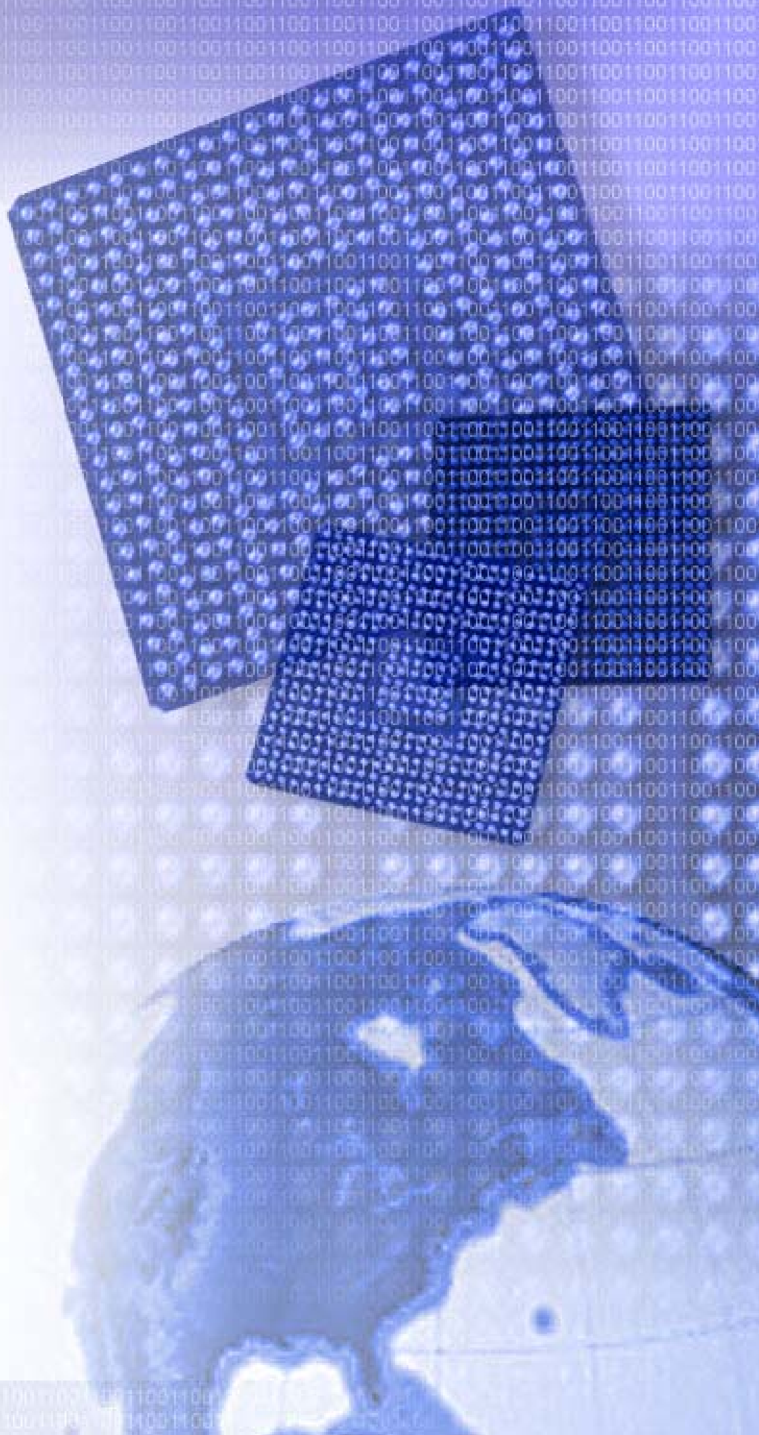
# PicoBlaze™

for Spartan-3, Virtex-II, Virtex-IIPRO and Virtex-4 devices

# JTAG Loader

*Quick User Guide*

