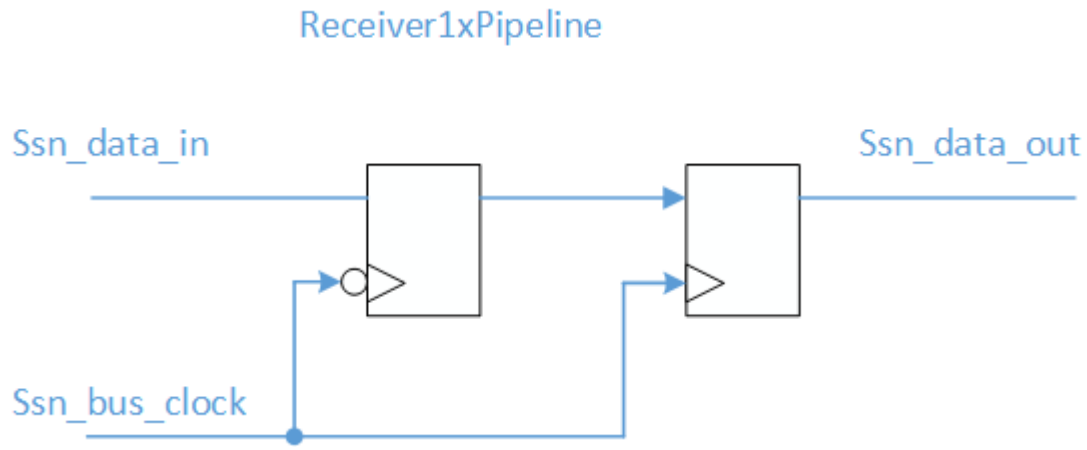
# 1. datapath

## 1.1 pipeline

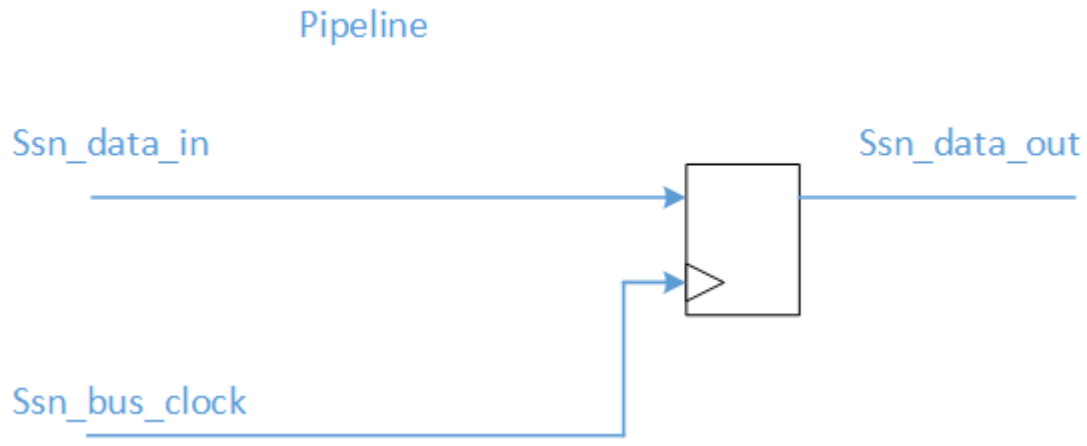
有三种pipeline, 这三种pipe从端口上来看时序是一样的，最终效果都是用正沿打一拍

