DATA STREAM MULTIPLEXER

Design Specifications:

Inputs

`clk`: Internal clock signal operating at 100 MHz.

`symbol_clk`: External clock synchronizing incoming data streams, frequency ranging from 1 kHz to 50 MHz.

`mode`: 2-bit input to determine the desired mode of operation (values 00, 01, 10, or 11).

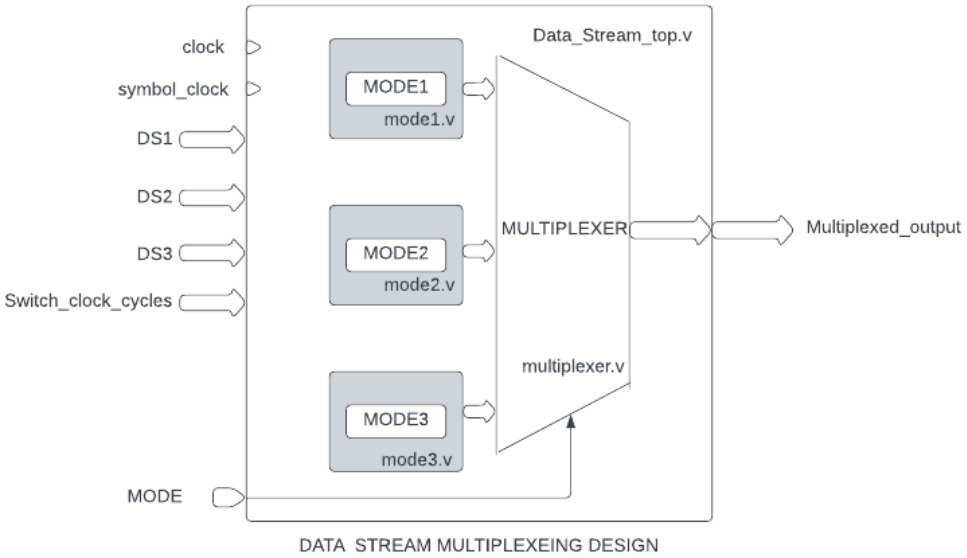
`switch_clk_cycles`: 32-bit input representing the number of `clk` cycles after which the data stream in the output should switch to the next one.

`DS1`, `DS2`, `DS3`: 16-bit data streams synchronized with the `symbol_clk`.

Outputs

`output_data`: 16-bit output data stream representing the selected data stream based on the mode of operation.

Block Diagram



Code

DataStream_top.v

```
`timescale 1ns / 1ps

module DataStream_top(input clk,
```

```
                input symbol_clk,
                input [31:0]switch_clock_cycles,
                input [1:0] mode,
                input [15:0] DS1,
                input [15:0] DS2,
                input [15:0] DS3,
                output  [15:0] multiplexed_data  );
 wire [15:0] out_1;
 wire [15:0] out_2;
 wire [15:0] out_3;
 mode1 U1(clk,symbol_clk,switch_clock_cycles,DS1,out_1);
 mode2 U2(clk,symbol_clk,switch_clock_cycles,DS1,DS2,out_2);
 mode3 U3(clk,symbol_clk,switch_clock_cycles,DS1,DS2,DS3,out_3);
 multiplexer U4(out_1,out_2,out_3,mode,multiplexed_data);
endmodule
```

Mode1.v

```
`timescale 1ns / 1ps
module mode1( input clk,
            input symbol_clk,
            input [31:0] switch_clk_cycles,
            input [15:0] DS1,
            output reg  [15:0] out_1
            );
 reg [15:0] count=1;
 always@(posedge clk) begin
 if (count <  switch_clk_cycles) begin
 out_1 <= DS1;
 count <= count+1;
 end
```

```
 else count <= 1;
 end
endmodule
```

Mode2.v

```
 `timescale 1ns / 1ps
module mode2(input clk,
             input symbol_clk,
             input [31:0] switch_clk_cycles,
             input [15:0] DS1,
             input [15:0] DS2,
             output reg  [15:0] out_2    );
 reg [15:0] count=1;
always @(posedge clk) begin

  if (count <= (switch_clk_cycles)+1) begin
 out_2<=DS1;
 count<=count+1;
 end
 else if (count < (2*switch_clk_cycles) ) begin
 out_2 <= DS2;
 count<=count+1;
 end
  else count<=1;
 end
endmodule
```