***Kris Chaplin*** ***and Ken Chapman***

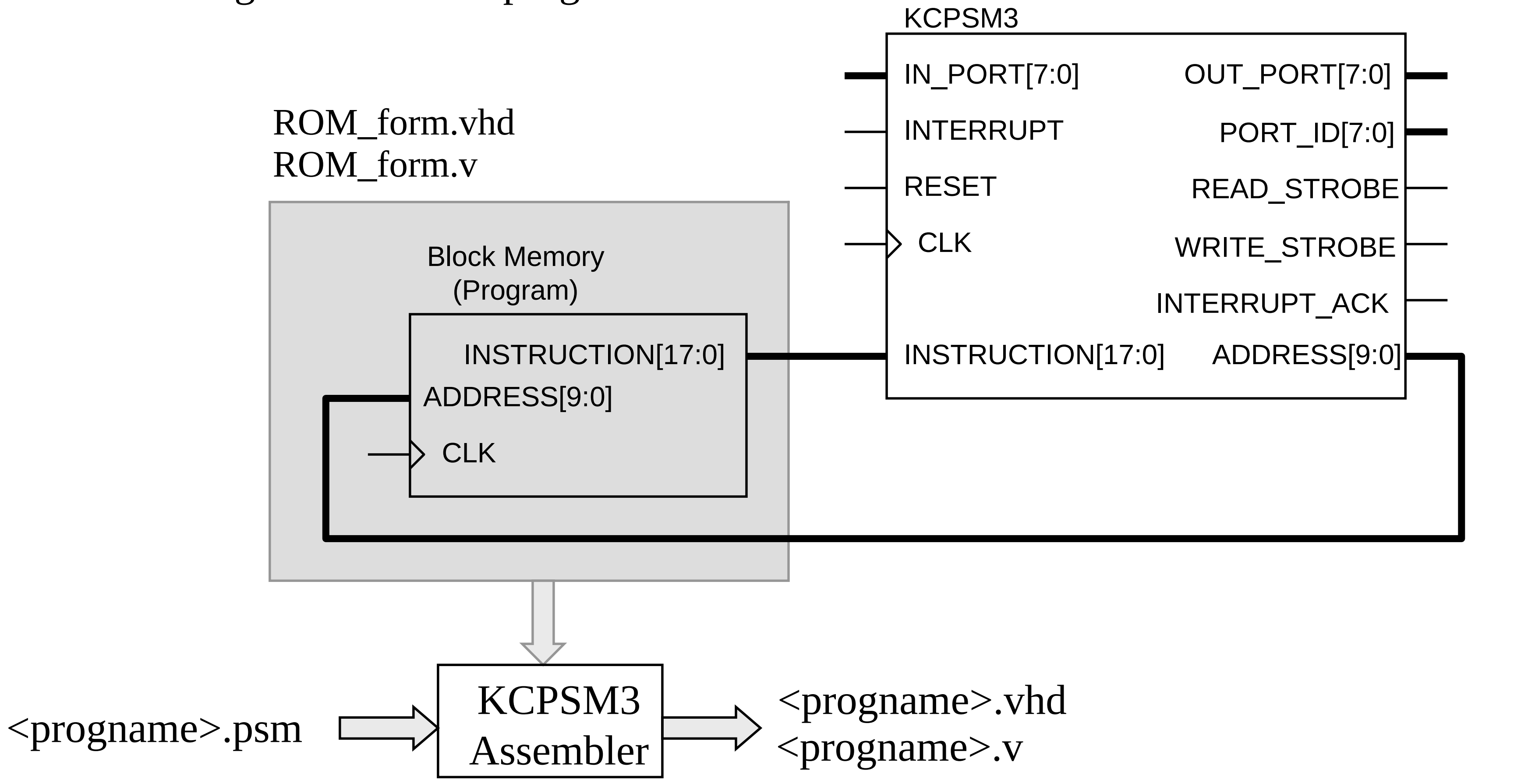


JTAG Loader prepared by
Kris Chaplin
Customer Applications Engineer
Xilinx Ltd
email:chaplin@xilinx.com

