Clock Tree Synthesis Analysis and Optimization in Physical Design Flow of Serial Peripheral Interconnect (SPI)

PEDDI ANUSHA¹, K.SATISH BABU²

¹PG Scholar, Dept of ECE, Aurora’s Technological and Research Institute, JNTU, Hyderabad, Telangana, India, Email: peddianushachowdary@gmail.com.
²Assoc Prof, Dept of ECE, Aurora’s Technological and Research Institute, JNTU, Hyderabad, Telangana, India, Email: kaapaarpu_satish@yahoo.co.in.

Abstract: Chip-level clock tree synthesis (CCTS) is a key problem that arises in complex system-on-a-chip designs. A key requirement of CCTS is to balance the clock-trees belonging to different IPs such that the entire tree has a small skew across all process corners. Achieving this is difficult because the clock tree is different IPs might be vastly different in terms of their clock structures and cell/interconnect delays. The chip-level clock-tree is expected to compensate for these differences and achieve good skews across all corners. Also, CCTS is expected to reduce clock divergence between IPs that have critical timing paths between them. Reducing clock divergence reduces the maximum possible clock skew in the critical paths between the IPs and thus improves yield. The design of the SPI is so complex that manual design would not be feasible. The only way to design and fabricate such complex designs is to use computers to automate portions of the design process. The focus of this paper is the numerous aspects of the physical design process and how those aspects are automated using computer-aided design (CAD) tools by Synopsys. The main focus of this project is to study the contemporary methods of IC Physical design and to solve the different problems of physical design with the application of Synopsys tools. Some of the main topics covered in this paper are: Design planning, Placement, Clock tree synthesis, Routing, Crosstalk and Chip finishing.

Keywords: CCTS, CAD, Design Planning, Placement, Clock Tree Synthesis, Routing, Crosstalk and Chip Finishing.

I. INTRODUCTION

This Paper provides specifications for the SPI (Serial Peripheral Interface) Master core. Synchronous serial interfaces are widely used to provide economical board-level interfaces between different devices such as microcontrollers, DACs, ADCs and other. Although there is no single standard for a synchronous serial bus, there are industry-wide accepted guidelines based on two most popular implementations: SPI (a trademark of Motorola Semiconductor) Microwire/Plus (a trademark of National Semiconductor) Many IC manufacturers produce components that are compatible with SPI and Microwire/Plus. The SPI Master core is compatible with both above-mentioned protocols as master with some additional functionality. At the host side, the core acts like a WISHBONE compliant slave device. The design of SPI is taken into consideration in the proposed paper serial to peripheral interface (SPI) [1] is a hardware/firmware communications protocol developed by Motorola and later adopted by others in the industry. Microwire of National Semiconductor is same as SPI. Sometimes SPI is also called a “four wire” serial bus. The Serial Peripheral Interface or SPI bus is a simple 4-wire serial communications interface used by many microprocessor/microcontroller peripheral chips that enables the controllers and peripheral devices to communicate each other. Even though it is developed primarily for the communications between host processor and peripherals, a connection of two processors via SPI is just as well as possible. The SPI bus [2], which operates with full duplex (means, signals carrying data can go in both directions simultaneously), is a synchronous type data link setup with a Master/Slave interface and can support up to 1 mega baud or 10Mbps of speed. Both single-master and multi-master protocols are possible in SPI. But the multi-master bus is rarely used and look awkward, and are usually limited to a single slave.