**Receiver1xPipeline**

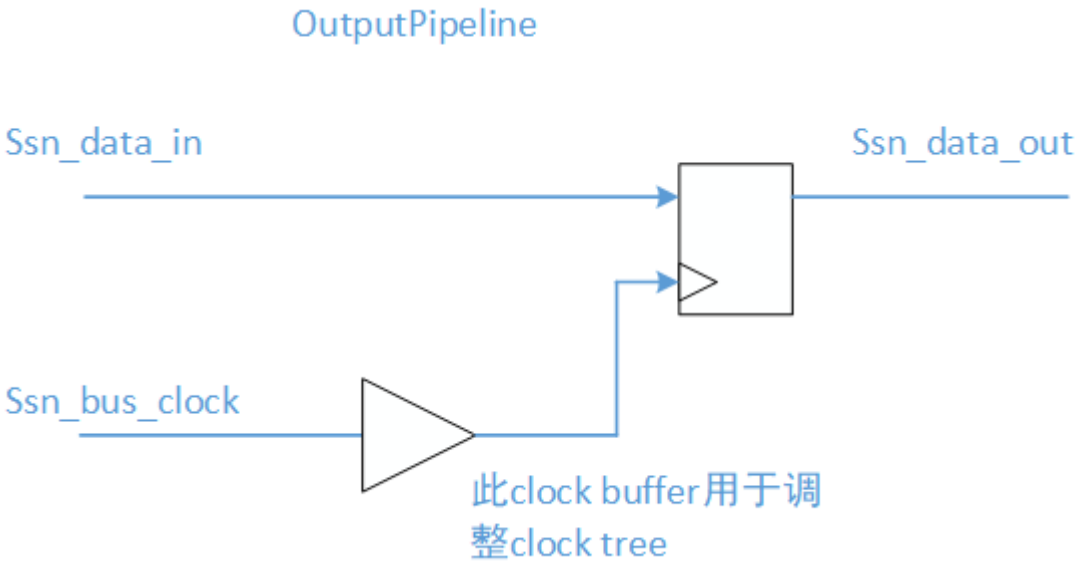
前一级是用正沿打出来的，这里先用负沿打一下是为了修hold 比如前级reg的clk tree比较短，后级的clk tree比较长，这样后级采样的时候hold容易出问题。



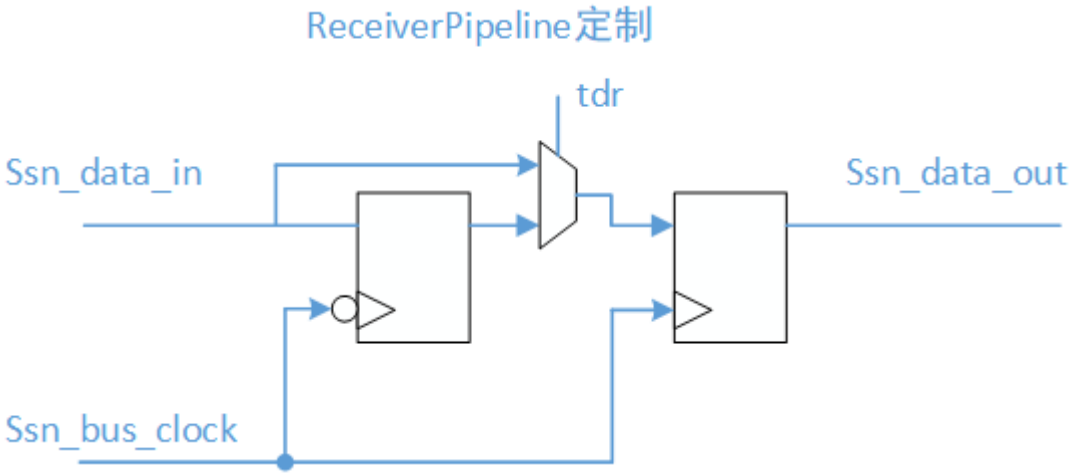

**Pipeline**




**OutputPipeline**




**RxPipeline**




## 1.2 example 1

```
DataPath(1) {
  output_bus_width       : 2;
  Pipeline(2) {
  }
  ScanHost(1) {
  }
  Pipeline(1) {
  }
}
```

