CMP United Business Media
EE Times

EE Times: Design News

Cluster-based approach eases clock tree synthesis

Udhaya Kumar
(11/14/2005 9:00 AM EST)
URL: http://www.eetimes.com/showArticle.jhtml?articleID=173601961

Clock network design is a critical task in the design of high performance circuits because both the performance and the functionality of the circuit depend directly on the characteristics of the clock network.

The physical realization of the clock tree network constrains the chip design, and it directly impacts the control of clock skew and jitter. Further, the clock network impacts overall chip power and area because of the large number of buffers and repeaters that are inserted during clock tree synthesis.

There are direct impacts on chip timing as the clock tree drives the overall timing of the chip. Crosstalk from the clock grid affects circuit speed. And the speed and accuracy of the clock net directly contribute to the minimum cycle time of the chip.

The speed of a design is often its costliest component. But clock skew is an important factor in deciding a circuit's cycle time. To achieve optimal clock path delays for high-speed designs, the skew should be reduced — or, alternatively — utilized to the maximum extent possible within the defined cycle time.

In earlier process nodes, choosing clock skew was an optimization trick. But in deep sub-micron (DSM) processes it is a challenge, as it is difficult to control the delay in the clock path due to process variation.

This article discusses strategies for skew management in DSM designs. It covers the role of skew in a design, the types of clock trees used in current technology, the effective use of customized cluster based clock-tree synthesis (CTS), the merits of cluster based CTS for skew controlling and, finally, realistic delay consideration for static timing analysis (STA).

Deep submicron effects

In the deep submicron era, design complexity increases each time there is a change in the technology. A shrink in the feature size, a reduction in operating voltage, dual threshold libraries, multi-voltage or multi-clock domains, diminished gate oxide thickness and manufacturing inaccuracies are examples of technology changes that lead to big challenges in achieving the desired performance.

The nature of the problem has changed as well. With the scaling of technology and die size, transistor speed has become less important and the interconnect delay now contributes 70~80% of the cycle time. Even so, the faster transistors provided in each successive process generation have led designers to expect that cycle times will shrink with each new process.

Further complicating design, the operating supply voltage at each new process node has been reduced to save the power and protect the increasingly thin gate oxides. With each reduction, problems such as ground bounce, cross talk, glitch propagation and so forth were exacerbated because of low noise margin and increased leakage power. These problems, in turn, can force designers to slow down the slew rates and clock rise/fall times in their circuits.

But as a generalization, the most difficult part of the implementation of a design today is not one of these problems. It is managing the clock skew, with its profound impact on circuit timing.

Role of skew in a design

Any factor that contributes delay to a clock net — a mismatch in the RC delay or buffering the clock network, for example — can contribute to skew. Variations in manufacturing also can contribute unwanted skew in the clock network. Since the maximum clock skew sets the lower bound for the circuit's clock period, the unwanted skew caused by process variation becomes the bottleneck for improving the clock frequency in high-frequency designs.

Let's start examining the problem by stating a definition. The clock delay of a register is the time needed for a clock-edge to propagate from the clock source to the register. Because of the net delays and/or buffer delays (together here called "insertion delay"), the clock edges reach the registers at different times. That is, each register receives the clock with a different insertion delay.

By definition, the clock skew is the difference between the longest insertion delay and the shortest insertion delay of a clock network. If d_l and d_c are the clock arrival time of launch and capture registers, then $d_l > d_c$ leads to negative skew and $d_l < d_c$ leads to positive skew. To make sure the clock operates within the required cycle time,

($d_c - d_l$) must be no more than (min logic delay + register delay – hold) for hold

And

($d_l - d_c$) must be no more than (cycle time – (max logic delay + register delay + setup)) for setup

One way to meet these constraints is simply to make the skew as close as possible to zero. But to achieve zero skew, the design needs to have a huge amount of buffer insertion and wire sizing, to make the insertion delay equal at all registers. Clearly, this may lead to unwanted area and power consumption.

Alternatively, a design may accept a small amount of skew in order to avoid blowing up area and power consumption. Controlling such non-zero skew is the critical part of clock tree synthesis. This article explains cluster-based clock tree synthesis, which delivers an optimal result on skew control.

Types of clock trees

There are many clock tree structures used widely in the design industry, each of which has its own merits and demerits. We will discuss four structures in this article: H-tree (figure 1), balance tree (figure 2), the spine tree (figure 3) and distributed driven buffer tree (figure 4).

