Techniques for Improving Design Performance of VLSI Circuits and Systems

Kirat Pal Singh
Assistant Professor
Department of Electronics and Communication Engineering
Surya World, Punjab, India
Email – kirat_addiwal@yahoo.com

ABSTRACT - The purpose of this paper is to increase the maximum clock frequency and improve the setup and hold timing by modifying the circuit design. This paper describes the digital gates and memory elements such as latches and registers and can analyze a circuit to find the maximum clock frequency. In this paper we performs to maximize the clock frequency by adding output registers, minimize the setup and hold window by adding input registers, adjust delay measurements when including a delay locked loop (DLL), recalculate the timing of the board-level system after timing modification.

Keywords- Delay, Frequency, Performance, Registers, Clock

1. INTRODUCTION

In Electronics, performance issues in digital systems such as clock skew and its effect on setup and hold time constraints, and the use of pipelining for increasing system clock frequency. This is followed by definitions for latency and throughput, with associated resource tradeoffs explored in detail through the use of dataflow graphs and scheduling tables applied to examples taken from digital signal processing applications. Also, design issues relating to functionality, interfacing, and performance for different types of memories commonly found in ASICs and FPGAs such as FIFOs, single-ports, and dual-ports are examined.

2. INCREASING MAXIMUM CLOCK FREQUENCY

The three types of delays paths through a circuit set the maximum clock frequency. The only way to increase the maximum clock frequency is to reduce the delay through these worst-case paths. Assuming the propagation delays of the gates and registers cannot be changed, only changing the circuit architecture can reduce the worst-case path delays.

Reducing the worst-case delays by adding circuit elements is not intuitive, but it is effective in increasing performance. For example, the pin-to-pin combinational delay through a circuit can be completely removed by ensuring there are no combinational paths from any input to any output. Likewise, $t_{c2q}$ can be minimized by reducing combinational paths between the clock input and the output. Both of these tasks can be accomplished by using the same method. Placing registers on all outputs of the circuit removes all combinational delay paths, and minimizes the combinational path of $t_{c2q}$.

Adding registers to the design may seem like it would reduce the clock frequency, but in fact it can often increase it. Analyzing the worst-case paths is the only way to set maximum clock frequency. If the worst-case path delay is reduced, then the circuit naturally can be clocked faster. While the pin-to-pin combinational delay is inherently removed from the analysis, the clock-to output is usually reduced to its minimum possible value. Since the registers are placed at the output of the circuit, there are no combinational circuits after this to add to the clock-to-output delay. The only clock-to-output delay paths possible are through these output registers, so the analysis is greatly simplified.

The output registers can only be added before the combinational output buffer delay because this is not an actual gate in the design. This delay represents the interface from the chip to the board. Often the output circuitry design has a significant delay because of the need for a high fan-out, larger voltage swing, and over-voltage protection. Therefore, placing the register immediately before this buffer is the optimum location. One consequence of this approach is the impact of $t_{r2r}$ through the circuit. Since there are more registers in the design, there are more register-to-register delays to be computed. Sometimes the worst-case $t_{r2r}$ will increase because of this. If the clock frequency is being limited by the pin-to-pin delay or the clock-to-output delay, and then these delays are reduced, the clock frequency will still increase if $t_{r2r}$ is not increased by a significant amount. If registers are added to the outputs, the worst-case $t_{r2r}$ will usually become the largest delay path of the circuit.

Another consequence of this approach is the impact on latency. Latency is the time required for an input to propagate through a circuit to the output. If a circuit is all combinational, then the latency is in the same clock period in which the data input is applied. By adding registers to the output of the circuit, the latency increases into the next clock period. Adding a set of registers to all outputs of a device means the latency of each input will increase to the beginning of the next clock period. While this is a disadvantage, the impact on performance is usually not significant. The latency has increased, but the clock period has decreased as well (usually). Therefore, the combination of these two effects often cancels each other out.

While latency may have increased by one clock cycle, the rate at which data is being input and output is the same. New data is input and output every clock cycle. The throughput of the data is the same, even though the latency has increased. Therefore, the overall computing performance of the device will increase. This effect is called pipelining.
The analysis for this circuit as shown in Fig. 1 is the same as for all maximum clock frequency calculations. The worst-case pin-to-pin combinational delay, clock-to-output delay, and \( t_{R2R} \) must be found. Since the output is now registered, there is no pin-to-pin combinational delay. This measurement can be excluded from the analysis, or set to zero for continuity in the final comparison.