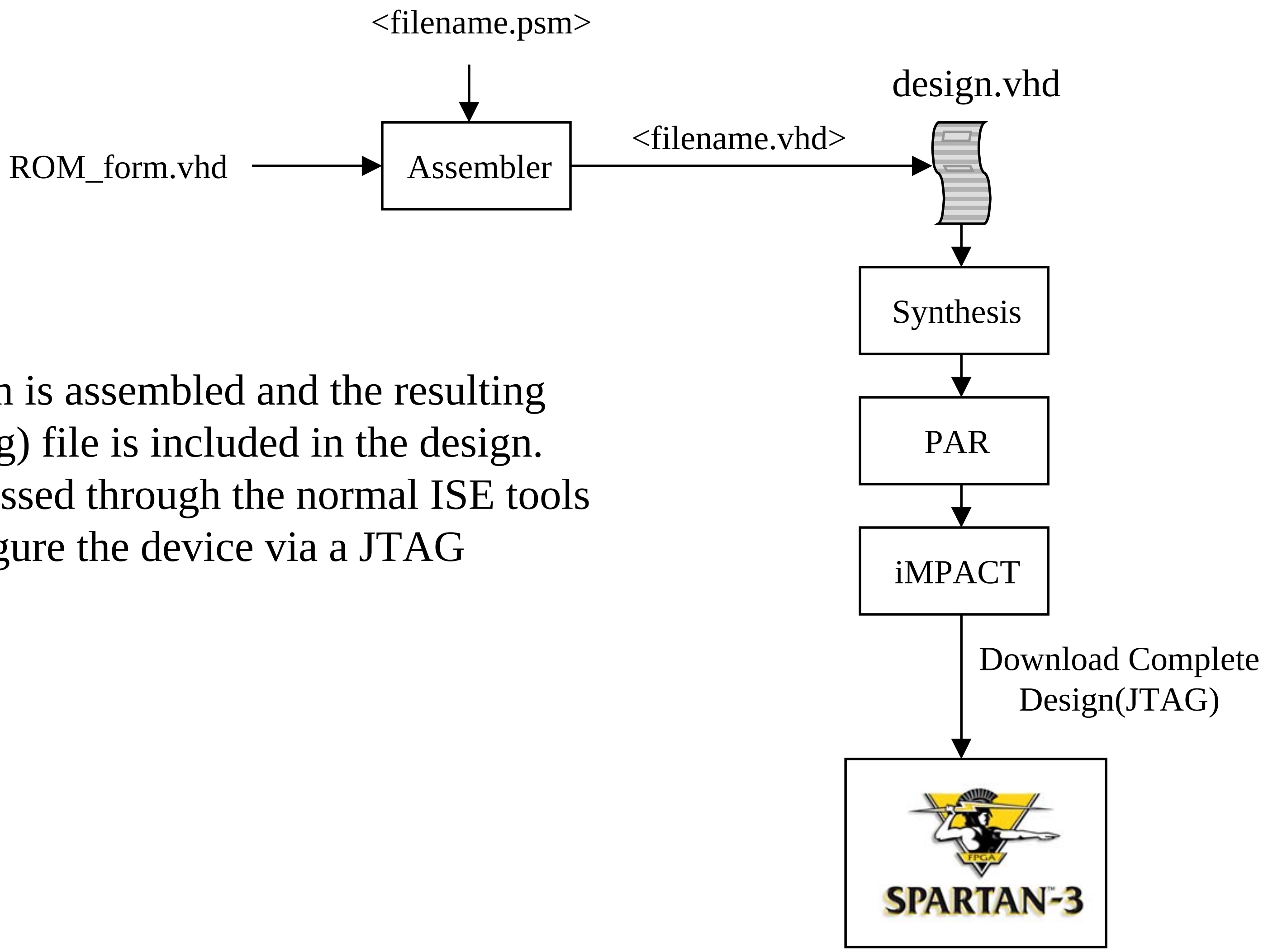
Please contact the author or Xilinx Technical support with any questions about this material.

