The Business Of Startups,Technology and Digital Healthcare

ASIC DESIGN

Debugging Logic Synthesis & Timing Optimization QOR issues

MAY 13, 2012 | KBULUSU | LEAVE A COMMENT |

Lets face it. You have run the logic synthesis/physical synthesis tool and you have a problem. You havent met slack. Now what? where do you start? Each issue depends on the design/process node, I'm going to keep it simple and will list few pointers/tips as to where we should be looking for. These are just for guidance and you as Designer/CAD engineer/applications engineer have to dig deep and find the root cause. I'm assuming you have stopped the flow right after global placement and routing and havent entered CTS (ofcourse it doesnt make sense even to do CTS when you failing timing)

Before , we start discussing about debugging timing issue/QOR, I'm assuming that the floorplan is of good quality. Please remember that a bad floorplan can give you a bad QOR no matter how hard the tool tries. Dont even both to do any debugging. Correct your floorplan first. If you are still in the early exploration phase, then dont complain about QOR, but rather concentrate on the correct by construction approach to give you best results.

Lets use the old and well known divide and conquerer approach. Ok now. lets break down the problem into 2 areas.
1. Your logic synthesis tool did the good job and its your physical synthesis which made the things worse.
2. The QOR after your logic synthesis is already bad.

Below is a sort of check list like what you want to do if you have issue with the above 2 items.

A1) What was the critical path looking like ? Is is a datapath or Register – IO path, IO-IO path or some macro-reg path?.

Look at the timing histogram and see how many paths fail .This gives you a good idea on how bad the timing looks like on your design. Check the top 15-20 critical paths. If it is a IO path, check the IO constraints. Also, relax the IO margings sothat they dont fail and become critical, do incremental timing optimization . Now check again, how the timing looks like. Is the critical path a reg-reg path? If it is a reg-reg path, then check step 1.1b

A2) For register-register path, check the detailed timing path and see
A2.1) Check if its a High Fanout issue, and if yes, whether the path has been buffered/cloned correctly?
A2.2) Also, check if the path is overbuffered/under buffered?
A2.2) Check if the tool has picked up correct drive strength cells? Yet times, tools dont upsize/down size the cells correctly and as a results, more buffers/inverters are added making the timing worse. Remember in 65nm a buffer has around 50ps delay.
A2.3) Also check if the tool has picked up a multi-stage cells. It is always a good idea to give the freedom to the tool to pick a cell and buffer it rather allowing the tool to select a available multistage cells. If you are trying to extract every pico second out of the tool, you might want to check this.

A3) Check if the tool complains about congestion. If you see congestion, then you might want to check the utilization. A quick visual inspection of the floorplan will tell you if there is more space available in the primary inner shape for the tool to move around. If this is the issue, then you have a placement issue.

A4) If Cloning/Buffering doesnt seem to the issue, then cross probe the timing path into the layout window using fly lines and see how the path is laid out. Is the path very long and jogging all around. If yes, then its a global placement/routing issue.

A5) Check the macro placement. How does it look? Does it look optimal? Did you create the halos around the macros? What about Placement blockages? If these are missing, please provide them and re-run the physical synthesis.

A6)If you are utilizing the auto floorplan capabilities of modern tools, then better check the quality of floorplan. Many tools have issue in creating a good and optimal rectilinear floorplans.

A7)If all else looks good, check the number of logic levels for register-register timing path. If this seems to high or suspicious, then check the logic levels at the of logic synthesis. IF you still the same thing there, then you have a logic synthesis issue rather than physical synthesis issue.

A8) To debug a logic synthesis issue, check the timing for the same path you see at the end of physical synthesis in front end STA. Check if it datapath or a control path or some kind of distributed logic sitting and getting shared between two modules.

A9)For datapath, check if the tool has picked the correct architecture. For example in case of adders,it might have selected ripple, but may be it could have selected carry-look ahead adder. similarly check if the tool can pick better architectures for other datapath components

A10) It is a control path, then check how the logic is being written in RTL. Whether if it is a deeply nested if-else logic with mutually exclusive conditions. should we really create a deep mux chain. How was the case logic written. May be the tool is inferring a big shifter, but only partial shiter is used. So the tool unnecessarily created the big shifter logic there.