![Fig. 1 Adding an output register to the sequential circuit](image)

The clock-to-output delay only has one path to compute. Since this delay can pass through at most one register, the only register it can now pass through to the output is the new added register. This path proceeds from the clock buffer \( C \), through the register \( U3 \), and through the output buffer \( D \). The improved clock-to-output delay is 13 ns.

\[
C \ t_{pd} + U3 \ t_{C2Q} + D \ t_{pd} = t_{C2Q} \ \text{SYS}
\]

\[2 + 5 + 6 = 13 \text{ns}
\]

The number of register-to-register paths has increased due to adding another register from two to four. The paths are listed in Table 1. The worst-case path is from \( U1 \), through gates \( E \) and \( H \), to the new output register \( U3 \) for a total delay of 25 ns.

**TABLE 1** Total set of new register-to-register propagation delays

| Starting input | Path          | Delay |
|----------------|---------------|-------|
| U1             | U1+E+H+U2    | 15 ns |
| U2             | U2+G+U1      | 16 ns |
| U1             | U1+E+H+U3    | 25 ns |
| U2             | U2+H+U3      | 17 ns |

Adding registers to the output of the circuit also changes \( t_{su} \) and \( t_{bd} \) for the circuit. If the circuit has a combinational path through the circuit and a register is added to the output, the longest combinational delay path from a circuit input to a register input could very likely be the newly added register. The setup and hold window could increase significantly because of the new output register. One way to minimize the effects of adding output registers is to place registers on the inputs of the circuit. This will reduce the combinational paths to the registers to minimize the setup and hold window. The input registers can only be placed after the input buffer delay since this is not an actual buffer much like the output buffer delay. Therefore, there will be an input buffer combinational delay to the register input.

![Fig. 2 Adding input registers to the sequential circuit](image)

For example, the \( t_{su} \) of the circuit shown in Fig. 2 before adding input registers is computed by finding the longest combinational path to any register in the design. The addition of the output register increases the worst-case delay to 18 ns from the circuit input \( X \) to the \( U3 \) register through gates \( A \), \( E \), and \( H \). The minimum clock delay remains the same. Therefore, the new circuit \( t_{su} \) increases to 19 ns.

\[
(t_{pd \ data \ U1} - t_{pd \ clk \ U1}) + t_{su} \ FF = t_{su} \ \text{TOTAL}
\]

\[(18 - 2) + 3 = 19 \text{ ns}
\]

**TABLE 2** Measured improvement of adding output registers

| Measurement | Original delay | Improved delay |
|-------------|----------------|----------------|
| P2P         | 24 ns          | 0 ns           |
| C2Q         | 30 ns          | 13 ns          |
| R2R         | 16 ns          | 25 ns          |
| Clock period| 30 ns          | 25 ns          |
| Clock frequency | 33.3MHz       | 40MHz          |

The clock period is set by taking the largest of the three worst-case paths, zero ns for the pin-to-pin combinational delay, 13 ns for the clock-to-output delay, and 25 ns for \( t_{R2R} \). Therefore, the minimum clock period is 25 ns, which corresponds to a maximum clock frequency of 40 MHz. Before adding the register on the output, the minimum clock period was set by the clock-to-output delay. Since this delay decreased to 13 ns, it is no longer limiting the clock period. The \( t_{R2R} \) has increased, but is still less than the previous limiting value of 30 ns. This means the maximum clock frequency has significantly increased by adding a single register to the design. The total comparison of measured values is present in Table 2.

3. IMPROVING SETUP AND HOLDTIMES

Adding registers to the output of the circuit also changes \( t_{su} \) and \( t_{bd} \) for the circuit. If the circuit has a combinational path through the circuit and a register is added to the output, the longest combinational delay path from a circuit input to a register input could very likely be the newly added register. The setup and hold window could increase significantly because of the new output register. One way to minimize the effects of adding output registers is to place registers on the inputs of the circuit. This will reduce the combinational paths to the registers to minimize the setup and hold window. The input registers can only be placed after the input buffer delay since this is not an actual buffer much like the output buffer delay. Therefore, there will be an input buffer combinational delay to the register input.

