

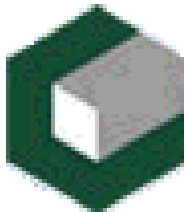
Depuración y verificación en sistemas empotrados

Maestría en Sistemas Digitales

Alejandro J. Cabrera Sarmiento

Dpto. de Automática y Computación
Universidad Tecnológica de La Habana “José Antonio Echeverría”
CUJAE

alex@automatica.cujae.edu.cu



Sumario

- Depuración y verificación en sistemas empotrados
- Modos de depuración de MicroBlaze
- Herramientas de depuración en EDK
 - XMD Xilinx Microprocessor Debugger (*XMD*)
 - GNU Debugger (*GDB*)
- Herramientas de verificación de Xilinx
 - ChipScope Pro



Depuración de aplicaciones en sistemas empotrados

- Un porcentaje elevado del ciclo de diseño de un sistema de procesamiento empotrado de cierta complejidad se invierte en:
 - **Depuración del software** ejecutado en el procesador.
 - **Verificación del hardware y las transacciones** en buses internos.
- Si consideramos el *Time_to_Market* como un factor clave del diseño, es preciso contar con herramientas que simplifiquen y aceleren las dos etapas anteriores.



Depuración de aplicaciones en sistemas empotrados (cont.)

- El entorno **EDK** proporciona:
 - Herramientas de **depuración de software** para programas que se ejecutan en el procesador implementado en una FPGA
 - Módulos **IP específicos** que permiten acceder a los procesadores, buses y lógica de usuario en el interior de la FPGA y **software** para visualizar y controlar estos bloques



Herramientas de depuración

GNU debugger

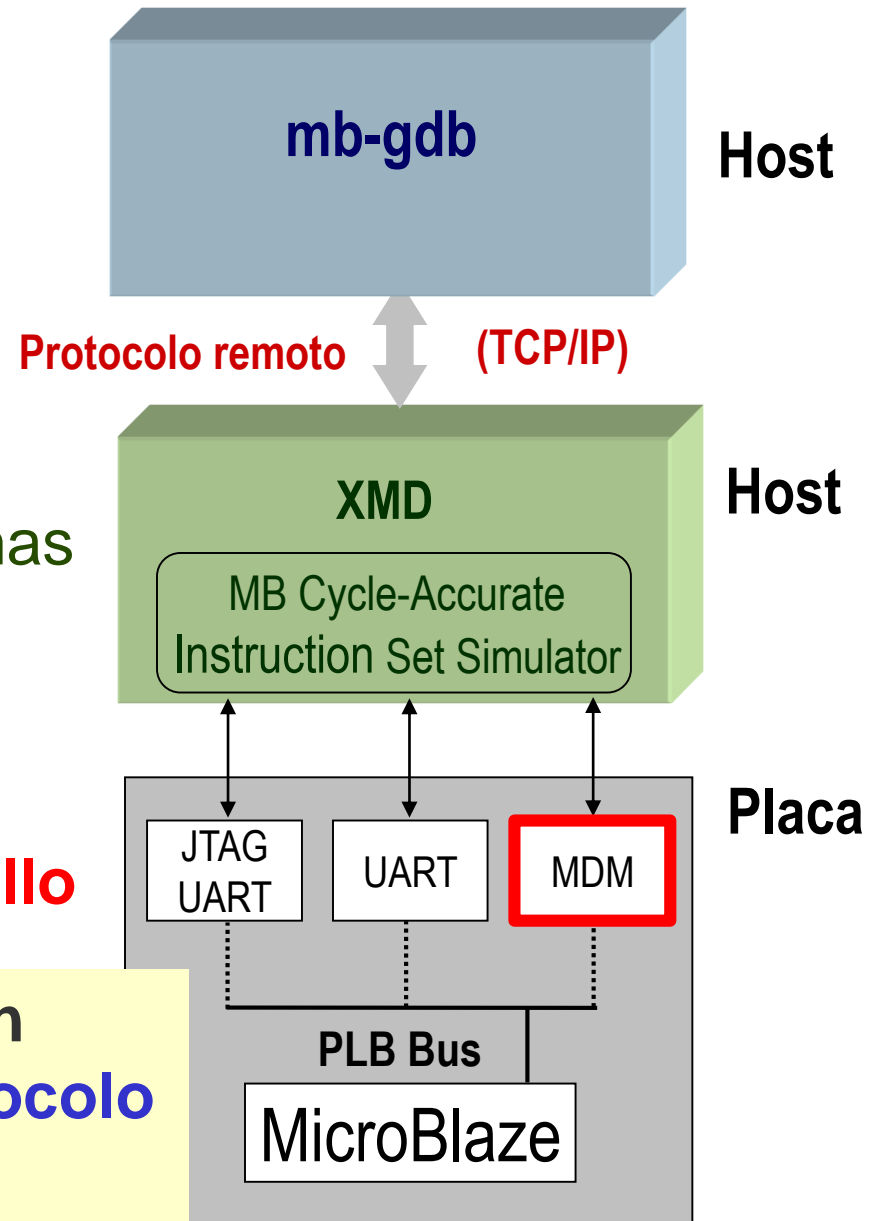
Herramienta típica de depuración simbólica de software

Xilinx Microprocessor debugger (XMD)

Facilita la depuración de programas mediante:

- **Simulación del conjunto de instrucciones**
- **Conexión a placa de desarrollo**

Ambas herramientas deben conectarse a través de un **protocolo remoto** basado en **TCP/IP**



Conexión de MDM a Microblaze

BEGIN microblaze

PARAMETER INSTANCE = microblaze_0

PARAMETER C_DEBUG_ENABLED = 1

...

BUS_INTERFACE DEBUG = microblaze_0_mdm_bus

END

BEGIN mdm

PARAMETER INSTANCE = mdm_0

PARAMETER C_MB_DBG_PORTS = 1

PARAMETER C_USE_UART = 1

...

BUS_INTERFACE SPLB = mb_plb

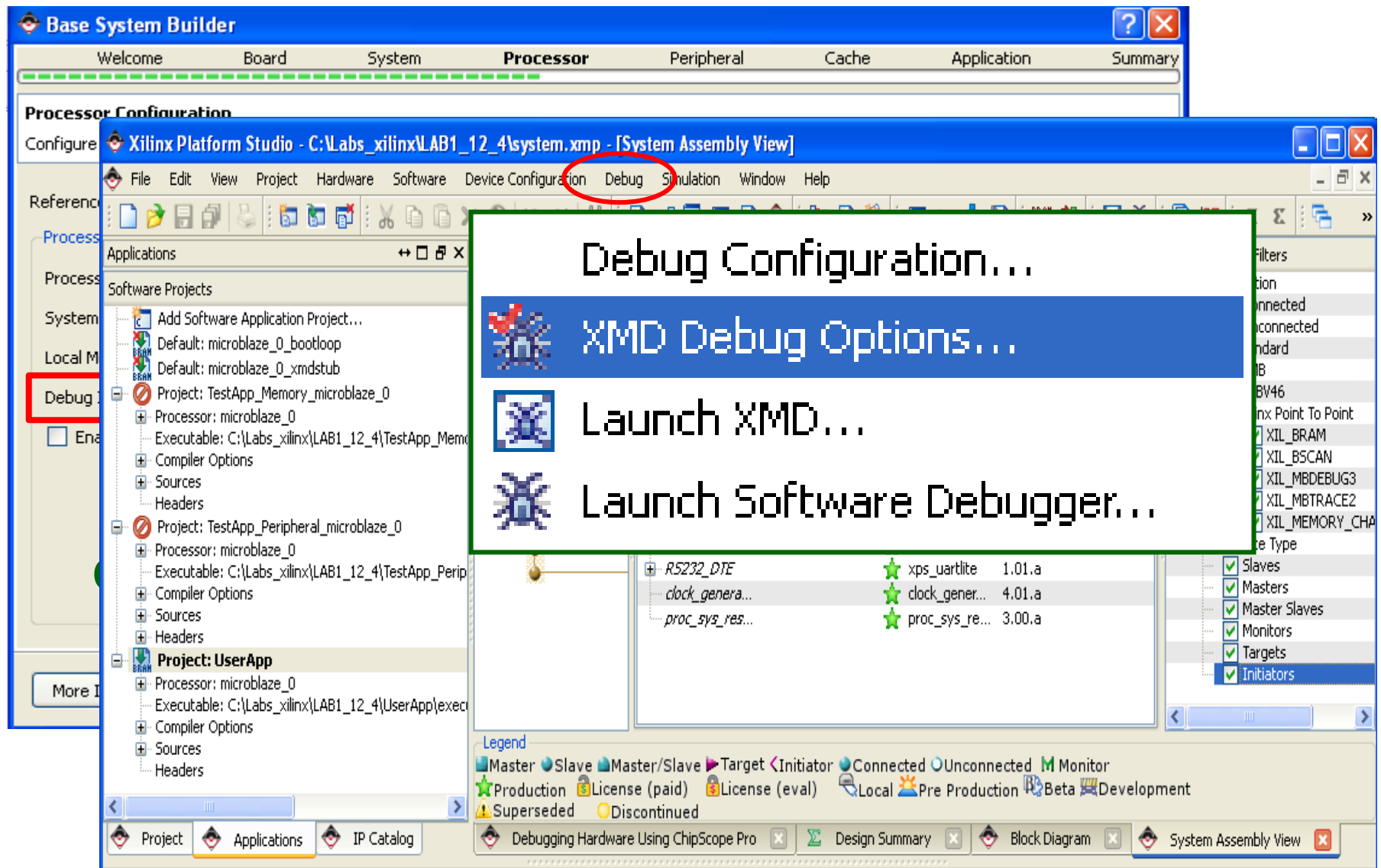
BUS_INTERFACE MBDEBUG_0 = microblaze_0_mdm_bus