# Normal PicoBlaze Design

A PicoBlaze (KCPSM3) program is stored in a BRAM configured as a ROM. The program is normally modified by a change to the configuration bit stream. The KCPSM3 assembler reads a VHDL or Verilog template describing the BRAM configuration and simply adds the initialization strings to define the program.



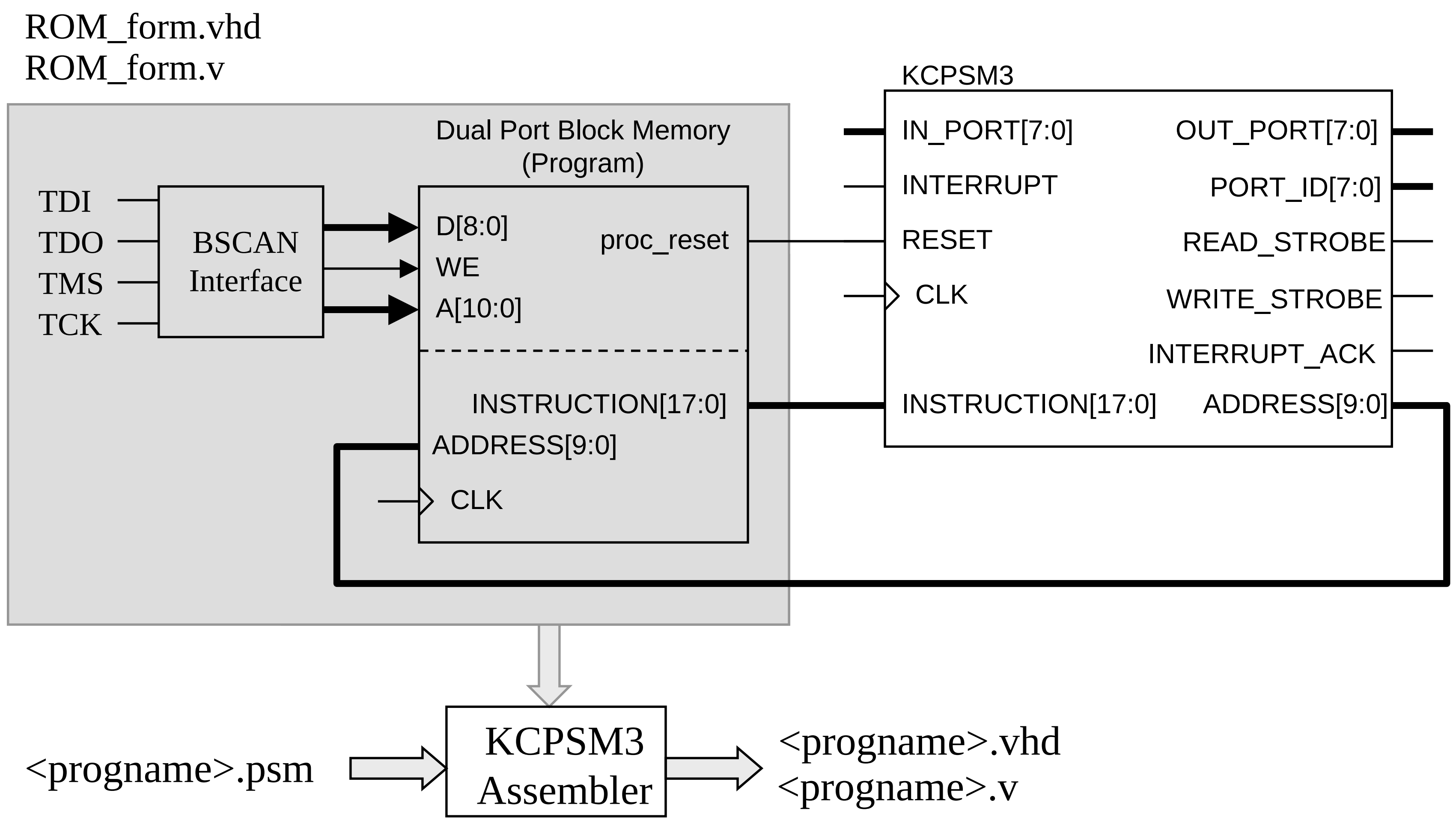
# Normal Design Flow



The PSM program is assembled and the resulting VHDL (or Verilog) file is included in the design. This is then processed through the normal ISE tools and used to configure the device via a JTAG download cable.

# PicoBlaze JTAG Program Loader

The ‘ROM_form’ template is replaced. This adds a few slices of logic to connect the second port of the BRAM to the JTAG controller inside the FPGA. It also adds a reset control.



# Insert JTAG Loader

1 - Replace the ROM_form.vhd (or ROM_form.v) file in your project directory.

2 - Assemble your program to create new VHDL (or Verilog) file.