Mode3.v

```
`timescale 1ns / 1ps
module mode3( input clk,
             input symbol_clk,
```

```
        input [31:0]switch_clk_cycles,
        input [15:0] DS1,
        input [15:0] DS2,
        input [15:0] DS3,
        output reg  [15:0]  out_3   );
reg [31:0] count=1;
always @(posedge clk) begin
if (count<(switch_clk_cycles)+1  ) begin
out_3<=DS1;
count<=count+1;
end
else if (count < (2*(switch_clk_cycles))+1 ) begin
out_3<=DS2;
count<=count+1;
end
else if ( count < 3*(switch_clk_cycles)+1) begin
out_3<=DS3;
count<=count+1;
end
else count<=1;
end
endmodule
```

Multiplexer.v

```
`timescale 1ns / 1ps
module multiplexer( input [15:0] out_1,
            input [15:0] out_2,
            input [15:0] out_3,
            input [1:0]  mode,
            output reg  [15:0]  multiplexed_data
```

```
                );
always @(*) begin
case(mode)
2'd1: multiplexed_data <=out_1;
2'd2:multiplexed_data <=out_2;
2'd3:multiplexed_data <=out_3;
endcase
end
endmodule
```

Test Bench

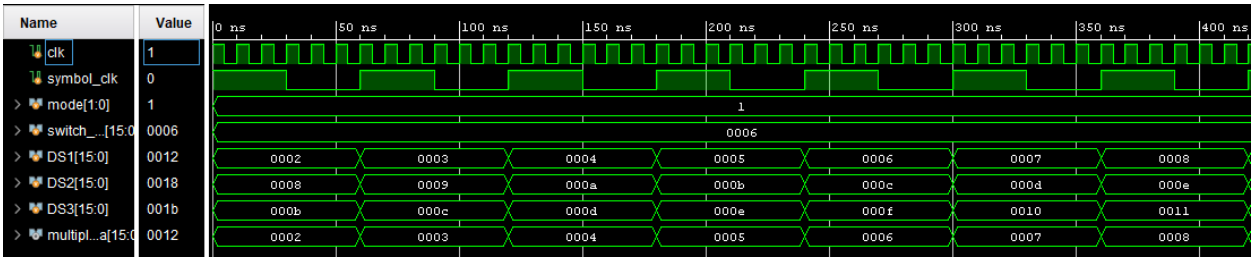
DataStream_top.v

```
`timescale 1ns / 1ps
module DataStream_tb ;
reg clk;
reg symbol_clk;
reg [1:0] mode;
reg [31:0] switch_clock_cycles;
reg [15:0] DS1;
reg [15:0] DS2;
reg [15:0] DS3;
wire [15:0] multiplexed_data;
DataStream_top DUT (
   .clk(clk),
   .symbol_clk(symbol_clk),
   .mode(mode),
   .switch_clock_cycles(switch_clock_cycles),
   .DS1(DS1),
   .DS2(DS2),
   .DS3(DS3),
```

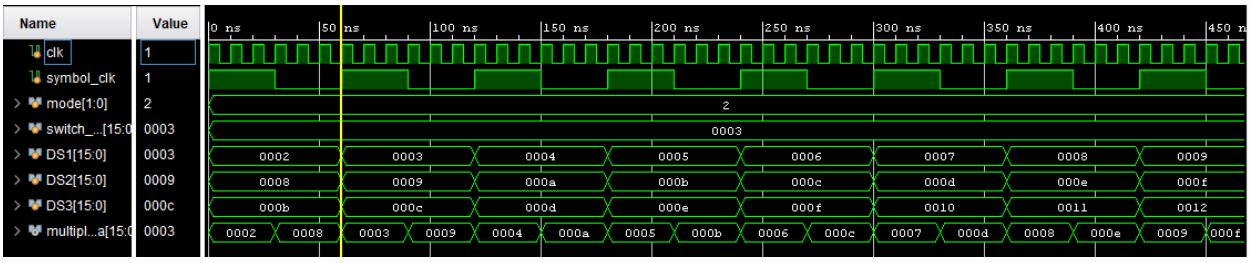
```
    .multiplexed_data(multiplexed_data)
);
integer t_clk=5;
integer t_sym_clk=10;
initial begin
clk=1;
symbol_clk=1;
//switch_clock_cycles=(t_sym_clk/t_clk);//mode1
switch_clock_cycles=(t_sym_clk/t_clk)/2;//
//switch_clock_cycles=(t_sym_clk/t_clk)/3;//mode3
mode=2'b10;
DS1=16'd1;
DS2=16'd7;
DS3=16'd10;
end
always #t_clk clk=~clk;
always #t_sym_clk symbol_clk=~symbol_clk;
always @(posedge symbol_clk) begin
DS1=DS1+1;
DS2=DS2+1;
DS3=DS3+1;
end
endmodule
```