END

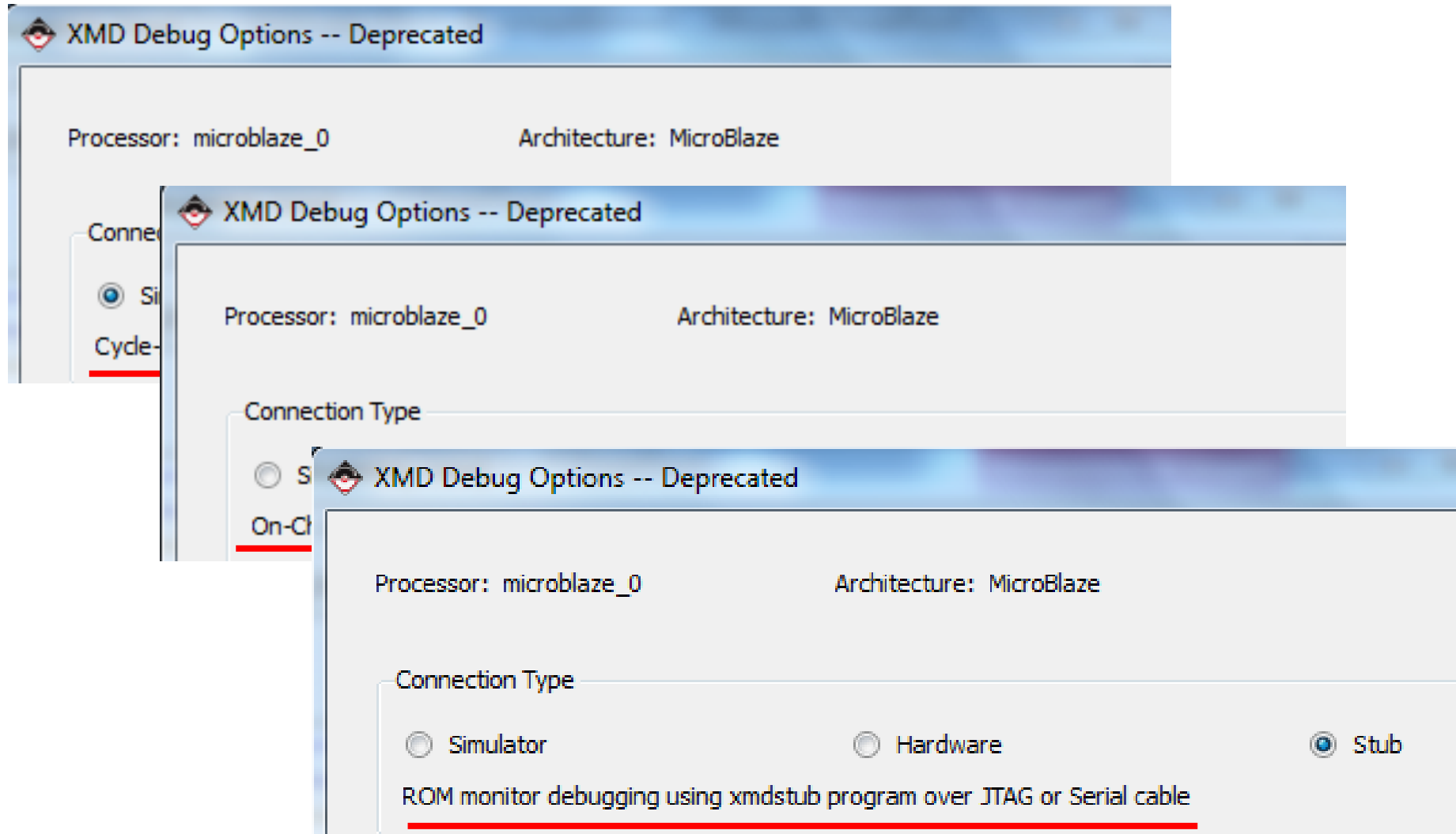
Xilinx Microprocessor Debugger (XMD)

- Facilita la depuración de programas mediante su simulación o ejecución controlada sobre una placa de desarrollo
- Proporciona:
 - Simulador de instrucciones (*Cycle-accurate Instruction Set Simulation*)
 - Mecanismo de conexión con el hardware
 - Una interfaz de comandos basada en TCL (*Tool Command Language*)
 - Un protocolo de comunicación remoto con GDB
- Admite tres modos de depuración (targets)
 - Simulación – sim
 - Hardware – mdm
 - Software – stub

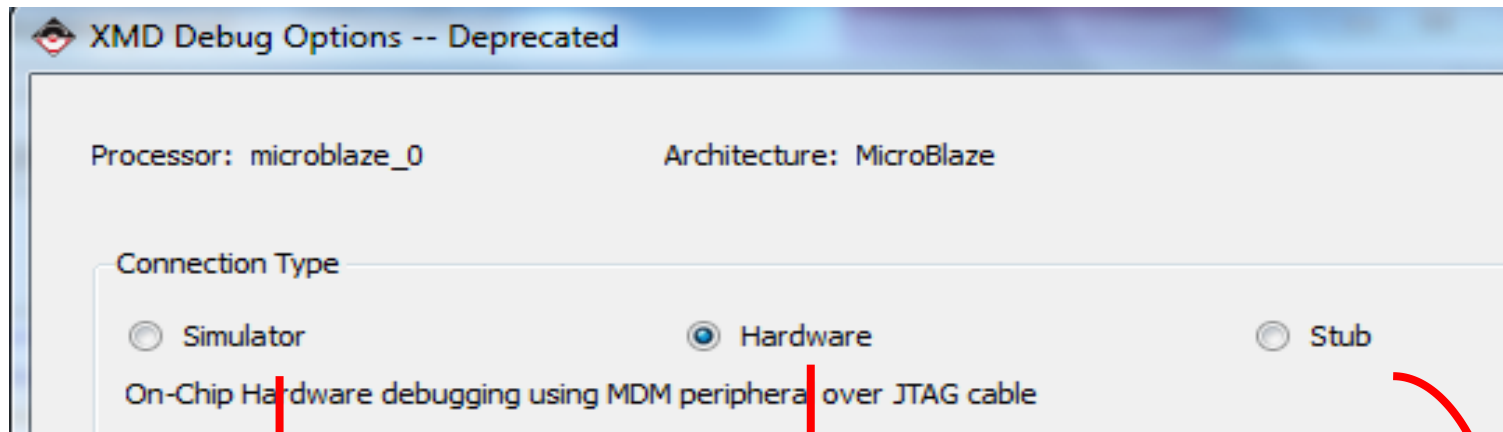
Modos de depuración de XMD



Modos de depuración de *XMD* (cont.)



Modos de depuración de *XMD* (cont.)



Basado en simulación:

- Mediante un **simulador de instrucciones (ISS)**
- No necesita conexión.

Basado en hardware:

- Mediante el módulo **MDM**
- Conexión a través de **JTAG**

Basado en software:

- Mediante **xmdstub** (un monitor ROM)
- Conexión a través de **UART** (basada en JTAG, basada en UART)
- En desuso (No habilitado en las últimas versiones de Xilinx)

MicroBlaze Simulator Target

XMD incorpora un simulador del conjunto de instrucciones (ISS) del procesador MicroBlaze

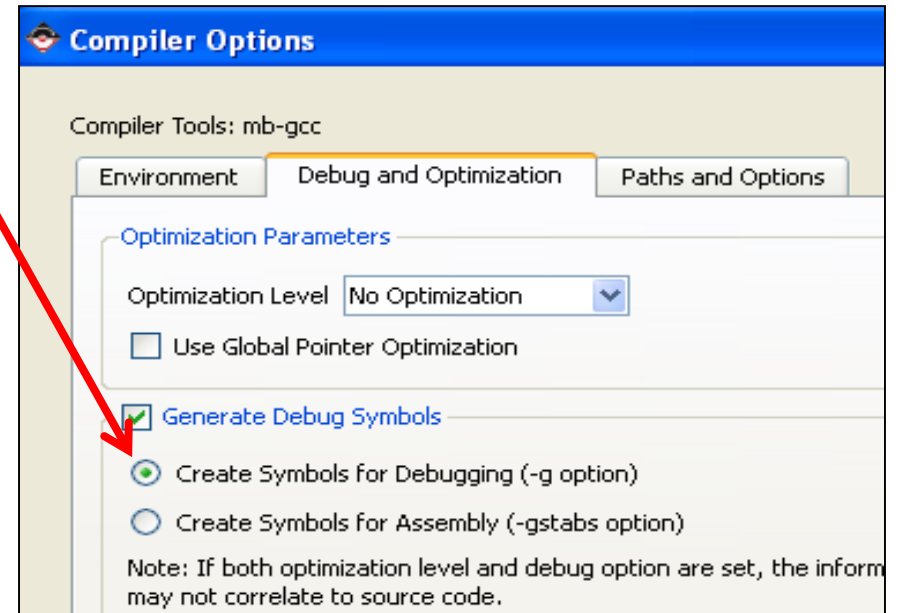
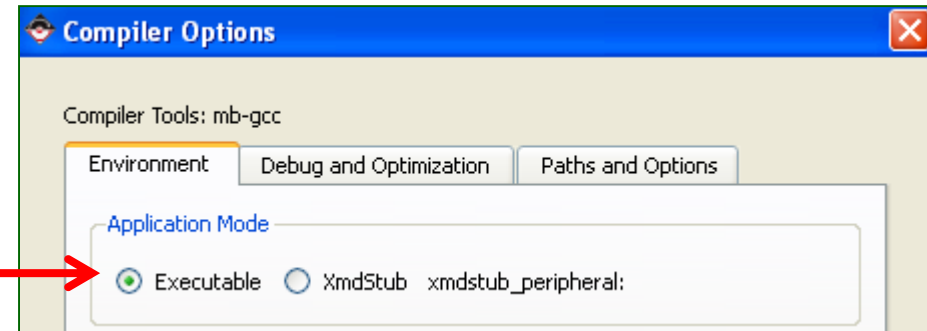
- **Requisitos:**