3 - Add the 'reset' to the instantiation of the the program ROM **<u>and connect</u> to the PicoBlaze**.

```
  component my_prog
     Port (      address : in std_logic_vector(9 downto 0);
             instruction : out std_logic_vector(17 downto 0);
              proc_reset : out std_logic;
                     clk : in std_logic);
     end component;
```

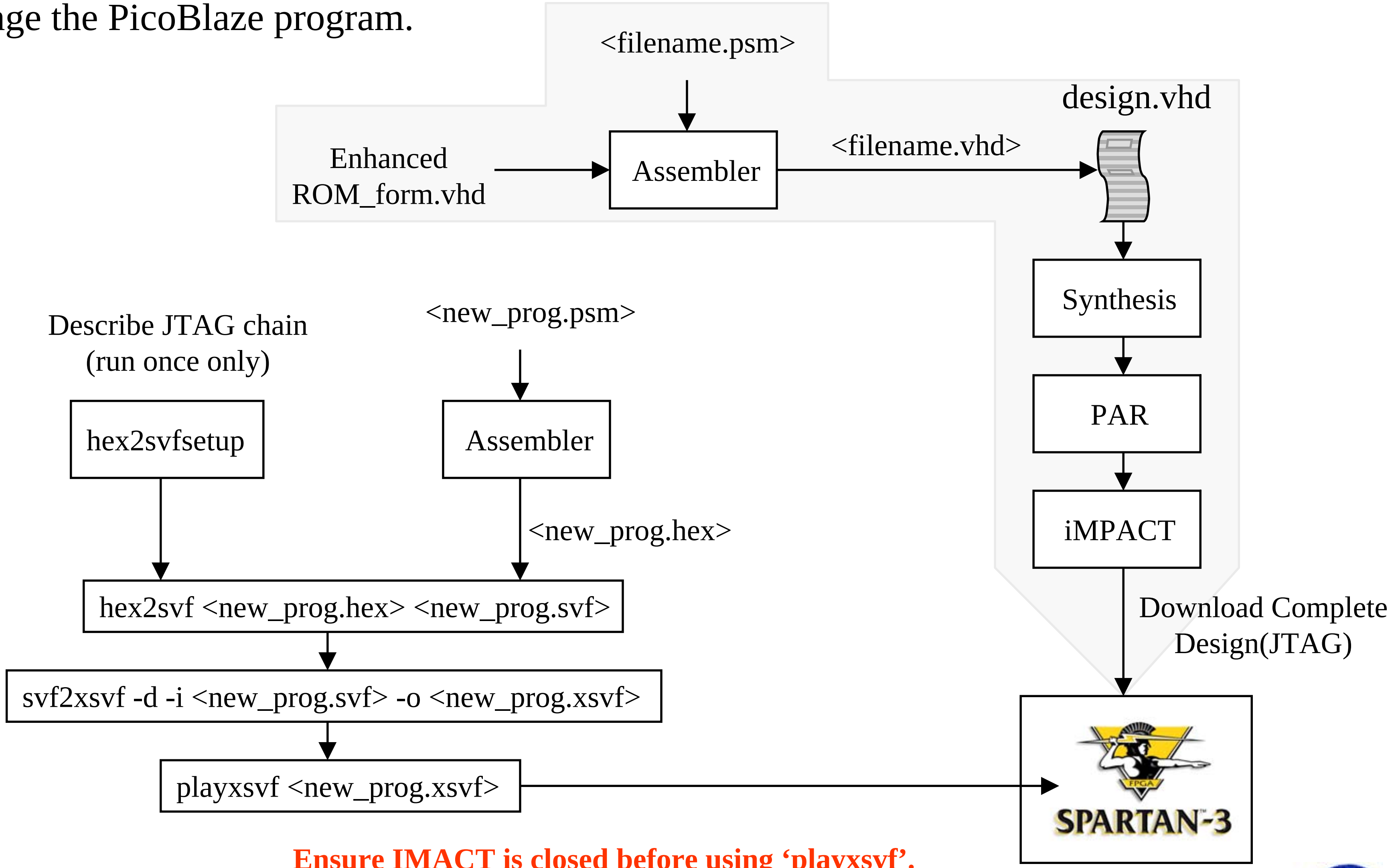
```
 processor: kcpsm3
     port map(      address => address,
                instruction => instruction,
                    port_id => port_id,
               write_strobe => write_strobe,
                   out_port => out_port,
                read_strobe => read_strobe,
                    in_port => in_port,
                  interrupt => interrupt,
              interrupt_ack => interrupt_ack,
                      reset => reset,
                        clk => clk);

   program_rom: my_prog
     port map(      address => pv_address,
                instruction => pv_instruction,
                 proc_reset => reset,
                        clk => clk);
```