For example, the \( t_{su} \) of the circuit shown in Fig. 2 before adding input registers is computed by finding the longest combinational path to any register in the design. The addition of the output register increases the worst-case delay to 18 ns from the circuit input \( X \) to the \( U3 \) register through gates \( A \), \( E \), and \( H \). The minimum clock delay remains the same. Therefore, the new circuit \( t_{su} \) increases to 19 ns.

\[
(t_{pd \ data \ U1} - t_{pd \ clk \ U1}) + t_{su} \ FF = t_{su} \ \text{TOTAL}
\]

\[(18 - 2) + 3 = 19 \text{ ns}
\]

**Fig. 2 Adding input registers to the sequential circuit**

The \( t_{su} \) of the circuit before adding input registers is computed by finding the shortest combinational path to any register in the design. The addition of the output register does not increase this value. The shortest path is the same as the previous analysis at 8 ns. This means \( t_{su} \) remains the same at \(-2\) ns, which should be set to zero since it is negative. The setup and hold window is now 19 ns because of the addition of the output registers.

\[
(t_{pd \ clk \ (MAX)} - t_{pd \ data \ (MIN)}) + t_{bd} \ FF = t_{bd} \ \text{TOTAL}
\]
Adding input registers after the input buffers simplifies the computations because the number of paths from each input is reduced to one per input. For this circuit, the combinational delay for each input is 1 ns, and the delay for the clock is 2 ns. This means the new \( t_{su} \) is 2 ns, and the new \( t_{hd} \) is 5 ns. This means the setup and hold window is now 7 ns. The comparison between \( t_{su} \) and \( t_{hd} \) is given in Table 3.

**TABLE 3** Measured improvement of adding input registers

| Measurement       | Original | Added output registers | Added input registers |
|-------------------|----------|------------------------|-----------------------|
| Setup time        | 10 ns    | 19 ns                  | 2 ns                  |
| Hold time         | 0 ns     | 0 ns                   | 5 ns                  |
| Setup and Hold window | 10 ns  | 19 ns                  | 7 ns                  |

\[(t_{pd} \text{ data U1} - t_{pd} \text{ clk U1}) + t_{su} \text{ FF} = t_{su} \text{ TOTAL}\]

\[(1 - 2) + 3 = 2 ns\]

\[(t_{pd} \text{ clk(MAX)} - t_{pd} \text{ data(MIN)}) + t_{hd} \text{ FF} = t_{hd} \text{ TOTAL}\]

\[(2 - 1) + 4 = 5 ns\]

The setup and hold window is nearly doubled when output registers were added to the design. When registers were added to the inputs, the setup and hold window decreased to the smallest possible window. The window cannot decrease below this because it is limited by the setup and hold window of the register, which is also 7 ns.

4. DELAY LOCKED LOOPS

Often modern designs that have internal clocks have some type of Phased Locked Loop (PLL) or Delay Locked Loop (DLL) to stabilize and adjust the clock. A PLL is a circuit that creates a completely new clock internal to the circuit, but based on the external clock provided to it. A DLL passes the external clock to the circuit, but adjusts its timing through a network of delays. There are significant differences between these two types of clock management schemes, but they are beyond the focus of this paper. For this section, the term DLL will be used to describe both PLLs and DLLs. The relevant feature to this material is how DLLs can adjust the phase of the internal clock. A clock signal can be easily manipulated because of its predictability. The clock will always have a repeating 1-0-1-0 pattern. Therefore, once the clock is active, the clock is the same from one clock period to the next. If the external clock signal is delayed by an input buffer, the internal clock will not be aligned with the external clock. A DLL can artificially make the clock appear to be aligned by inserting additional delay to the clock. For example, an external clock with a period of 8 ns passes through an input buffer that delays the signal by 1 ns as in Fig. 3. The DLL measures that the two clocks are not aligned, and then it inserts additional delay to the internal clock until they are aligned. In this example, the DLL would add a 7 ns delay to make the two clocks aligned.

A DLL can change the phase of the internal clock either manually or automatically. The advantage of this is that the active clock edge can be placed anywhere. This means the clock delay in the clock-to-output calculations and \( t_{su} \) and \( t_{hd} \) calculations can be set to whatever needed. Typically the DLL will align the internal clock with the external clock to remove any delays added by the input buffer for the clock signal. The input buffer will add a fixed delay to the clock signal, and the DLL will effectively reduce the delay by that same amount. Note that this technique is not possible to reduce the delays on the data signals because they don’t have a predictable repeating pattern.