- Compilar en modo **ejecutable**

- Compilados con la opción **-g**

- **Limitaciones:**

- **No** soporta simulación de **periféricos**
- Tamaño máximo de programas **64 kB**



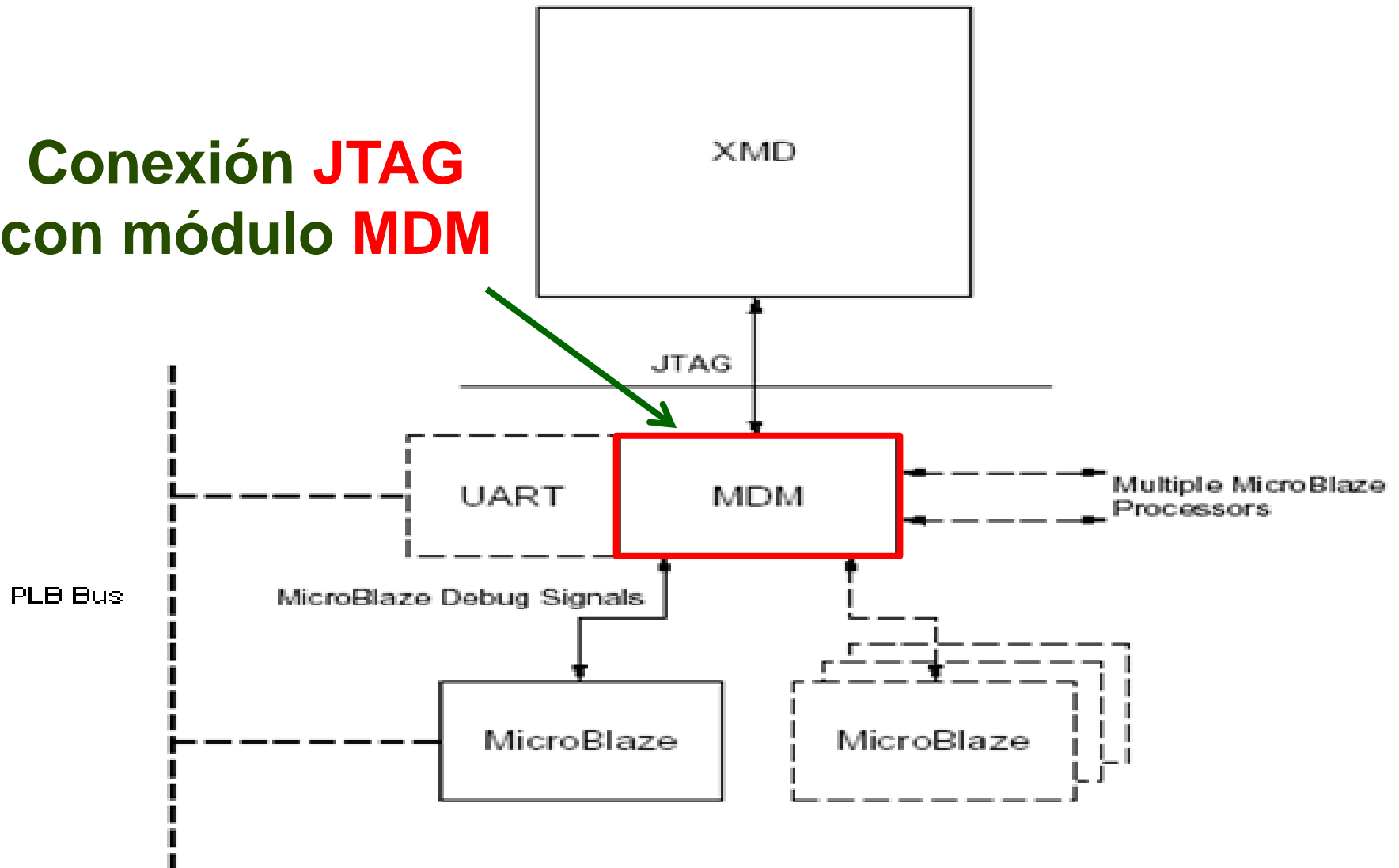
Microblaze HW Target

- ❑ Permite depurar los programas sobre el hardware sin tener que modificar el software
- Requisitos:
 - Incluir **MDM** en el hardware del sistema
 - Programas en modo **ejecutable**
 - **No** es preciso inicializar **xmdstub**
- Comando:
 - **XMD%** connect mb mdm



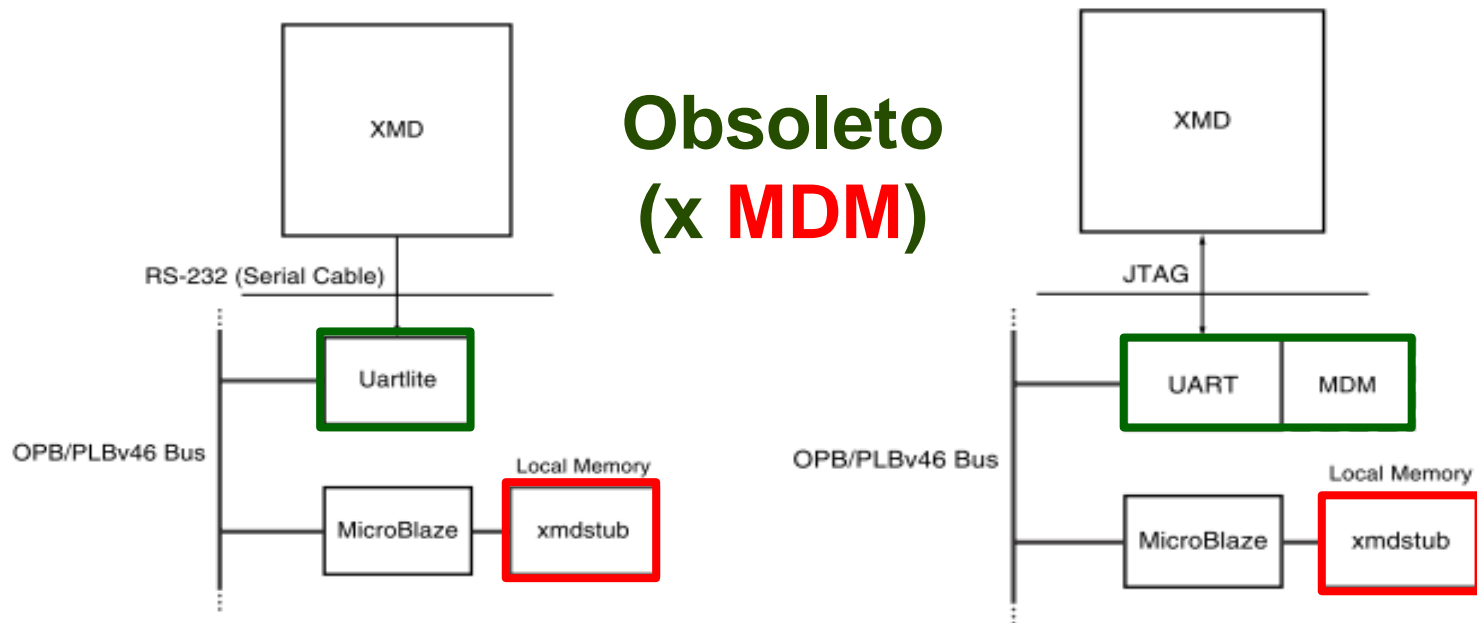
Microblaze HW Target (cont.)