II. SPI (SERIAL PERIPHERAL INTERFACE)

A. SPI Protocol

Standard SPI [6] is a high-speed, full-duplex, synchronous communication bus [4]. For saving the chip ports and space on PCB layout, the ports of the SPI only take four lines. As this simple-to-use feature, more and more chips integrated SPI protocol. SPI protocol communication principle is very simple. It is working in the master-slave full duplex mode which usually has a master device and one or more slave devices and requires four lines whose components are SDI (data in), SDO (data out), SCK (clock), CS (chip select) at least. When the SPI master wants to send data to a slave or more slaves, it will pull the CS line low for selecting slaves, and activates the clock signal which usable between the master and the slave at same time. The master transmits the
data to the MOSI (master’s SDO and slave’s SDI) line and receives the data from the MISO (master’s SDI and slave’s SDO) line at the time. SPI is a serial communication protocol, that data is transmitted bit by bit. The clock pulse is provided by SCK and SDI, SDO is based on this pulse to making the data transmission. Data output through the master’s SDO line at the rising or falling edge of the clock, and be read by slave in the falling or rising edge followed. So 8-bit data transfer need at least 8 times the clock signal changes [3]. Based on the analysis of the SPI timing, a soft-core IP which the device number is selected by parameters is implemented. Increase little registers in order to fit 4, 8 and 16 numbers of three kinds for user to choose. The design can meet the need of high-performance. The IP [7] can be use to design SOC as intelligence property.

B. SPI Bus Interface

The Serial Peripheral Interface or SPI bus is a synchronous serial data link (fig 1), a de facto standard, named by Motorola that operates in full duplex mode. It is used for short distance, single master communication, for example in embedded systems, sensors, and SD cards. Devices communicate in master/slave mode where the master device initiates the data frame. Multiple slave devices are allowed with individual slave select lines. Sometimes SPI is called a four-wire serial bus, contrasting with three-, two-, and one-wire serial buses. SPI is often referred to as SSI (Synchronous Serial Interface).

Fig 1: SPI bus-single master and single slave.

1. Features
- Full duplex synchronous serial data transfer
- Variable length of transfer word up to 128 bits
- MSB or LSB first data transfer
- Rx and Tx on both rising or falling edge of serial clock independently
- 8 slave select lines
- Fully static synchronous design with one clock domain
- Technology independent Verilog
- Fully synthesizable

C. Serial interface

The serial interface consists of slave select lines, serial clock lines, as well as input and output data lines. All transfers are full duplex transfers of a programmable number of bits per transfer (up to 64 bits). Compared to the SPI/Microwire protocol, this core has some additional functionality. It can drive data to the output data line in respect to the falling (SPI/Microwire compliant) or rising edge of the serial clock, and it can latch data on an input data line on the rising (SPI/Microwire compliant) or falling edge of a serial clock line. It also can transmit (receive) the MSB first (SPI/Microwire compliant) or the LSB first. It is important to know that the RxX and TxX registers share the same flip-flops, which means that what is received from the input data line in one transfer will be transmitted on the output data line in the next transfer if no write access to the TxX register is executed between the transfers.

1. Modes of operation

The SPI functions in three modes, run, wait, and stop.

Run Mode: This is the basic mode of operation. It is used to run the program.

Wait Mode: SPI operation in wait mode is a configurable low power mode, controlled by the SPISWAI bit located in the SPICR2 register. In wait mode, if the SPISWAI bit is clear, the SPI operates like in Run Mode. If the SPISWAI bit is set, the SPI goes into a power conservative state, with the SPI clock generation turned off. If the SPI is configured as a master, any transmission in progress stops, but is resumed after CPU goes into Run Mode. If the SPI is configured as a slave, reception and transmission of a byte continues, so that the slave stays synchronized to the master.

Stop Mode: The SPI is inactive in stop mode for reduced power consumption. If the SPI is configured as a master, any transmission in progress stops, but is resumed after CPU goes into Run Mode. If the SPI is configured as a slave, reception and transmission of a byte continues, so that the slave stays synchronized to the master.

2. Wishbone Bus Interface

Wishbone Bus Interface is a system on a chip (SOC) interconnection method that allows integrating digital circuits together in a chip. It is flexible simple and portable way of interconnecting “IP Cores” to circuit which means that a new functionality can be added quickly and easily in the design.it is specified as a “logical bus” i.e., in terms of “signs”, clock cycles and high and low levels. It presents a Master/Slave architecture with four different types of interconnection.