Simulation Results

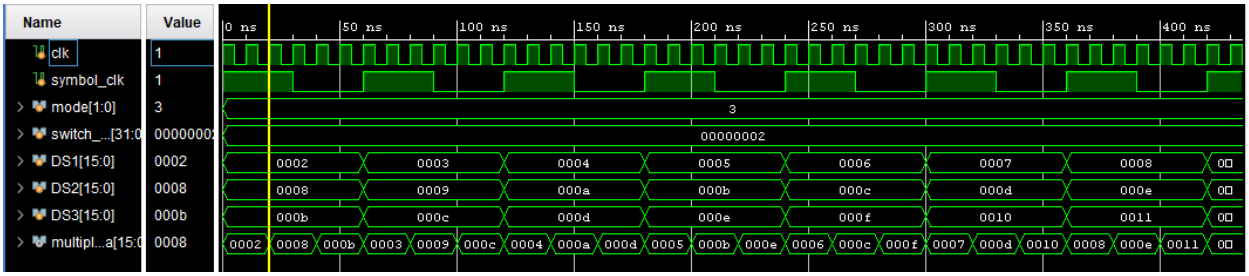
Mode1:



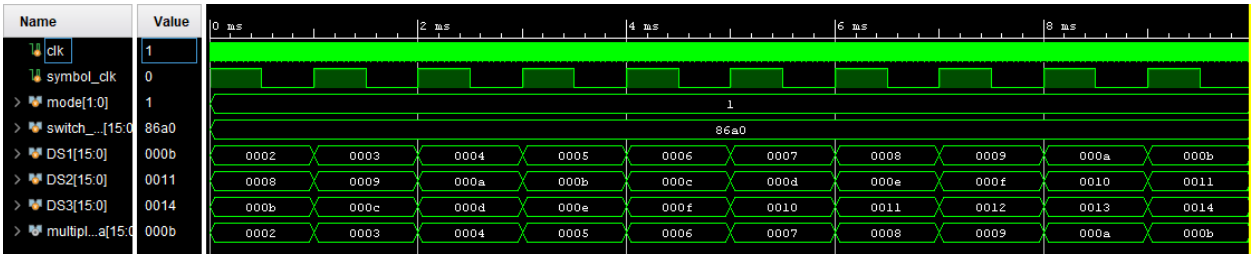
Mode2:

