WHITE PAPER

Chronos Latency- Pole Position Performance

By G. Rinaldi and M. T. Moreira, Chronos Tech

 © Chronos Tech LLC – All Rights Reserved

Introduction

Modern SoC performance is often limited by the capability to exchange information at high speed between the various IPs. It is in fact common to find a NoC, or even multiple NoCs, integrated inside high-end SoCs to facilitate the exchange of data, with the intent of enhancing performance. Two major metrics define how efficiently information travels between the two ends of a communication channel: throughput and latency.

Throughput is defined as the rate of data delivery over a communication channel, typically measured in bits per seconds. Broadly speaking, in synchronous terms, it is directly related to the clock frequency.

Latency, on the other hand, is referred as the time it takes for a circuit to generate a result after it reads the values in the associated primary inputs. Traditionally, this figure is given in number of clock cycles, as the clock is the signal that controls the sampling and propagation of data in synchronous circuits.

Pipelines are necessary in modern SoCs to guarantee throughput requirements, as data busses travel long distances within a die. Meeting throughput requirements is typically achieved by adding pipeline stages to reduce logic path delay between registers. Latency requirements, however, are more complicated, because the addition of an extra pipeline stage translates in a penalty of an extra clock cycle in latency.

As devices shrink, and datapath speed increases, busses travel equivalent longer distances and pipelines grow bigger. Furthermore, as clock margins become more significant, more pipeline stages are required, resulting in larger latencies. In this scenario, latency gains more relevance as a performance metric and synchronous pipelines start to be prohibitively constrained.

Chronos pipelines, instead, take advantage of a more flexible pipeline stage insertion and do not suffer from clock penalties in their performance metrics. Moreover, they can be optimized for throughput and latency independently, providing a boost in both performance metrics.

Latency in Chronos Links

In traditional synchronous pipelines, the latency metric is easily calculated as the period of the clock multiplied by the number of pipeline stages in the datapath. The latency of asynchronous circuits, on the other hand, is measured as forward latency, and it is only affected by the delays of the components in the pipeline. This is because data flow is not governed by a global clock signal, but rather by local handshake transactions between neighboring components.

Figure 1 shows a visual representation of this concept. In synchronous circuits, all margins added to accommodate clock uncertainties contribute to an increase in latency penalties. Furthermore, in these circuits, the clock period must accommodate worst case path delays, which are data dependent and specific to a given pipeline stage. Therefore, additional latency penalties are imposed, as faster data paths still must wait for the clock to propagate data to the next pipeline stage.

© Chronos Tech LLC – All Rights Reserved

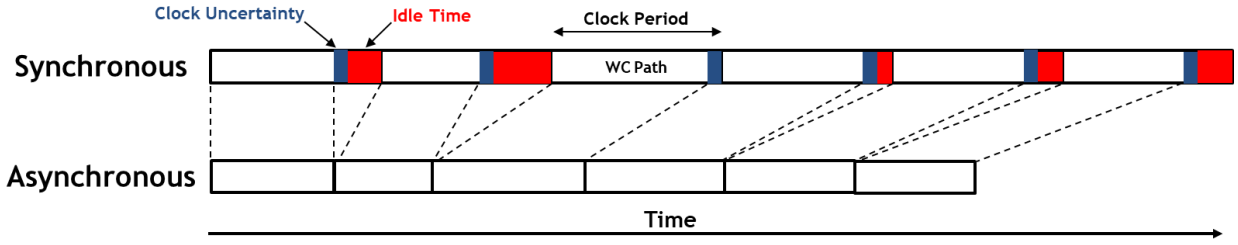

Figure 1 – Latency on a synchronous pipeline compared to latency on an asynchronous pipeline.

In asynchronous circuits these penalties are waived, and the result is that latency is affected only by the delay of logic paths. In practical terms, the latency of an asynchronous system is measured assuming the injection of a single data transmission, to avoid any possibility of congestion caused by previous data. This is because, such congestion is created by the cycles of the handshake protocol that governs the system and is important for throughput analysis, but not for latency. For latency in a Chronos pipeline with no compression, the analysis must assume that the system is ready to receive a new data and measure the time it takes for this system to respond to this data input.