Point to Point: It is the simplest interconnection. Allows only one master to communicate with only one slave.

Data Flow Interconnection: It is used when data can be processed in a sequential way. Every IP has a master & slave interface and the data “flows” through the line of interconnected cores. It is a kind of pipelining because exploits parallelism and therefore decreasing execution time.

Shared Bus Interconnection: It is the multi master/multi slave interconnection in this situation more than one master is connected, it is necessary arbitration.

Cross bar switch interconnection: It allows two or more master to communicate at the same time with two or more
Clock Tree Synthesis Analysis and Optimization in Physical Design Flow of Serial Peripheral Interconnect (SPI)

Slaves (each slave can only be addressed by one master, two masters cannot address the same slave at the same time, but it is possible for different slaves at the same time. In this case, the arbiter indicates when a master can gain access to a specific slave.

This core is an SPI and Microwire/Plus compliant synchronous serial controller (fig 2). At the host side, it is controlled via registers accessible through a WISHBONE rev. B1 interface.

![WISHBONE Interface Diagram](image)

**Fig 2: SPI interface of multi slave.**

3. Data transmission

![Data Transfer Diagram](image)

**Fig 3: Data Transfer.**

A typical hardware setup using two shift registers to form an inter-chip circular buffer (fig 3). To begin a communication, the bus master first configures the clock, using a frequency less than or equal to the maximum frequency the slave device supports. Such frequencies are typically up to a few MHz. The master then transmits the logic 0 for the desired chip over the chip select line. A logic 0 is transmitted because the chip select line is active low, meaning its off state is a logic 1, on is asserted with a logic 0. If a waiting period is required (such as for analog-to-digital conversion), then the master must wait for at least that period of time before starting to issue clock cycles. During each SPI clock cycle, a full duplex data transmission occurs:

- The master sends a bit on the MOSI [9] line; the slave reads it from that same line.
- The slave sends a bit on the MISO line; the master reads it from that same line.

Not all transmissions require all four of these operations to be meaningful, but they do happen. Transmissions normally involve two shift registers of some given word size, such as eight bits, one in the master and one in the slave; they are connected in a ring. Data is usually shifted out with the most significant bit first, while shifting a new least significant bit into the same register. After that register has been shifted out, the master and slave have exchanged register values. Then each device takes that value and does something with it, such as writing it to memory. If there is more data to exchange, the shift registers are loaded with new data and the process repeats. Transmissions may involve any number of clock cycles. When there is no more data to be transmitted, the master stops toggling its clock. Normally, it then deselects the slave. Transmissions often consist of 8-bit words. A master can initiate multiple such transmissions if it wishes/needs. However, other word sizes are also common, such as 16-bit words for touch screen controllers or audio codecs, like the TSC2101 from Texas Instruments or 12-bit words for many digital-to-analog or analog-to-digital converters. Every slave on the bus that has not been activated using its chip select line must disregard the input clock and MOSI signals, and must not drive MISO. The master must select only one slave at a time.

### III. PROPOSED THEME

CTS is the process of insertion of buffers or inverters along the clock paths of ASIC design in order to achieve zero/minimum skew or balanced skew. The goal of CTS is to minimize skew and insertion delay. Clock is propagated after placement because the exact physical location of cells and modules are needed for the clock’s propagation which in turn impacts in dealing with accurate delay and operating frequency and clock is propagated before routing because when compared to logical routes, clock routs are given more priority. This is because clock is the only signal switches frequently which in acts as source for dynamic power dissipation. To run in accordance with the advancing technologies and to know what has to be done, knowing the process of physical layout provides an insight into the functioning and physical devices that are used to design SPI or any other device used in the mother board. It involves firstly writing the code which suits to the qualities required in a serial bus or in any other design and then synthesizing it to convert into net list. The net list is used to design physical layout of the serial computer bus. This can be done with the assistance of CAD [8] (computer aided design) tools.