A11) Yet times, it is problem of optimization itself. May be the tool couldnt have knocked off the extra registers by doing more agressive constant flop optimizations and dead code removal .

A12) Some times mistakenly users set unnecessary dont touch (synopsys) or force keep (magma) or they mess with the configs where they insist the tool the retain floating logic. Be careful in what you want to retain or what can be knocked off.

A13) Yes, the world is not flat always . Some users want to keep all the hierarchy. You dont want even that. Many optimization alogorithms works best when there is no hierarchy. Whenever there is a hierarchy, the scope of the optimizations is limited to within that module. So, only retain the hierarchy on which the timing contraints or present or when there are special requirements from other 3rd party tools regarding maintaining the hierarchy they introduced. So, check if any of the logic in the critical path can be flattened.

A14) Sometimes hiding the high drive strength cells in the library or preventing the tool from using very complex gates like XOR/XNOR and in some cases AOI/OAI cells helps to improve the timing. But this should not be done blidnly. Check the library and the cells the tool is picking. Then decide whether selective hiding is the way to go.

A15)Over Synthesis: Many users blindly push the tool to meet a high target slack in the design. I'm not referring to the clock uncertainity normally you account for. This target slack is applied only for front end synthesis. Be reasonable in what you want the target slack to be. Normally 15% of the clock freq is decent enough.

A16) Over Constraining : This is a setup margin you apply all across the flow till CTS is done. Again be reasonable in how much you constrain for. Over contraining can sometime break the alogorithms. The optimization alogorithms sees more negative slack that it really is and so inorder to meet slack, it tries over buffering/cloning/bad sizing and gives it up. Just for the record, this is not a bug in the tool. But it has to do more with constraining the design correctly.

A17) One more thing to check for is the slew limits and fanout limits and check if the tool is honouring them.

A18)Some times, because of the incorrect false paths or multi cycle paths set , you are mis guiding the tool. Remember folks, over exceptions kill the design. Dont set a false path unless it is needed . Perhaps setting a multi cycle path is way to go.

A19) Perhaps this should have been mentioned in the beginning. Quality of Contraints dictate your timing results. Bad constraints leads to bad QOR. So check your timing contraints before you start your timing analysis. There are many tools out there which can help you in this. As a thumb rule,

A19.1) check your IO constraints,

A19.2) check your exceptions (multi cycle/false paths)

A19.3) Check if there are unconstrained nodes in the design? You should not have any uncontrained nodes.

A19.4) Check if there are more events happening on a given node? Say 12 or 16 timing events happening on the same node is not good sign. Check the node.

A19.5) Also check the timing event density . This will tell you if you have over lapping or conflicting contraints or if you have high number of timing events in the design.Either way, its not good. For example, some 3rd party DFT tools whether they write our post scan SDC, it some times large timing events on some nodes and this causes havoc on timing algorithms.

A19.6)Also check if there are any nodes where there are zero timing events. This is not same as uncontraining the design.

A19.7)Check the clock definitions and units .

A19.8)Check the generated clock definitions and whether the source clock is mentioned correctly.

A19.9) Check to see if there exists any cases in the design where clock becomes data. If yes, then timing analysis tools, will treat this as data node as opposed to clock node .

A19.10)Check if the case analysis contraints have been setup correctly.

A19.11). If you have a clock gating cell say CKG1 between 2 registers say FF1/Q and FF2/D , then the tool will see two paths : path from FF1/Q to CKG1/EN and CKG/OUT to FF2/D . But in reality, there is only one path FF1/D to FF2/Q and setup checks are done on FF2/Q. So, you have to tell the tool somehow to consider the delay through the clock gate and that the clock cycle time is from FF1/CK to FF2 . So the way you do it is you want to apply a negative margin at the clk pin of the clock gating cell and setup margin on the clock gating output pin and constrain the path.

A19.12) The cleaner and better the constraints are , the timing results will that much better.

Misc Tips: It is always good idea to study the library . It gives good idea on what cells and of what strengths are available in the library. It helps to fine tune your optimization and guide your implementation tools.