The reset will ensure the program is executed from the beginning following each new download.

4 - Synthesize and download the new design.

# JTAG Loader Programs

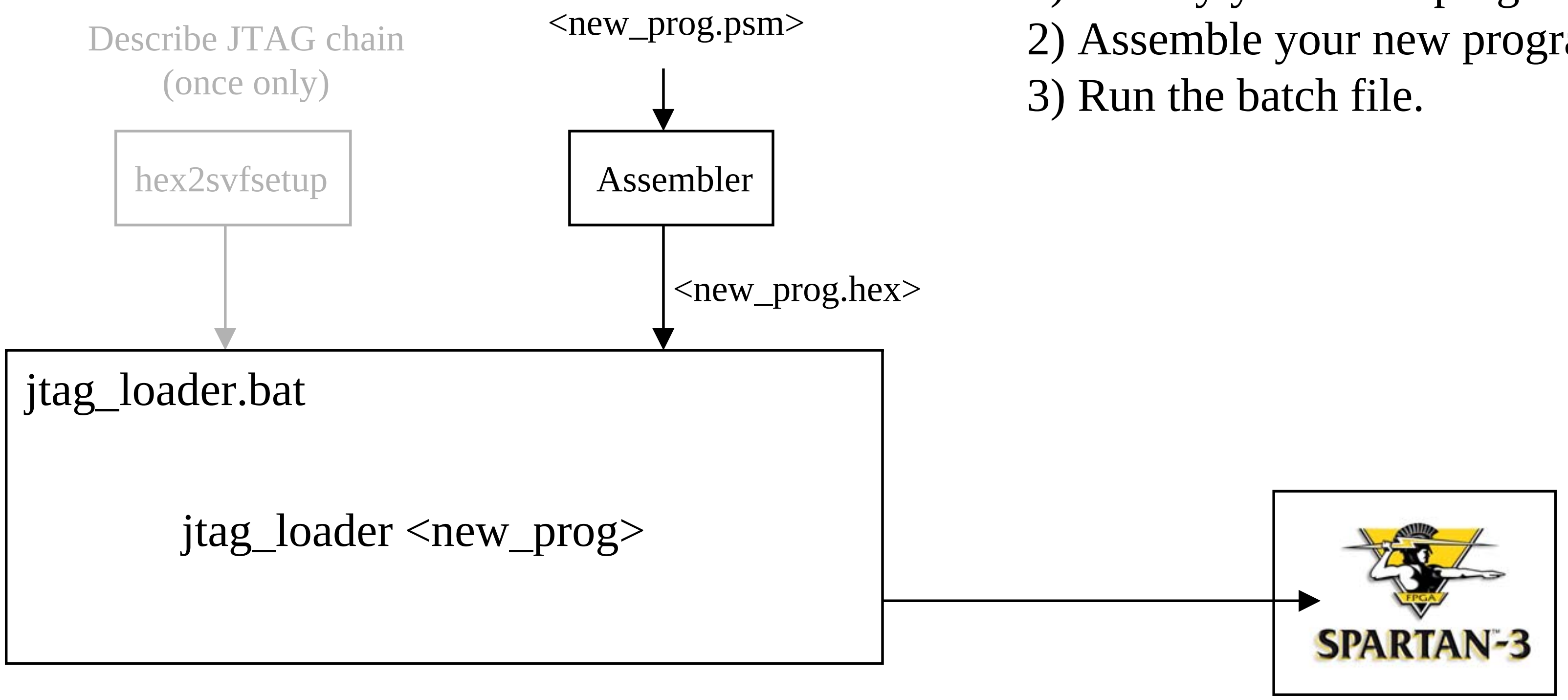
Once the new design is configured into the device, a new set of programs can be used to rapidly change the PicoBlaze program.



