Guidelines for FPGAsIC Designs

by Vinay Sharma

Integrated circuits have changed the way of our life. You name one gadget and will find the power of silicon which has made such complex electronic circuits possible.

Integrated circuits, come in many different flavors these days. User designed chips in particular, CPLDs and FPGAs have revolutionized the way of system design.

But ASIC remains in lead, due to their speed, power and performance advantages. Every critical system design is flagged with ASICs.

To learn the IC design process, techniques & 'critical' requirement handling, engineers practice for hours and hours on EDA tools to master know-how of design fundamentals.

Modern ASIC design tools like DSCH and MICROWIND provides very easy to go-through design flow for CMOS IC designs. It supports traditional schematic circuit building methods, layout editing, various analysis & verification methods, and fab sign off.

But more than *rights* & *lefts* of IC design flow, it's the basic design methodology and circuit building techniques which leads to success in fabrication. But many of times, engineers face hurdles during simulation and failures in prototyping.

This article is aimed at explaining the basic design styles and recommendation for building digital CMOS circuits for ASICs, along with list of suggestions for building digital circuits which can be compatible for FPGA prototyping.

This article would certainly leave you in a better position to make the necessary decisions to embark with confidence on the route to custom silicon after FPGA prototyping.

Design Tools and need of Prototyping

Modern ASIC techniques give ordinary design engineers the opportunity to exploit some of today's most advanced electronic technology. This has been made possible by powerful software tools which take care of the technology-specific aspects of integrated circuit design. This high technology inevitably means that there are many aspects to consider in weighing up the best path to take. The quality of the tools, how well they are documented and supported and what they are capable of producing are in many ways as important as the price and performance of the finished circuits.

EDA tools like MICROWIND & DSCH, which offers a complete IC design flow, which starts with schematic building of digital circuits and then converting into verilog file for compilation in CMOS layout using MICROWIND layout compiler. We would discuss a little about the design flows little ahead in article.

Every engineer needs to verify his/her circuit before going for Fab. sign off; and FPGAs are best available platform for ASIC prototyping.

A prototype is *"A system model to test and develop the product before its final implementation."*

Field Programmable Gate Arrays (FPGA) are build around using Look-Up Tables (LUTs) and switch matrix, and are rich in resources. Advantages like high gate density, flexibility, moderate speed, etc. gives ideal platform to ASIC designers for prototyping their designs before going for fabrication of ASIC.

Engineers can use any evaluation kits available in market with required gate density and I/O features for prototyping of their ASIC designs.

Design Flow for ASIC & FPGA

MICROWIND, which is an CMOS layout and simulation tool, offers a complete ASIC design flow, covering all necessary steps of the design flow. Figure 1 shows the design flow for ASIC along with FPGA design flow. The figure also shows the FPGA design flow which can be linked with MICROWIND design flow.

