Motorola MC68040 High-Speed Design Using Altera EPM5000 Erasable Programmable Logic Devices

Hui-Chien Shen and Stephen M. Becker
Sandia National Laboratories
Department 2274
P.O. Box 5800
Albuquerque, NM 87185
Phone (505) 844-6335

Abstract --- Many designs use EPLD's (Erasable Programmable Logic Devices) to implement control logic and state machines. If the design is slow, timing through the EPLD is not crucial so designers often treat the device as a black box. In high speed designs, timing through the EPLD is critical. In these cases a thorough understanding of the device architecture is necessary. This paper discusses lessons learned in the implementation of a high-speed design using the Altera EPM5130.

Introduction

A Sandia National Laboratories developed multiprocessor flight computer, the Sandia Airborne Computer V (SANDAC V), uses 32 bit microprocessors for applications such as flight tests, robotics and image processing. The CPU module employs either the Motorola MC68020 or MC68040 microprocessor. The architecture allows any processor to access another processor's memory space. This is implemented via a global bus and a bus arbitration scheme that allows up to 15 processor modules to carry out inter-processor communication.

An Altera EPLD was used to implement glue logic for the MC68040 based processor module. In addition it had to provide bi-directional bus translation for global accesses to MC68020 based processor modules. This meant that synchronous MC68040 bus cycles and asynchronous MC68020 bus cycles had to be compatible. The EPLD design consisted of five modules: 68040 / 68020 bus translator, bus arbitration module, 2 megabyte RAM controller, watchdog timer and a control module for manipulating 32, 16 and 8 bit data bus multiplexers. The initial design was done in an Altera EPM5128 EPLD. The design was recast into a larger EPM5130 EPLD to eliminate five additional IC's. This freed up critical board space and made the board easier to manufacture.

All papers must include the following statement:
This work performed at Sandia National Laboratories is supported by the U.S. Department of Energy under contract DE-AC04-76DP00789.

This paper focuses on lessons learned in developing and implementing an EPLD design. They include:

1. Black Box Rules. How to avoid contention when using bi-directional signals.
2. Expander Terms. How expander gates can increase product terms.
3. Floor planning the design. Delays from one logic array block to another can result in excessive timing delays. Implementation of global signals can minimize the problem.
4. Pros and Cons of using a global clock versus an array clock.
5. Timing. What you see is not what you get.
6. Rules to follow when prototyping the design. Issues on wire wrapping and probing internal nodes with an oscilloscope.
7. Delay lines.
8. Implementation of test pins.
9. Why the compiler, not the designer, should choose pin assignments during layout.
10. What to do if the design does not fit.

Black Box Rules

From our experience, the Altera EPLD can be treated as a black box if all of the following conditions are met:

1. Input to output propagation delay is greater than two and a half to three times the EPLD prop delay. For example, if you are using a 25ns EPLD then substitute 75ns as the "real" prop delay of this part. If your design can withstand this amount of delay you probably don't have to worry about tweaking the architecture. The 75ns time is worst case for one feedback path plus the use of expander terms of the same LAB (Logic Array Block) or one feedback path thru the PIA to other LABs.

2. On bi-directional signals, the time required between tri-stating the output driver to asserting the input is greater than twice the prop delay of the EPLD. On a 25ns EPLD

you will need 50ns (worst case) for the output enable to negate the tristate buffer before reconfiguring the pin as an input.

3. On bi-directional signals, time required between input negation and output assertion is greater than twice the prop delay of the EPLD. If conditions two and three are met you avoid contention between an input changing to an output and vice versa inside the EPLD.

4. Macrocell utilization is less than 50% of available macrocells and number of spare pins is greater than 10% of the total I/O pin count. This will keep multiple feedback paths to a minimum.

If all the conditions described above are not met then you need to understand the architecture.