**Ensure IMACT is closed before using ‘playxsvf’.**

# The JTAG Loader "1-2-3"

Once the set up program has been used once, a batch file (provided) makes the execution of the remaining 3 programs much easier and faster.



1) Modify your PSM program.
2) Assemble your new program.
3) Run the batch file.

## Working or Not Working?

When the ‘playxsvf’ part of the JTAG loader works the process completes in a few seconds and a few simple messages are displayed….

```
XSVF Player v5.01, Xilinx, Inc.
XSVF file = led_lab.xsvf
SUCCESS - Completed XSVF execution.
Execution Time = 1.061 seconds
```

However, if the player can not access the JTAG cable, the process will appear to take a long time and the DOS window will be filled TCK, TMS and TDI values. These appear so fast that you will not notice the message about not being able to connect to the parallel cable.

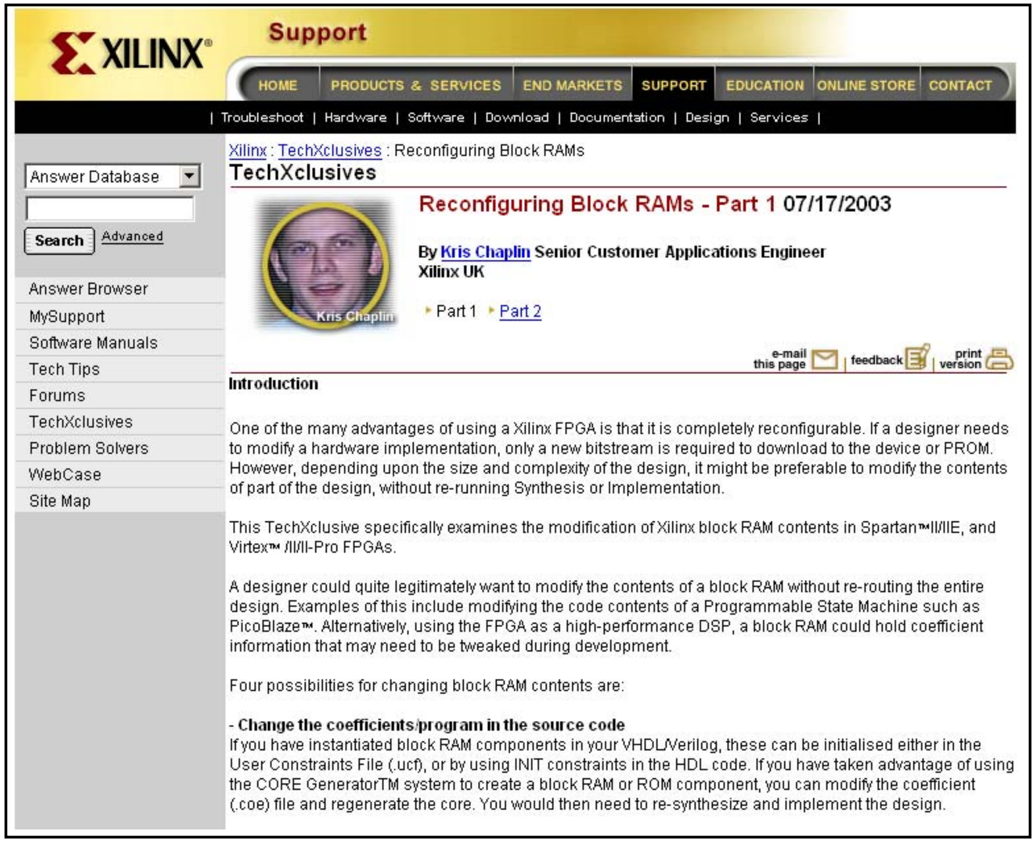
```
XSVF Player v5.01, Xilinx, Inc.
INFO:  XSVF file = blink.xsvf
ERROR:  Xilinx Parallel Cable is not connected to parallel port.
TCK = 0;  TMS = 1;  TDI = 0
TCK = 1;  TMS = 1;  TDI = 0
TCK = 0;  TMS = 1;  TDI = 0
TCK = 1;  TMS = 1;  TDI = 0
TCK = 0;  TMS = 1;  TDI = 0
TCK = 1;  TMS = 1;  TDI = 0
TCK = 0;  TMS = 1;  TDI = 0
TCK = 1;  TMS = 1;  TDI = 0
TCK = 0;  TMS = 1;  TDI = 0
TCK = 1;  TMS = 1;  TDI = 0
```

Make sure the parallel JTAG cable is connected and close all other programs which use the JTAG cable such as iMPACT as these may be preventing access.

# Further Reading

To read more about the JTAG mechanism being used by this utility, please visit the TechXclusive web site…….

http://support.xilinx.com/xlnx/xweb/xil_tx_home.jsp