Conexión JTAG
con módulo MDM



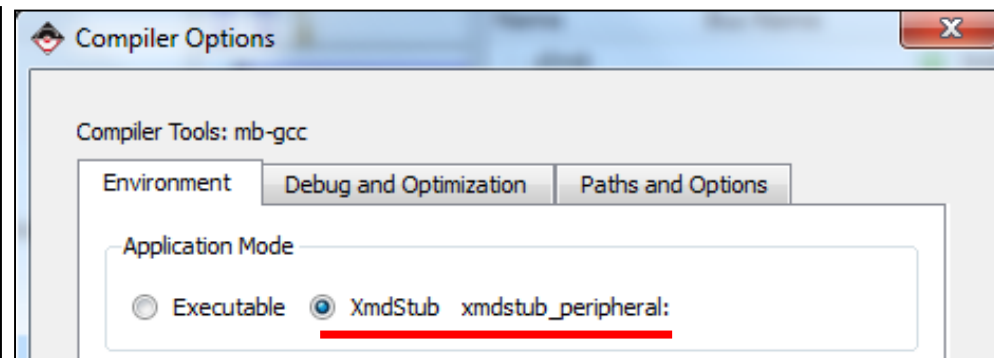
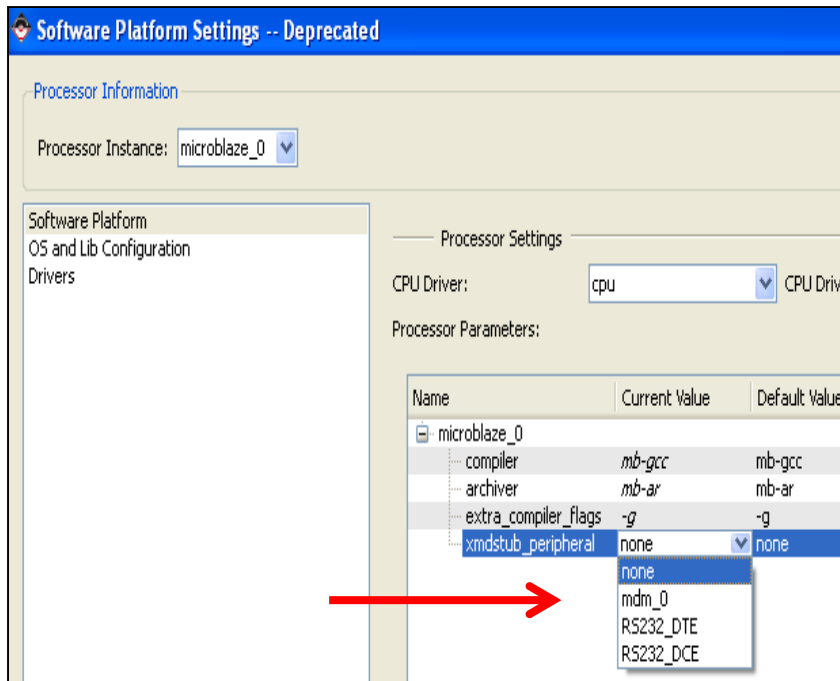
MicroBlaze Stub Target

- También permite depurar los programas directamente sobre el hardware **añadiendo SW específico (xmdstub)**
- **Requisitos:**
 - Incluir **Uartlite** o **MDM** en el hardware del sistema.



MicroBlaze Stub Target (cont.)

- Requisitos (cont.):
 - Definir **XMDSTUB_PERIPHERAL**
 - Programas en modo **xmdstub**



Obsoleto
(x MDM)



MicroBlaze Stub Target (cont.)

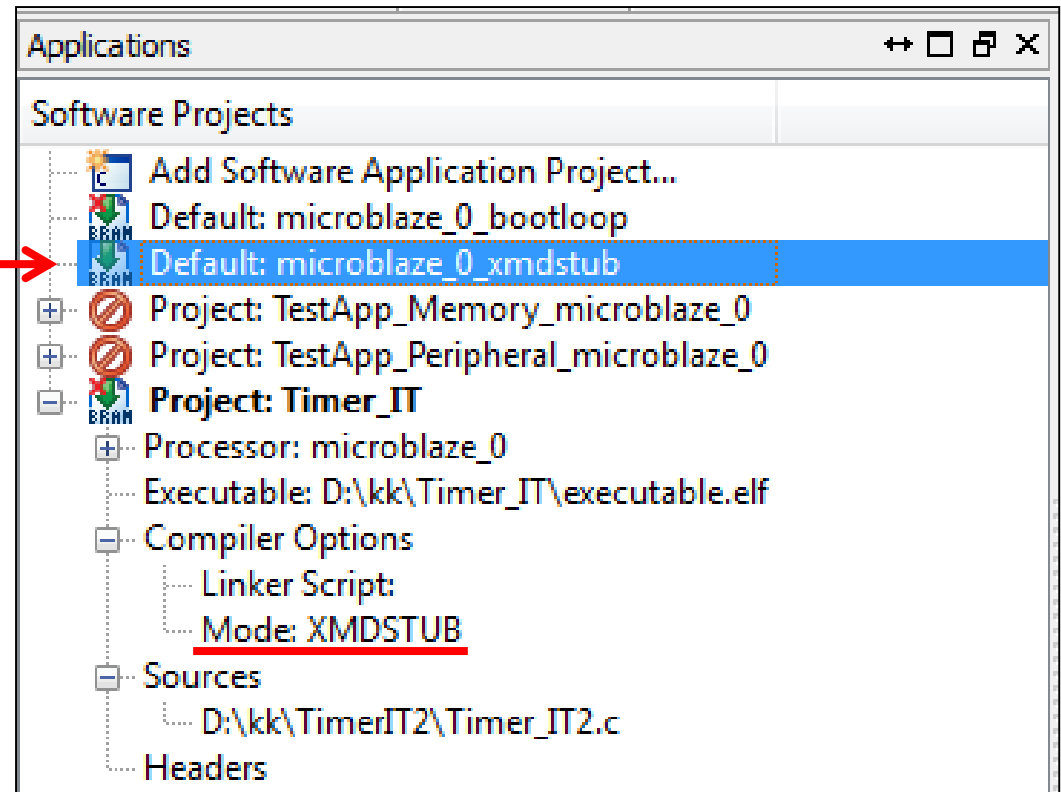
- **Requisitos (cont.):**

- Inicializar la memoria local con el ejecutable **xmdstub**

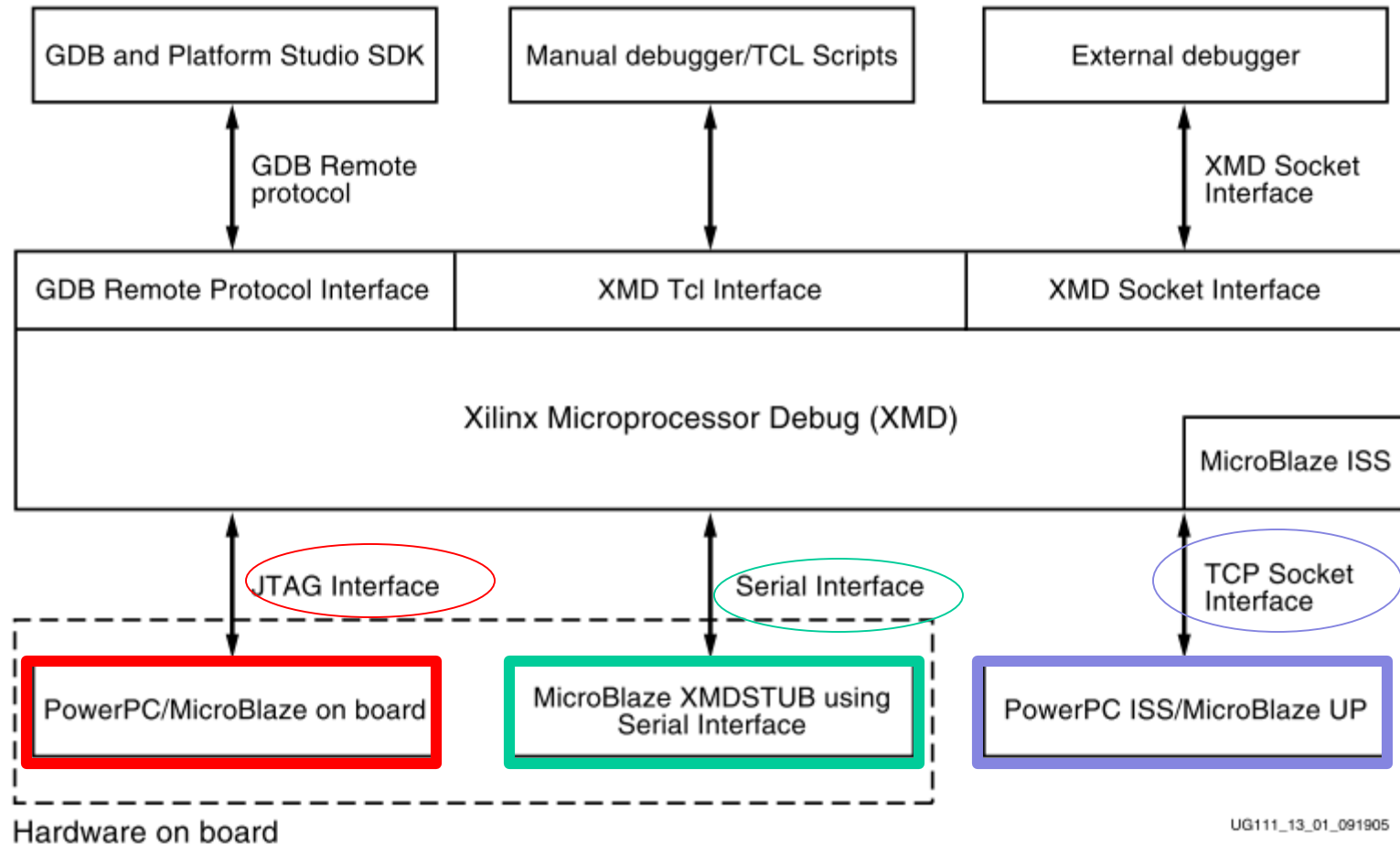
- **Comando:**

XMD% connect mb stub

**Obsoleto
(x MDM)**



XMD Targets



UG111_13_01_091905



Ejecución de *XMD*

- *XMD* se ejecuta desde una “Xilinx Bash Shell”
- Proporciona comandos para:
 - Conectar a un target específico
 - **connect mb** <sim|stub|mdm> [opciones], **disconnect...**
 - Inicializar y descargar programas
 - **dow** <fichero>, **xload** <xmp|mhs|mss> <fichero>
 - Establecer puntos de ruptura
 - **bps, bpr, bpl, ...**
 - Ejecutar programas
 - **run, con, stp, rst, stop, ...**
 - Ver y modificar registros y memoria
 - **rrd, rwr, mrd, mwr, ...**



Ejecución de XMD

The screenshot shows the Xilinx Platform Studio interface with the 'System Assembly View' active. The 'Ports' tab is selected, and the 'Launch XMD...' option is highlighted in the context menu. A red circle highlights the 'Launch XMD...' icon in the toolbar, and a red arrow points to the 'Launch XMD...' option in the context menu.