Expander Terms

Expander terms (for EPM5000 series parts) can only be shared between macrocells that reside within the same LAB. Expander terms are a "sea of NAND" gates which the compiler will use to create more product terms since each macrocell is limited to 3 "AND" gates feeding one "OR" gate. You can, in effect, increase the number of product terms since there are 32 expander term NAND gates in each LAB. There are 16 macrocells in each LAB that share the expander terms. Recall you can implement any logic with just NAND gates. This is the building block of "real" gate arrays. The expander term NAND gates can be used to create D Flip-flops, D latches and SR latches, however, there is a speed penalty. Also it takes 6 NAND gates to build a D F/F so there is a resource penalty as well. Expander terms can never be shared throughout the entire chip. The compiler will duplicate expander logic in other LABs but will never share expander logic between LABs. This can be confusing when looking at the report file of the design. It may list some expander equations as being shared between LABs. In reality, the logic is **duplicated** in these LABs, not shared. All JK, SR and T flip flops are implemented as D flip flops for EPM5000 series parts.

As a side note there is a "Balancer" algorithm used in the router. It's job is to keep expander term to macrocell usage at a 2:1 ratio. Buried macrocells may be used instead of expander terms for implementing combinatorial logic.

You can lump logic together as one megafunction by specifying the CLIQUE option. The compiler will try to put the megafunction logic into one LAB to avoid the PIA delay between multiple LABs.

Floor Planning

You must pay strong attention to floor planning the design. Delays from one LAB to another can be excessive. Any input signal used by multiple LABs should be implemented as a global signal. Examples would be clock, read, write, address lines etc. Use the dedicated input pins of the EPLD for global signals. Connect all unused **dedicated** input pins to ground to prevent high current draw and noise from floating inputs. Unused I/O pins, normally called reserved pins, must be left floating. There is one exception to floating reserved pins. If you have a signal that you might want to use in the future do the following: (1) Run the trace on the PC board to the reserved pin. (2) Go into the schematic editor and connect a tristate buffer to the reserved pin. (3) Tie the EPLD tristate buffers' input signal and output enable to ground to keep it tristated. You can then change the design later to use the reserved signal as an input or output.

Clocks

Each LAB has two options for implementing clocked flip flops. Array clock which can be generated by an internal product term, and global clock which must come from the dedicated clock pin.

An SCLK buffer must be placed to designate the clock as a global clock. If a global clock is used in a LAB, all 16 macrocell flip-flops associated with that LAB are tied to the global clock line. Use it very carefully. This means you cannot use multiple clocks in that LAB to drive different flip flops.

Delay from global clock to output is much shorter than an array clock but it requires a longer setup time which may offset its benefits. If you need to use more than one clock in a LAB you will be forced to use array clocking. As always, don't use gated clocks in your design.

Timing

There are times when what you see is not what you get. The circuit in Figure 1 is designed to assert DSACK1 and DSACK0 at the same time. If you look at the design at face value it appears to work correctly. There is one problem. The EPLD architecture only allows one output buffer per macrocell. The compiler routes the output of the DSACK1 flip flop thru an internal feedback path to an adjacent macrocell for DSACK0. The delay thru the feedback path to the buffer causes DSACK0 to assert later than DSACK1. A better way to implement this circuit is shown in Figure 2. Here the feedback path is eliminated since the input signal drives the front end of both macrocells. It is assumed that

both flip flops reside in the same LAB. If they reside in separate LABs there will be skew due to interconnect delays between LABs.