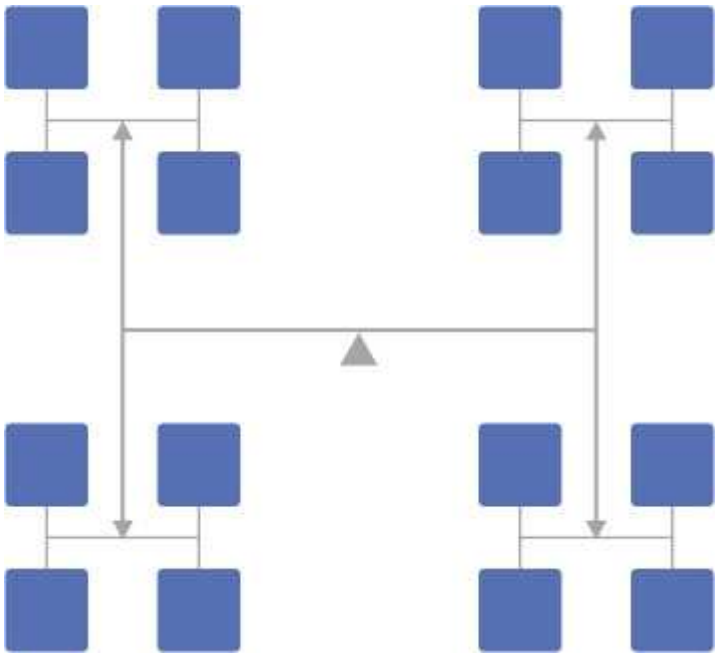

Figure 1 — Because of the balanced construction, it is easy to reduce clock skew in the H-tree clock structure. A disadvantage to this approach is that the fixed clock plan makes it difficult to fix register placement. It is rigid in fine-tuning the clock tree.

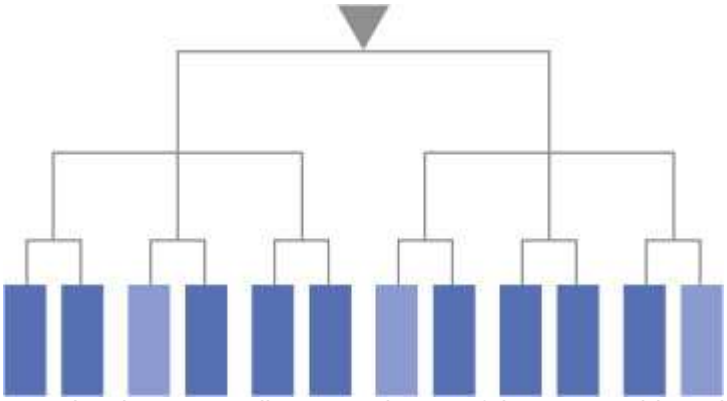

Figure 2 — Balanced tree makes it easy to adjust capacitance of the net to achieve the skew requirements. But the dummy cells used to balance the load increase area and power.

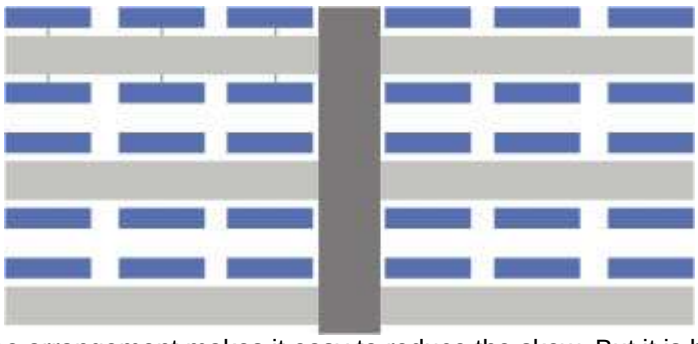

Figure 3 — The spine tree arrangement makes it easy to reduce the skew. But it is heavily influenced by process parameters, and may have problems with phase delay.

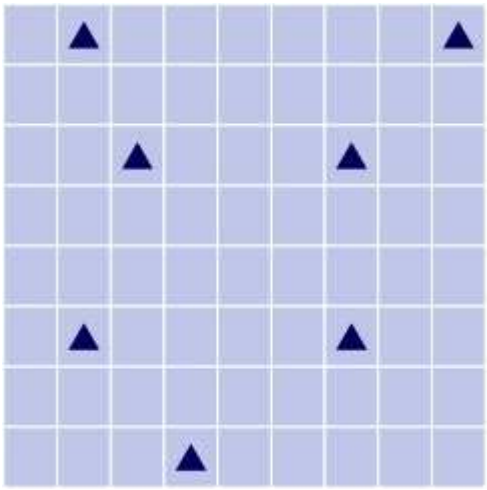

Figure 4 — Distributed driven buffer tree. Distributed buffers make it easy to reduce skew and power. Clock routing may not be an issue. However, since buffering is customized, it may be a less area efficient method.

Cluster based CTS

A better plan in principle would be to individually balance and distribute the clock from the source to each of the logic elements. If we were simply to lay out the logic and then route a clock signal to each logic element, using constant wire width and no buffers, then obviously the register furthest from the source will receive the clock with the greatest delay.