### IV. PHYSICAL DESIGN

In integrated circuit design, physical design is a step in the standard design cycle which follows after the circuit design. At this step, circuit representations of the components (devices and interconnects) of the design are
converted into geometric representations of shapes which, when manufactured in the corresponding layers of materials, will ensure the required functioning of the components. This geometric representation is called integrated circuit layout. This step is usually split into several sub-steps, which include both design and verification and validation of the layout.

A. ASIC Physical Design Flow
   The main steps in the ASIC physical design flow are
   - Design Netlist (After Synthesis)
   - Floor planning
   - Placement
   - Clock-tree Synthesis (CTS)
   - Routing

1. Design Netlist
   IC Compiler is a single, convergent Netlist-to-GDSII synthesis design tool for chip designers developing very deep submicron designs. It takes as input a Gate-level Netlist, a detailed Floorplan, Timing constraints, physical and Timing libraries, and foundry-process data, and it generates as output either a GDSII-format file of the layout.

2. Floor planning
   The first step in the Physical Design flow is Floor Planning. Floor planning is the process of identifying structures that should be placed close together, and allocating space for them in such a manner as to meet the sometimes conflicting goals of available space (cost of the chip), required performance, and the desire to have everything close to everything else. Based on the area of the design and the hierarchy, a suitable floor plan is decided upon. Floor Planning takes into account the macro's used in the design, memory, other IP cores and their placement needs, the routing possibilities and also the area of the entire design. Floor planning also decides the IO structure, aspect ratio of the design. A bad floor-plan will lead to waste-age of die area and routing congestion.

3. Placement
   In placement first we have to define the placement blockages. Placement blockages are areas that leaf cells must avoid during placement and legalization, including overlapping any part of the placement blockage. Placement blockages can be hard or soft.
   - A hard blockage prevents cells from being put in the blockage area.
   - A soft blockage restricts the coarse placer from putting cells in the blockage area, but optimization and legalization can place cells in a soft blockage area.
   Hard and soft placement blockages in this design, the hard placement blockages take priority over the soft placement blockages in places where they overlap.

B. Performing Placement and Optimization
   After finishing design planning and power planning, we can perform placement and optimization on design. To perform placement and optimization, use the place_opt command or choose Placement > Core Placement and Optimization in the GUI. The place_opt command performs coarse placement, high-fanout net synthesis, physical optimization, and legalization. During area recovery phase, removing cells on noncritical timing paths increases the fanout. The paths sometimes become timing-critical during post-route optimization to correct signal integrity problems. To avoid removing cells from potential timing-critical paths, you can specify a maximum fan out threshold limit for optimization by setting the psynopt_high_fanout _legality_limit variable. By default, this variable has a value of 0 and there is no high-fanout limit. If you set this variable to a nonzero value, optimizations that result in a fanout greater than the specified value are not performed.

1. Clock Tree Synthesis (CTS)
   The goal of CTS is to minimize skew and insertion delay. Clock is not propagated before CTS as shown in below Figure 4.

   ![Fig 4: CTS.](image)

   After CTS hold slack should improve. Clock tree begins at .sdc defined clock source and ends at stop pins of flop. There are two types of stop pins known as ignore pins and sync pins. ‘Don’t touch’ circuits and pins in front end (logic synthesis) are treated as ‘ignore’ circuits or pins at back end (physical synthesis). ‘Ignore’ pins are ignored for timing analysis. If clock is divided then separate skew analysis is necessary.

D. Optimization in CTO
   The different options in CTO to reduce skew are described in the following list.

1. Buffer and Gate Sizing
   - Sizes up or down buffers and gates to improve both
   - Skew and insertion delay.
   - Design can impose a limit on the type of buffers and gates to be used.
   - No new clock tree hierarchy will be introduced during this operation (fig 5).
2. Buffer and Gate Relocation
   - Physical location of the buffer or gate is moved to reduce skew and insertion delay.
   - No new clock tree hierarchy will be introduced during this operation.