To further clarify this concept, Figure 2 shows the schematic of a section of a Chronos Pipeline, where 3 repeaters are connected in series through data and acknowledge channels. Repeaters are logic blocks capable of holding data and performing handshake transactions with neighboring components to allow data flow.

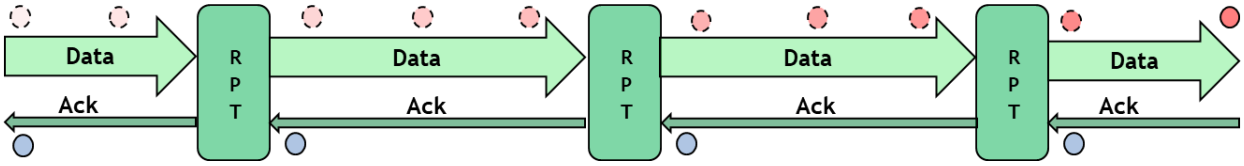

Figure 2 – Basic Chronos Pipeline token flow.

In this example, the pipeline is in a reset state and the blue circles at the end of each acknowledge channel represent tokens available for the repeater to consume, or simply the readiness to start a new handshake transaction. The red circles in the pipeline represent one data word travelling through the repeaters. The time it takes for this data to travel from the input of this pipeline to its output is defined as the latency of this circuit.

Because every repeater is ready to receive new data (has a blue token in the acknowledge bus), there is no congestion caused by handshake cycles involved in this calculation. However, when considering temporal compression, the latency of a Chronos Pipeline is affected by the cycles of the handshake protocol, due to the serial communication nature in such circuits. In these cases, the cycles required to inject all temporally compressed code words that represent one data word must be accounted for at the input of the pipeline only. Then, the forward latency must be added to those cycle times, referred to as serialization penalty. Nevertheless, there is still no need to account for the cycle time of a global clock and its margins and pessimism.

© Chronos Tech LLC – All Rights Reserved

Another interesting fact about the latency of Chronos Pipelines is that they can be optimized by the addition of repeaters. Figure 3 shows an example of how the latency of an example Chronos Link is affected by the number of repeaters in its pipeline.

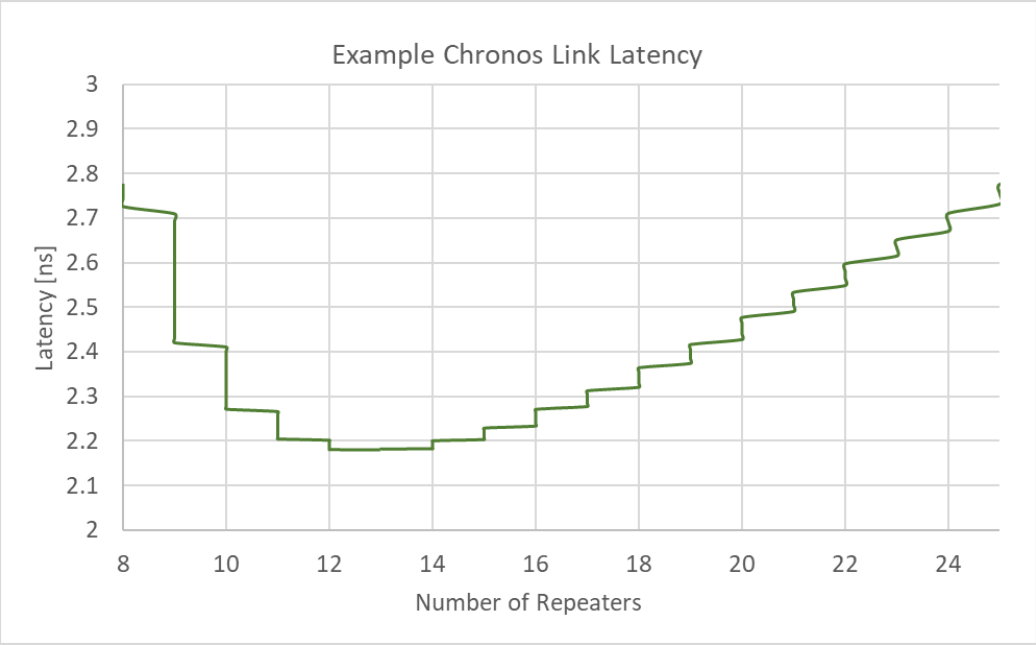

Figure 3 – Example Chronos Link latency as a function of the number of repeaters in the Chronos pipeline.