Table: microblaze_0's Address

Instance	Address	High Address	Size	Bus Interface(s)	Bus	
dlmb_cntlr	C_BASEADDR	0x00000000	0x00003FFF	16K	SLMB	dlm
ilmb_cntlr	C_BASEADDR	0x00000000	0x00003FFF	16K	SLMB	ilm
DDR2_SDRAM	C_MPMC_BASE...	0x40000000	0x43FFFFFF	64M	XCL0:XCL0_B	mic
xps_bram_if_cntlr_0	C_BASEADDR	0x50000000	0x50001FFF	8K	SPLB	mb
LEDs_8Bit	C_BASEADDR	0x81400000	0x8140FFFF	64K	SPLB	mb
DIPs_4Bit	C_BASEADDR	0x81420000	0x8142FFFF	64K	SPLB	mb
BTNs_4Bit	C_BASEADDR	0x81440000	0x8144FFFF	64K	SPLB	mb
RS232_DCE	C_BASEADDR	0x84000000	0x8400FFFF	64K	SPLB	mb
mdm_0	C_BASEADDR	0x84400000	0x8440FFFF	64K	SPLB	mb

Legend

- Master (blue square)
- Slave (green circle)
- Master/Slave (blue square with green circle)
- Target (purple triangle)
- Initiator (pink triangle)
- Connected (blue circle)
- Unconnected (light blue circle)
- Monitor (green M)
- Production (green star)
- License (paid) (yellow star)
- License (eval) (yellow star)
- Local (blue star)
- Pre Production (orange star)
- Beta (blue star)
- Development (blue star)
- Superseded (yellow triangle)
- Discontinued (yellow circle)

Ejecución de *XMD* (mdm)

The screenshot shows the Xilinx Microprocessor Debugger (XMD) console. The title bar indicates the path: C:\Xilinx\12.4\ISE_DS\EDK\bin\nt\xbash.exe. The console output includes copyright information, platform configuration details, JTAG chain configuration, and MicroBlaze processor configuration. Red arrows point from yellow callout boxes to specific parts of the console output.

Información del HW

Checking platform configuration...
IPNAME:plb_v46 INSTANCE:master(s) : 4 slave(s)
IPNAME:lmb_v10 INSTANCE:master(s) : 1 slave(s)
IPNAME:lmb_v10 INSTANCE:master(s) : 1 slave(s)

Identifica la cadena JTAG

Device	ID Code	IR Length	Part Name
1	22228093	6	XC3S700A
2	f5046093	8	XCF04S

Configuración de MicroBlaze

MicroBlaze Processor Configuration :
Version.....8.00.b
Optimization.....Area
Interconnect.....PLB_v46
MMU Type.....No_MMU
No of PC Breakpoints.....1
No of Read Addr/Data Watchpoints...0
No of Write Addr/Data Watchpoints..0
Instruction Cache Support.....off
Data Cache Support.....off
Exceptions Support.....off
FPU Support.....off
Hard Divider Support.....off
Hard Multiplier Support.....on
Barrel Shifter Support.....on
MSR clr/set Instruction Support...on
Compare Instruction Support.....on
Data Cache Write-back Support.....off

Inicia servidor GDB

Connected to "mb" target. id = 0
Starting GDB server for "mb" target (id = 0) at TCP port no 1234
XMD% _

Prompt de XMD

Descarga y ejecución de programas

```
C:\Xilinx\12.4\ISE_DS\EDK\bin\nt\xbash.exe
XMD% dow UserApp/executable.elf
Downloading Program -- UserApp/executable.elf
section, .vectors.reset: 0x00000000-0x00000003
section, .vectors.sw_exception: 0x00000008-0x0000000b
section, .vectors.interrupt: 0x00000010-0x00000013
section, .vectors.hw_exception: 0x00000000
section, .text: 0x00000050-0x0000116b
section, .init: 0x0000116c-0x0000118f
section, .fini: 0x00001190-0x000011ab
section, .ctors: 0x000011ac-0x000011b3
section, .dtors: 0x000011b4-0x000011bb
section, .rodata: 0x000011bc-0x00001625
section, .data: 0x00001628-0x0000174f
section, .eh_frame: 0x00001750-0x0000175f
section, .jcr: 0x00001754-0x00001757
section, .bss: 0x00001758-0x00001797
section, .stack: 0x00001798-0x00001b97
Setting PC with Program Start Address 0x00000000
System Reset .... DONE
XMD% rrd
r0: 00000000    r8: 00001780    r16: 00000000    r24: 00000000
r1: 00001b4c    r9: 00000002    r17: 00000000    r25: 00000000
r2: 00001628    r10: 00000000    r18: 0098967f    r26: 00000000
r3: 00112939    r11: 00000000    r19: 00001b4c    r27: 00000000
r4: 00000008    r12: 00000000    r20: 00000000    r28: 00000000
r5: 00001780    r13: 00001758    r21: 00000000    r29: 00000000
r6: 00000001    r14: 00000000    r22: 00000000    r30: 00000000
r7: 0000000d    r15: 00000220    r23: 00000000    r31: 00000000
pc: 00000000    msr: 00000000
XMD% con
Processor started. Type "stop" to stop processor
RUNNING> XMD% stop
XMD% User Interrupt, Processor Stopped at 0x00000244
XMD%
```

Cargar programa
Visualizar registros
Continuar ejecución
Detener procesador

XMD como terminal serie

- **MDM** puede ser configurado como una interfaz serie
 - **PARAMETER C_USE_UART = 1**
 - Comunicación a través de **JTAG**
 - Las funciones son las mismas del **UARTLite**
- Puede configurarse **MDM** como E/S estándar
- **XMD** puede ser configurado como terminal serie
 - Conexión **JTAG**
 - Comando: **terminal -jtag_uart_server**
 - Ej: ***xil_printf*** ("-- Ejecutando main() --\r\n");



GNU Debugger (*mb-gdb*)

- Depurador simbólico (*source-code debugger*) para C y C++
- Facilita la **depuración del software** al proporcionar, a través de una **interfaz gráfica de usuario**, mecanismos para :
 - Cargar y ejecutar programas
 - Establecer puntos de ruptura (*breakpoints*)
 - Examinar el contenido de: **Registros, Memoria, Stack, Variables, Expresiones**
 - **Modificar el programa** con objeto de provocar determinadas situaciones o corregir el efecto de un error de código

No corre sobre Windows7 +

Ejecución de *mb-gdb*

- mb-gdb* puede lanzarse desde una Shell de comandos o desde la interfaz gráfica de *XPS*.

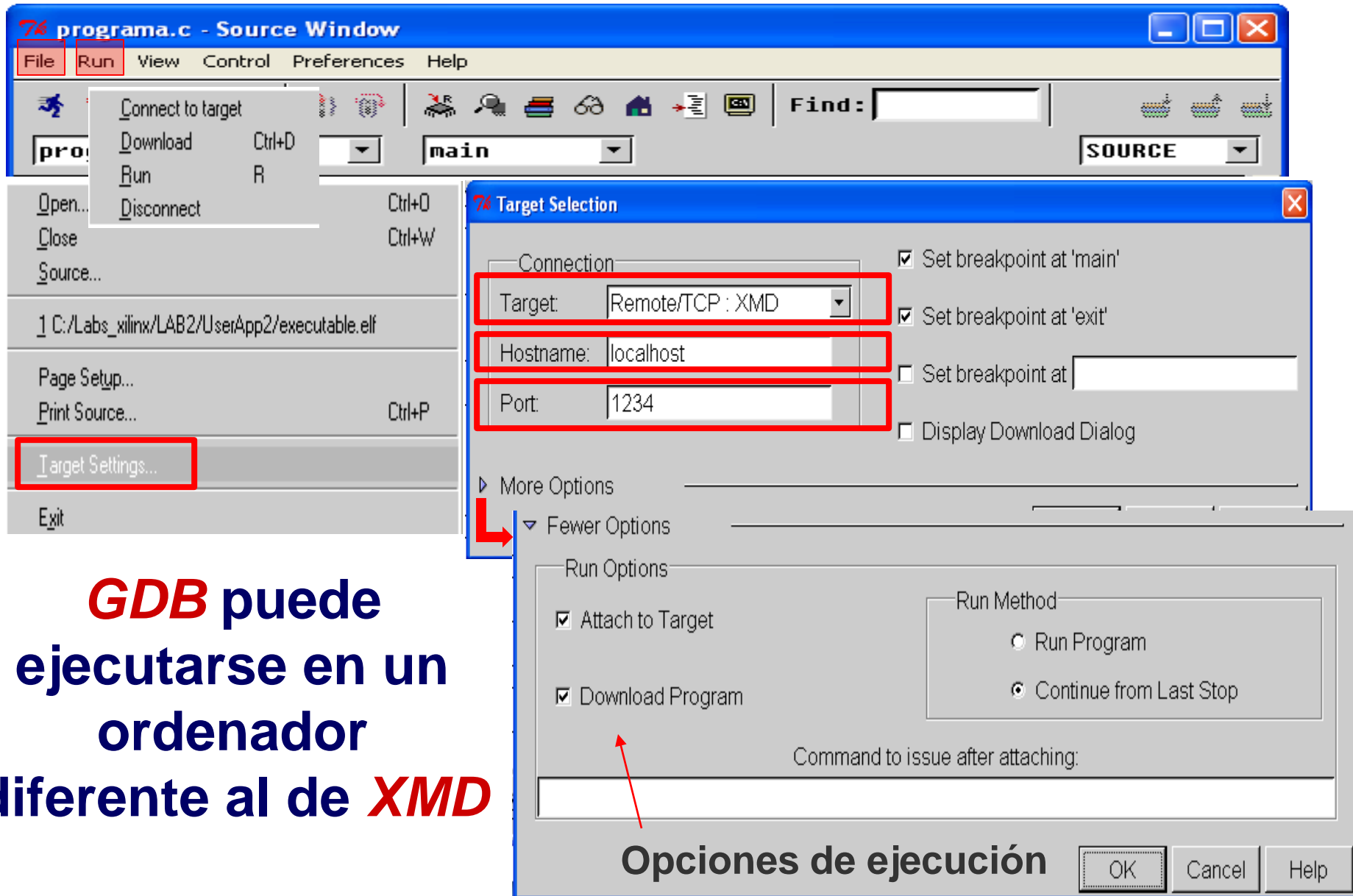
The screenshot shows the Xilinx Platform Studio interface. The 'Debug' menu is open, and the 'Launch Software Debugger...' option is highlighted with a red rectangle. A red circle highlights the 'Launch Software Debugger...' icon in the toolbar. The 'programa.c - Source Window' is open, showing the following C code:

```
30 Status = XGpio_Initialize(&leds, XPAR_LEDS_8BIT_DEVICE_ID);
31 if (Status != XST_SUCCESS) print ("GPIO ERROR: error de inicializa
32
33 // Se establecen los puertos de salida del GPIO de los leds
34 XGpio_SetDataDirection(&leds, 1, 0x00000000);
35
36 // Bucle del programa principal
37 print("***** Inicio del programa\r\n");
38
39 contador=0;
40
41 while(1){
42
43     // Se escribe en los leds
44     XGpio_DiscreteWrite(&leds, 1, contador);
45     for (j=1; j<100000000; j++); // tiempo de espera
46
47     xil_printf("**** contador= %d\r\n", contador);
48
49     if (contador < 256) contador++;
50     else contador=0;
51
52 }
53 }
```