3. Level Adjustment
   - Adjust the level of the clock pins to a higher or lower part of the clock tree hierarchy.
   - No new clock tree hierarchy will be introduced during this operation (fig 6).

4. Reconfiguration
   - Clustering of sequential logic
   - Buffer placement is performed after clustering.
   - Longer runtimes.
   - No new clock tree hierarchy will be introduced during this operation.

5. Delay Insertion
   - Delay is inserted for shortest paths.
   - Delay cells can be user defined or can be extracted from by the tool. By adding new buffers to the clock path the clock tree hierarchy will change.

6. Dummy Load Insertion
   - Uses load balancing to fine tune the clock skew by increasing the shortest path delay.

V. RESULTS
A. Simulation: Waveforms

Fig 5: Buffer/gate sizing.

Fig 6: Buffer/gate relocation and level adjustment.

Fig 7: Reconfiguration.

Fig 8: Dummy load insertion and Delay insertion.

Fig 9: SPI Top Module Waveform.
Fig 10: SPI Output result submodule spi_slave.

Based on the analysis of the SPI protocol get the basic structure of high-speed SPI bus, including: the clock generator module, data transfer module & shift module (fig 9 and 10). The important role of the top-level module is to ensure sub-module work smoothly. Therefore, the top of the SPI module needs the control word, the normal operation of clock generation module and data transmission & shift module. In this stage, after applying the constraints to the design, a technology mapped gate level net-list is obtained as output, and also the area, timing, qor reports are generated at this step of the procedure. The schematic view of the design is shown in figure 12. Net-list schematic is the representation of the design in the form of gates and flip-flops.

B. Synthesis Result

The figures from 11 to 13 show the synthesis results and generated netlist of a SPI_top.

C. Clock Tree Synthesis

Fig 14: Optimized CTS.
Clock Tree Synthesis Analysis and Optimization in Physical Design Flow of Serial Peripheral Interconnect (SPI)

After placing all blocks after the placement an actual clock is assigned to the design and also a clock tree will be formed accordingly replacing the virtual clock which has been in the design till now, and the clock tree bifurcates to individual blocks. In clock tree optimization (CTO) clock can be shielded so that noise is not coupled to other signals. But shielding increases area by 12 to 15% (fig 14). Since the clock signal is global in nature the same metal layer used for power routing is used for clock also.

**TABLE 1: COMPARISON TABLE**

<table>
<thead>
<tr>
<th>Cell Count</th>
<th>Before optimization</th>
<th>After optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hierarchical Cell Count</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>Hierarchical Port Cell Count</td>
<td>188</td>
<td>327</td>
</tr>
<tr>
<td>Leaf Cell Count</td>
<td>116</td>
<td>1170</td>
</tr>
<tr>
<td>Buff Cell Count</td>
<td>156</td>
<td>110</td>
</tr>
<tr>
<td>Buff Cell Count</td>
<td>42</td>
<td>......</td>
</tr>
<tr>
<td>Inv Cell Count</td>
<td>114</td>
<td>......</td>
</tr>
<tr>
<td>CT Buff Inv Cell Count</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>Combinational Cell Count</td>
<td>933</td>
<td>941</td>
</tr>
<tr>
<td>Sequential Cell Count</td>
<td>229</td>
<td>229</td>
</tr>
<tr>
<td>Macro Count</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

VI. CONCLUSION

SPI is a high speed synchronous data transfer mode for communication with many devices. It transfers the data in full duplex mode i.e., received and transmitted data both at a time. It is designed for a high speed aspect Serial communication between components is a vital part to reducing size of circuitry. Within the physical design stage, a complete flow is implemented as well. This flow will be described more specifically, and as stated before, several EDA companies provide software or CAD tools for this flow. The overall goal of this tool/software is to combine the inputs of a gate-level net-list, standard cell library, along with timing constraints to create and Floor plan placement and cts. The Serial peripheral interface (SPI) is implemented using the physical design process with the help of IC compiler to analyze and optimize.

VII. REFERENCES


