

Partitioning, Floor Planning, Detailed Placement and Routing Techniques for Schematic Generation of Analog Netlist

Bikram Garg
Bikram_garg@mentor.com
Mentor Graphics, India

Ashish Agrawal
Ashish_agrawal@mentor.com
Mentor Graphics, India

Rajeev Sehgal
Rajeev_Sehgal@mentor.com
Mentor Graphics, India

Amarpal Singh
Amarpal_singh@mentor.com
Mentor Graphics, India

Manish Khanna
Manish_khanna@mentor.com
Mentor Graphics, India

Abstract

The schematic generator has become a very powerful debug tool in EDA in all of the flows whether it is RTL, Synthesis, formal verification or at the layout level.

The paper talks about various challenges in showing a “good” schematic for a spice netlist and then proposes novel algorithms used to generate the schematic. The paper also links various schematic generation stages with techniques used by P&R tools such as floor planning, global routing, and detailed placement and routing.

The transistor level schematic generation differs from the gate level schematic, in that a set of transistors in spice netlist forms a logic and these set of transistors need to be placed together in some symmetry to represent the logic. The paper addresses the challenges that lie in placing and extracting these components which represent logic or a current flow or cascade effect.

1. Introduction

The increase in circuit complexities has increased the number of components that are present in the netlist. The number of components goes on exploding as we move along the flow from behavioral to gate level and from gate level to transistor level. The design complexity increases further if we move from hierarchical to flat designs. Use of schematic viewer has become mandatory for the success of an EDA tool which requires analysis of gate level or transistor level netlist. Looking at the netlist in the form of schematic gives a designer a better insight into the design that he has tried to generate. He can correlate the blocks

written at RTL level seeing the schematic at gate level and layout level. The schematic generator has also become a very powerful debug tool in all the flows whether it is RTL, Synthesis, formal verification or at the layout level. It can help the designer to debug the issues faster and more efficiently. The schematic can also help in the documentation of the design. But to achieve the above goals we need to generate a “good” schematic.

Several methods have been reported to get schematic-like information from the CMOS circuit netlist [1,2] and there are also many approaches which talks about automatic gate level netlist schematic generator[3] but there is no complete approach which tries to bring together the two concepts and generates a schematic level view of a transistor level circuit using circuit network partitioning.

The partitioning algorithms we have added are novel and considered logical grouping, signal flow and visual observation to come up with heuristics. As the netlist has logic inside it, some partitioned circuit we detect can be represented as a Boolean equation. Our methods work for both CMOS netlists and bipolar netlists. For convenience we will refer to only CMOS circuits but one can easily map bipolar devices on transistors.

In this paper, heuristic algorithms used in partitioning of netlist and placement of components for the CMOS netlist, are discussed. Section 2 talks about the “good” schematic generator. Section 3 explains the initialization steps required on netlist to achieve deterministic, “good” schematic. Section 4 describes the heuristic used for partitioning of the circuit. Section 5 elucidates the placement and routing

algorithms for component placement. Section 6 discusses the placement of blocks in the grid based layout and Section 7 explains the detailed placement and routing.

2. Problem Statement

Given an Analog netlist, the aim is to generate aesthetically “good” schematic. The definition of good schematic for an analog netlist is as below:

- [1] The netlist generated should be comparable with the manually drawn schematic. This is important for the transistor level netlist because the group of transistors represents logic.
- [2] It should be deterministic. It should give the same results for two same netlists, given random connectivity and component inputs.
- [3] The signal flow from top to bottom and left to right should be honored.
- [4] The wire crossovers should be minimized.
- [5] The wires should have minimum jogs.

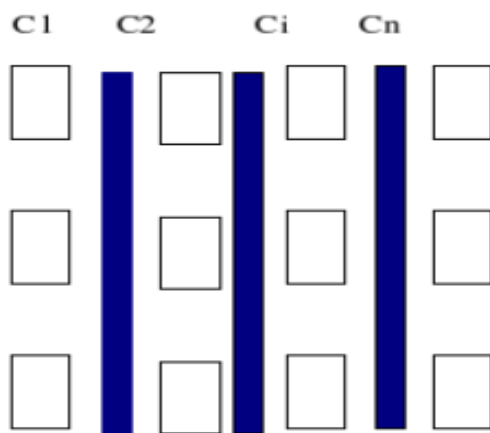


Figure 1 Schematic layout model

Figure 1 shows the layout model of a schematic. The layout is grid based with several columns and rows. The left to right ordering of components based on their net connectivity is known as leveling. The level algorithm determines the level of each component. Each component which is in the same level is assigned the same column. The rows of a module represent the relative position of the components in the column.

As the analog schematic not only contains a left to right flow from input but it also contains a top to down power/ground flow. The problem task is placing the analog devices on this layout. The broad outline of the solution is as following:

We partition the netlist to identify the individual logic blocks from the netlist. These logic blocks are placed on the grid based layout shown in Figure 1. The device components are then placed internally within the block. These blocks are then connected to each other through external wire routing. Finally the internal wire routing within the block connects the device components to each other.

