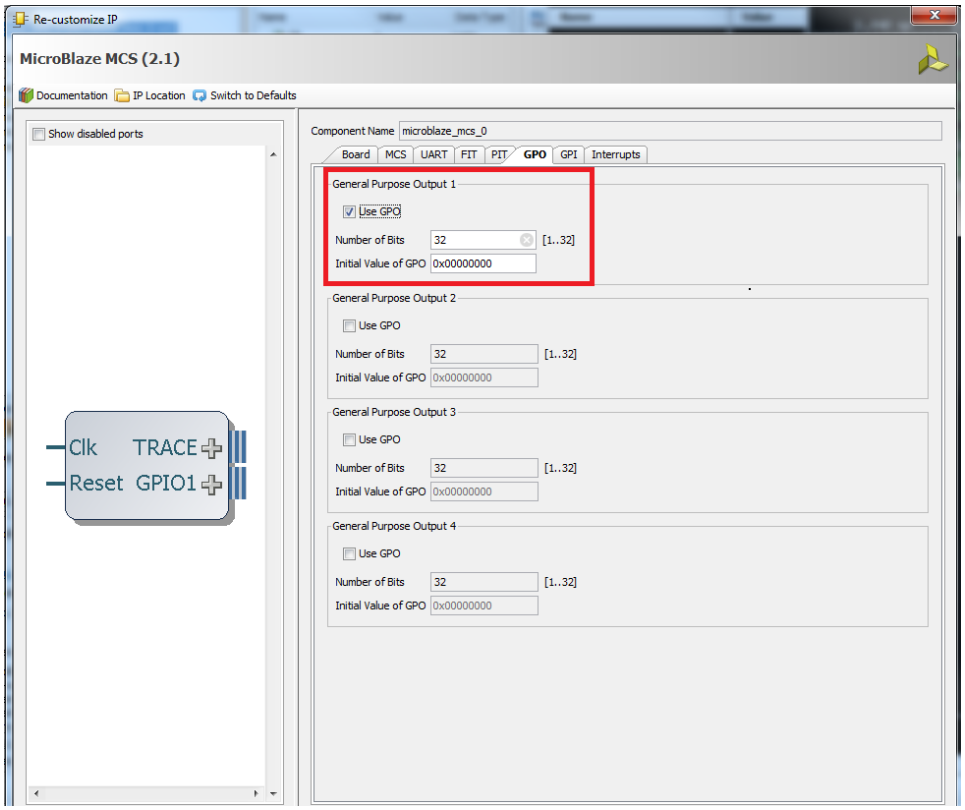
I used Vivado 2013.4 for this test (For ISE skip to page 4). I created a Vivado project targeting a KC705. I added the MCS from the catalog. I configured the MCS as follows (all the rest is left default)





I generated the Output Products (Right Click on the MCS in design sources and select Generate Output Products)

Next, I launched SDK and created a New BSP (File -> New -> Board Support Package). This prompted me to add the XML file. I used the one in the sources directory (similar to below):

project_1\project_1.srcs\sources_1\ip\microblaze_mcs_0\microblaze_mcs_0_sdk.xml

I then created an empty application. I created my application to do a write to the GPO1. My code is shown below:

```
#include <stdio.h>
#include "xil_io.h"


int main()
{

    Xil_Out32(0x80000010, 0x12345678);

    return 0;
}
```

I got the GPO1 register address from the MCS datasheet:

MicroBlaze MCS Register Descriptions

Table 14: **MicroBlaze MCS Address Map**

Address (hex)	Name	Access Type	Description
0x0 - C_MEMSIZE-1	Local Memory	RW	Local Memory for MicroBlaze software
C_MEMSIZE - 0x7FFFFFFF	Reserved		
0x80000000	UART_RX	R	UART Receive Data Register
0x80000004	UART_TX	W	UART Transmit Data Register
0x80000008	UART_STATUS	R	UART Status Register
0x8000000C	IRQ_MODE	W	Interrupt Mode Register
0x80000010	GPO1	W	General Purpose Output 1 Register

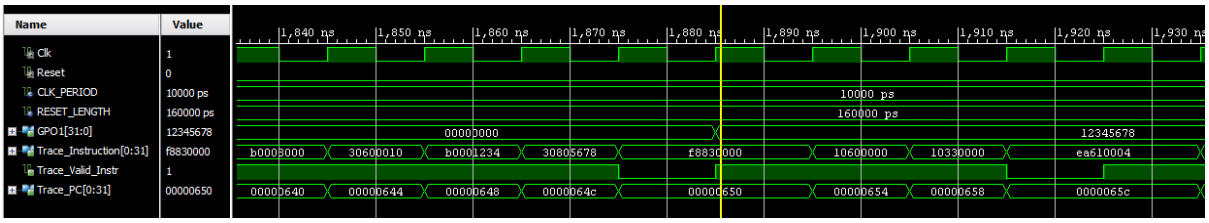
I generated the elf file. To create the mem file. I used my own script shown below:

C:/Xilinx/14.7/ISE_DS/ISE/bin/nt64/data2mem.exe -
bm ../../project_1.srcs/sources_1/ip/microblaze_mcs_0/microblaze_mcs_0.bmm -
bd ../../workspace/test_gpo/Debug/test_gpo.elf -bx

C:/Xilinx/14.7/ISE_DS/EDK/gnu/microblaze/nt64/bin/mb-objdump.exe -
S ../../workspace/test_gpo/Debug/test_gpo.elf > disassembled.txt

This will generate the mem files, and a disassembled.txt file. I then added the mem files as simulation sources. I created a simple Testbench that simply generated the clk, and reset.

