

Interface and Timing Analysis with Two Clocks

Interface

Assume that the interface of your top-level circuit (shown in Fig. 1) includes two clocks:

- `io_clk`: input/output clock used for loading input data (message blocks) and unloading the result (hash value), and
- `clk`: clock used for internal data processing.

Both clocks are synchronized, so there is no shift in phase between them.

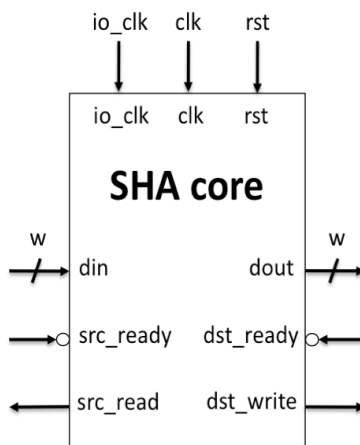


Fig. 1: Input/output interface of a SHA core

The frequency ratio defined below is an integer, greater or equal than one:

$$r = f_{\text{io_clk}}/f_{\text{clk}} \geq 1$$

In a special case, when $f_{\text{io_clk}}=f_{\text{clk}}$, one clock can be used for both input/output and internal processing. Although this situation may happen for some SHA-3 candidates, a typical ratio must be greater than one. This condition must be met in order to keep an input word size, w , at a manageable level, and thus assure that the circuit can be implemented using majority of modern FPGAs, including low-cost FPGAs with limited number of pins.

`io_clk` is used only to load input data to the SHA unit, and unload/store output hash value. `clk` is used for all internal data processing.

Timing analysis

In order to determine the maximum frequency of both clocks, run synthesis, implementation, and timing analysis as usual.

Close to the end of the static timing analysis report, you should see results in the following format:

Clock to Setup on destination clock clk

Source Clock	Src:Rise	Src:Fall	Src:Rise	Src:Fall	Dest:Rise	Dest:Rise	Dest:Fall	Dest:Fall
clk	47.335							

Clock to Setup on destination clock io_clk

Source Clock	Src:Rise	Src:Fall	Src:Rise	Src:Fall	Dest:Rise	Dest:Rise	Dest:Fall	Dest:Fall
io_clk	2.999							

Based on these results, you can determine the maximum value of the ratio

$$r_{\max} = \lfloor f_{\text{io_clk}}/f_{\text{clk}} \rfloor = \lfloor T_{\text{clk}}/T_{\text{io_clk}} \rfloor$$

For example, according to the results of the static timing analysis listed above,

$$r_{\max} = \lfloor 47.335/2.999 \rfloor = 15.$$

Please note that this ratio does not take into account all possible limitations on the frequency of the input/output clock. For example, input/output FIFOs, shown in Fig. 2, may have their own maximum clock frequency, which is smaller than the maximum clock frequency of the input/output shift registers. In this case, the maximum frequency of the io_clk and the corresponding ratio r must be adjusted accordingly.

For the purpose of our project, we assume that no such extra limitations exist.

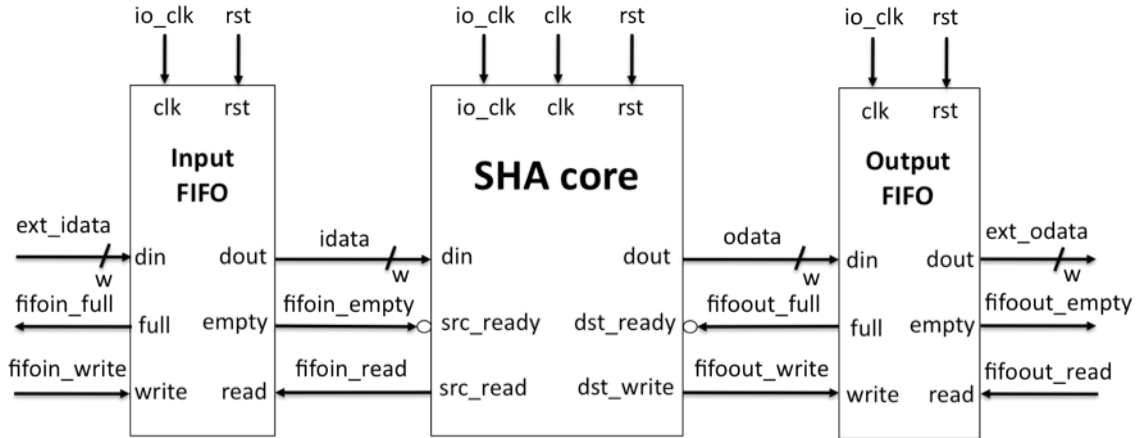


Fig. 2: A typical configuration of a SHA core connected to two surrounding FIFOs. The Input FIFO serves as a source of input data, and the Output FIFO as a destination for output data.

In order to select the optimum values of the parameters r and w , we perform the following analysis.

1. Determine a formula for the time of hashing one message block as a function of T_{clk} and T_{io_clk} .

Assume that this time does not include the time to store the result (unload output register).

