Platform Independent 8-bit Soft-core for SoPC

Farhad Merchant[1], Shashank Pujari[2], Manish Patil[3]

Abstract— the theme of this paper is to have a platform independent 8-bit soft core. Paper speaks about implementation of the Picoblaze Processor, a 8-bit soft processor core from Xilinx Inc. on FPGA platforms of other vendors. PicoBlaze is based on a RISC architecture of 8 bits. The design was originally named KCPSM which stands for "Constant (K) Coded Programmable State Machine" (formerly "Ken Chapman's PSM"). It is a well optimized processor designed by Xilinx, which occupies 2% resources of Xilinx Spartan III FPGA. Original design uses the Xilinx primitives, which are replaced by standard generic logics like gates, flipflops, multiplexers etc.. Authors have used KCPSM3 (PicoBlaze3) version of the processor. As a test case the generic design is implemented on Altera FPGA platform. Altera Development and Education (DE2) board is used to verify the platform independent PicoBlaze design, which provides an ideal vehicle for learning about digital logic, computer organization, and FPGAs, Featuring an Altera Cyclone® II FPGA, DE2 board offers state-of-the-art technology suitable for wide range of design projects, as well as sophisticated digital system development. Altera offers NIOS II soft processor with SOPC builder software suite. This paper presents the comparison of PicoBlaze and Nios II processor on Altera platform for small applications. Authors have shown implementation of data transfer through UART using PicoBlaze and Nios II on Altera DE2. Xilinx synthesis tool version 9.1 and Altera quartus II version 7.1 are used. KCPSM3 assembler is used for compilation of PicoBlaze assembly language programs. A small utility program is developed for porting the ROM code generated by the assembler to Altera compatible code.

Keywords: ***platform independent processors, PicoBlaze, reconfigurable hardware, soft core.***

I. Introduction

A Soft processor is a microprocessor core that can be wholly implemented using logic synthesis. It can be implemented via different semiconductor devices containing programmable logic (e.g., FPGA, CPLD)*[1]*. There are several soft cores available in the market, e.g. PicoBlaze, MicroBlaze, Openfire, Nios, Nios II, Cortex-M1, Mico8, Mico32 and AEMB. Out of all listed here PicoBlaze and mico8 are the 8-bit open source soft-cores available in the market. Authors have chosen PicoBlaze because of availability of Xilinx tool chain. INV, LUT2, LUT3, LUT4, MUXCY, MUXF5, XORCY, FDR, FDS, FD, FDE, FDRE, FDRSE, RAM64 and RAM32 primitives are used in the original design of PicoBlaze, which are replaced by the same functionality blocks using standard digital logics.

Organization of the paper is as follows. Section II tells the reason for choosing PicoBlaze soft-core, section III speaks about architecture of PicoBlaze. Section IV is about signals of top module of KCPSM. Section V tells about platform independent implementation of PicoBlaze. Section VI tells about benchmarking of PicoBlaze and Nios II on Altera platform with small application of UART running on it. Section VII is a future work of the idea and concludes the work.

F. A. Farhad is with the ATS Infotech Pvt. LTd, Pune.. (e-mail: farhadmerchant@gmail.com).

S. B. Prof. Shashank Pujari is with International Institute of Information Technology, Pune as a Head of the Department of Embedded System Design.. (e-mail: pujarishashank@gmail.com).

T. C. Prof. Manish Patil is with the International Institute of Information Technology, Pune as a Head of the Department of VLSI Design and Techlology. (e-mail: manishp@isquareit.ac.in)

II. Reason for Choosing PicoBlaze

There are literally dozens of 8-bit microcontroller architectures and instruction sets. Modern FPGAs can efficiently implement practically any 8-bit microcontroller, and available FPGA soft cores support popular instruction sets such as the PIC, 8051, AVR, 6502, 8080, and Z80 microcontrollers.

The PicoBlaze microcontroller is specifically designed and optimized for the Spartan-3, Virtex-II, and Virtex-II Pro FPGA architectures. Its compact yet capable architecture consumes considerably less FPGA resources than comparable 8-bit microcontroller architectures within an FPGA. Furthermore, the PicoBlaze microcontroller is provided as a free, source-level VHDL file with royalty-free re-use within Xilinx FPGAs. Even wide variety of Windows and Linux based assemblers and simulators are freely available for PicoBlaze *[2][3]*

III. Background

The PicoBlaze microcontroller is a compact, capable, and cost-effective fully embedded 8-bit RISC microcontroller.The PicoBlaze microcontroller provides cost-efficient microcontroller-based control and simple data processing. The PicoBlaze microcontroller is optimized for efficiency and low deployment cost. It occupies just 96 FPGA slices, or only 12.5% of an XC3S50 FPGA and a miniscule 0.3% of an XC3S5000 FPGA. In typical implementations, a single FPGA block RAM stores up to 1024 program instructions, which are automatically loaded during FPGA configuration. Even with such resource efficiency, the PicoBlaze microcontroller performs a respectable 44 to 100 million instructions per second (MIPS) depending on the target FPGA family and speed grade.