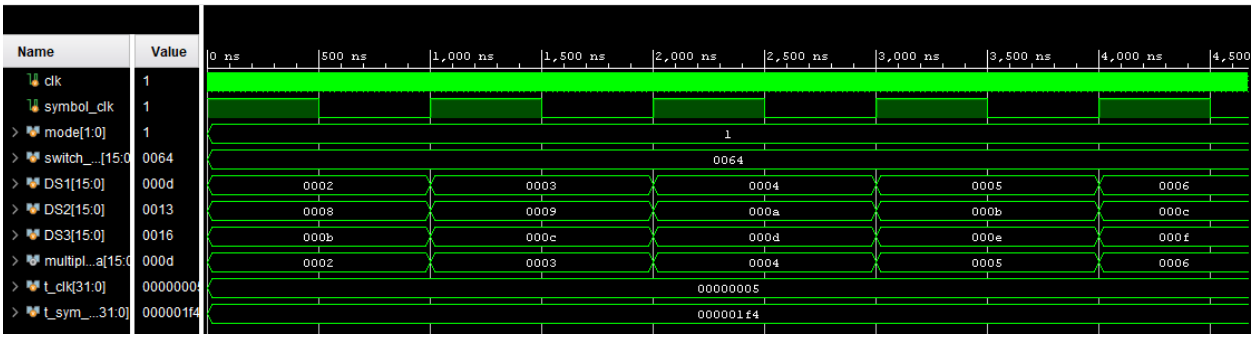


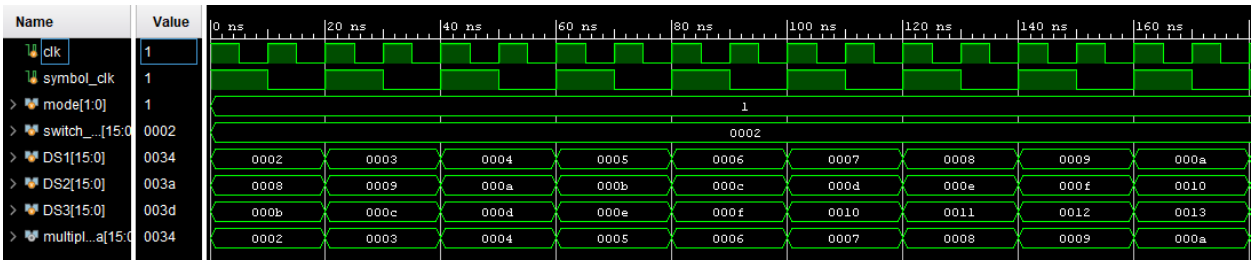
Mode3

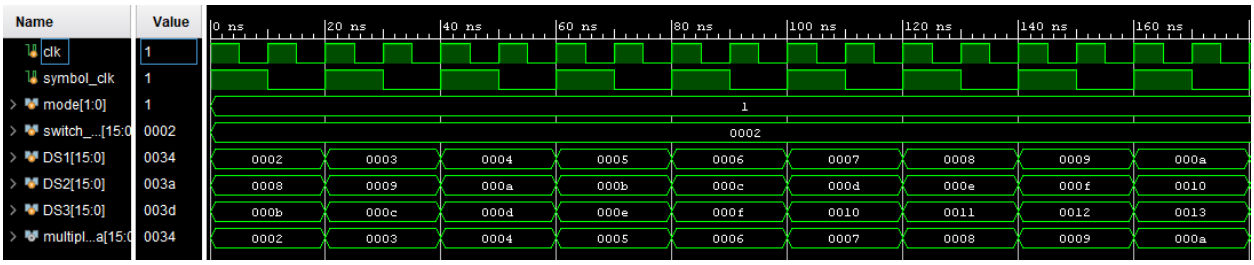


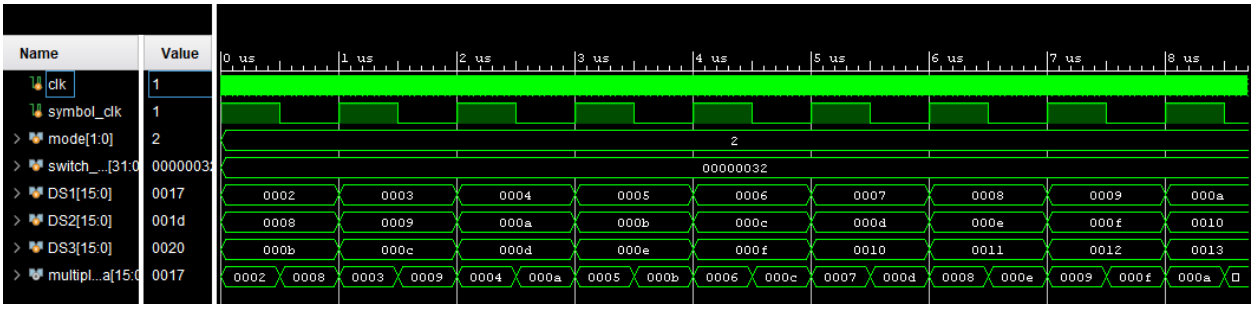
Simulation results with different symbol_clk frequencies