Xilinx : TechXclusives : Reconfiguring Block RAMs

TechXclusives

Reconfiguring Block RAMs - Part 1 07/17/2003

By Kris Chaplin Senior Customer Applications Engineer
Xilinx UK

Part 1 Part 2

**Introduction**

One of the many advantages of using a Xilinx FPGA is that it is completely reconfigurable. If a designer needs to modify a hardware implementation, only a new bitstream is required to download to the device or PROM. However, depending upon the size and complexity of the design, it might be preferable to modify the contents of part of the design, without re-running Synthesis or Implementation.

This TechXclusive specifically examines the modification of Xilinx block RAM contents in Spartan™II/IIE, and Virtex™ /II/II-Pro FPGAs.

A designer could quite legitimately want to modify the contents of a block RAM without re-routing the entire design. Examples of this include modifying the code contents of a Programmable State Machine such as PicoBlaze™. Alternatively, using the FPGA as a high-performance DSP, a block RAM could hold coefficient information that may need to be tweaked during development.

Four possibilities for changing block RAM contents are:

- **Change the coefficients/program in the source code**
If you have instantiated block RAM components in your VHDL/Verilog, these can be initialised either in the User Constraints File (.ucf), or by using INIT constraints in the HDL code. If you have taken advantage of using the CORE GeneratorTM system to create a block RAM or ROM component, you can modify the coefficient (.coe) file and regenerate the core. You would then need to re-synthesize and implement the design.