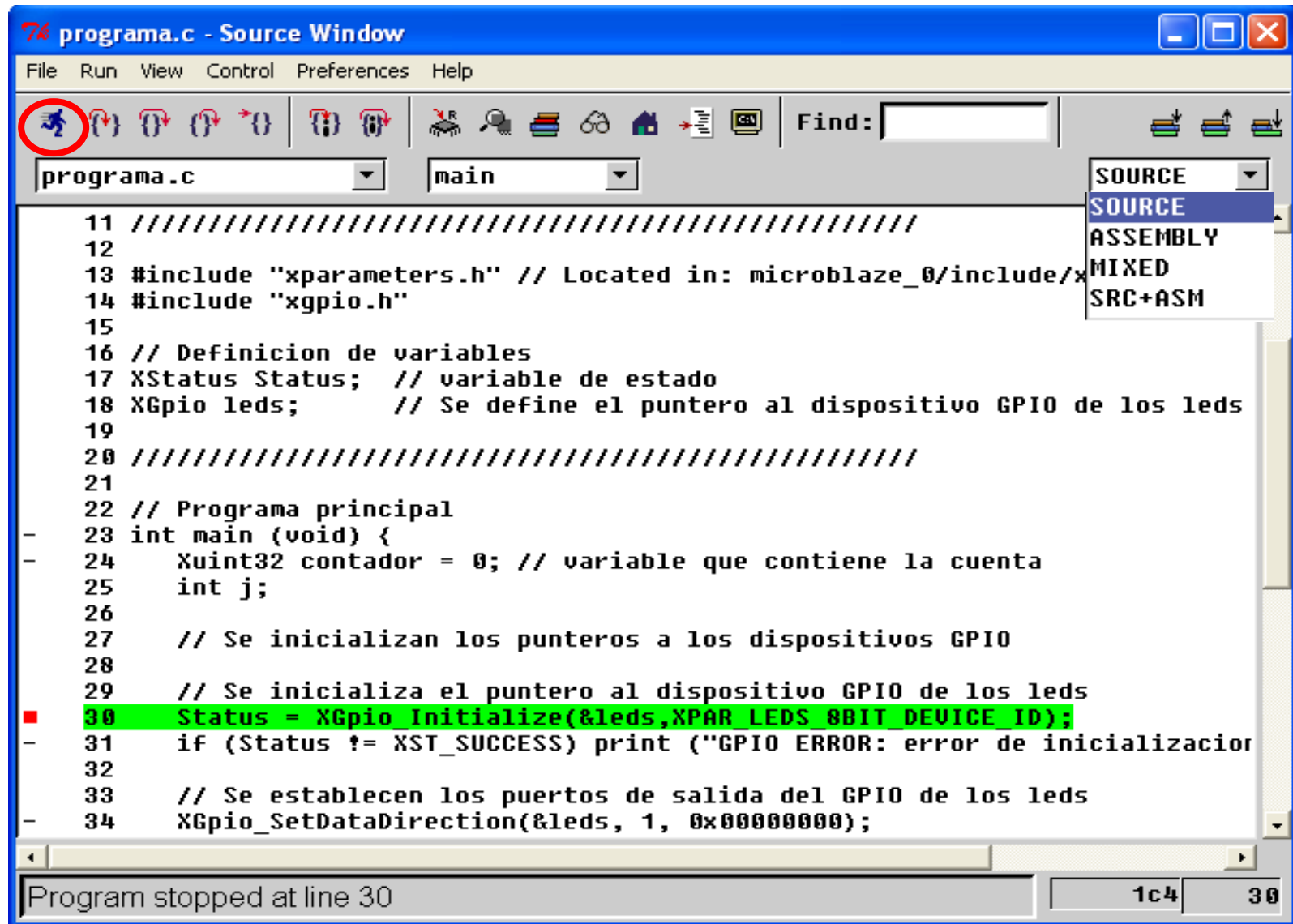
At the bottom of the window, a status bar indicates: "Program not running. Click on run icon to start." The page number "1c4" and "30" are visible in the bottom right corner.

¡ Previamente debe haberse iniciado el servidor GDB mediante *XMD* !

Ejecución de *mb-gdb*



Visualización de código fuente



74 programa.c - Source Window

File Run View Control Preferences Help

programa.c main

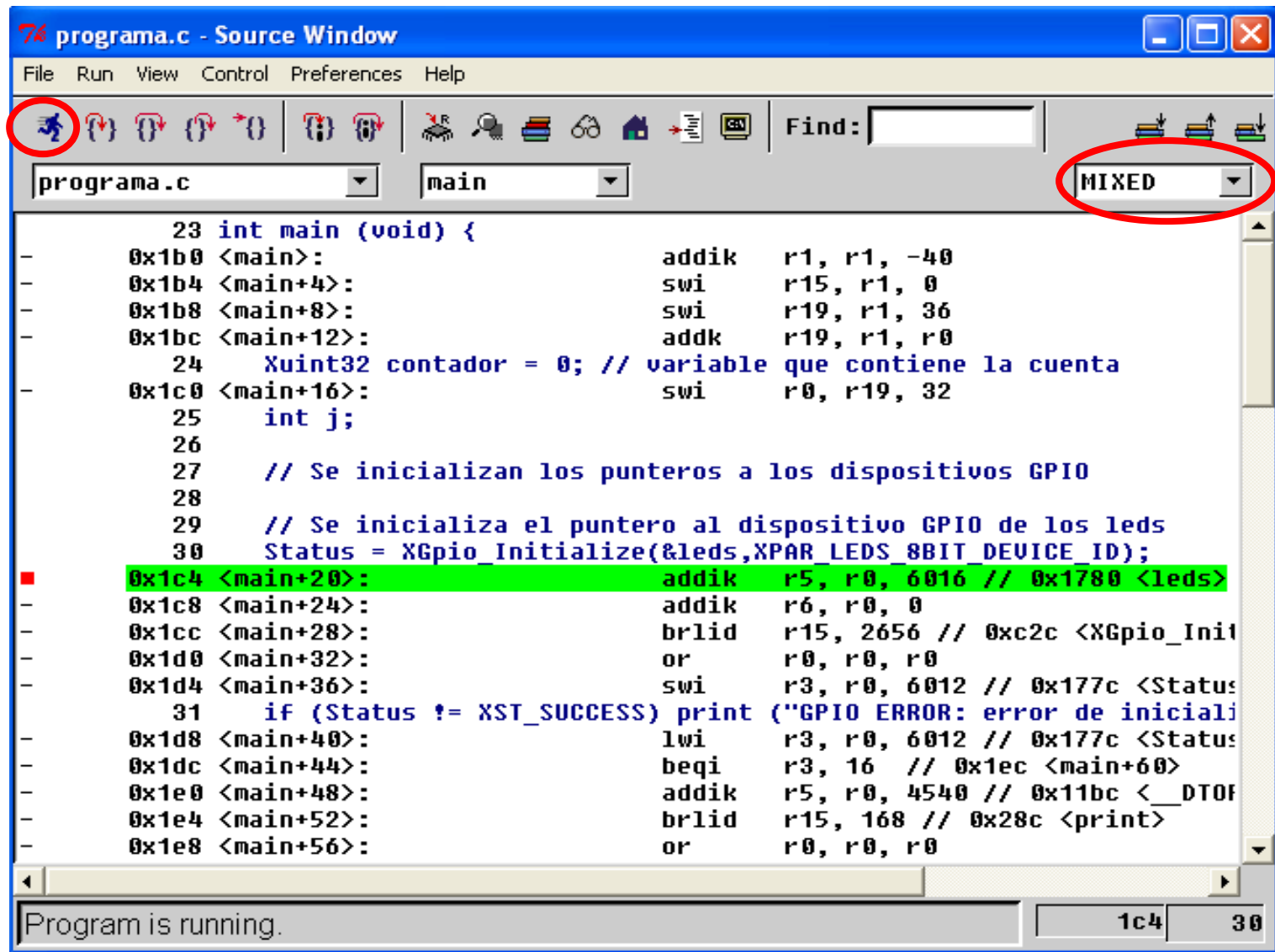
SOURCE
SOURCE
ASSEMBLY
MIXED
SRC+ASM

```
11 //////////////////////////////////////////////////
12
13 #include "xparameters.h" // Located in: microblaze_0/include/x
14 #include "xgpio.h"
15
16 // Definicion de variables
17 XStatus Status; // variable de estado
18 XGpio leds; // Se define el puntero al dispositivo GPIO de los leds
19
20 //////////////////////////////////////////////////
21
22 // Programa principal
23 int main (void) {
24     Xuint32 contador = 0; // variable que contiene la cuenta
25     int j;
26
27     // Se inicializan los punteros a los dispositivos GPIO
28
29     // Se inicializa el puntero al dispositivo GPIO de los leds
30     Status = XGpio_Initialize(&leds, XPAR_LEDS_8BIT_DEVICE_ID);
31     if (Status != XST_SUCCESS) print ("GPIO ERROR: error de inicializacio
32
33     // Se establecen los puertos de salida del GPIO de los leds
34     XGpio_SetDataDirection(&leds, 1, 0x00000000);
```