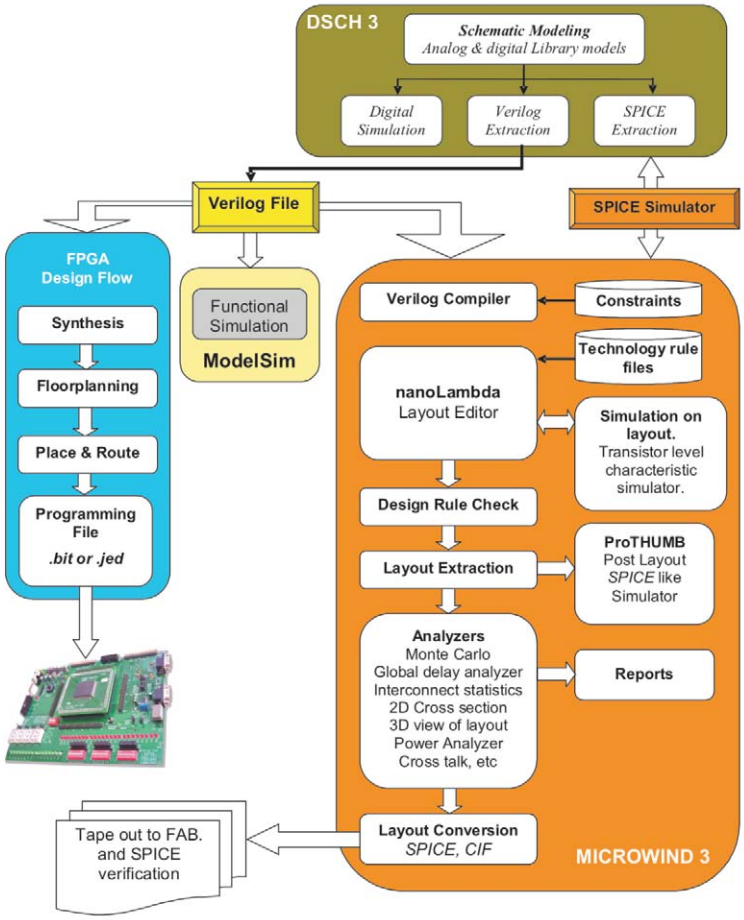


Figure 1: MICROWIND & FPGA Design Flow

DSCH : Digital schematic editor & simulator

DSCH offers lots of library symbols for schematic building, and hierarchical design features. It can simulate the digital circuits using pattern based simulator and even can generate SPICE files for 3rd party EDA tool interface. Once the schematic is ready, user can generate the verilog file for the schematic, containing all the symbols and I/O definitions.

MICROWIND: CMOS layout editor & simulator

MICROWIND supports complete IC design flow using full-custom & semi-custom design styles over a P-well process.

Building CMOS circuit in MICROWIND can be started with custom drawing of transistors or user can take the verilog file extracted from the schematic, which can be specified to MICROWIND verilog compiler and it will generate a complete error free CMOS layout in few seconds. This layout can be organized for different area sizing and different options for auto routing.

After drawing the CMOS layout, different checks can be performed using design rules files. Over the next stage, layout can be verified for functional checks using inbuilt simulator. User can analyze for power consumption, FFT, current, delay analysis, eye diagrams, etc using different plot functions in simulation window.

MICROWIND provides various parametric analysis methods for CMOS layouts. Power dissipation, temperature analysis, Monte Carlo, global delay, FEM for interconnect to be named for few.

The FPGA design flow accepts inputs like HDL or schematic, but as schematic file format and library are different for FPGA and DSCH, which make them incompatible at schematic level. But the verilog file extracted from DSCH can be specified to the synthesizer of any FPGA design tool.

Once the design is synthesized, it can simulated using Model-sim for pre-layout simulation. The synthesized netlist is implemented over a target FPGA after pin locking of I/Os for target board.

After implementation process, which covers the mapping, place & route processes, the programming file can be generated for the target FPGA. There after programming FPGA on board, the design can be verified for functional working.

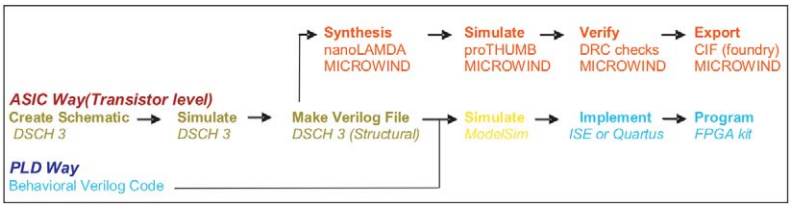


Figure 2 : ASIC and FPGA design flow using MICROWIND

The figure 2 explains the design flow steps which has to be carried from DSCH to FPGA or custom ASIC.

But many of times engineers while using the verilog file generated from DSCH have failed while implementing over FPGA. The circuit was tested, simulated but failed at FPGA prototyping.

Generally there are 3 reasons for the failures;

1) Wrong design style

2) Wrong design style

3) Wrong design style.

As FPGA are not recommended for asynchronous design styles, even your ASIC can fail for the same reason. Alongwith that combinational feedback design and gated clocks generally causes failure of design; and you would keep wondering over the reasons.

We would discuss over few suggestions and guidelines which can help in building circuits in ethical way. Over the guidelines, we would take an example which would be designed completely synchronous way.

Synchronous Design Techniques

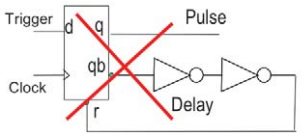
Here are some global design guideline for successful implementation of digital circuits for ASIC and FPGA platforms.