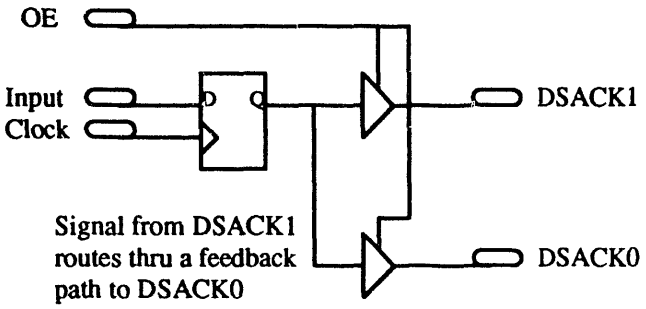

Figure 1. Skew due to internal feedback delay

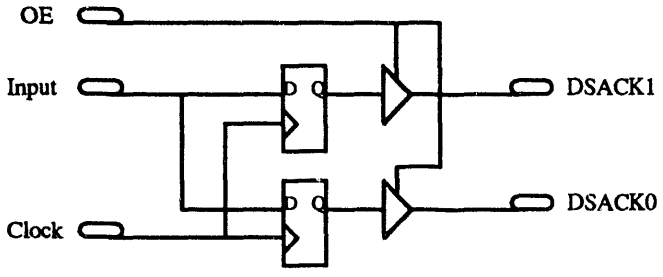

Figure 2. No feedback delay

Wire Wrap

It is best to wire wrap all pins on the first prototype EPLD. Exception to this might be the clock signals. If the pins are wire wrapped, a multilayer prototype board can be fabricated while the EPLD is being designed. This means you may have to make preliminary pin assignments for clock, power and ground on the prototype board. Be sure to use pull up resistors on EPLD driven signals such as output enable and chip selects. This will prevent bus contention and short circuit conditions in case the board is powered up without the EPLD in place. An advantage to wire wrapping is the ability to change pin assignments on the fly for hand routing of critical nets. The wire wrap pins can also be used as test points for instrumenting the design with a logic analyzer and oscilloscope. This can be invaluable for surface mount prototype boards. Be sure to sprinkle at least 2 VCC and 4 ground test points around the EPLD for instrumentation purposes.

NOTE: If a lot of outputs switch state at the same time, wire wrapping may induce noise (voltage glitches) due to the increased ground path inductance. Use bypass capacitors and a ground plane around the EPLD to minimize this problem. Altera recommends using 0.2 uF capacitors from **each** VCC pin to ground on the EPLD. You can also use a 10 to 30 ohm series resistor in the output signals. This will slow down the rise time and minimize noise.

Delay Lines

Because the Altera architecture has fixed routing delays you could use the feedback paths as delay lines. This is **NOT** an acceptable practice since faster parts could be used which would offset the delay line. Also, if you change the design at a later date, the routing may not be the same as the original design. It is best to implement a synchronous design to avoid this pitfall. If you go against conventional wisdom and use the feedback path as a delay line be sure to:

1. Use an MCELL buffer to prevent minimization.
2. Document delay on the schematic page.
3. Verify the delay in the timing simulator or timing analyzer, not the functional simulator.

Test Pins

If the macrocell of a spare I/O pin is used for buried logic, the compiler will automatically attach its output to the spare I/O pin and you can monitor the signal with a scope. This is why you sometimes see the output of an unused I/O pin toggle. If the macrocell of a spare I/O pin is not used for buried logic, it can be used as a test pin to probe internal nodes of the chip with an oscilloscope. You must always remember that timing can change when test pins are used in this type of implementation due to signal delay thru the macrocell.

Layout

Let the compiler choose pin assignments for speed critical designs. It will do a better job of minimizing timing delays than can be done by hand. Once pin assignments have been done, critical nets may need to be assigned to specific macrocells based on timing of feedback routes inside the EPLD. You may want to change pin assignments based on this timing information. You can specify the layout of critical nets for both I/O macrocells and buried macrocells. I/O macrocells can be specified thru pin number assignments. Buried macrocells can be specified with the PIN/MC CHIP option in the assignments menu. The problem with tweaking is you may create a timing delay in another portion of the design.

Many times it is hard to estimate how large of an EPLD you need. When working with a prototype board, consider using two EPLD footprints as illustrated in Figure 3. The small

footprint is for the EPLD you expect to use. The large footprint is available in case a larger EPLD is required. Since the EPLD pins are wire wrapped it is easy to change configurations and you might save an iteration of a prototype board.