For example:

$$H_{time}(1, T_{clk}, T_{io_clk}) = 8 \cdot T_{io_clk} + 24 \cdot T_{clk} .$$

2. Determine a formula for the time of hashing N message blocks as a function of T_{clk} and T_{io_clk} .

Assume that this time does not include the time to store the result (unload output register).

For example:

$$H_{time}(N, T_{clk}, T_{io_clk}) = 8 \cdot T_{io_clk} + 16 \cdot N \cdot T_{clk} + 8 \cdot T_{clk} .$$

3. Determine a formula for the time of processing of one message block as a function of T_{clk} .

$$\begin{aligned} \text{Block_processing_time}(T_{clk}) &= \\ &= H_{time}(N+1, T_{clk}, T_{io_clk}) - H_{time}(N, T_{clk}, T_{io_clk}) \end{aligned}$$

For example:

$$\text{Block_processing_time}(T_{\text{clk}}) = 16 \cdot T_{\text{clk}}$$

4. Determine a formula for the throughput for long messages as a function of T_{clk} .

$$\text{Throughput}(T_{\text{clk}}) = \text{Block_size} / \text{Block_processing_time}(T_{\text{clk}})$$

For example:

$$\text{Throughput}(T_{\text{clk}}) = 512 / (16 \cdot T_{\text{clk}})$$

5. Determine all possible values of the parameters (r, w) , which are possible in your circuit.

The values of (r, w) must fulfill the following conditions:

$$\text{Input_block_loading_time}(\text{block_size}, w, T_{\text{io_clk}}) \leq \text{Block_processing_time}(T_{\text{clk}})$$

From here:

$$(\text{block_size}/w) \cdot T_{\text{io_clk}} \leq k_P \cdot T_{\text{clk}},$$

where k_P is a block processing time in clock cycles.

The value of k_P should be determined based on the analysis of your block diagram, and if possible confirmed by simulation.

From here,

$$r = T_{\text{clk}} / T_{\text{io_clk}} \geq (\text{block_size}/w) / k_P$$

and thus

$$r_{\text{min}} = \lceil (\text{block_size}/w) / k_P \rceil$$

For example:

For parameters used in the example above, and $w=8$, we have

$$r_{\text{min}} = \lceil (512/8)/16 \rceil = 4.$$

The additional conditions are:

$w=2^i$, where $i=1..6$, i.e., $w \leq 64$

$r \geq 1$, i.e., $f_{\text{io_clk}} \geq f_{\text{clk}}$.

Please specify values of all pairs (w, r) fulfilling the above requirements.

For example:

$$w=64, r \in [\max(1, r_{\min}), r_{\max}] = [\max(1, \frac{1}{2}), 15] = [1, 15]$$

$$w=32, r \in [\max(1, 1), r_{\max}] = [1, 15]$$

$$w=16, r \in [\max(1, 2), r_{\max}] = [2, 15]$$

$$w=8, r \in [\max(1, 4), r_{\max}] = [4, 15]$$

$$w=4, r \in [\max(1, 8), r_{\max}] = [8, 15]$$

$$w=2, r \in [\max(1, 16), r_{\max}] = \text{empty set.}$$

6. Determine the optimum pair of parameters (w, r)

Throughput for long messages is not a function of either w or r, so all values determined in point 5 are equivalent.

The time required to load one message block,

$$\text{Input_block_loading_time}(\text{block_size}, w, T_{\text{io_clk}}) = (\text{block_size}/w) \cdot T_{\text{io_clk}}$$

Thus,

$$\text{Input_block_loading_time}(\text{block_size}, w, T_{\text{clk}}) = (\text{block_size}/w) \cdot (T_{\text{clk}}/r).$$

Thus, in order to minimize this time, one can always choose the largest possible values of w and r, i.e.,

$$w_{\max} = 64 \quad \text{and} \quad r = r_{\max}.$$

However, in practical circuits, there exist benefits of minimizing both w and r.

A smaller w means a narrower bus, and thus a smaller number of pins in case of a stand-alone cryptographic module, and a smaller number of interconnects in a system-on-chip.

If $\text{block_size} \leq w_{\max} = 64$, then w should be chosen to be equal to

$$w = \text{block_size}.$$

Similarly, if a hash function supports loading data gradually during processing, using smaller internal word size, w_{int} , (which is the case for SHA-1 and SHA-2), then it is natural to set

$$w = w_{\text{int}}.$$

r=1 is preferred because it means that one clock can be used as both input/output clock and internal clock.

Additionally, the smaller r , the smaller requirements on the frequency supported by surrounding logic, such as FIFOs. Taking into account that such requirements may not be known in advance, and they may vary among various FPGAs, it might be wise to choose the smallest possible $r \geq 1$, i.e.,

$$r = \max(1, r_{\min}(w_{\text{opt}})),$$

where w_{opt} is w chosen according to the above recommendations, and

$$r_{\min} = \lceil (\text{block_size}/w_{\text{opt}})/k_P \rceil.$$

In your code, please define w and r as constants.

Then, demonstrate that your code can support at least two different pairs of (w, r) :

- a. $w=w_1=\min(w_{\max}=64, \text{block_size}), r=r_1=\max(1, r_{\min}(w_1))$
- b. $w=w_2=\min(w_{\max}=64, \text{block_size}), r=r_2=r_{\max}$.