The PicoBlaze microcontroller core is totally embedded

within the target FPGA and requires no external resources. The PicoBlaze microcontroller is extremely flexible. The basic functionality is easily extended and enhanced by connecting additional FPGA logic to the microcontroller's input and output ports.

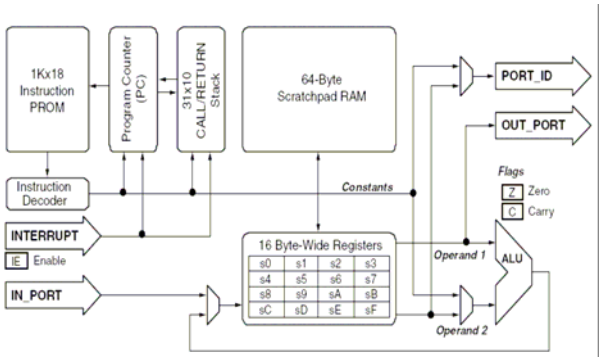

Figure 1 PicoBlaze Embedded Microcontroller Block Diagram*[4]*

Figure 1 shows the architecture of PicoBlaze microcontroller. Blocks of the diagram are as follows:

- General-Purpose Registers
- 1,024-Instruction Program Store
- Arithmetic Logic Unit (ALU)
- Flags
- 64-Byte Scratchpad RAM
- Input/Output
- Program Counter (PC)
- Program Flow Control
- CALL/RETURN Stack
- Interrupts
- Reset

Description of the blocks in detail is widely available in the user manual of the PicoBlaze Processor.

IV. PICOBLAZE SIGNALS

The top-level interface signals to the PicoBlaze microcontroller appear in figure 2.

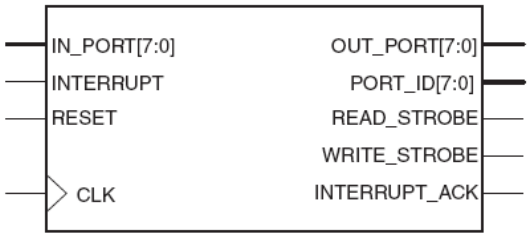

Figure 2 PicoBlaze Embedded Microcontroller Block Diagram*[4]*

A. Inputs

IN_PORT[7:0]: Input Data Port: Present valid input data on this port during an INPUT instruction. The data is captured on the rising edge of CLK.
INTERRUPT: Interrupt Input: If the INTERRUPT_ENABLE flag is set by the application code, generate an INTERRUPT Event by asserting this input High for at least two CLK cycles. If the INTERRUPT_ENABLE flag is cleared, this input is ignored.
RESET: Reset Input: To reset the PicoBlaze microcontroller and to generate a RESET Event, assert this input High for at least one CLK cycle. A Reset Event is automatically generated immediately following FPGA configuration.
CLK: Clock Input: The frequency may range from DC to the maximum operating frequency reported by the Xilinx ISE. All PicoBlaze synchronous elements are clocked from the rising clock edge. There are no clock duty-cycle requirements beyond the minimum pulse width requirements of the FPGA.

B. Outputs

OUT_PORT[7:0]: Output Data Port: Output data appears on this port for two CLK cycles during an OUTPUT instruction. Capture output data within the FPGA at the rising CLK edge when WRITE_STROBE is High.
PORT_ID[7:0]: Port Address: The I/O port address appears on this port for two CLK cycles during an INPUT or OUTPUT instruction.
READ_STROBE: Read Strobe: When asserted High, this signal indicates that input data on the IN_PORT [7:0] port was captured to the specified data register during an INPUT instruction. This signal is asserted on the second CLK cycle of the two cycles INPUT instruction. This signal is typically used to acknowledge read operations from FIFOs.
WRITE_STROBE: Write Strobe: When asserted high, this signal validates the output data on the OUT_PORT [7:0] port during an OUTPUT instruction. This signal is asserted on the second CLK cycle of the two-cycle OUTPUT instruction. Capture output data within the FPGA on the rising CLK edge when WRITE_STROBE is high.
INTERRUPT_ACK: Interrupt Acknowledge: When asserted High, this signal acknowledges that an INTERRUPT Event occurred. This signal is asserted during the second CLK cycle of the two-cycle INTERRUPT Event. This signal is optionally used to clear the source of the INTERRUPT input.

V. PLATFORM INDEPENDENT IMPLEMENTATION OF PICOBLAZE

Original PicoBlaze design uses the following primitives of Xilinx FPGA which are .

