Characterization of GasP cells & analyzing the effects of the operating environments on timing verification

Prasad Joshi

08/21/2008

Collaborative Effort
From
Sun Microsystems &
University of Southern California

Introduction:

Almost all asynchronous circuit families have internal timing constraints where a particular signal is required to arrive before another signal. Such timing constraints are termed as Relative Timing (RT) constraints.

The GasP family of circuits used in the design of Fleet can be represented as follows:

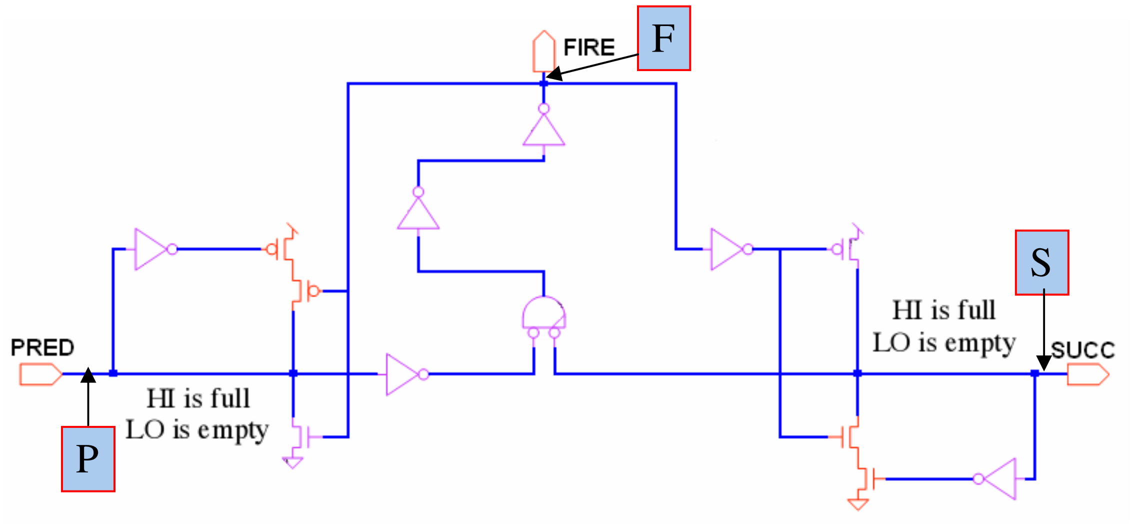


Figure 1 : GasP Plain

In order to ensure that the GasP control circuits work as desired, a set of RT constraints need to be identified and verified. These RT constraints can be grouped as follows:

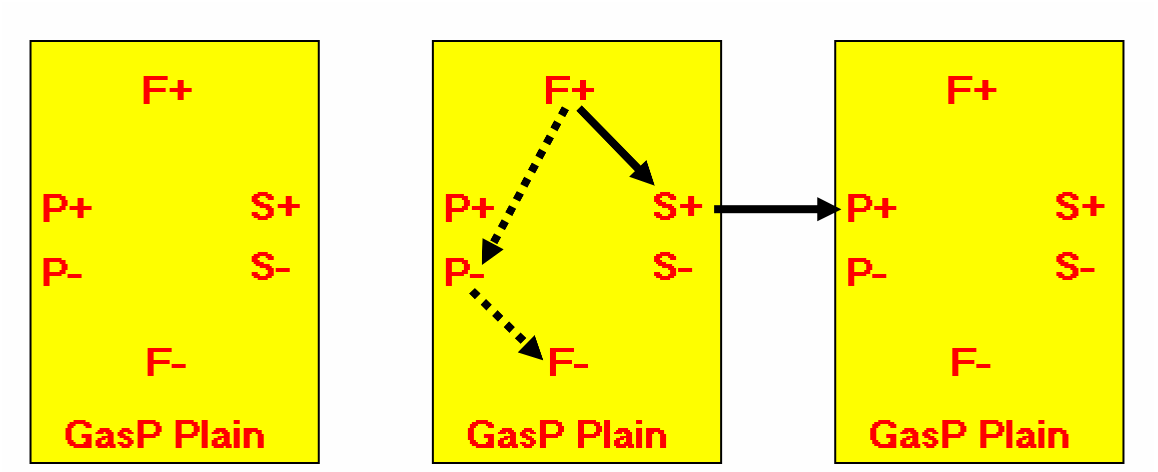
1. Rail to Rail (RR) Constraints.
2. Short Circuit (SC) Constraints.