Program stopped at line 30

1c4 30

Visualización de código fuente



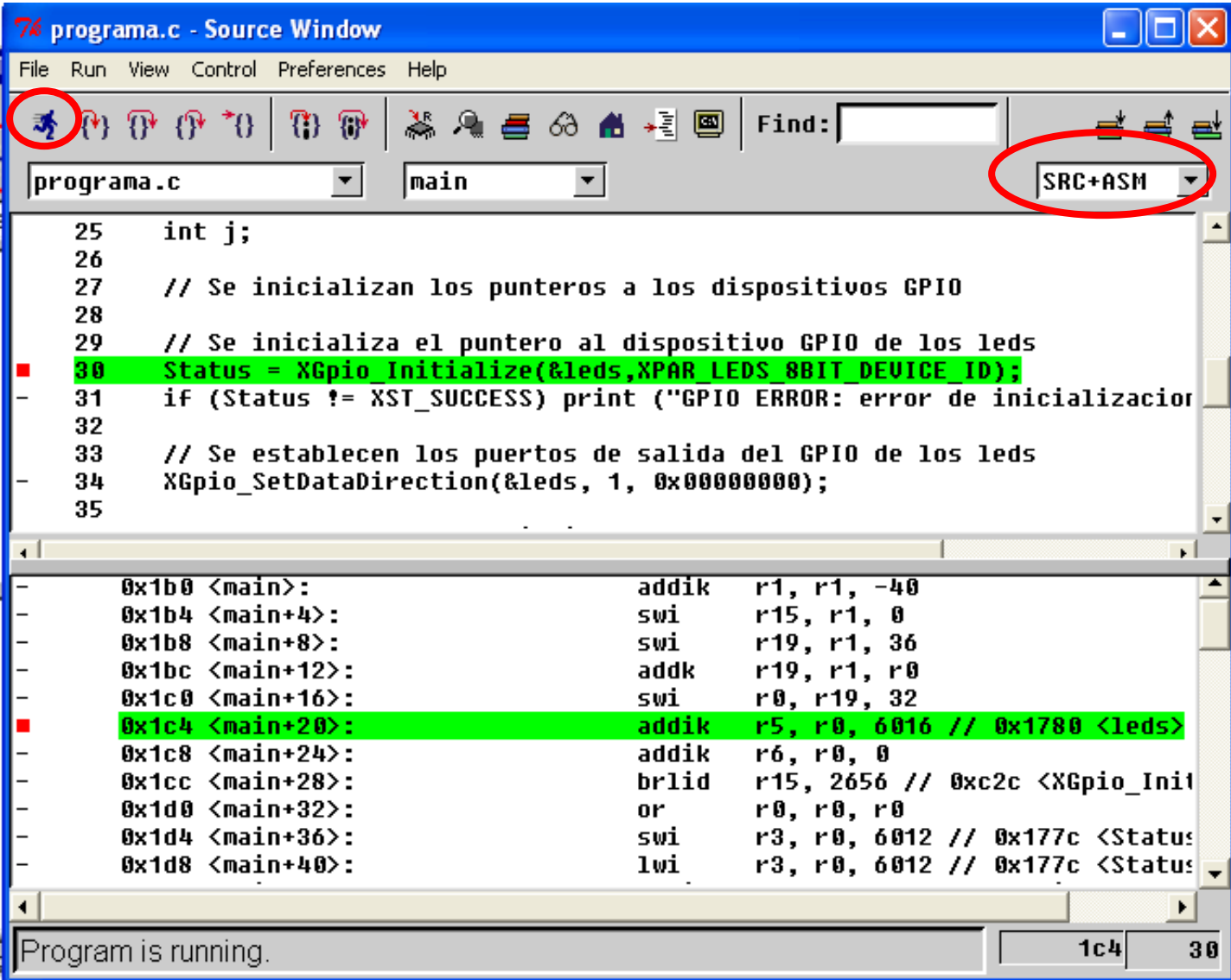
```
programa.c - Source Window
File Run View Control Preferences Help

programa.c main MIXED

23 int main (void) {
- 0x1b0 <main>:          addik    r1, r1, -40
- 0x1b4 <main+4>:        swi      r15, r1, 0
- 0x1b8 <main+8>:        swi      r19, r1, 36
- 0x1bc <main+12>:       addk     r19, r1, r0
24     Xuint32 contador = 0; // variable que contiene la cuenta
- 0x1c0 <main+16>:       swi      r0, r19, 32
25     int j;
26
27     // Se inicializan los punteros a los dispositivos GPIO
28
29     // Se inicializa el puntero al dispositivo GPIO de los leds
30     Status = XGpio_Initialize(&leds, XPAR_LEDS_8BIT_DEVICE_ID);
- 0x1c4 <main+20>:       addik    r5, r0, 6016 // 0x1780 <leds>
- 0x1c8 <main+24>:       addik    r6, r0, 0
- 0x1cc <main+28>:       brlid    r15, 2656 // 0xc2c <XGpio_Init
- 0x1d0 <main+32>:       or       r0, r0, r0
- 0x1d4 <main+36>:       swi      r3, r0, 6012 // 0x177c <Status
31     if (Status != XST_SUCCESS) print ("GPIO ERROR: error de iniciali
- 0x1d8 <main+40>:       lwi      r3, r0, 6012 // 0x177c <Status
- 0x1dc <main+44>:       beqi     r3, 16 // 0x1ec <main+60>
- 0x1e0 <main+48>:       addik    r5, r0, 4540 // 0x11bc <__DTOF
- 0x1e4 <main+52>:       brlid    r15, 168 // 0x28c <print>
- 0x1e8 <main+56>:       or       r0, r0, r0

Program is running. 1c4 30
```

Visualización de código fuente



The screenshot shows a source code editor window titled "programa.c - Source Window". The menu bar includes File, Run, View, Control, Preferences, and Help. The toolbar contains various icons, with the first icon (a blue bug) circled in red. Below the toolbar, there are dropdown menus for the file name "programa.c" and the function "main". To the right of these is a "Find:" text box and a dropdown menu labeled "SRC+ASM", which is also circled in red. The main text area displays C code with line numbers 25 to 35. Line 30 is highlighted in green. Below the C code, the assembly code is shown, with line 0x1c4 highlighted in green. The status bar at the bottom indicates "Program is running." and shows the current line and column as "1c4" and "30".

```
25  int j;
26
27  // Se inicializan los punteros a los dispositivos GPIO
28
29  // Se inicializa el puntero al dispositivo GPIO de los leds
30  Status = XGpio_Initialize(&leds, XPAR_LEDS_0BIT_DEVICE_ID);
31  if (Status != XST_SUCCESS) print ("GPIO ERROR: error de inicializaci
32
33  // Se establecen los puertos de salida del GPIO de los leds
34  XGpio_SetDataDirection(&leds, 1, 0x00000000);
35
```

```
0x1b0 <main>:      addik    r1, r1, -40
0x1b4 <main+4>:    swi      r15, r1, 0
0x1b8 <main+8>:    swi      r19, r1, 36
0x1bc <main+12>:   addk     r19, r1, r0
0x1c0 <main+16>:   swi      r0, r19, 32
0x1c4 <main+20>:   addik    r5, r0, 6016 // 0x1780 <leds>
0x1c8 <main+24>:   addik    r6, r0, 0
0x1cc <main+28>:   brlid    r15, 2656 // 0xc2c <XGpio_Init
0x1d0 <main+32>:   or       r0, r0, r0
0x1d4 <main+36>:   swi      r3, r0, 6012 // 0x177c <Status
0x1d8 <main+40>:   lwi      r3, r0, 6012 // 0x177c <Status
```

Program is running. 1c4 30

Control de ejecución

programa.c - Source Window

File Run View Control Preferences Help

Find:

programa.c main SRC+ASM

```
25 int j;  
26  
27 // Se inicializan los punteros a los dispositivos GPIO  
28  
29 // Se inicializa el puntero al dispositivo GPIO de los leds  
30 Status = XGpio_Initialize(&leds, XPAR_LEDS_8BIT_DEVICE_ID);  
31 if (Status != XST_SUCCESS) print ("GPIO ERROR: error de inicializaci  
32  
33 // Se establecen los puertos de salida del GPIO de los leds  
34 XGpio_SetDataDirection(&leds, 1, 0x00000000);  
35
```

Run

- Connect to target
- Download Ctrl+D
- Run R
- Disconnect

Control

- Step S
- Next N
- Finish F
- Continue C

Program is running.