A. LUT2, LUT3, LUT4

LUT2, LUT3 and LUT4 respectively means 2-input, 3-input and 4-input LUTs. Behavior of an LUT depends on the INITSTATE value given to LUT. Following piece of code shows how the Xilinx FPGA dependent LUTs are replaced by independent logic.

e.g.

```
attribute INIT : string;
attribute INIT of int_value_lut label is "04";
int_value_lut: LUT3
  generic map (INIT => X"40")
      port map( I0 => a,
                I1 => b,
                I2 => c,
```

O => o);

The technology schematic of the above code is shown in figure 3.

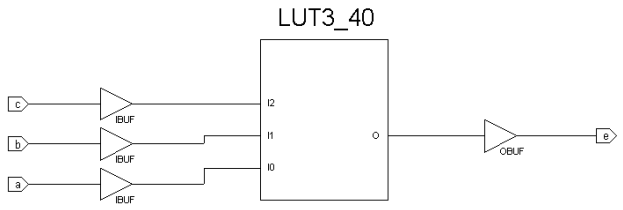

Figure 3 technology schematic of Xilinx primitive(LUT3).

The piece of code shown above is replaced by the following dataflow statement in the architecture body..

O<=(not a) and b and c;

The technology schematic of above dataflow statement is shown in figure 4.

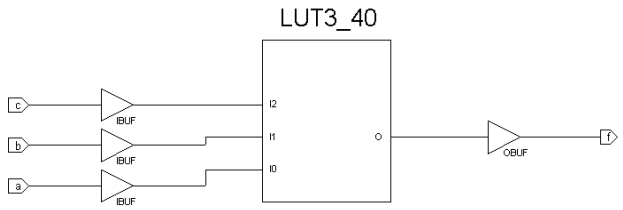

Figure 4 technology schematic of dataflow statement in eq(1).

This way all the LUTs are replaced by the independent logic of same functionality depending on their INIT value.

B. LUT1, LUT2, LUT3, LUT4 reevaluated expressions

Table I depicts the expression for LUT1, LUT2, LUT3 and LUT4 and their replacement expressions to make the design independent from the platform.

The rest of the modules are not in the table as each of them can be replaced with different logic with the behavioral type of modeling in VHDL. They are explained in C.2 with example.

At the end of the synthesis the expresses ions are evaluated with the corresponding FPGA platform. Here that is Altera. All the expressions in the third column of the table are platform independent dataflow statements.

TABLE I
XILINX PRIMITIVES AND THEIR COUNTERPART EXPRESSIONS FOR ALTERA PLATFORM

Xilinx Primitives	INIT Value	Expression to be Evaluated (A, B, C and D are assumed as inputs and O as output)
LUT1	X"1"	O<= !A
	X"2"	O<=A
LUT2	X"D"	O<= B*(!A)
	X"8"	O<= A*B
	X"C"	O<=B
	X"3"	O<=!B
LUT3	X"04"	O<=(!A)*B*(!C)
	X"2F"	O<=A*(!B)+(!C)
	X"3F"	O<=(!B)+(!C)
	X"F3"	O<=C+(!B)
	X"E4"	O<=(A*C)+((!A)*B)
	X"1F"	O<=(!A)*(!B)+(!C)
	X"6C"	O<=(A*(!B)*C)+((!A)*B)+(B*(!C))
	X"96"	O<=((!A)*B*(!C))+((!A)*(!B)*C)+(A*B*C)+(A*(!B)*(!C))
	X"FE"	O<=A+B+C
LUT$	X"0080"	O<=A*B*C*(!D)
	X"EAAA"	O<=(B*C*D)+A
	X"7400"	O<=((!B)*C*D)+((!A)*B*D)
	X"5A3C"	O<=((!B)*C*(!D))+(A*(!C)*D)+((!A)*C*D)+(B*(!C)*(!D))
	X"1000"	O<=((!A)*(!B)*C*D)
	X"5400"	O<=((!A)*C*D)+((!A)*B*D)
	X"41FC"	O<=(C*(!D)+((!A)*(!B)*(!C)*D)+((!A)*B*C)+(B*(!D))
	X"0001"	O<=((!A)*(!B)*(!C)*(!D))
	X"6996"	O<=((!A)*B*(!C)*(!D))+((!A)*(!B)*C*(!D))+(A*B*C*(!D))+((!A)*(!B)*(!C)*D)+(A*B*(!C)*D)+(A*(!B)*C*D)+((!A)*B*C*D)+(A*(!B)*(!C)*(!D))
	X"F3FF"	O<=((!B)*C*(!D))
	X"0145"	O<=((!A)*B*(!D))+((!A)*(!B)*(!C))
	X"0400"	O<=((!A)*B*(!C)*D)
	X"8000"	O<=A*B*C*D
	X"FFE2"	O<=(B*C)+D+(A*(!B))
	X"6E8A"	O<=(A*B*(!D))+((!A)*B*D)+(A*(!C))
	X"0002"	O<=(1*(!B)*(!C)*(!D))
	X"0010"	O<=((!A)*(!B)*C*(!D))
	X"4000"	O<=((!A)*B*C*D)
	X"0100"	O<=((!A)*(!B)*(!C)*D)
	X"6555"	O<=((!A)*(!C))+(A*(!B)*C*D)+((!A)*B)+((!A)*(!C))
	X"A999"	O<=(A*B)+((!A)*(!B)*(!C))+(A*C*D)+((!A)*(!B)*(!D))