I then run the simulation and I added the Trace signals, and the GPO1.



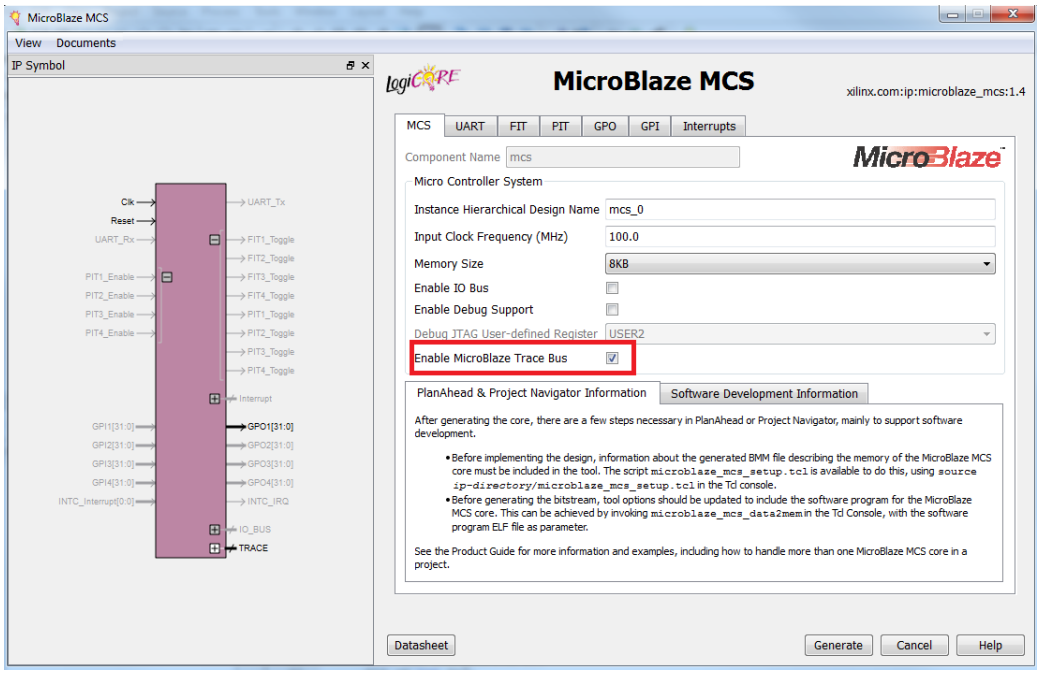
Here, you can see that the GPO1 is written to as expected. Here, you can compare use the disassembled elf and verify against the simulation results:

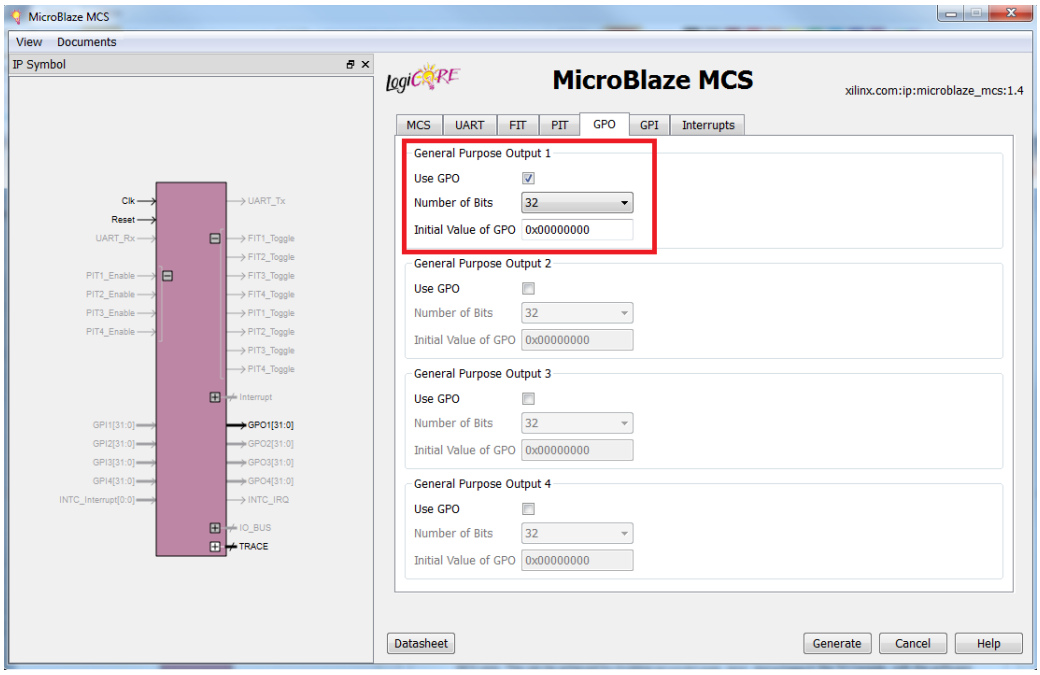
```
int main()
{
 634:   3021fff8        addik   r1, r1, -8
 638:   fa610004        swi     r19, r1, 4
 63c:   12610000        addk    r19, r1, r0

    Xil_Out32(0x80000010, 0x12345678);
 640:   b0008000        imm     -32768
 644:   30600010        addik   r3, r0, 16
 648:   b0001234        imm     4660
 64c:   30805678        addik   r4, r0, 22136
 650:   f8830000        swi     r4, r3, 0

    return 0;
 654:   10600000        addk    r3, r0, r0
}
 658:   10330000        addk    r1, r19, r0
 65c:   ea610004        lwi     r19, r1, 4
 660:   30210008        addik   r1, r1, 8
 664:   b60f0008        rtsd    r15, 8
 668:   80000000        or      r0, r0, r0
```