For example, Use a DLL to align the internal clock to the external clock in Fig. 2. Any equation that uses the delay of the input buffer C must be recalculated with that value set to zero. The first change is in the calculation of the clock-to-output delay for the circuit. There is only one clock-to-output path through the circuit through the output register. The new clock-to-output delay for this circuit is reduced by 2 ns to 11 ns.

\[C \ t_{pd} + U3 \ t_{CQ} + D \ t_{pd} = t_{CQ} \ \text{SYS} \]

\[0 + 5 + 6 = 11 \text{ ns}\]

The pin-to-pin combinational delay and the register-to-register delay are not affected by the change to the clock because they do not include the clock buffer C. The maximum clock frequency must be checked because this change might affect it if the clock-to-output delay was the limiting factor. Typically \( t_{max} \) limits the maximum clock frequency, so often the clock frequency will not change when adding a DLL.

The \( t_{su} \) and \( t_{hd} \) also depend on the clock delay, so they will be affected by adding a DLL. The minimum and maximum clock delay is set to zero and \( t_{su} \) and \( t_{hd} \) are recalculated.

\[(t_{pd} \text{ data U1} - t_{pd} \text{ clk U1}) + t_{su} \text{ FF} = t_{su} \text{ TOTAL}\]

\[(1 - 0) + 3 = 4 \text{ ns}\]

\[(t_{pd} \text{ clk(MAX)} - t_{pd} \text{ data(MIN)}) + t_{hd} \text{ FF} = t_{hd} \text{ TOTAL}\]

\[(0 - 1) + 4 = 3 \text{ ns}\]

The new \( t_{su} \) is 4 ns, and the new \( t_{hd} \) is 3 ns. The setup and hold window has not changed from 7 ns.

5. BOARD-LEVEL TIMING IMPACT

The final calculation of the chip is to analyze how well the circuit will improve the board-level performance. The datasheet for the improved circuit is listed in Table 4. The new calculations include both input and output registers and a DLL for clock adjustment.
TABLE 4 Datasheet for the improved circuit

| Parameter  | Description               | Old min | Old max | New min | New max |
|------------|---------------------------|---------|---------|---------|---------|
| $T_{clk}$  | Clock period              | 30ns    | 25ns    |         |         |
| $F_{clk}$  | (MHz)                     | 33.3    | 40      |         |         |
| $T_{su,y}$ | Y setup time              | 10ns    | 3ns     |         |         |
| $T_{hd,y}$ | Y hold time               | 0ns     | 4ns     |         |         |
| $X_{t_{pd, P2P}}$ | Combinational delay     | 24ns    | N/A     |         |         |
| $T_{pd, C2Q}$ | Clock to output delay   | 30ns    | 11ns    |         |         |

For example, using the circuit in Fig. 4, maximum clock frequency can be calculated. Each chip has the same circuit as in Fig. 2 and uses the timings in Table 4.

Fig. 4 Board-level schematic to compute maximum clock frequency

First, since there is no combinational path through the chip, there is no calculation for the pin-to-pin combinational path for the board. This value is excluded when computing maximum clock frequency. One clock-to-output delay exists for this circuit. This path passes only through the clock input of U2. If there is no clock delay, the clock-to-output for the board is the same as the clock-to-output of the chip. This delay is 11 ns. Two register-to-register delays exist for this circuit. The first is through the U1 clock-to-output to either input on U2. The third is through the U2 clock-to-output to the input of Y on U1. Both paths have the same delay of $11 + 4 = 15$ ns.

U1 $t_{C2Q} + U2 t_{su} = t_{C2Q}^{SYS}$

$11 + 4 = 15$ ns

The three worst-case paths and the chip minimum clock period limit the clock frequency for the board-level system. The largest of these four values (0 ns, 11 ns, 15 ns, 25 ns) is 25 ns, which is also the minimum clock period for the chip. This means the board can operate at the same frequency as the chips on the board. Note the removal of the combinational paths greatly reduces the delays at the board level.

CONCLUSION

The paper concludes the parameters that indicate the maximum clock frequency of a circuit. The design can be modified to reduce the longest delays to improve circuit performance. Reducing the combinational delay paths increases the maximum clock frequency by targeting the worst-case paths. By registering all inputs and outputs, the circuit can operate at its maximum frequency within a larger system. Using additional technologies like DLLs can further increase the circuit performance within a larger system.

REFERENCES

[1] R. Ginosar, “Fourteen ways to fool your synchronizer”, Proc. of the Ninth International Symposium on Asynchronous Circuits and Systems, 12-15 May 2003, pp 19-26.

[2] Justin Davis and Robert Reese, “Finite State Machine Datapath Design, Optimization, and implementation”, Morgan & Claypool, 2008.