3. Initialization of the netlist

This step is important for the generation of “good” schematic. On the netlist, detect the presence of analog devices. Sort the components and the nets in an order to ensure deterministic view for same connectivity of the netlist. The user can populate the netlist database in any fashion i.e. instead of adding first component say C1 he can start with populating with component C2 or C3.... So having the sorting ensures that for the same connectivity, the view is the same. This above step will ensure deterministic view of the netlist. The second important thing that needs to be done on the netlist is searching for the presence of bulk, power and ground nets in the netlist. Bulk, power, ground nets are common feature in an analog netlist. We need to determine these global nets and do special handling of these nets for routing and leveling. This will ensure optimal and minimal crossings in the schematic. Third important processing required on an analog netlist is looking for the terminal connectivity of transistors with respect to power and ground. The connection given in the netlist is correct but transistor’s source and drain terminal might be connected opposite from the viewing perspective. So, on the netlist, search for components, whose “drain” pin is connected to power or the “source” pin is connected to ground. The connection of these components is reversed to ensure a “logical” flow of current from power to ground and correct display of schematic. These steps are performed prior to partitioning of the designs.

4. Partitioning of the CMOS Netlist

We are using a set of methodologies based on the logic, visual view and flow to partition the CMOS netlist to achieve “good” schematic. These methodologies are represented in the order of the flow that we are using to partition the design.

AND-OR logic detection approach: We detect the transistors which are connected in parallel or series with each other and form bigger blocks out of it. The parallel components in the NMOS transistors represents the OR logic and the series grouping in NMOS transistors represents the AND logic. These

sets of series, parallel components are then replaced with the blocks in the netlist and we continue with the next stage of our partitioning. These blocks detected during partitioning can be represented as a functional logic. In Figure 2 shown below the pull-up network represents set of pmos S1(p3, p4, p2, p1) and pull-down network represents set of nmos S2(p8, p7, p6, p5) with each member in S1 has relation with each member in S2 {(p3, p7), (p4, p8), (p2, p6), (p1, p5)}. The expression represents the logic $!(a + b.(c+d))$ which could be deduced based on the above reasoning.

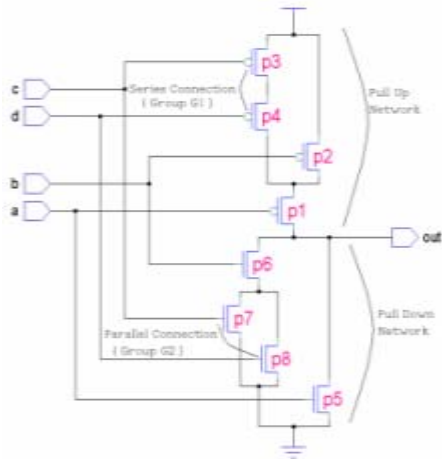


Figure 2 AND-OR logic detection

Cascading effect (visual observation)

Based on the current flow the terminals of the blocks, devices, transistors connected to the gate of the transistors or blocks and the other end connected to the common net will be represented together. This methodology will show the cascaded circuits both at the top and at the bottom. This partitioning keeps the cascaded blocks together and show the flow in a better way. This effect is more prevalent in bipolar devices.

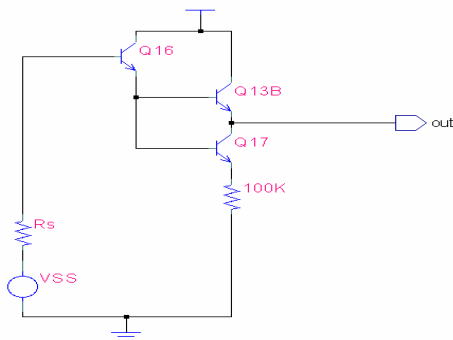


Figure 3 Cascading Effect

Driver logic effect

We are creating a different form of partitioning to represent the series group which also has an output pin connected to the gate of the transistor or a block. This partitioning helps in determining left to right flow of the blocks and provide current flow information to the placement algorithms.

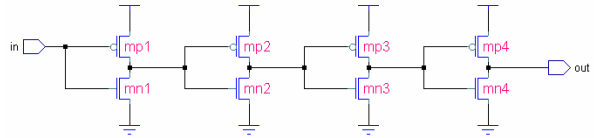


Figure 4 Driver logic effect

Transmission Gate Detection

We are detecting the transmission gates as an independent and separate part of the circuit as its detection adds logic to the circuit. If the drain pin of a pmos is connected to a drain pin of nmos and the source pin of the pmos is connected to the source pin of nmos, then the heuristic is that if the input at the gate pin of these two components is different then they together form a transmission gate. (Notice M1, M8 and M2, M9 in figure5)

5. General algorithm for partitioning the CMOS circuit

Partitioning(Graph G)

```
{
  Transmission_Gate_Detection(G);
  AND_OR_logic_detection(G);
  CASCADE_logic_detection(G);
  Driver_logic_detection(G);
}
```

The graph formed after partition is in the form

$$G(V,E) = G_{new}(V,E) = (B_i \cup \Sigma C_i \cup \Sigma D_i \cup \Sigma \text{Original Components})$$

Here B_i represents blocks formed from the AND_OR partition logic, C_i represents blocks formed from the Cascade logic and D_i represents blocks formed during DRIVER logic. There could be some original components also present. So a new graph G_{new} is formed.