* represents AND operation,
\+ represents OR operation,
! represents NOT operation

C.2 FDR, FDS, FD, FDE, FDRE, MUXCY, XORCY, INV, FDRSE, RAM16X1D, RAM64X1S, MUXF5, RAM32X1S

All these modules used as Xilinx primitives in original designs are replaced by their behavioral model in the platform independent design. Such a replacement doesn't cause any calamity in the model.

i. e.

```
process(clk)
begin
   if clk='1' and clk'event then
      if reset='1' then
            q<='0';
      else
            q<=d;
```

```
        end if;
      end if;
    end process;
```

The technology schematic of above piece of code is shown in figure 4, which is same as using FDR Xilinx primitive.

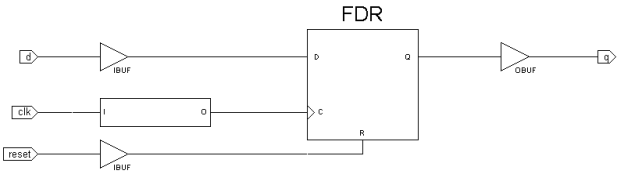

Figure 4 technology schematic of FDR(when implemented without using Xilinx primitive)

C. HEX to MIF converter

Xilinx soft-cores use the HEX format to program the processor ROM while Altera platform based soft cores use the Memory Initialization File format. To convert HEX format into MIF format C programming is used with command line argument.

VI. BENCHMARKING

Table I shows the comparative resources utili1zation of Xilinx PicoBlaze (dependent on Xilinx Platform), PicoBlaze(Independent of Platform) on Xilinx platform, PicoBlaze on Altera platform, and NIOS on Altera platform.

TABLE II
RESOURCE UTILIZATION SUMMARY

Device & platform	PicoBlaze Original Xilinx Spartan 3	PicoBlaze platform independent Xilinx Spartan 3	PicoBlaze platform independent ALTERA CYCLONE II	NIOS ALTERA CYCLONE II
Slices in Xilinx Logic Elements in Altera	99 out of 3584 2%	236 out of 3584 6%	389 out of 33,216 1%	3084 out of 33,216 9%

VII. CONCLUSION AND FUTURE WORK

At the end of the synthesis a minion becomes a totally free from the platform and can be implemented on any platform occupying very less resources (1% in CYCLONE II).
For small application it is recommended to use the PicoBlaze on Altera platform rather than going for Nios or Nios II. Other suggested applications by authors are automotive embedded systems, multiprocessor applications such as Advanced Encryption Standard (AES – a block cipher algorithm) and applications based on tiny OS *[9]*.

ACKNOWLEDGEMENT

Special thanks go to Ken Chapman, for designing the PicoBlaze microcontroller and keeping the source code open. I am thankful to Xilinx and Altera for providing tool chains. I am thankful to ISquareit, Pune for providing me space for research.

REFERENCES

[1] PicoBlaze 8-bit Embedded Microcontroller User Guide for Spartan-3, Virtex-II and Virtex-II Pro FPGAs.
[2] Sdasdsa http://www.xs4all.nl/~marksix/
[3] Ivybridge Simulation (2008, November, 10). Picoblaze Projects [Online] Available: http://www.ivysim.com/kits/spartan3e/picoblaze/
[4] Ken Chapman (2008, November, 12). PicoBlaze Manual [Online] Available:http://courseware.ee.calpoly.edu/cpe-269/ExternalDocs/KCPSM3_Manual.pdf
[5] http://en.wikipedia.org/wiki/Soft_microprocessor
[6] Industrial embedded systems (2008, October, 10). Product Search [Online] Available:http://www.industrial-embedded.com/products/search/fm/id/?29678
[7] Press Release (2008, November, 5). [Online] Avalable: http://www.embeddedstar.com/press/content/2002/12/embedded6562.html
[8] VHDL Primer by J. Bhasker
[9] *Pengyuan Yu, Patrick Schaumont* "EXECUTING HARDWARE AS PARALLEL SOFTWARE FOR PICOBLAZE NETWORKS"