graph LR ssn_busin-->pipe1 pipe1-->ssh ssh-->pipe2 pipe2-->ssn_busout

# 1.3 example

```
    DataPath(1) {
      output_bus_width       : 2;
      Pipeline(2) {
      }
      ScanHost(1) {
      }
      Multiplexer(1) {
          Connections {
              secondary_bus_data_in  : ssn_s0_bus_data_in[1:0];
          }
      }
      Pipeline(1) {
        ExtraOutputPath {
            Connections {
              bus_clock_out      : ssn_s0_bus_clock_out;
              bus_data_out       : ssn_s0_bus_data_out[1:0];
            }
        }
      }
    }
```

graph LR ssn_busin-->pipe1 pipe1-->mux1 pipe1-->ssn_s0_bus_data_out; ssn_s0_bus_data_out -.-> ssn_s0 ssn_s0-->mux1 mux1-->ssh ssh-->pipe2 pipe2-->ssn_busout

# 1.4 example

```
    DataPath(1) {
      output_bus_width       : 2;
      ScanHost(1) {
        OnChipCompareMode {
        }
      }
      Multiplexer(2) {
          Connections {
              secondary_bus_data_in  : ssn_s1_bus_data_in[1:0];
          }
      }
      Pipeline(2) {
        ExtraOutputPath {
            Connections {
              bus_clock_out      : ssn_s1_bus_clock_out;
              bus_data_out       : ssn_s1_bus_data_out[1:0];
            }
        }
      }
      Multiplexer(1) {
```

```
        Connections {
            secondary_bus_data_in  : ssn_s0_bus_data_in[1:0];
        }
    }
    Pipeline(1) {
      ExtraOutputPath {
          Connections {
            bus_clock_out      : ssn_s0_bus_clock_out;
            bus_data_out       : ssn_s0_bus_data_out[1:0];
          }
      }
    }
  }
```

graph LR ssn_busin-->pipe1 pipe1-->mux1 pipe1-->ssn_s0_bus_o; ssn_s0_bus_o -.->ssn_s0 ssn_s0-->mux1 mux1-->pipe2 pipe2-->mux2 pipe2-->ssn_s1_bus_o; ssn_s1_bus_o -.->ssn_s1 ssn_s1-->mux2 mux2-->ssh ssh-->ssn_busout

## 1.5 example

```
  DataPath(1) {
    output_bus_width       : 2;
    Pipeline(8) {
    }
    ScanHost(1) {
      OnChipCompareMode {
      }
    }
    Multiplexer(1) {
        Connections {
            secondary_bus_data_in  : ssn_s0_bus_data_in[1:0];
        }
        Pipeline(22) {
        }
    }
    Pipeline(1) {
      ExtraOutputPath {
          Connections {
            bus_clock_out      : ssn_s0_bus_clock_out;
            bus_data_out       : ssn_s0_bus_data_out[1:0];
          }
      }
    }
  }
```

graph LR ssn_busin-->pipe1 pipe1-->mux1 pipe1-->ssn_s0_bus_o; ssn_s0_bus_o -.->ssn_s0 ssn_s0-->pipe22 pipe22-->mux1 mux1-->ssh ssh-->pipe8 pipe8-->ssn_busout

# 1.6 example

```
    DataPath(1) {
      output_bus_width       : 2;
      Pipeline(8) {
      }
      ScanHost(1) {
        OnChipCompareMode {
        }
      }
      Multiplexer(1) {
          Connections {
              secondary_bus_data_in  : ssn_s0_bus_data_in[1:0];
          }
          Pipeline(22) {
          }
      }
      Pipeline(1) {
        ExtraOutputPath {
            Connections {
              bus_clock_out      : ssn_s0_bus_clock_out;
              bus_data_out       : ssn_s0_bus_data_out[1:0];
            }
            Pipeline(11) {
            }
        }
      }
    }
```

graph LR ssn_busin-->pipe1 pipe1-->mux1 pipe1-->pipe11; pipe11-->ssn_s0_bus_o; ssn_s0_bus_o -.->ssn_s0 ssn_s0-->pipe22 pipe22-->mux1 mux1-->ssh ssh-->pipe8 pipe8-->ssn_busout