Here, I created a simple ISE project targeting the KC705. I added a new source (Coregen -> MCS). I Configured this as follows (everything else was left as default):





I then launched SDK. I created a new BSP (File -> New -> Board Support Package). This prompted my to add the XML file (I used the one in the ISE project):

mcs_sim\ISE\top\ipcore_dir

I then created an empty application. I created my application to do a write to the GPO1. My code is shown below:

```
#include <stdio.h>
#include "xil_io.h"


int main()
{

    Xil_Out32(0x80000010, 0x12345678);

    return 0;
}
```

I got the GPO1 register address from the MCS datasheet:

MicroBlaze MCS Register Descriptions

Table 14: **MicroBlaze MCS Address Map**

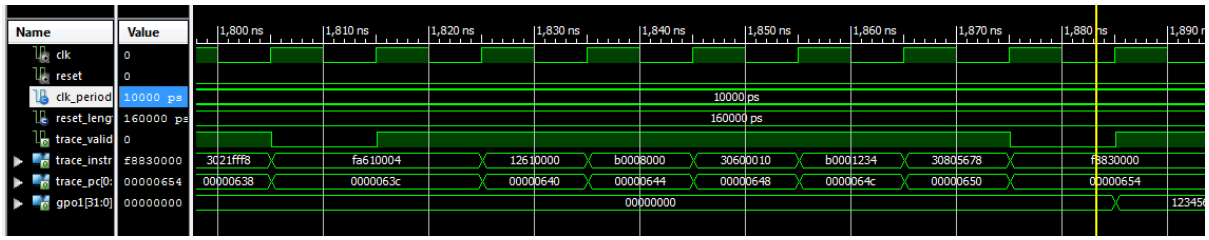
Address (hex)	Name	Access Type	Description
0x0 - C_MEMSIZE-1	Local Memory	RW	Local Memory for MicroBlaze software
C_MEMSIZE - 0x7FFFFFFF	Reserved		
0x80000000	UART_RX	R	UART Receive Data Register
0x80000004	UART_TX	W	UART Transmit Data Register
0x80000008	UART_STATUS	R	UART Status Register
0x8000000C	IRQ_MODE	W	Interrupt Mode Register
0x80000010	GPO1	W	General Purpose Output 1 Register

I generated the elf file. To create the mem file. I used my own script (placed in the project directory) shown below:

C:/Xilinx/14.7/ISE_DS/ISE/bin/nt64/data2mem.exe -bm ipcore_dir/mcs.bmm -bd ../workspace/test_gpo/Debug/test_gpo.elf -bx

C:/Xilinx/14.7/ISE_DS/EDK/gnu/microblaze/nt64/bin/mb-objdump.exe -S ../workspace/test_gpo/Debug/test_gpo.elf > disassembled.txt

This will generate the mem files, and a disassembled.txt file. I then added the mem files as simulation sources. I created a simple Testbench that simply generated the clk, and reset.



Here, you can see that the GPO1 is written to as expected. Here, you can compare use the disassembled elf and verify against the simulation results:

```
int main()
{
 638:   3021fff8        addik   r1, r1, -8
 63c:   fa610004        swi     r19, r1, 4
 640:   12610000        addk    r19, r1, r0

    Xil_Out32(0x80000010, 0x12345678);
 644:   b0008000        imm     -32768
 648:   30600010        addik   r3, r0, 16
 64c:   b0001234        imm     4660
 650:   30805678        addik   r4, r0, 22136
 654:   f8830000        swi     r4, r3, 0

    return 0;
 658:   10600000        addk    r3, r0, r0
}
 65c:   10330000        addk    r1, r19, r0
 660:   ea610004        lwi     r19, r1, 4
 664:   30210008        addik   r1, r1, 8
 668:   b60f0008        rtsd    r15, 8
 66c:   80000000        or      r0, r0, r0
```