1. Use a single master clock and maintain synchronous flow of data.

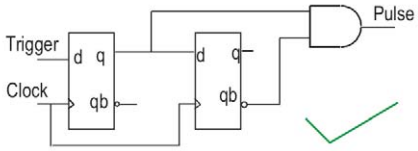
2. Use a single master set or reset. Preferably, use asynchronous resets because they work independently from the clock. When an asynchronous reset establishes the initial state, it puts the entire circuit into a known state and helps make logic simulation and manufacturing test easier. Keep in mind that CMOS ASIC technology prefers active-low asynchronous set or reset, but often FPGAs use active high.

3. Avoid race conditions on de-asserting concurrent set and preset signals. You cannot predict in simulation how the flip-flop will behave when both set and reset are de-asserted close in time.

4. Do not use delayed logic or monostable pulse generators, which relies on delays for its operation (they are unpredictable in ASIC & FPGA). Instead use synchronous pulse generators which have known timings and does not generates glitches.

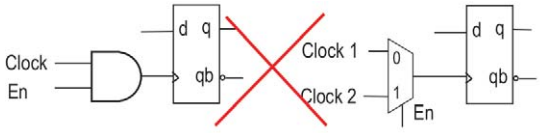


Incorrect style of Pulse generator

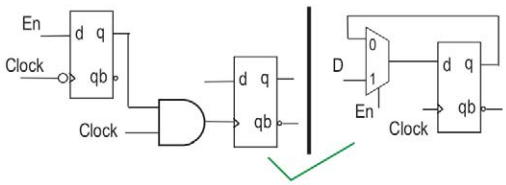


Recommended Synchronous Pulse generator

5. Use clock-enabled flip-flops. Do not use gated clocks. Synthesizing the gated clocks is dangerous. This configuration is very sensitive to glitches and simultaneous switching inputs, and may even delay the clock. Further in the design process, problems with testability may also occur.



Gating of clocks or muxing different clocks



Recommended gating styles and enabling

6. Use clock-enabled flip-flops for clock division. In many FPGA implementations, ripple clock dividers are popular. Not only can ripple clock dividers cause problems with EDA tools, the generated clock will experience a phase delay.

7. Use clock-enabled flip-flops to avoid glitching state decoders. FPGAs are sometimes tolerant when a state decoder goes through "11" while changing from "01" to "10." To ASICs, this causes implementation-dependent glitches. Using clock-enabled flip-flops not only avoids glitches but also adds no additional clock delays.

8. Have resets and transition states for Finite State Machines. Although FSMs are usually synchronous, they still can have issues during the conversion process. Make sure there are no dead states because during power-up the FSM can enter an unused state. Make sure reset is also available on your FSM to make life easier during simulation and test vector generation.

9. Synchronize all asynchronous inputs in your FSM. When input data and clock change at the same time, setup and hold time requirements may cause metastability problems. It is recommended 2 serial synchronization flip-flops to remove metastability problems on asynchronous inputs.

10. Avoid latches; use flip-flops instead. Latches cause complications with static timing and timing-driven layout tools. Latches are difficult to analyze, and the gate savings between a latch and a flop are less important with submicron technology.

11. Do not use combinational feedback loops. They would surely would cause failures while FPGA prototyping. Also setup and hold times cannot be determined by logic timing analysis or simulation.

12. Avoid the use of three-state logic in the core of your design. Use muxes instead.

13. Do not use clocks generated from combinational feedback circuits or logic. When clocks are generated by feeding clocks through some gates in the FPGA, the frequency will change during the ASIC conversion. The timing changes because the technology inherent in the ASIC is different from that in the FPGA.

The above 13 guidelines to circuit design would always ensure that you have followed the ethical method of designing circuit for FPGA and ASIC implementation.

Guideline for Schematics

Many of time while drawing schematics, engineers face hurdles over management of schematics and land up into problems later. Here are some basic guidelines to follow for successful management of schematic and design using DSCH schematic editor.

1. Use multiple pages or symbols for big and complex circuits, as it gets difficult to keep track over signals and logic.

2. Use hierarchical approach using symbols of tested blocks. Engineers can create symbols of tested circuits and use them in another page.

3. Do not mix analog symbol with digital, as FPGA do not supports analog symbols used by DSCH.

4. Keep limit on the number of I/O ports for schematic symbols, which can help in ease of interfacing the symbols with other ones.

5. DSCH uses all basic primitive components used in verilog LRM, except D-Flip-flop and Mux which are added extra in library. So user has to use components for D flip-flop and Mux while implementing in FPGA.

