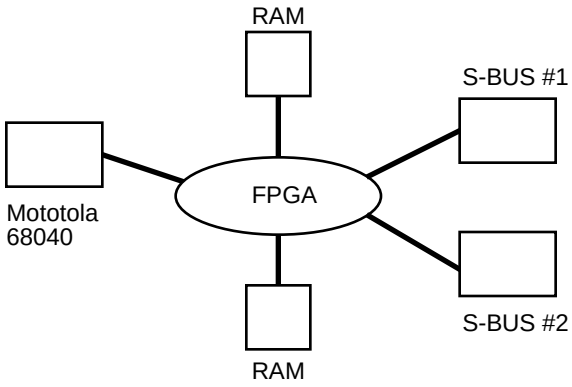
Bus Translation Design Using FPGAs

Venkata Ramana Kalapatapu, Design Engineer
Sand Microelectronics, Inc.

Abstract

This paper discusses the use of a 6K gate FPGA to implement a design that controls and manages the communication between a Motorola 68040 bus, two SUN S-BUS devices, and two static RAMs at the rate of 25 MHz.



Introduction

The above diagram describes the configuration of the board with multi-processors and the role of the FPGA to control the communication between all the devices.

Each processor can be a master or a slave in a particular configuration while both RAMs have a slave role at all times. When the Motorola 68040 is the master, either of the SBUS devices or both can be a slave. When either of the SBUS devices is the master, only the 68040 can be the slave and the other SBUS device is idle.

The chip controlling the communication between the various devices on the board must determine where an incoming address has initiated, the destination device of choice, perform bus translation, and detect and correct any parity errors that occur during transmission.

Requirements

The choice of the device had to meet the following criteria:

- Data transfer rate of 25 MHz.
- High I/0 count (157 I/Os) with a high degree of flexibility for ease of board layout (pins had to be fixed early in the design and cannot be restricted to be of one type or another).
- Reliability and high level of signal integrity.
- Fast prototyping ability.
- Synthesizability (verilog code).
- Low cost.
- Ease of debugging, and ability to re-place and route without changing pin assignments.

FPGA Details

The device of choice for this design was Actel's A1460A-1 in a 208-pin PQFP package. The device had a core capacity of 6000 gates. The device has three high fanout clock networks for the core and one for the I/Os. Two of the core clock networks was used in this design. The HCLK hard wired high speed internal clock ran at 40 MHz. It clocked the control state machines, the address decoding, and the data transmission. Another clock network (CLKA) ran at 25 MHz. It controlled the parity error checking and correction. The fine granularity of this FPGA was highly suitable for synthesis and the closest in performance to a full ASIC.

The core of the chip was partitioned into two major blocks:

- Control logic and address decoding.
- Data Path.

The control logic section occupied the majority of the core. It included the functions of detecting the Master device, determining the slave device, controlling data transmission, and checking and correcting any parity errors that occurred during bus translation and transmission. The decoding logic took care of address decoding and of translation.

The Data Path block consisted of the bus translation function between the Motorola 68040 and the S-bus devices (both the S-bus and the Motorola bus are 32 bits wide). Also, the Data Path section included the Read/Write operations from any of the master devices to the RAMs and visa versa.

The S-Bus operates in a burst transfer mode while the Motorola bus is not capable of burst transfer mode operation. Thus a data transfer from the S-Bus device to the Motorola 68040 is a single burst cycle operation.

Paper published in the 1995 PLD Conference Proceedings

The RAMs are 16-bit wide. Thus a Read/Write operation involving any of the master devices and a RAM is done in two cycles, each cycle covering 16 bits of data.

Design Encoding and Simulation

The design was encoded using standard verilog VHDL. It was then simulated using a zero-delay verilog behavioral model simulator. The design was then synthesized using the Synopsys standard compiler. The gate level netlist was then simulated using Actel's macrocells verilog models library (unit delay models).

The design was then placed and routed using Actel's ALS software. The ALS timer was used to check the post place and route timing on the critical paths as well as overall chip performance.

In the encoding process some specific flip-flops were directly instantiated. All the I/Os were also directly instantiated. Our experience has been that the performance one gets from an FPGA using synthesis is highly dependent on proper coding that makes use of the particular architecture of that FPGA (mux based in the case of the A1460).

Post layout delays were back annotated to the verilog gate level netlist using Actel's SDF interface. The back annotated design was then simulated using the verilog-XL simulator. The design was simulated by the same set of input vectors for both the behavioral design and the back annotated gate level design producing the same result.

Sample Code

```
module SRamCtrl ( SC_NIO_CLK,
SC_MEZZ_RESET, SC_ICE2SRAM,
SC_ISP2SRAM, ......................,
SC_SRAM_UB_L, SC_SRAM_LB_L );
input [1:0] SC_NIO_SIZE;
input [2:0] SC_SBUS_ACK_L;
input [2:0] SC_SBUS_SIZE;
input SC_NIO_CLK, SC_MEZZ_RESET,
SC_ICE2SRAM, SC_ISP2SRAM, SC_INT_TS,
SC_AS_L,
  SC_NIO_RW, SC_SBUS_RW,
SC_NIO_ADR:
output SC_SRAMCYC_L, SC_CSDELAY,
SC_SRAM_CS_L, SC_SRAM_OE_L,
SC_SRAM_WE_L,
  SC_SRAM_UB_L, SC_SRAM_LB_L;
  wire n640, n641, n642, n630, n631, n632,
n633, n634, n635, n636, n637,
  n638, n639, n627, n628, n629,
IspSRamCyc, \*cell* 111/Z_0,
     \*cell*106/CONTROL1;
  DFP1 SC_SRAM_CS_L_reg (
.PRE(SC_MEZZ_RESET),
.CLK(SC_NIO_CLK), .D(n637),
    .Q(SC_SRAM_CS_L) );
  DFP1 SC_SRAM_WE_L_reg (
.PRE(SC_MEZZ_RESET),
.CLK(SC_NIO_CLK), .D(n640),
    .Q(SC_SRAM_WE_L) );
.....................
.....................
  DFP1 SC_SRAMCYC_L_reg (
.PRE(SC_MEZZ_RESET),
.CLK(SC_NIO_CLK), .D(n639),
    .Q(SC_SRAMCYC_L) );
endmodule;
```

Results

The design went through two iterations. In the first pass the core utilization was 98 percent and the design included JTAG testing. The results were satisfactory but a decision was made to get rid of the JTAG. The design was modified in a major manner to push the performance further. The pinout was fixed from the first round. Re-placing and re-routing the design went through without any problems at all. The final core utilization was 70 percent. The final design was backannotated and simulated at gate level. It was fully functional at a data transfer rate of 30 MHz.

Although a "–2" speed grade was available for this part allowing for another 10 percent speed improvement. The design was finalized in a A1460-1 in a 208-pin PQFP package.

After the design was verified a fuse file was generated in the ALS environment and Actel's activator was used to program prototypes.

Conclusions

Using FPGAs for bus translation applications proved to be feasible, easy, and reliable. FPGAs provided high I/0 counts needed for such an application. Also, sufficient core gates were available for control and arbitration logic. I/0 instantiation and writing code that makes use of the available architecture is instrumental in obtaining optimal results.