The Business Of Startups,Technology and Digital Healthcare

ASIC DESIGN, **EDA**

Logic Synthesis Primer

MAY 13, 2012 | KBULUSU | LEAVE A COMMENT

I have recently seen many folks asking questions like "How to write a synthesis script" or "Whats is in a synthesis environment" . Many freshers right out of the school claim to know about synthesis but in reality doesnt have a clue where to start.

When I refer to Synthesis in this post…I mean Logic Synthesis and I mostly cover only logic synthesis and it doesnt include any STA (Timing analysis). I will cover some points which overlap between logic synthesis and front end timing optimization. I will write a seperate post on Front-end timing optimization and Physical Synthesis (Floorplan , Global Placement and Routing , CTS).

Synthesis is not just a script you write for a specific tool. It is actually much more than that. I have seen many folks who loosly couple it with a specific tool like Synopsys DC or Magma Blast Create. It is in a sense a methodology which evolves over the time by doing many runs as the block/chip evolves. Before you start synthesis , One has to know

1. What are the area goals? If area is one of the important criteria, then one needs to know what are the area reduction techniques available from the tool . I'm assuming that RTL has conforms to best design practices. Check the number of logic levels and cells used after first iteration. Accordingly decide what your strategy for area reduction should be. One benefit of reducing the area will help to comeup with a relatively smaller floorplan. Ofcourse most EDA folks prefer bigger floorplans sothat their floorplanning or placement tools can easily do the job 😃 , easily avoid the congestion and cross-talk issues :). But I strongly suggest synthesis folks to have some understanding of physical synthesis.

2. What are the power goals? Some tools have advanced power savings schemes in synthesis itself like Power gating flops/Retention flops. They do this when the RTL designer uses some special pragmas in their RTL code and they capture this when the tools parse the RTL.

3. If clock gating allowed? Are there any modules/blocks for which clock gating has to be disabled? You need to understand why it has to enabled/disabled and should be aware of its impact. You should know if the technology library has the support for ICG (Integrated
clock gating cells) .

4.How much hierarchy has to be kept? Keep in mind that Logical Hierarchy is different from Physical Hierarchy. When you decide to maintan hierarchy, it has to be considered that some optimization algorithms are limited by the module hierarchies/boundaries and so care should be taken as if and how much QOR can be sacrificed.

5. Is flattening allowed ? The answer to this question depends partly on the decision you make on the above question. If yes , is it allowed on entire design ? If not, can you atleast do flattening selectively. Other relevant information you need to know is , if rtl inferred models can be flattened .

6. What is your DFT strategy and methodology. Many might wonder why is it important to consider at Logic Synthesis stage. This is especially important when you plan for DFT during RTL development. Like you might have declared the test/scan ports in RTL and since scan insertion will not be done till synthesis has been done, many optimization algos in the synthesis engine will see them as floating and will blow them away. So, you might need to instruct the tool not to touch them.

7. Resource sharing and Operator Merging : If the these options are available in the synthesis and if you dont have any constraint or reason for not using them, then it is highly recommended to take advantage of these . But care has to be taken as some formal tools either dont have good support for this or dont support them at all .

8. Datapath architecture selection : Some advanced synthesis tool allows you to configure/pre-select the datapath architectures . If timing is critical for a particular block, then you might want to overide the area optimization steps by selecting the fastest architecture available for all the datapath components in that particular block. Do remeber that selecting fastest architecture might blow up area sometimes.

9. Formal Verification : FV (Formal Verification) tools dont do agressive optimizations (or should I say, it is not a good idea to so 😀) as synthesis tools do. So, it is highly important that you let your FV tool know about the synthesis options when exists & possible. You should try to mimic the synthesis env in your FV environment. Else you can see some false failures.

10. Hard Macros : When macros are used and if some of the inputs/outputs are unused, they might be removed. So if any macros or hard instantiated gates are present , you should set the appropriate commandslike force keep or set_dont_touch .

11. Spare gates/registers : When spare registers are described in the RTL itself, synthesis tools should be instructed to preserve them else they are treated as a part of unreachable registers and might be thrown off during synthesis (deadcode removal). Some people use spare registers in the backend and sprinkle them evenly.

12. It is important to analyze the technology library for the cell delays and area. One another important factor most people forget is to consider the effect of EM (electro-migration) and yield . All the bad cells (which have high delays , or bad for EM or yield, bad area) should be hidden or disabled from being used by synthesis engine. Forgetting to disable cells bad for EM or yield effects timing closure with cross-talk/SI during backend.

Apart from this, make sure all complex cells like AOI, OAI, Full Adders and Half Adders, XOR etc are available to the synthesis tool. It helps save area and increases drive load capability resulting less buffering.

13. Dont use highest effort levels in synthesis by default (unless you know what you are doing). Some optimization algorithms might hurt your design by doing agressive optimizations. Synthesis knobs have to be used with care and by studying what it does to the design.

14. Designware usage: Sometimes, it makes sense to use Designware components in RTL . Make sure the synthesis tool used can detect the Designware components , understand and synthesize them. Some synthesis tool vendors change them to their equivalent models (for example, Lavaware components from magma) . If not , you might need to black box them and read in the gate level netlist of those designware components after synthesis is done.

15. Pipe-lining : Almost all synthesis tool support this . So where necessary, the designer or synthesis expert has to know which block/module needs pipe-lining ; how many stages are required and what is the latency at each stage etc.

16. Re-Timing : Sometimes when RTL has those designware or lavaware components , some synthesis tools automatically apply re-timing and some synthesis tools require you to explicilty set the relevant re-timing configs . But keep in mind that re-timing is not supported that well in FV tools.

17. Dontcare optimization : If the RTL contains dont-cares (X) , many synthesis tools allow you to choose whether you want x to be treated as “0” or “1”. I suggest it would be better if we leave it to the synthesis tool to decide. Most dontcare algorithms select the value of x which will result in smaller ckt area if area optimization is enabled or better ckt with better timing if timing mode is enabled.

18. Clock Edge mapping: Some synthesis tool map to neg edge flops and add a inverter if they see that is has better area savings than pickingup a pos edge flop. Some design methodologies especically back-end teams dont prefer this sometimes. So, you need to set the configs accordingly.

19. If there are any complex cells like Full Adders, Half Adders etc with multiple outputs in your library , then most synthesis tools dont utilize them and so if your want to synthesis tool to use them , then you have to hand instantiate them in RTL. Remeber these cells will be timed, but will not infered or decomped to simpler cells/logic during optimization

phases .
With all these said, I cant stress enough how important it is for RTL coders to follow best practices. There is lot of information out there or one can refer to STARC methodology guide or Design-Reuse methodology manual for information .