Implementation of 8-bit Serial Adder for FPGA Prototyping using DSCH & MICROWIND

To understand the design flow and safe design techniques, we design a serial 8-bit adder, which would use a single bit full adder to perform addition of two 8-bit numbers. The adder would generate 9-bit result after 8 clock cycles. By serial addition we get the benefit of maximum frequency and re-usage of resources.

Architecture:

The block diagram (figure 3) below shows the implementation technique of the serial adder.

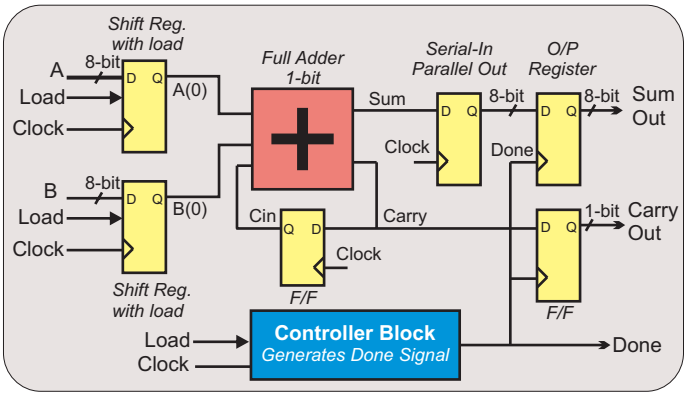


Figure 3: Block diagram of serial adder

Working:

The serial adder works with control signals like load, Reset and Clock. On active high load signal, it registers the two 8-bit numbers in shift registers, and when load goes low, it starts shifting right the data in shift registers.

Parallely, a 1-bit adder keeps grabbing bits from the LSBs of both shift registers and perform addition on those bits along with carry generated. The sum is stored in another 8-bit shift register which keep on shifting right from MSB.

Implementation:

Figure 4 shows the implemented circuit of serial adder using schematic diagram. This circuit works with single clock with synchronous flow data. The load for shift registers works as enable using multiplexers at 'D' input. The carry generated from 1-bit adder is fed back to carry input using a flip-flop, which avoids the race conditions and ensures the correct operation of circuit.

The clock network is not gated anywhere, circuit is build around with logic at only 'D' input of the flip-flops.

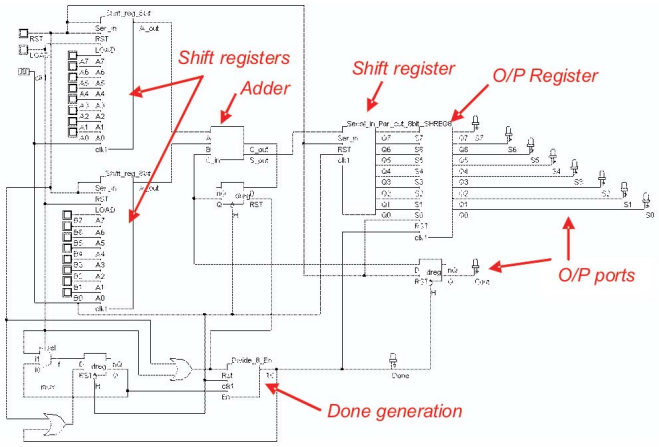


Figure 4: Serial Adder implemented in DSCH

Eventually after 8-clock cycles when the addition is performed, the controller generates a done signal, which indicated the end of addition operation and also stores the results in O/P registers for user access.

Results:

The resources consumed by Serial Adder on FPGA are as follows:

Implementation Results from Xilinx ISE for *XC2S200* FPGA

Number of Slice Flip Flops	: 31 out of 4,704
Number of 4 input LUTs	: 33 out of 4,704
Number of bonded IOBs	: 28 out of 140
IOB Flip Flops	: 9
Clock Frequency	: 216.6 MHz

The serial adder utilizes following resources when implemented on silicon:

PMOS Transistors	: 354
NMOS Transistors	: 434
Surface area	: 2682.8 um^2 (45nm technology)
	: 16012.9 um^2 (0.12um technology)
Clock Speed	: 1.8 GHz (45nm technology)
	: 1.2 GHz (0.12um technology)

Synchronous way of design always safe guards the circuits from glitches and timing errors.

We saw an illustrated way of integrating design flows for FPGA and ASIC which can be easily achieved using DSCH + MICROWIND + ISE. Only engineers have to follow few guidelines for successful implementation of circuits.