Question 1.1: A pipelined version cannot use the same interface as the combinational circuit. Explain why not. We've provided a new interface for you in RightShifterTypes.bsv.

```
In a combinational circuit, the output is solely determined by the current inputs, and the circuit produces an
immediate output response. The inputs can be changed at any time, and the circuit will produce the
corresponding output without any delays. Therefore, the combinational circuit can have a simple interface
where inputs are directly provided, and the output is immediately available.

On the other hand, a pipelined version breaks the computation into multiple stages, where each stage performs
a portion of the overall computation. Each stage requires a certain amount of time to complete its operation
before passing the intermediate result to the next stage. This introduces a delay in the processing of inputs
and outputs. The pipeline stages need to be synchronized and coordinated to ensure correct operation, and the
overall system performance is influenced by the pipeline depth and the timing of each stage.

To accommodate the pipelined design, a different interface is required to manage the flow of data and
synchronization between pipeline stages.
```

Question 1.2: In the test file, the methods start and result of the shifter are being called from two different rules. Explain why there had to be two rules instead of one. (Hint: try to call both methods from the same rule).

```
Having two separate rules allows for proper coordination between initiating the pipeline operation and
capturing the final output. It ensures that the shifter is appropriately started with the input data and that
the output is obtained only after the entire pipeline computation is completed. This separation of rules
facilitates the synchronization and correct operation of the pipelined shifter.
```

Question 1.3: What is the throughput of your shifter (throughput is the number of full shifts per cycle)? Depending on the conditions, you may get different results. For each throughput level, explain under what conditions you are able to achieve that throughput and what you needed to do in order to get that throughput.

```
10/13
In a pipelined shifter, we place multiple multiplexers (MUX) in different pipeline stages to increase the
utilization of components.

A pipelined shifter is a circuit design that performs shift operations on data. To improve the efficiency of
the shifter, we can allocate multiple MUXs to different pipeline stages. Each pipeline stage is responsible
for processing a portion of the shift operation and passing the result to the next stage.

By using MUXs in different stages, we can process multiple data elements simultaneously and complete partial
shift operations within each stage's clock cycle. This parallel processing approach greatly enhances the
utilization of components, thereby increasing the throughput of the entire pipelined shifter.

By allocating MUXs to different pipeline stages, we ensure that each stage can perform shift operations
concurrently without interference or conflicts. This staged design ensures the correctness and sequencing of
the pipelined shifter while improving processing speed without compromising latency.

In summary, by appropriately allocating multiple MUXs to different pipeline stages in a pipelined shifter, we
can effectively increase the utilization of components and improve the overall performance of the shifter
without introducing additional delays.
```