6. Floor planning of partitioned blocks of the netlist

The placement of blocks plays a significant role in the routing achievement. The above partitioning algorithm used to partition the design, generates a new netlist which have blocks representing various AND_OR, CASCADE, DRIVERS partitions. We take this netlist having blocks and run placement algorithms (leveling, which will assign a column to each component/block, relative Y ordering of components in the column, this is done to have minimum crossing of the nets on the view [4] and finally an exact Y placement of components so as to have minimum wire length of the net connecting components). The generated placement of blocks/components on running the placement algorithm is the floor plan of the netlist. The coordinates generated represent the block and component positions. Using the block positions, we can now place the components present inside the blocks discussed in next subsection.

The routing algorithm is conventional one. We used Yoshimura, Kuh channel router [5] during floor planning. The algorithm basically involved track assignment based on the net connectivity in a channel. Once the tracks are determined, we know the wire points for a net.

7. Detail Placement of components in the blocks and internal routing

In the identified, blocks components are placed from top to bottom in the direction of current flow from power to ground. We had developed a vertical placement algorithm for the logical placement of components within a partitioned functional block. All the power connected components are assigned the fixed initial level 1. As identified during circuit partitioning, the components connected to these power connected components are assigned an incremented level 2 and so on. This process continues for each functional block except the parallel block for which all components are at the same level, till we reach the ground level. In this way the components constituting functional blocks are vertically leveled. The top level components (power connected components) are placed at the top. The lower level components are placed below the top level component with fixed vertical spacing. Fixed horizontal spacing is also maintained for parallel block components. This horizontal and vertical spacing is maintained to avoid net overlapping and net crossing through instances thereby producing a “nice”, logical and clean connection between the components. For the internal component placement,

we made our own wire algorithms as in the block we do not require tracks[5] for AND_OR and DRIVER blocks but we need to have good wiring to remove congestion. So based on symmetry and placement of connectivity we determined wire points inside the block.

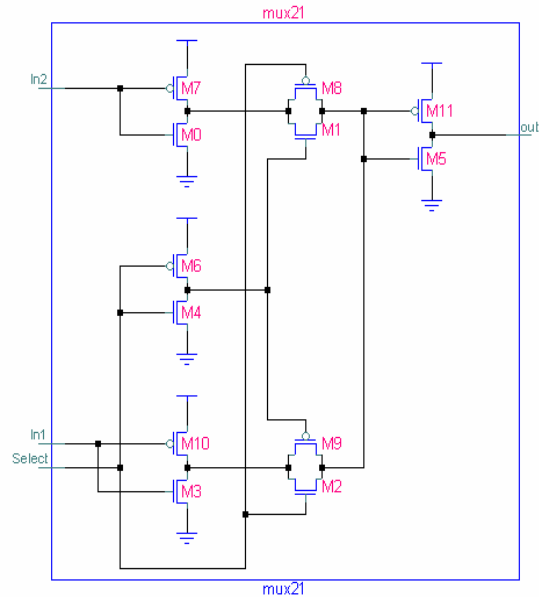


Figure 5 Detailed placement and routing

8. Conclusion

A novel method of transistor level schematic generation and logic detection has been presented in the paper. A much easier to infer visualization of a design is produced with minimized crossing and no overlapping. The compact and meaningful placement of components ensures the wire length to be minimized. The experimental results shows the effectiveness of our method.

9. References

- [1] Wen-Jeng Lue and L.P. McNamee, *Extracting schematic-Like Information from CMOS Circuit Net-Lists*
- [2] N.P. Jouppi. *Derivation of signal flow direction in MOS VLSI*. IEEE Transactions of Computer-Aided Design, CAD-6(3):480-490, 1987
- [3] B.Naveen, A.Savargaonkar and K.S.Raghunathan, *N2S: Automatic Netlist to schematic generator*
- [4] B. Kernighan and S. Lin, " *An Efficient Heuristic Procedure for Partitioning Graphs,*" Bell Systems Technical Journal, 49(2), pp. 291-308, 1970
- [5] Yoshimura and Kuh, *Efficient Algorithms for channel-routing*, IEEE Tran on CAD of ICs and systems, Vol-CAD 1,Jan 1982, pp25-35