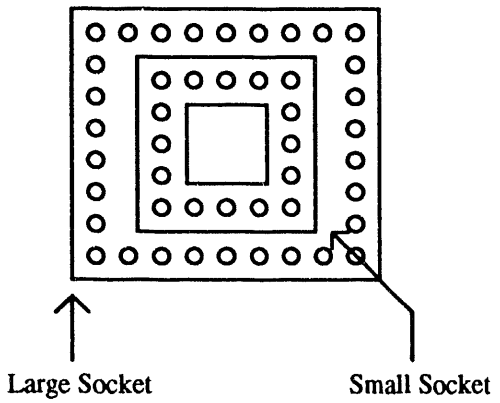

Figure 3. Dual Footprint

Report File

A Report file is generated once the fitter has routed the design. The report file contains everything you would ever want to know about your design. However, there are portions that are cryptic such as the macrocell inter-connectivity maps and the PLA equations. Sometimes a design will not fit. The inter-connectivity maps can be helpful in partitioning the design with MCELL buffers so it will fit. The synthesizer will not minimize an MCELL buffer, however, an MCELL buffer does introduce additional signal delay.

The PLA equations are sometimes DeMorganized. Keep this in mind if you have to dig into them. The equations will tell you how macrocell and expander terms are used. This can be helpful when tracking down critical nets. You can specify macrocell assignments and force the fitter to layout the design in a specific manner. An EXP in the PLA equation indicates an expander gate. Keep in mind that equation numbers found in the report file are generated randomly and have no correlation to specific macrocells or LABs.

Design won't fit

There are several reasons why a design won't fit.

1. Pin number assignments are poorly defined requiring multiple feedback paths when implementing the design. Let compiler make pin assignments.

2. SCLK buffer was used in a LAB that could not take advantage of global clocking for all 16 Macrocells. This wastes flip flop resources which could result in too few flip flops available for the design.

SCLK can safely be used if: (A) All 16 flip flops in the LAB take advantage of the global clock. (B) You have enough spare pins that wasting resources is not a problem.

3. Combinatorial logic function is too complex to fit in one macrocell. Place a SOFT buffer or a MCELL buffer in the logic to break it up. If the logic function uses too many expander terms a buffer will force the logic to be implemented across 2 or more Macrocells within the same LAB. Keep in mind that a SOFT or MCELL buffer adds a delay to the signal path.

4. If a large combinatorial function feeds multiple LABs use an MCELL buffer to keep the "source" logic in one LAB. Otherwise the compiler will tend to duplicate the logic in each LAB which wastes resources.

There is one major difference between a SOFT buffer and an MCELL buffer. A soft buffer can be minimized out by the logic synthesizer. If the compiler decides that the soft buffer is not needed then it takes it out. There are cases where you don't want the compiler to do this. In those cases you will want to insert an MCELL buffer. The compiler will not remove an MCELL buffer during logic synthesis. This allows you to maintain control in partitioning the design. When performing simulation there are times you will want to attach a probe at a node that may be minimized out by the synthesizer. In those cases use an MCELL buffer to prevent the node from disappearing. You must realize that timing information is no longer accurate when this is done but it does help in troubleshooting functional logic problems.

5. Of course you can always let the compiler choose a bigger EPLD or partition the design across multiple EPLDs. Rule of thumb is if you partition across several EPLDs you lose speed due to input / output pad delays and increased signal capacitance.

Summary

In high speed applications, an in-depth knowledge of the EPLD architecture is beneficial in minimizing timing problems. By paying close attention to internal feedback paths and macrocell layout, many problems can be avoided before a design is verified at the prototype stage.

References and Sources

1. Altera 1992 Applications Handbook.
2. Altera 1992 Data Book.

END

DATE
FILMED
10/15/93