# 4-to-1 Priority Multiplexer Design Overview

This document gives an overview of the functionality of the 4-to-1 priority multiplexer implemented in *PriorityMultiplexer.sv*.

The module performs a 4-to-1 priority mux by instantiating three 2-to-1 combinatorial multiplexers, implemented in *two\_to\_one\_priority\_mux.sv*. These arbitrate between the first and second inputs, between the third and fourth inputs, and between the outputs of the first-level multiplexers. The outputs of the second-level combinatorial multiplexer are then registered and passed to the module output.

This module has a latency of 1 clock tick with an FMax of 281.7 MHz when implemented on a Virtex 7-class device.

## Summary of operation

### PriorityMultiplexer.sv

This module performs a 4-to-1 priority mux by instantiating three 2-to-1 combinatorial multiplexors, *two\_to\_one\_priority\_mux.sv*. These sub-modules arbitrate between the first and second inputs, between the third and fourth inputs, and between the outputs of the first-level multiplexors.

If only one input is valid, that input is passed. If more than one input is valid, the input with the highest priority is passed. If more than one input is valid with the highest priority, the lower index input is passed. The *dataIndex* output is set to the active output channel index.

The outputs of the second-level multiplexor are registered before passing the outputs out of the module.

### Two\_to\_one\_priority\_mux.sv

This module performs a combinatorial 2-to-1 priority mux. If only one input is valid, that input is passed. If both inputs are valid, the input with the higher priority is passed. If both inputs are valid with the same priority, the first input is passed. The *dataIndex* output is set to 0 to indicate that the first input has been passed, and 1 to indicate that the second input has been passed.

## Critical Path Analysis

The critical path in the design is between the *inPriority* input ports and the output registers *outPriorityReg*. This is because the *Priority* signal undergoes the most levels of logic and has the highest fan out. This is due to it being used to determine which input to be passed, as well as being passed itself.

If a higher FMax is required, the critical path could be broken by adding a register stage in *PriorityMultiplexer.sv* between the outputs of the first-stage multiplexers and the inputs of the second-stage multiplexer. This should allow for an FMax of around 500 MHz but would increase latency to 2 clock ticks.

## Notes on design methodology

In order to achieve the best latency, the following considerations were taken while designing the module:

* Minimize the levels of logic in the critical path
* Minimize the highest fanouts in the design
* Ensure the use of complete *if* and *case* statements in order to avoid inferring latches
* Only use register stages as needed to allow the critical path to meet timing