Delay can then be equalized by adding buffers/repeaters or lengthening the wires in the shorter nets so that the clock reaches all the registers at the same time. But this is not always true in practice, because in real life routing the clock directly from the source may not be possible in high speed and multi clock domains. A practical approach for such problems can be customized cluster-based clock tree synthesis.

In a customized cluster-based clock tree plan, the logic elements operated in a single clock domain or the logic with the same input timing registers are combined together to form a group. A reference register for each group is selected to establish a reference arrival time.

Initially these clusters are placed to meet the latency requirements based on the clock insertion delay from the source. These clusters can be individually optimized to minimize skew, so that the skew will be within the allowable range for the desired cycle time.

The cluster level routing can use any of the routing topologies mentioned above, based on the priorities of the design. For example, a cluster sensitive to skew can use H-tree or balanced-tree designs.

Obviously, the trade off between power, area and amount of buffers added into the network are to be preplanned before selecting a method for each group. It is not advisable to limit the performance based on the clock tree topology.

To implement the top-level clock network, it is better to use multiple-driven clock pins to form a spine over the entire clock tree network for the chip. This reduces the length of wire for buffers, which in turn means the design will need very few levels of buffer hierarchy. So this approach reduces the skew as well as latency. The following methods can be used for clock tree plan at either cluster level or top level to balance in-between clusters.

At the top-level clock routing, the load balancing of clusters will result in less skew. The balance points at the top-level clock tree are defined, and wire length is also considered for identifying the optimal place for buffer insertion to improve the slew. Either a mesh topology that spans all the clusters or, alternatively, a classic tree structure with scaled metal widths in the branches can accomplish this purpose.

Advantages of cluster based CTS

In a deep submicron process, millions of gates and very high frequencies — sometimes multiple GigaHertz — are becoming normal. In order to achieve such high frequency requirements, the clock tree network

needs to be very well planned and elaborated. The designers should be able to plan for the skew requirements to achieve the minimum cycle period, instead of trying to force the skew to zero in the presence of perhaps poorly-characterized process variations.

One of the biggest advantages of doing cluster-based CTS is that the delay due to voltage drop in the interconnect can be modeled or incorporated into static timing analysis. A voltage drop of 10% of Vdd may lead to more than 15% delay variation in nanometer technologies.

Typically the voltage drop will be more severe at the center of the chip, so the standard cells characterized for either worst case or best can not give accurate delay values — they cannot account for voltage variations that are happening on the fly. Though simultaneous, mixed min-max or on-chip variation (OCV) kinds of analysis are derived for such situation, there is unfortunately no static timing engine that considers the delay of standard cells due to the variation in the voltage. A cluster near the center of a chip, for example, may not receive 100% of Vdd, so the delay also will vary at the center since the voltage of those standard cells are less than Vdd.

When we use a cluster-based customized clock tree synthesis method, we always treat the cluster based on its priority within the chip. If the voltage drop at the center of the die is a potential problem, then the clusters at the center would always have timing priorities with setup and hold margins sufficient to prevent the voltage drop from causing timing problem. There always needs to be a trace-back analysis with actual voltage drop numbers to define the setup and hold margin. In such cases, the clusters can be operated based on the weighting of timing, area, power, and other factors.

Conclusion

For many designers, timing closure is the major issue in high-speed designs, which leads to any number of iterations and schedule-consuming tasks. At 350nm or 250nm technology, the EDA companies understood timing closure issues and they updated their tools to a certain extent to meet the designers' needs.

But in the current scenario, timing closure is the major part of the design cycle. Because of time-to-market pressure, new design methodologies at the system integration level like system on chip and network on chip are using on-chip buses and multiple clock domains. This kind of design requirement pushes designers to use new design tricks like timing driven placement, useful-skew concepts, or customized cluster-based CTS.

Since there are plenty of IP cores available in the market, simply designing an SoC may not consume much time. But after the blocks have been selected, the physical implementation into hardware consumes more time and effort. We may have to develop a customized design flow for each and every stage of the design cycle to address the current challenges in chip implementation.

Clock tree synthesis is a case in point. As we have discussed throughout the paper, the designers should not be limited to any fixed topology of clock routing algorithms. Rather, they should be encouraged to mix up any of the described clock routing schemes into a single chip. This customized cluster-based clock tree synthesis utilizes the best topology to meet requirements like skew, area, and power at every stage, and it improves the top-level system performance.

Udhaya Kumar is project manager for physical design at eInfochips Ltd. He has over 8 years of experience in chip design. He started his career in chip design as a verification engineer and then moved to physical design. He has worked as layout implementation engineer for SoC designs like digital video processors, ARM/RISC processor based SoCs, and communication controllers for aeronautic applications.