Rail to Rail Constraints:

The Rail to Rail Constraints ensure that the state wire is able to reach the power and ground rails. This ensures that every circuit in the GasP control pipeline successfully completes the handshake with its nearest neighbors. The two Rail to Rail constraints that one should to be aware of are as follows:

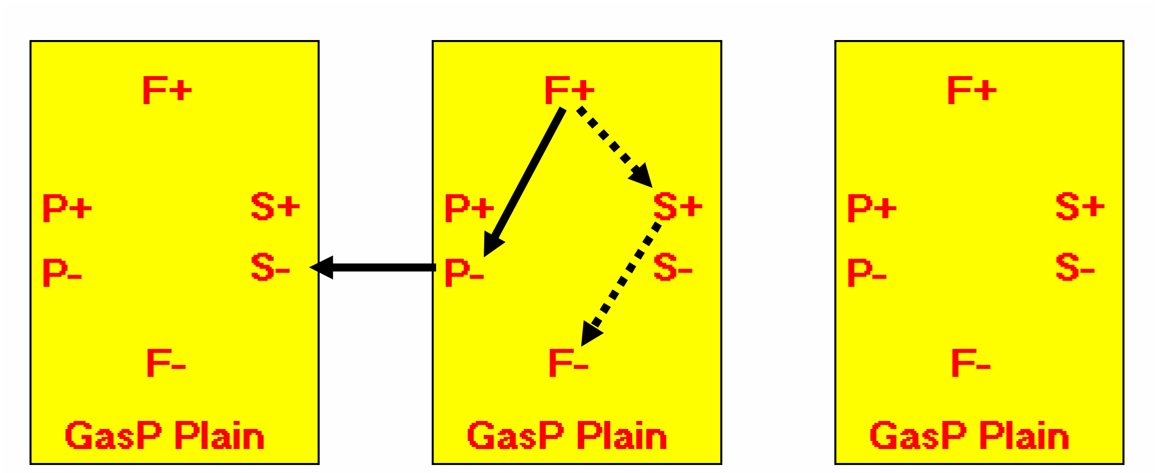
1. Predecessor Loop Constraint on the Successor State Wire.
2. Successor Loop Constraint on the Predecessor State Wire.

Predecessor Loop Constraint on the Successor State Wire:



The driver PMOS which pulls the successor state wire high can be turned off by action of the predecessor loop as well as the successor loop. In the above figure the delay of the solid path has to be less than that of the dotted path. This timing constraint ensures that the successor state wire is completely pulled high before the driver PMOS turns off because of the predecessor loop. This constraint has a margin of 3 gate delays. However this margin gets reduced when there is a large load on the successor state wire and a small load on the predecessor state wire. If the above constraint is violated, then control tokens can be lost and may even lead to deadlock.

Successor Loop Constraint on the Predecessor State Wire:



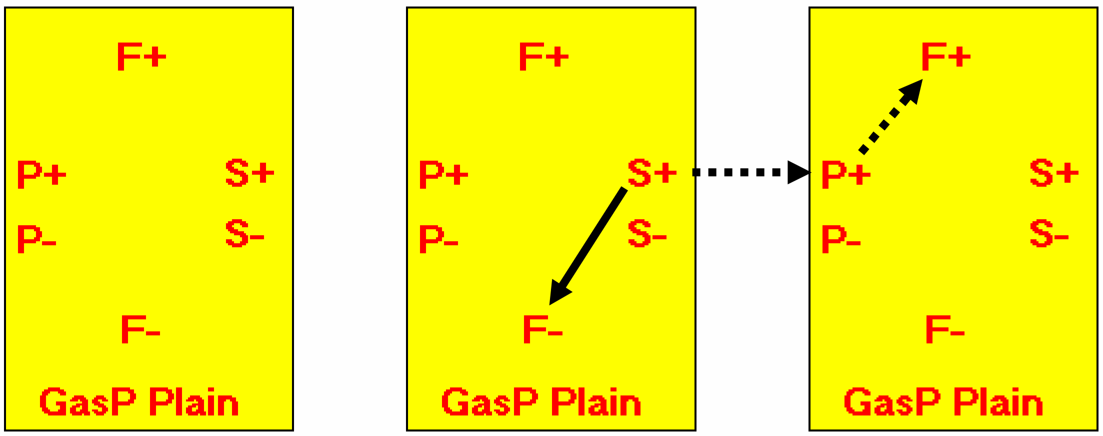
The driver NMOS which pulls the predecessor state wire low can be turned off by the action of the predecessor loop as well as the successor loop. In the above figure the delay of the solid path has to be less than that of the dotted path. This timing constraint ensures that the predecessor state wire is completely pulled low before the driver NMOS turns off because of the state wire loop. This constraint has a margin of 4 gate delays. However this margin gets reduced when there is a large load on the predecessor state wire and a small load on the successor state wire. If the above constraint is violated, then the predecessor state wire will be stuck at high and repeated tokens will be generated at the successor state wire. This can also lead to a deadlock.

Short Circuit Constraints:

The short circuit constraints ensure that the bidirectional state wire between two GasP control cells is driven by one cell at a time. Due to the symmetry of the GasP cells we have one short circuit constraint for each state wire:

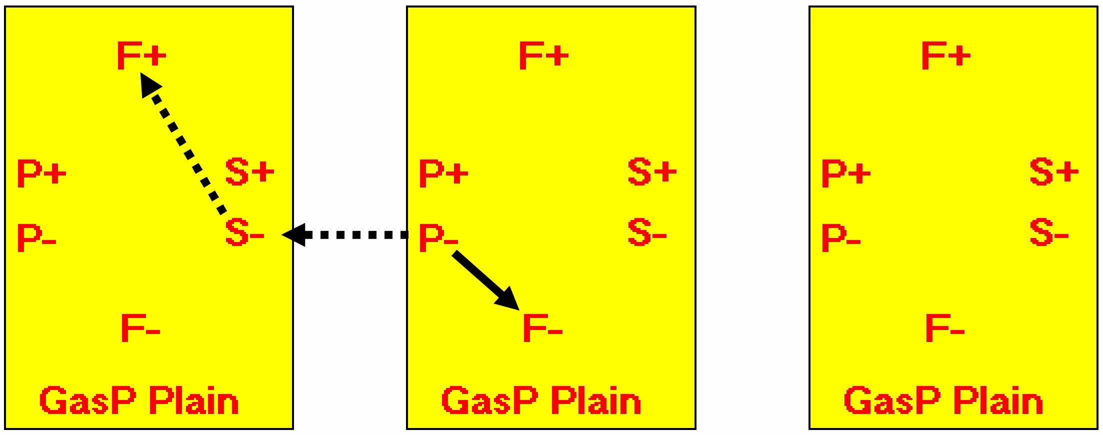
1. Short Circuit Constraint on the Successor State Wire.
2. Short Circuit Constraint on the Predecessor State Wire.

Short Circuit Constraint on the Successor State Wire:



In the above figure the delay of the solid path must be less than that of the dotted path. The driver PMOS of the current cell must turn off before the driver NMOS of the next cell turns on. This constraint has a margin of zero gate delay and hence is really tight.

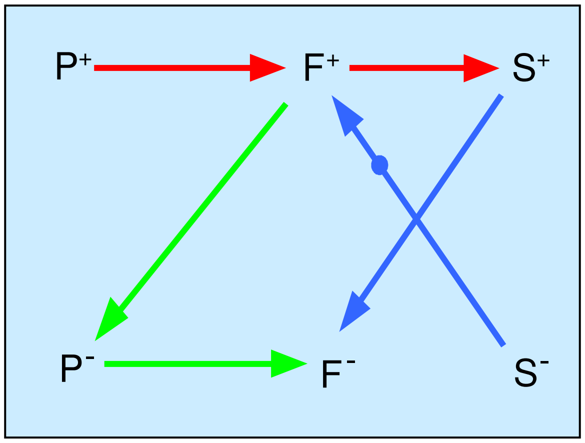
Short Circuit Constraint on the Predecessor State Wire:



In the above figure the delay of the solid path must be less than that of the dotted path. The driver NMOS of the current cell must turn off before the driver PMOS of the previous cell turns on. This constraint has a margin of zero gate delay and hence is really tight.

Characterization of the GasP control cells

The functional behavior of the GasP Control Circuits can be described using a State Transition Diagram as follows:



It is important to note that the arc S- to F+ has an initial token on it which enables the transition of P+ to F+. As seen from the above figure the various timing arcs that need to be characterized are as follows:

1. P+ to F+
2. F+ to S+
3. F+ to P-
4. P- to F-
5. S+ to F-
6. S- to F+

Characterizing the timing arcs:

In-order to make sure that the GasP control circuits satisfy all the RT constraints, we need to make sure that the worst region of operation is always taken into account. In other words, the characterization procedure of the GasP cells must make sure that the each of the timing arcs defined previously are characterized in the right way.

All the RT constraints defined previously are of the same form where one timing path A has to be shorter than the other timing path B. In order to make sure that the worst case is taken into account, we need to verify that the max of timing path A has to be smaller than the min of timing path B.

The above discussion makes it obvious that we should have two timing libraries:

1. Fast .lib for Fast Circuit Corners
2. Slow .lib for Slow Circuit Corners

Each of the above arcs needs to be characterized using different initial conditions for achieving the fast and slow circuit corners. These arcs are characterized by sweeping the input slews and the output loads. This is discussed in detail as follows:

1. P+ to F+ :

   a. Fast .lib: S pin of the DUT is held at 0 with no initial conditions on the other pins.

   b. Slow .lib: S pin of the DUT changes from 1 to 0 at the same time as the inverted P signal. This can be done by putting an inverter on the S pin which is of the same strength as the inverter to which the P signal is applied.

2. F+ to S+:

   a. Fast .lib: S pin is set to 0 by using .ic command in spice.

   b. Slow .lib: Same as above.

3. F+ to P-:

   a. Fast .lib: P pin is set to 1 by using .ic command in spice.

   b. Slow .lib: Same as above.

4. P- to F-:

   a. Fast .lib: S pin of the DUT changes from 0 to 1at the same time as the inverted P signal. This can be done by putting an inverter on the S pin which is of the same strength as that of the inverter to which the P signal is applied.

   b. Slow .lib: S pin of the DUT is held at 0 with no initial conditions on the other pins.

5. S+ to F-:

   a. Fast .lib: The inverted P of the DUT changes from 0 to 1 at the same time as the S pin changes from 0 to 1. This can be done by pulling out the pin Pbar in the sub-circuit definition and applying the same input signal on the Pbar pin as the S pin.

   b. Slow .lib: P pin of the DUT is held at 1 with no initial conditions on the other pins.

6. S- to F+:

   a. Fast .lib: P pin of the DUT is held at 1 with no initial conditions on the other pins.

   b. Slow .lib: The inverted P of the DUT changes from 1 to 0 at the same time as the S pin changes from 1 to 0. This can be done by pulling out the pin Pbar in the sub-circuit definition and applying the same input signal on the Pbar pin as the S pin.

Evaluating the effects of operating environments:

The GasP control pipelines operate in the following three regions:

1. Bubble limited (BL) region.
2. Data limited (DL) region.
3. Full throughput (FT) region.

The timing arcs defined previously have their minimum and maximum delays in different regions as shown in the following table:

Timing Arcs	Minimum Delay		Maximum Delay	
	Region	Characterized in which .lib?	Region	Characterized in which .lib?
P+ to F+	DL (note 1)	fast .lib	FT	slow .lib
F+ to S+	--- (note 2)	both	---	both
F+ to P-	--- (note 2)	both	---	both
P- to F-	note 3	fast .lib	FT	slow .lib
S+ to F-	note 3	fast .lib	FT	slow .lib
S- to F+	BL (note 4)	fast .lib	FT	slow .lib

Note1: The arc P+ to F+ has a minimum delay when the other input of the NOR gate is a constant low. In other words the minimum delay is achieved when the successor state wire is empty. This happens in the data limited case.

Note 2: The actual timings of the arcs F+ to S+ and F+ to P- in a real circuit depends on the load on the successor and predecessor state wires respectively. The timings on these arcs remain the same, irrespective of the operating region.

Note 3: The arc P- to F- has a maximum delay when the other input of the NOR gate is a constant low. This happens when the predecessor loop completes before the successor loop and hence it can be attributed to the fact that predecessor state wire load is less than the successor state wire load.
The arc S+ to F- has a maximum delay when the other input of the NOR gate is a constant low. This happens when the successor loop completes before the predecessor loop and hence it can be attributed to the fact that successor state wire load is less than the predecessor state wire load. However, the timings on both these arcs remain the same, irrespective of the operating region.

Note 4: The arc S- to F+ has a minimum delay when the other input of the NOR gate is a constant low. In other words the minimum delay is achieved when the predecessor state wire is full. This happens in the bubble limited case.

Note 5: The arcs P+ to F+ and S- to F+ pass through the NOR gate. Hence by Charlie effect, they have their maximum delays when both the inputs of the NOR gate change at the same time. This happens in the full throughput case.

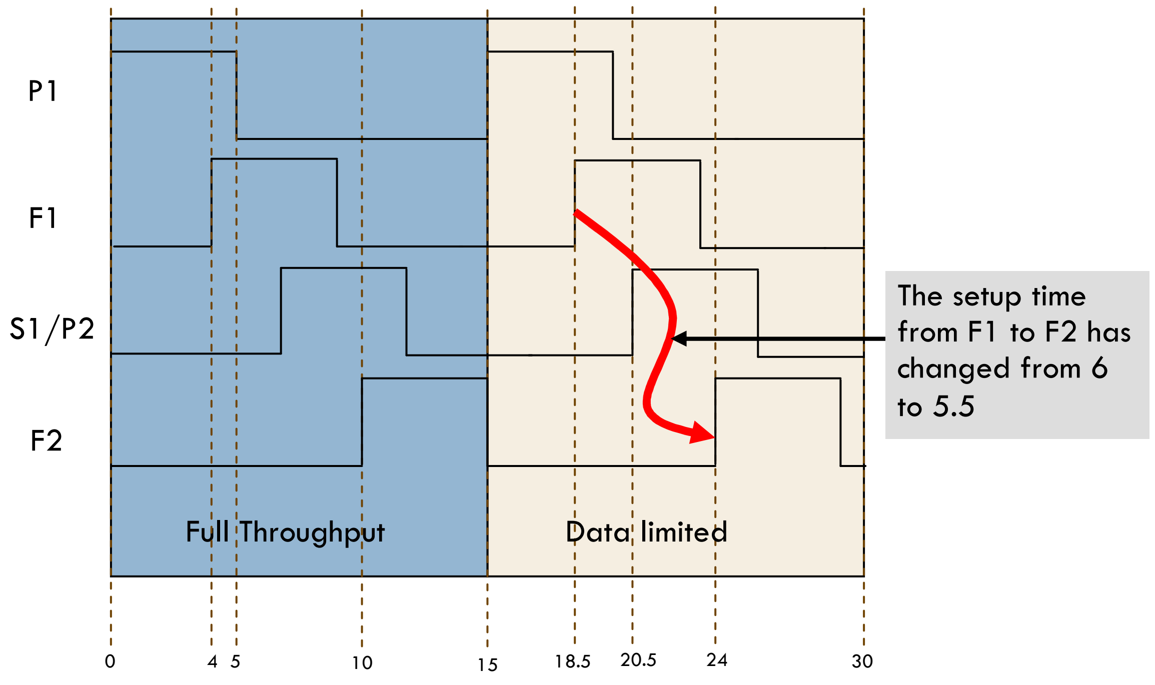
As seen in the above table all the maximum delay values are located in the slow .lib and all the minimum delay values are located in the fast .lib. Hence we can safely assume that if the RT constraints are verified using these two .lib's, then the GasP control circuits can be guaranteed to function as desired. This might be conservative, but it is safe.

Finding clocks for the data-path

After the RT constraints are verified, it is necessary to verify whether the data-path meets the setup and hold checks. The first step is to find out the phase difference between two neighboring fire signals. After determining the phase differences, local clocks can be defined in PrimeTime to perform the setup and hold checks.

Finding the worst case for Setup checks

Let us consider a transition phase of a GasP pipeline from the full throughput case into the data limited case. This will help us evaluate the difference between the two cases. As mentioned previously in this document, the NOR gate has a maximum delay in the full throughput case. Let us take this delay to be 1 gate delay. We also know that in the data limited case, the NOR gate operates slightly faster. Let us take this delay to be 0.5 gate delay. After making these assumptions, consider the following figure:



As seen in the figure, the setup check from F1 to F2 is 6 gate delays in the full throughput case and is reduced to 5.5 in the data limited case due to a slightly faster NOR gate. This leads us to the conclusion that the data limited case is worse than the full throughput case. The next question that comes into mind is "what happens in the bubble limited case?"

In the bubble limited case, as the successor state wire of a particular cell is stuck at full, that cell can't fire again till its successor state wire becomes empty. Hence the phase difference between F1 going high and F2 going high will always be greater than 6 gate delays. Hence we can conclude that the data limited case is the worst case for the setup checks.

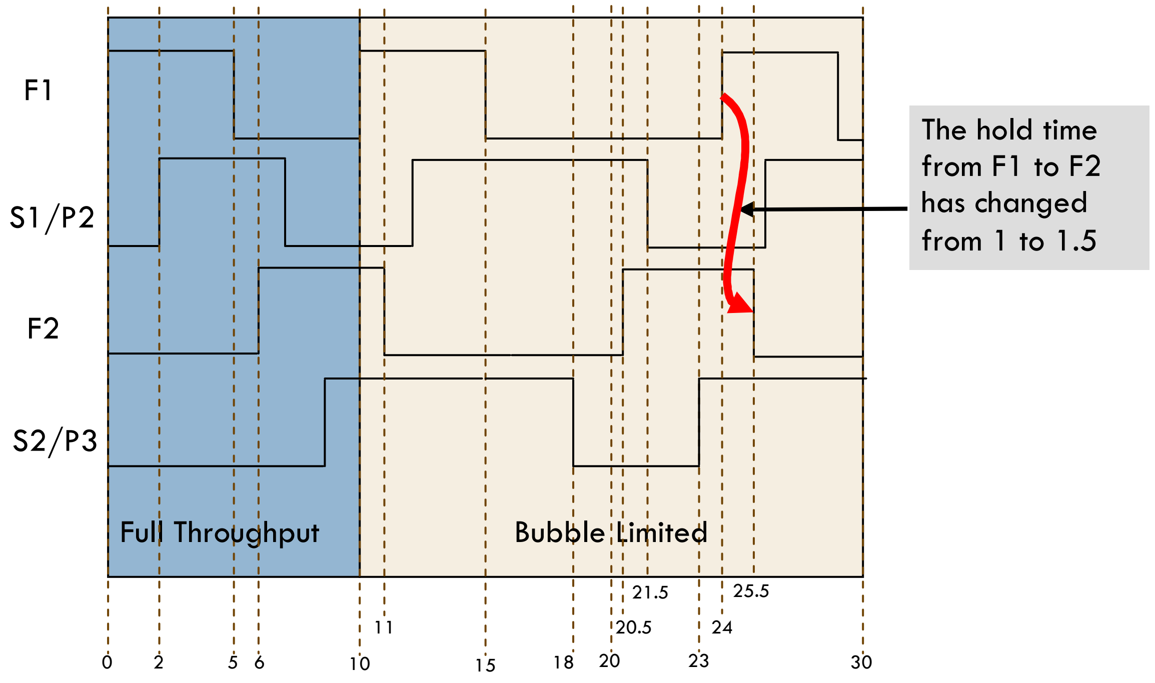
Finding clock for setup checks:

The phase difference between the rising edges of the neighboring clocks F1 and F2 is the summation of the delays F1+ to S1+ and P2+ to F2+. Now for the worst case, we must ensure that these delay values are picked up by PrimeTime from the data limited case.

Recall the discussion made while characterizing the various arcs. We had said that the fast .lib has the delay from P+ to F+ corresponding to the data limited case and F+ to S+ is the same for all three cases. Hence we can safely use the fast .lib to find out the phase differences between the rising edges of the neighboring clocks. Using these phase differences while defining the clocks, we can make PrimeTime verify the setup checks.

Finding the worst case for the Hold checks:

Let us now consider a transition phase of a GasP pipeline from the full throughput case into the bubble limited case. This will help us evaluate the difference between the two cases. Similar to the previous case we will assume the NOR gate delay to be 1 gate delay in the full throughput case. We also know that in the bubble limited case, the NOR gate operates slightly faster. Let us take this delay to be 0.5 gate delay. After making these assumptions, consider the following figure:



As seen in the figure, the hold time check from F1 to F2 has changed from 1 to 1.5 for the bubble limited case. Hence as shown in the figure, the third data token launched by F1

gets more time to corrupt the second data token that is getting latched by F2. This can result in a hold time violation. Thus we can conclude that the bubble limited case is worse than the full throughput case.
The hold time violations occur because the second data tries to corrupt the first data. This happens because of the overlapping transparent periods of the adjacent fire signals. However in the data limited case, the second fire signal arrives later than the usual 5 gate delays. This results in no overlap between the transparent periods of F1 and F2. Hence the data limited case can never be the worst case for hold time violations.

Finding clock for hold checks:

In order to verify the hold time violations, we need to find when F1+ can happen at the earliest after F2+ has happened. The phase difference between the rising edges of the neighboring clocks F2 and F1 is the summation of the delays F2+ to P2- and S1- to F1+. In other words we are finding the adjacent clocks in the backward direction. Now for the worst case, we must ensure that these delay values are picked up by PrimeTime from the bubble limited case.

Recall the discussion made while characterizing the various arcs. We had said that the fast .lib has the delay from S- to F+ corresponding to the bubble limited case and F+ to P- is the same for all three cases. Hence we can safely use the fast .lib to find out the phase differences between the rising edges of the neighboring clocks. Using these phase differences while defining the clocks, we can make PrimeTime verify the hold checks.

Conclusions:

1. RT constraints should be verified using the fast .lib for the minimum delay paths and the slow .lib for the maximum delay paths.
2. Data Limited case is the worst case for setup checks.
3. Bubble Limited case is the worst case for hold checks.
4. Setup and Hold checks should be performed using the fast .lib