As the chart shows, the minimum number of repeaters (8) allowed in this example is not the best case for latency. As we increase the number of repeaters to 12, the latency drops from around 2.75ns to lower than 2.2ns. Although this is counter intuitive, as one can expect that adding more repeaters to a pipeline will increase its latency, Chronos Pipelines rely on repeaters that exchange data using local handshakes. Furthermore, logic path delays are not a linear function with respect to wirelength. This way, cycle times, which are an inverse function of the number of repeaters in a Chronos Pipeline, are automatically adjusted to accommodate logic path delays as repeaters are inserted in the pipeline.

Example

Figure 4 shows the latency in an example bus in its original version and after applying Chronos technology. As the chart shows, the original bus had a latency of 6 clock cycles. The latency of the Chronos design, on the other hand, depends on the employed compression ratio. This is because, the serialization process adds a penalty proportional to the employed ratio. Nevertheless, in all cases of this example, Chronos was able to save at least 1 clock cycle of latency, and 2 clock cycles in the best case.

© Chronos Tech LLC – All Rights Reserved

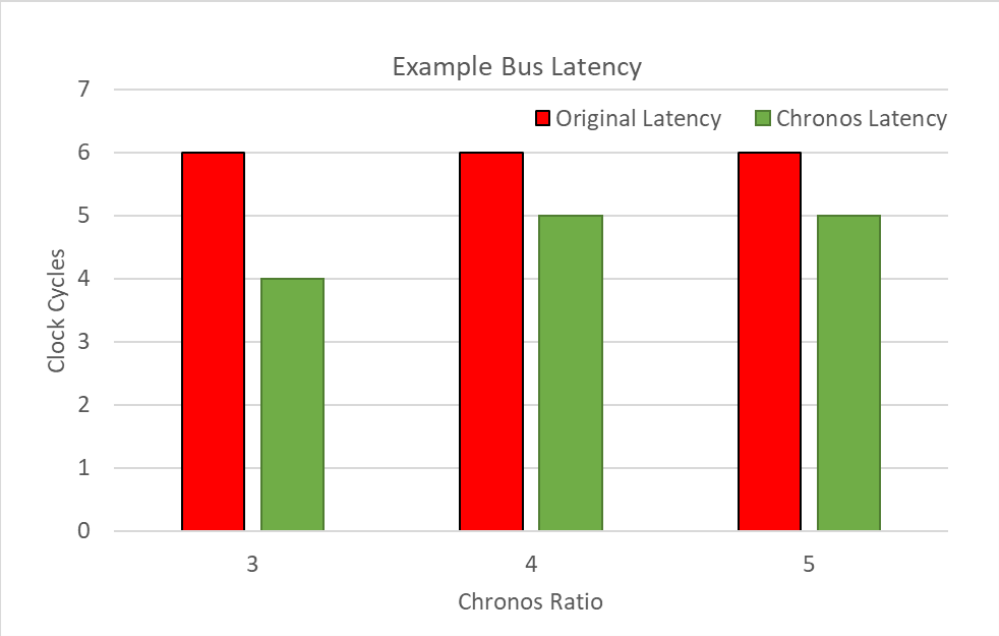

Figure 4 – Latency comparison in an example bus in its original version and after applying Chronos technology.

Note that latency metrics are given here as clock cycles to allow a fair comparison to the original design and they consider synchronization penalties and clock alignment at the receiver. These optimizations are on top of the savings on bus width. More specifically, the Chronos version using a ratio 3 allows reducing bus width by 17% and latency by 2 clock cycles. Ratios 4 and 5 allow reducing bus width by 38% and 50%, respectively, and latency by 1 clock cycle.

Such unique characteristic, of enabling savings in different areas of a design, allows SoCs not only to optimize area, congestion and other layout related metrics, but also to achieve pole position latency performance when using Chronos technologies.

© Chronos Tech LLC – All Rights Reserved