Visualización de información

74 programa.c - Source Window

File Run **View** Control Preferences Help

Stack Ctrl+S
Registers Ctrl+R
Memory Ctrl+M
Watch Expressions Ctrl+T
Local Variables Ctrl+L
Breakpoints Ctrl+B
Console Ctrl+N

74 Memory

Addresses

Address 0x001628 Target is BIG endian

	0	4	8	C	ASCII
0x00001628	0x00000000	0x00000000	0x000011b8	0x00000d20
0x00001638	0x00000000	0x00000001	0x00000001	0x00000000
0x00001648	0x81400000	0x00000000	0x00000000	0x00001520	.@.....
0x00001658	0x0000165c	0x00000000	0x00000000	0x00000000	... \.....
0x00001668	0x00000000	0x00000000	0x00000000	0x00000000
0x00001678	0x00000000	0x00001624	0x00000000	0x00000000\$......
0x00001688	0x00000000	0x00000000	0x00000000	0x00000000
0x00001698	0x00000000	0x00000000	0x00000000	0x00000000

74 Registers

Group: all

r0	0x0	r16	0x0	rpc
r1	0x1b4c	r17	0x0	rms
r2	0x1628	r18	0x0	reap
r3	0xffffffff	r1	0x0	
r4	0x0	r2	0x0	
r5	0x0	r2	0x0	
r6	0x0	r2	0x0	
r7	0x0	r2	0x0	
r8	0x1780	r2	0x0	
r9	0x1	r2	0x0	
r10	0x0	r2	0x0	
r11	0x0	r2	0x0	
r12	0x0	r28	0x0	rpor6
r13	0x1758	r29	0x0	rpor7
r14	0x0	r30	0x0	rpor8
r15	0x184	r31	0x0	rpor9

74 Local Variables

```
contador = (Xuint32) 0  
j = (int) 6988
```

74 Breakpoints

Breakpoint Global

	Address	File	Line	Function
<input checked="" type="checkbox"/>	d64		0	exit
<input checked="" type="checkbox"/>	1c4	programa.c	30	main

Herramientas de verificación de Xilinx

Debug Configuration

System

- Monitor Hardware Signals
- Debug Software Application
 - microblaze_0
- Miscellaneous
 - JTAG UART

Basic

Monitor Hardware Signals

The Xilinx® Embedded Development Kit (EDK) provides following Xilinx® ChipScope™ Pro cores for hardware debugging:

- chipscope_icon — Provides communication to other ChipScope cores
- chipscope_plbv46_iba — Facilitates monitoring of Processor Local Bus (PLB) transactions
- chipscope_ila — Facilitates monitoring individual non-bus signals in the processor design
- chipscope_vio — Facilitates virtual I/O to probe FPGA signals via JTAG

Information:

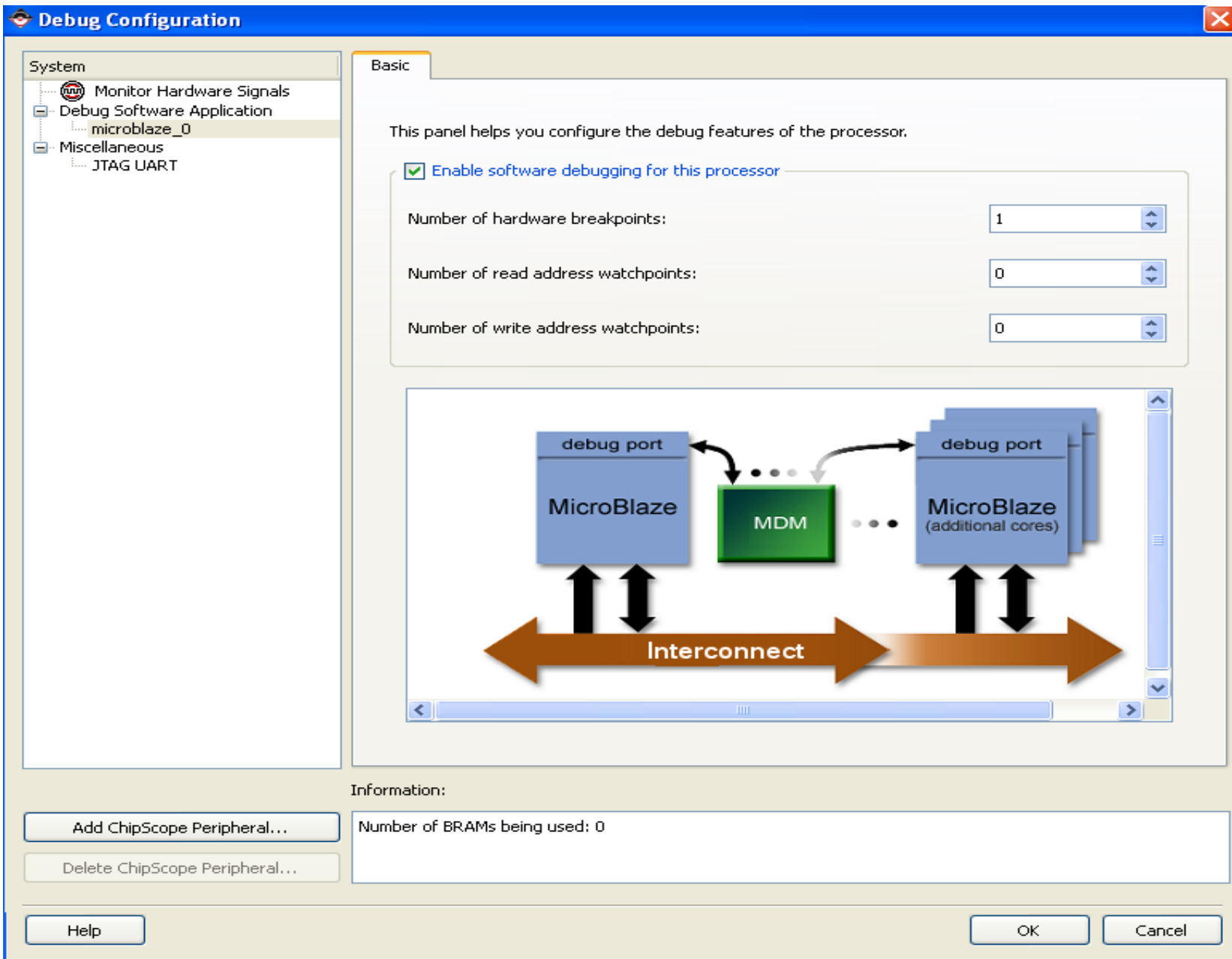
Add ChipScope Peripheral...

Delete ChipScope Peripheral...

Help

Number of BRAMs being used: 0

OK Cancel



Debug Configuration

System

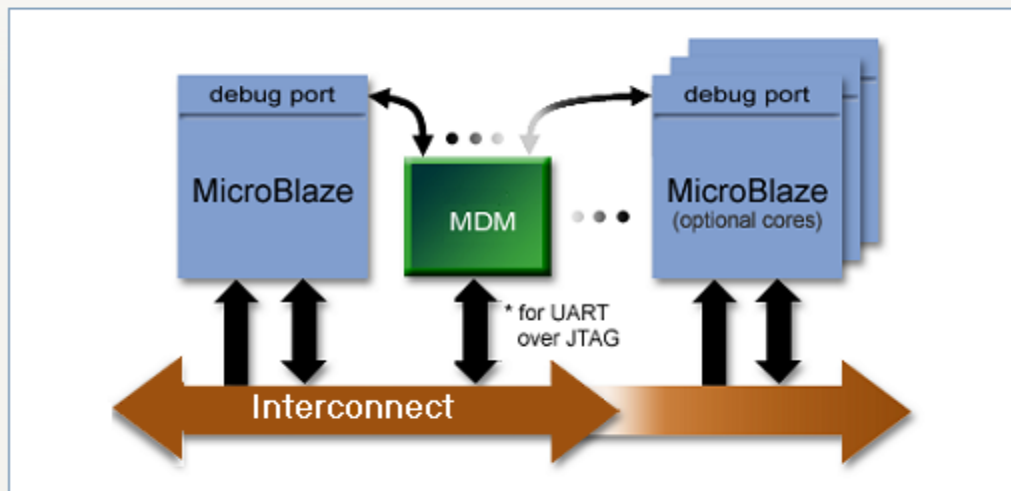
- Monitor Hardware Signals
- Debug Software Application
 - microblaze_0
- Miscellaneous
 - JTAG UART

Basic

Configure the JTAG interface for a specific processor, so that the UART signals can be transported over JTAG ports to the Xilinx Microprocessor Debugger (XMD) tool.

☒ Enable JTAG UART interface for this processor: microblaze_0

Note: Only one processor can use this feature, and that processor must be able to access MDM through bus.



Information:

Add ChipScope Peripheral...

Delete ChipScope Peripheral...

Number of BRAMs being used: 0

Help

OK

Cancel

Documentación

○ Manuales

- *Platform Studio User Guide → Debugging in EDK*
- *Embedded System Tools Ref. Manual → GNU Debugger*
- *Embedded System Tools Ref. Manual → Xilinx Microprocessor Debugger (XMD)*
- *Xilinx Drivers*

○ Soporte Web

- EDK
 - <http://www.support.xilinx.com/edk>