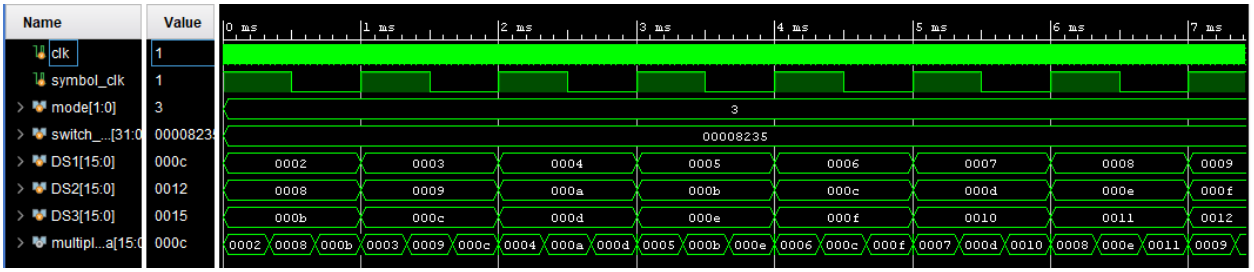


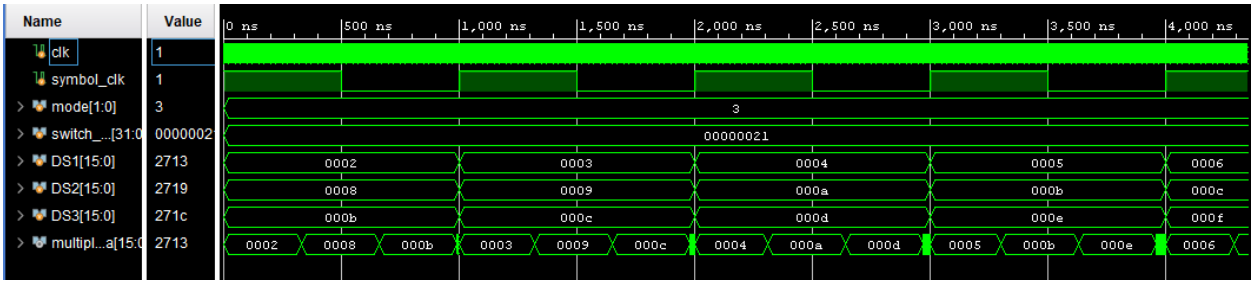

Name
Value
clk
1
symbol_clk
1
mode[1:0]
1
switch_...[15:0]
0064
DS1[15:0]
000d
DS2[15:0]
0013
DS3[15:0]
0016
multipl...a[15:0]
000d
t_clk[31:0]
00000005
t_sym_...31:0]
000001f4
0 ns
500 ns
1,000 ns
1,500 ns
2,000 ns
2,500 ns
3,000 ns
3,500 ns
4,000 ns
4,500
0002 0003 0004 0005 0006
0008 0009 000a 000b 000c
000b 000c 000d 000e 000f
0002 0003 0004 0005 0006


Name
Value
clk
1
symbol_clk
1
mode[1:0]
1
switch_...[15:0]
0002
DS1[15:0]
0034
DS2[15:0]
003a
DS3[15:0]
003d
multipl...a[15:0]
0034
0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
160 ns
0002 0003 0004 0005 0006 0007 0008 0009 000a
0008 0009 000a 000b 000c 000d 000e 000f 0010
000b 000c 000d 000e 000f 0010 0011 0012 0013
0002 0003 0004 0005 0006 0007 0008 0009 000a


Name
Value
clk
1
symbol_clk
1
mode[1:0]
1
switch_...[15:0]
0002
DS1[15:0]
0034
DS2[15:0]
003a
DS3[15:0]
003d
multipl...a[15:0]
0034
0 ns
20 ns
40 ns
60 ns
80 ns
100 ns
120 ns
140 ns
160 ns
0002 0003 0004 0005 0006 0007 0008 0009 000a
0008 0009 000a 000b 000c 000d 000e 000f 0010
000b 000c 000d 000e 000f 0010 0011 0012 0013
0002 0003 0004 0005 0006 0007 0008 0009 000a


Name
Value
clk
1
symbol_clk
1
mode[1:0]
2
switch_...[31:0]
00000032
DS1[15:0]
0017
DS2[15:0]
001d
DS3[15:0]
0020
multipl...a[15:0]
0017
0 us
1 us
2 us
3 us
4 us
5 us
6 us
7 us
8 us
0002 0003 0004 0005 0006 0007 0008 0009 000a
0008 0009 000a 000b 000c 000d 000e 000f 0010
000b 000c 000d 000e 000f 0010 0011 0012 0013
0002 0008 0003 0009 0004 000a 0005 000b 0006 000c 0007 000d 0008 000e 0009 000f 000a


Name
Value
clk
1
symbol_clk
1
mode[1:0]
3
switch_...[31:0]
00008235
DS1[15:0]
000c
DS2[15:0]
0012
DS3[15:0]
0015
multipl...a[15:0]
000c
0 ms
1 ms
2 ms
3 ms
4 ms
5 ms
6 ms
7 ms
0002 0003 0004 0005 0006 0007 0008 0009
0008 0009 000a 000b 000c 000d 000e 000f
000b 000c 000d 000e 000f 0010 0011 0012
0002 0008 000b 0003 0009 000c 0004 000a 000d 0005 000b 000e 0006 000c 000f 0007 000d 0010 0008 000e 0011 0009



Elaborated design: