HLS Lab A

Chapter 9
Advanced Hierarchical Design

Team 2
b07902123 蔡奇夆
b07209016 鐘晨瑋

Outline

- Introduction
- Coding Style
- Deadlock

Github:
https://github.com/ChungChenWei/HLS_LabA_Team2_Ch9

Introduction

Introduction

- Why blocking?
  - Read empty channel
- Different library between **Xilinx HLS** and **Mentor's Catapult**
  - **hls::stream** v.s. **ac_channel**
- Why deadlock?
- Ways to solve deadlock

Coding Style

Coding Style – In the book

BAD 👎

```
1  ac_channel<int> input0;
2  ac_channel<int> input1;
3  int data;
4  ...
5  if(input0.size()>4)
6      data = input0.read();
7  else if(input1.size()>2)
8      data = input1.read();
```

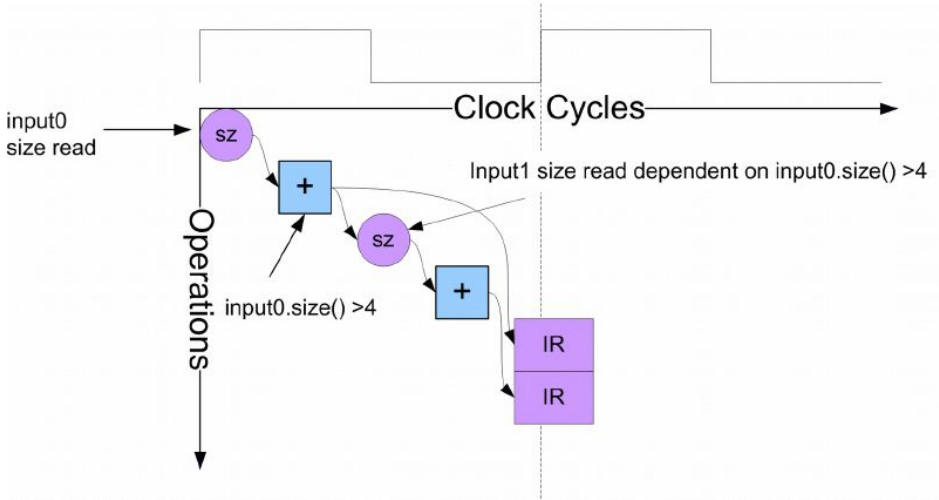
GOOD 👍

```
1  ac_channel<int> input0;
2  ac_channel<int> input1;
3  int data;
4  bool p[2];
5  ...
6  p[0] = input0.size()>4;
7  p[1] = input1.size()>2;
8  if(p[0])
9      data = input0.read();
10 else if(p[1])
11     data = input1.read();
```

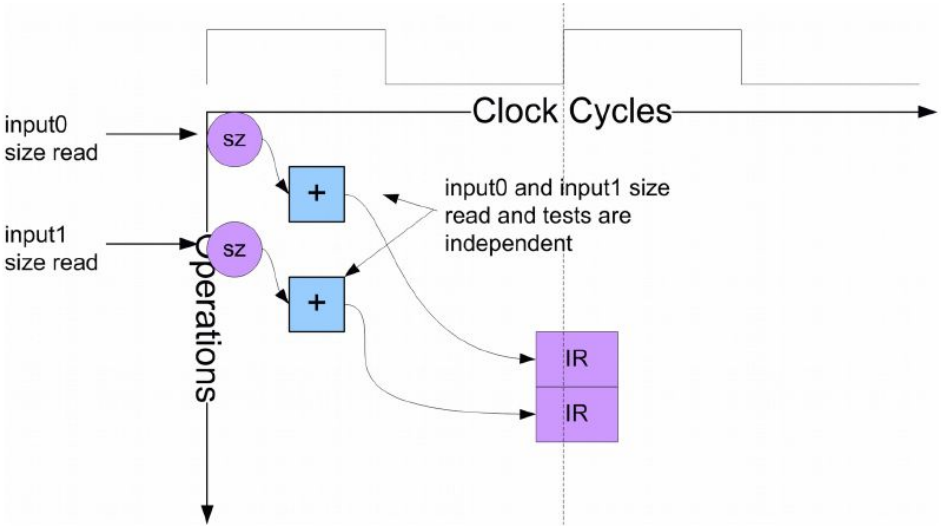
Ref: Mentor HLS Bluebook

Coding Style - In the book

BAD 👎



GOOD 👍



Ref: Mentor HLS Bluebook

Coding Style - What we do

BAD 👎

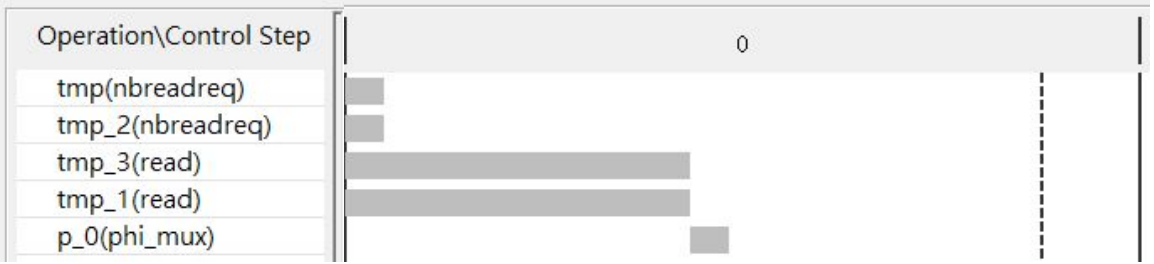
```
#include "bad-stream.hpp"

int top (stream_t &in1, stream_t &in2) {
    if (!in1.empty()) {
        return in1.read();
    }
    else if (!in2.empty()) {
        return in2.read();
    }
    return 0;
}
```

GOOD 👍

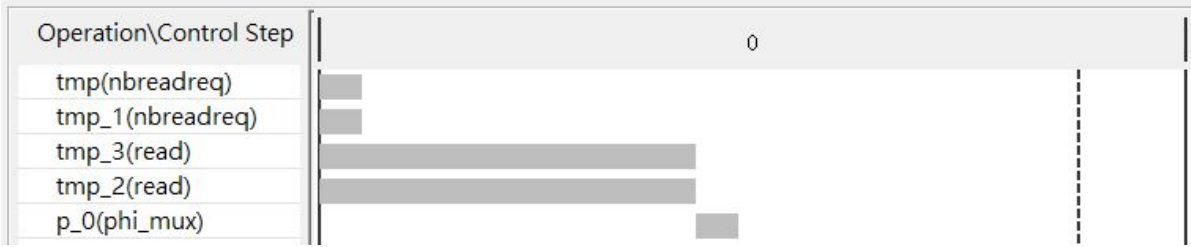
```
#include "good-stream.hpp"

int top (stream_t &in1, stream_t &in2) {
    bool cond1, cond2;

    cond1 = in1.empty();
    cond2 = in2.empty();

    if (!cond1) {
        return in1.read();
    }
    else if (!cond2) {
        return in2.read();
    }
    return 0;
}
```

Coding Style - Result

BAD 👎



GOOD 👍

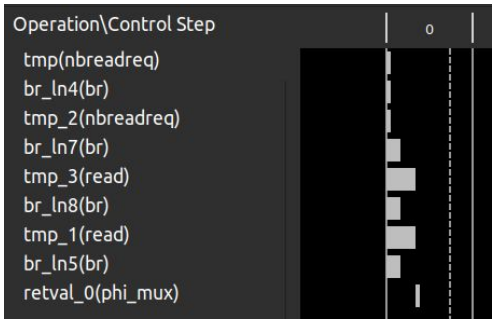
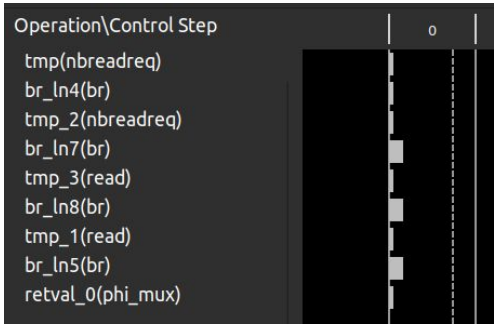
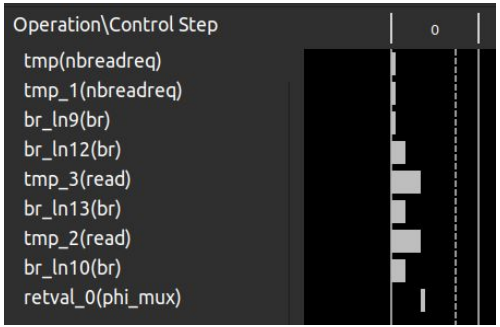
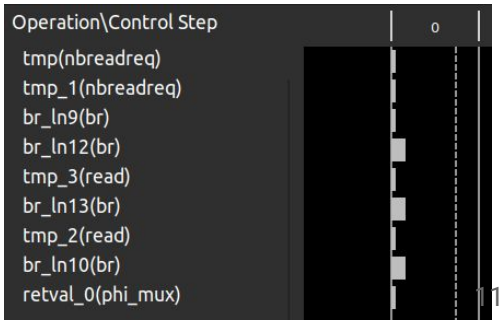


Coding Style - Result

```
always @ (*) begin
    if ((~((ap_start == 1'b0) | ((ap_predicate_op12_read_state1 == 1'b1) & (in2_V_empty_n == 1'b0)) | ((tmp_nbreadreq_fu_24_p3 == 1'd1) &
        in1_V_read = 1'b1;
    end else begin
        in1_V_read = 1'b0;
    end
end

always @ (*) begin
    if ((~((ap_start == 1'b0) | ((ap_predicate_op12_read_state1 == 1'b1) & (in2_V_empty_n == 1'b0)) | ((tmp_nbreadreq_fu_24_p3 == 1'd1) &
        in2_V_read = 1'b1;
    end else begin
        in2_V_read = 1'b0;
    end
end
```

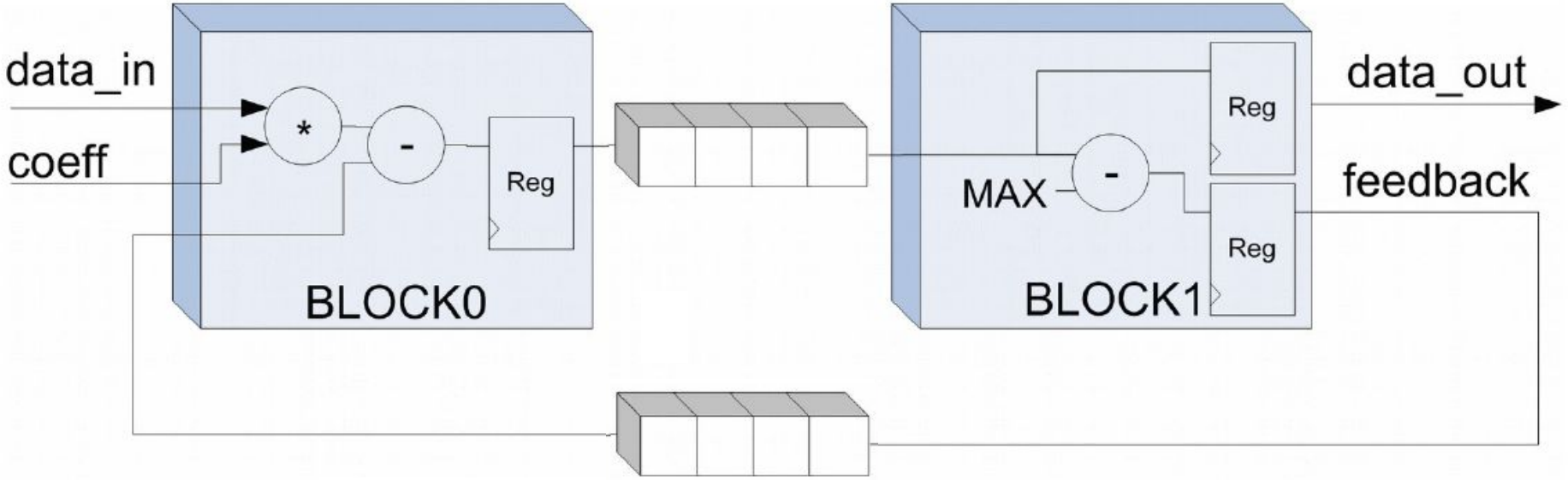
Coding Style – Extra

The default implementation of *hls::stream<>* is implemented as a FIFO with a default depth of 2

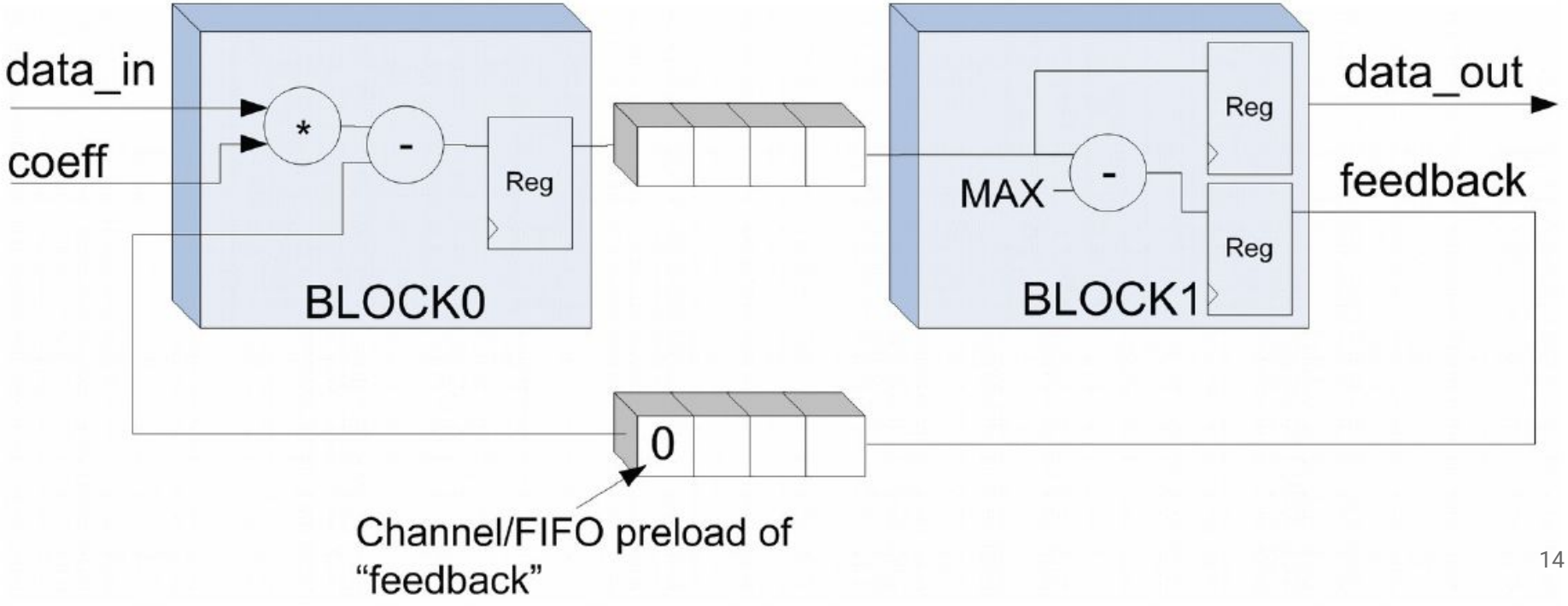
	FIFO	AXI4-Stream
bad style		
good style		

Deadlock

Deadlock - In the book



Deadlock - In the book



Deadlock – What we do

```cpp
#include "has-deadlock.hpp"

void block0 (stream_t &data_i, stream_t &coef_i, stream_t &feedback_i, stream_t &data_o) {


    data_o.write(data_i.read() * coef_i.read() - feedback_i.read());
}

void block1 (stream_t &data_i, stream_t &data_o, stream_t &feedback_o) {
    int val = data_i.read();
    int fb = (val > MAXFEEDBACK) ? MAXFEEDBACK : val;

    data_o.write(val);
    feedback_o.write(fb);
}

void top (stream_t &data_i, stream_t &coef_i, stream_t &data_o) {
#pragma HLS PIPELINE
    static stream_t data ("intermediate_data");
    static stream_t feedback ("feedback");

    block0(data_i, coef_i, feedback, data);
    block1(data, data_o, feedback);
}
```

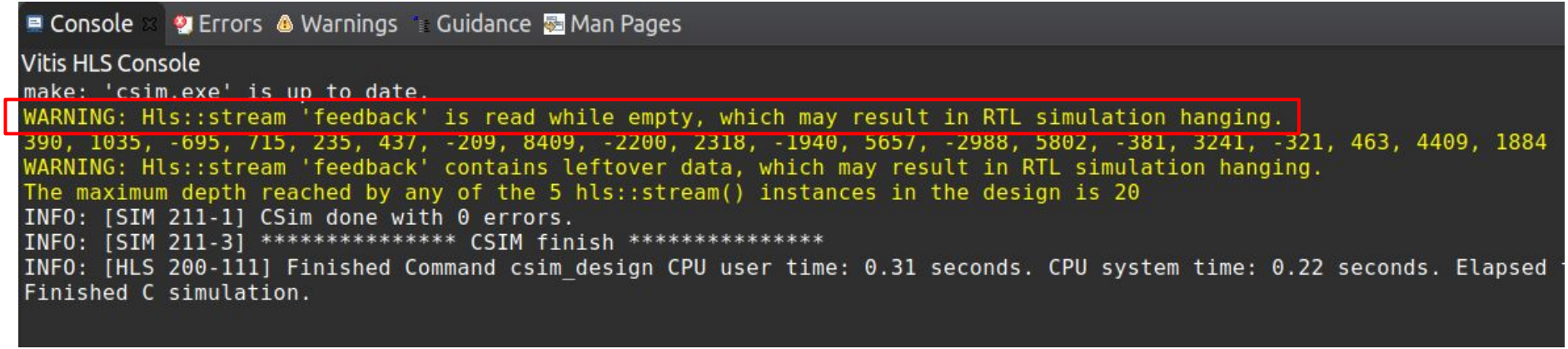
```
typedef hls::stream<int> stream_t;
```

Deadlock – What we do

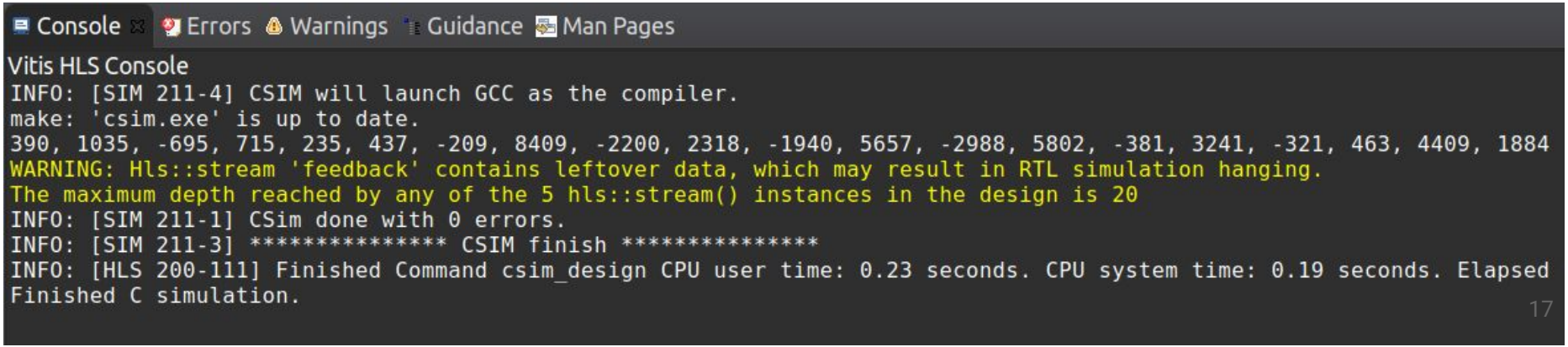
```
1  #include "no-deadlock.hpp"
2
3  void block0 (stream_t &data_i, stream_t &coef_i, stream_t &feedback_i, stream_t &data_o) {
4      int fb = feedback_i.empty() ? 0 : feedback_i.read();
5
6      data_o.write(data_i.read() * coef_i.read() - fb);
7  }
8
9  void block1 (stream_t &data_i, stream_t &data_o, stream_t &feedback_o) {
10     int val = data_i.read();
11     int fb = (val > MAXFEEDBACK) ? MAXFEEDBACK : val;
12
13     data_o.write(val);
14     feedback_o.write(fb);
15 }
16
17 void top (stream_t &data_i, stream_t &coef_i, stream_t &data_o) {
18 #pragma HLS PIPELINE
19     static stream_t data ("intermediate_data");
20     static stream_t feedback ("feedback");
21
22     block0(data_i, coef_i, feedback, data);
23     block1(data, data_o, feedback);
24 }
```


typedef hls::stream<int> stream_t;

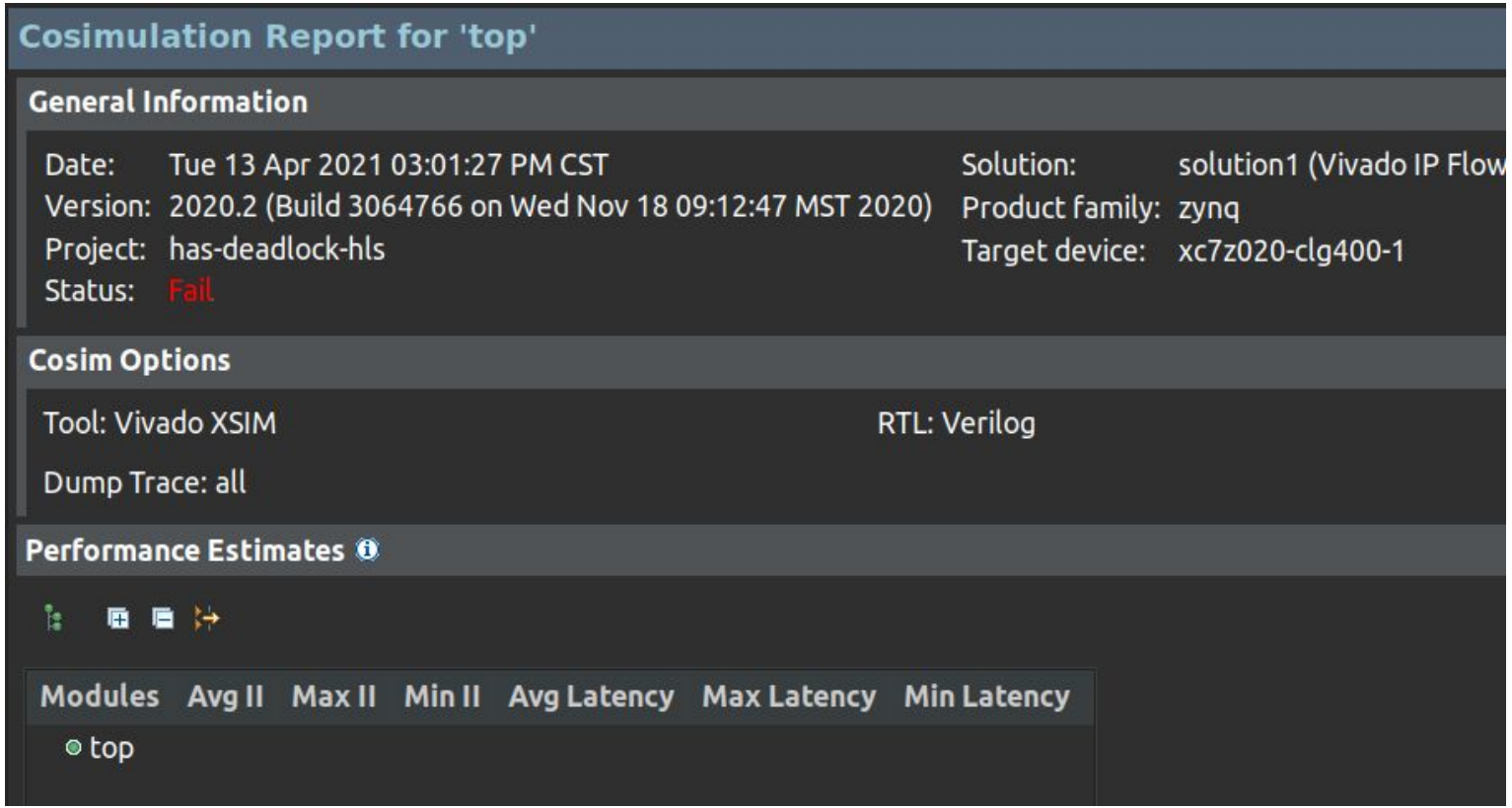
deadlock



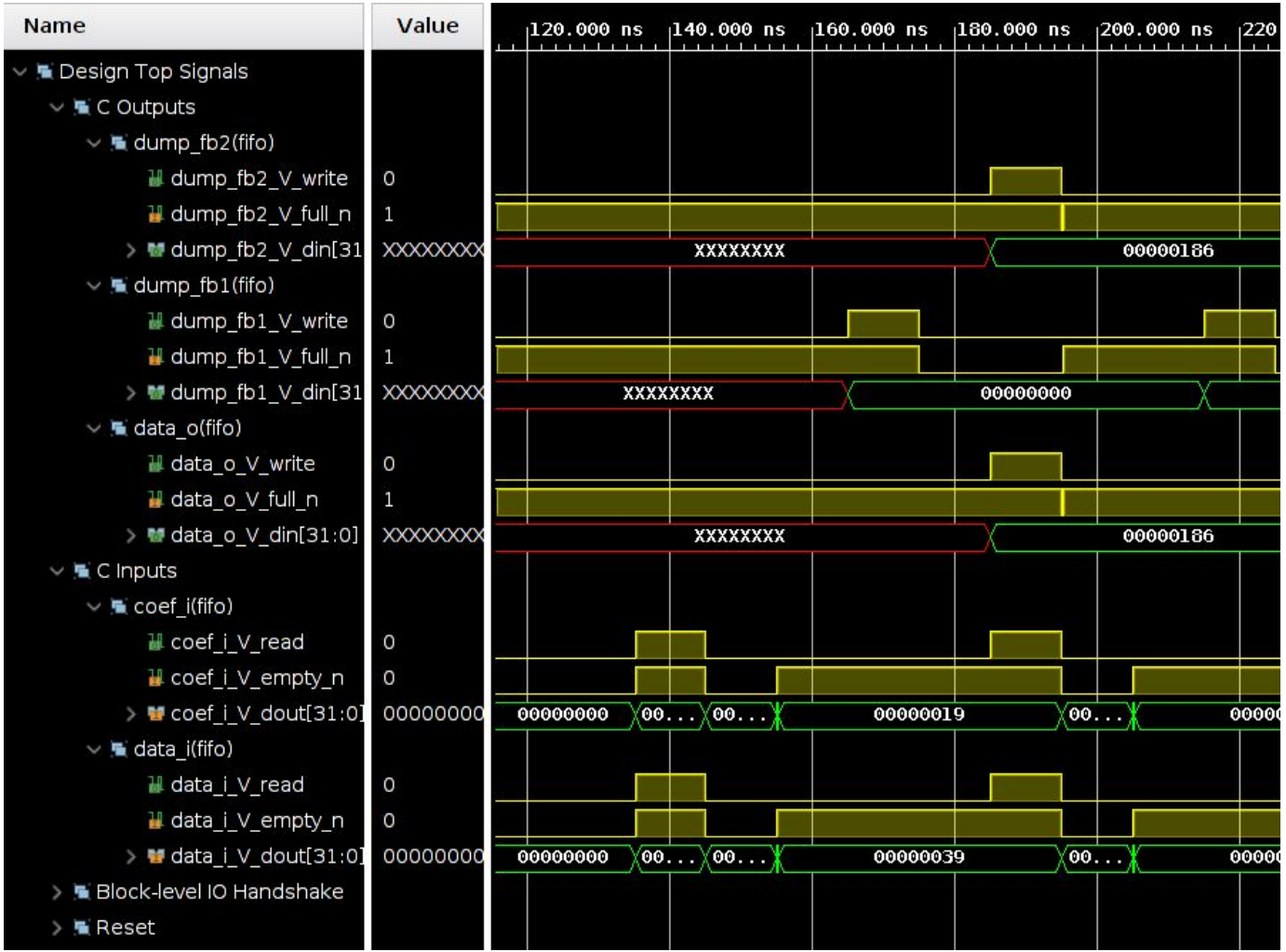
non-blocking



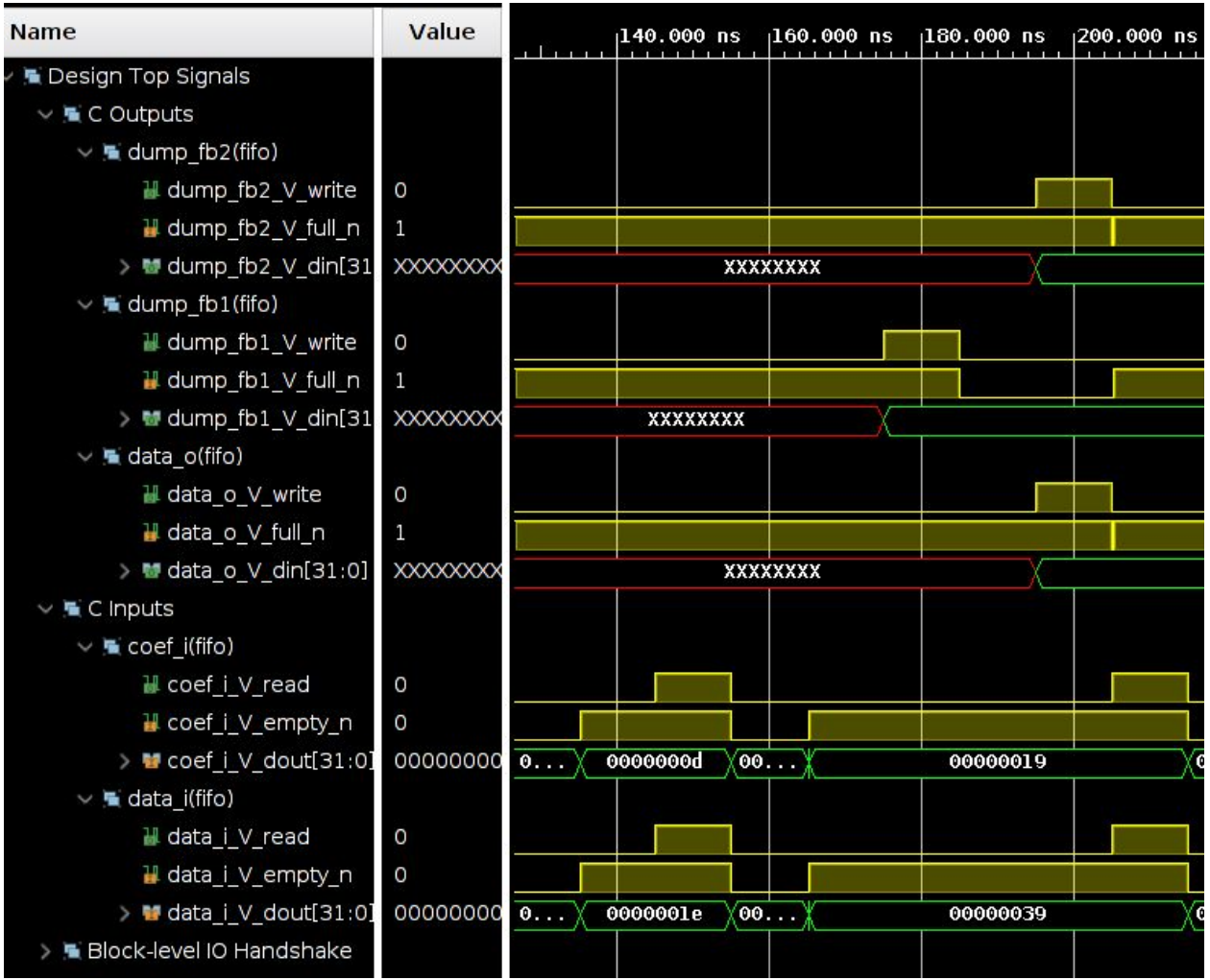
without
preload



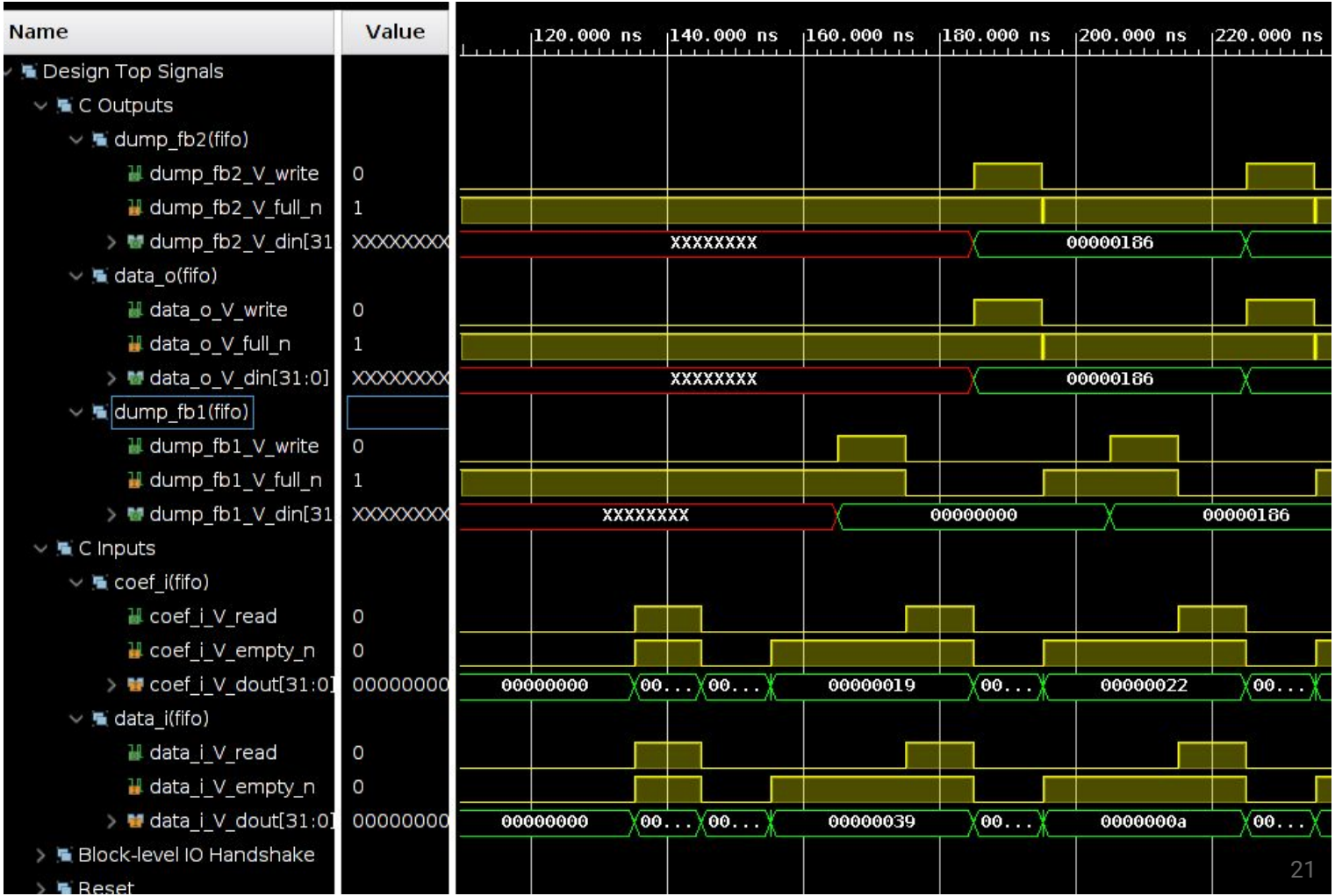
preload=1


Name
Value
120.000 ns
140.000 ns
160.000 ns
180.000 ns
200.000 ns
220
Design Top Signals
C Outputs
dump_fb2(fifo)
dump_fb2_V_write 0
dump_fb2_V_full_n 1
dump_fb2_V_din[31 XXXXXXXX
XXXXXXXX
00000186
dump_fb1(fifo)
dump_fb1_V_write 0
dump_fb1_V_full_n 1
dump_fb1_V_din[31 XXXXXXXX
XXXXXXXX
00000000
data_o(fifo)
data_o_V_write 0
data_o_V_full_n 1
data_o_V_din[31:0] XXXXXXXX
XXXXXXXX
00000186
C Inputs
coef_i(fifo)
coef_i_V_read 0
coef_i_V_empty_n 0
coef_i_V_dout[31:0] 00000000
00000000
00...
00...
00000019
00...
00000
data_i(fifo)
data_i_V_read 0
data_i_V_empty_n 0
data_i_V_dout[31:0] 00000000
00000000
00...
00...
00000039
00...
00000
Block-level IO Handshake
Reset

preload=2


Name
Value
140.000 ns
160.000 ns
180.000 ns
200.000 ns
Design Top Signals
C Outputs
dump_fb2(fifo)
dump_fb2_V_write
0
dump_fb2_V_full_n
1
dump_fb2_V_din[31
XXXXXXXX
dump_fb1(fifo)
dump_fb1_V_write
0
dump_fb1_V_full_n
1
dump_fb1_V_din[31
XXXXXXXX
data_o(fifo)
data_o_V_write
0
data_o_V_full_n
1
data_o_V_din[31:0]
XXXXXXXX
C Inputs
coef_i(fifo)
coef_i_V_read
0
coef_i_V_empty_n
0
coef_i_V_dout[31:0]
00000000
0...
0000000d
00...
00000019
data_i(fifo)
data_i_V_read
0
data_i_V_empty_n
0
data_i_V_dout[31:0]
00000000
0...
0000001e
00...
00000039
Block-level IO Handshake

non-blocking



Details: scheduling

preload = 1

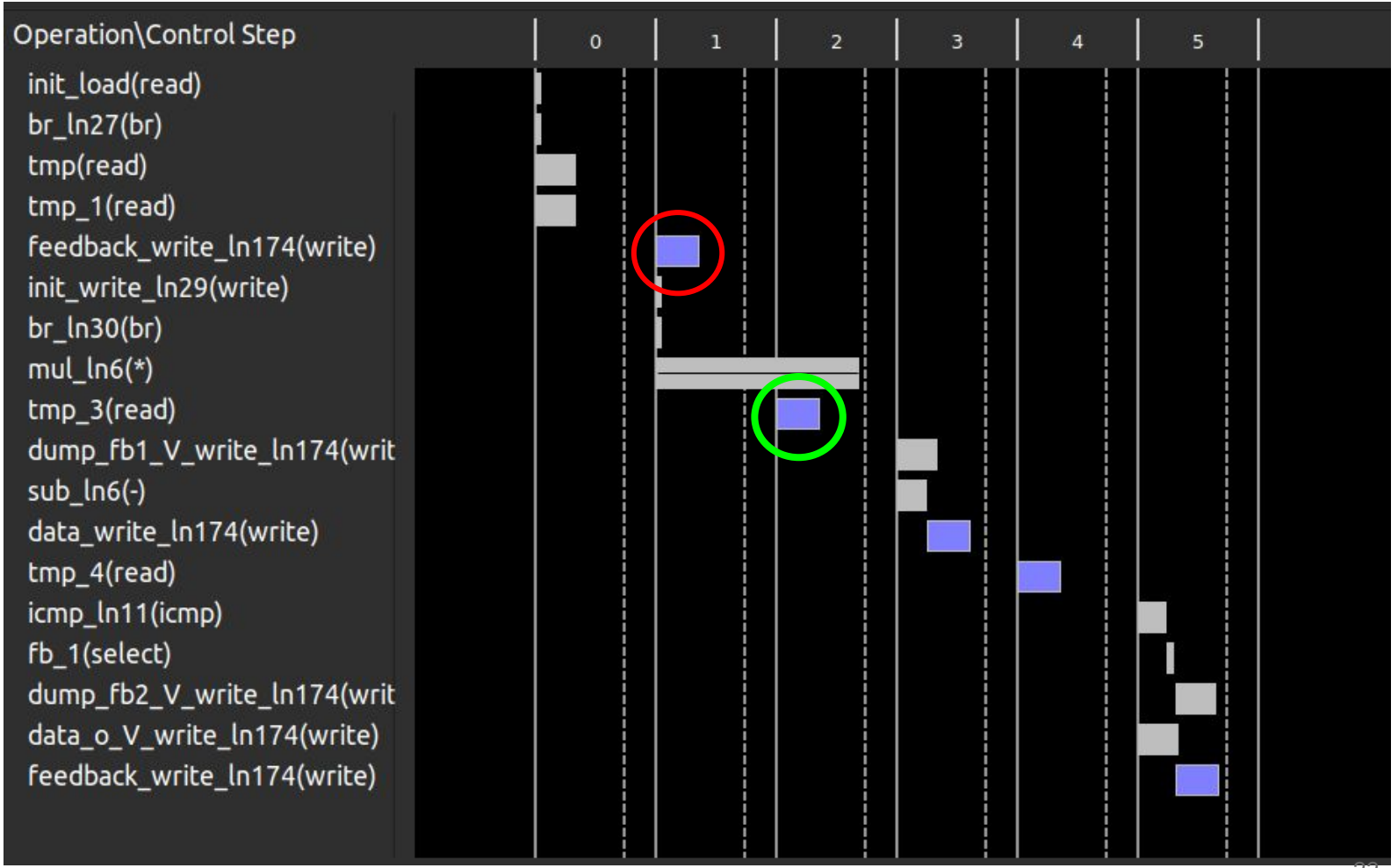
```
void top (stream_t &data_i, stream_t &coef_i, stream_t &data_o, stream_t &dump_fb1, stream_t &dump_fb2) {
#pragma HLS PIPELINE
#pragma HLS INTERFACE ap_fifo depth=20 port=dump_fb1
#pragma HLS INTERFACE ap_fifo depth=20 port=dump_fb2

    static stream_t data ("intermediate_data");
    static stream_t feedback ("feedback");
    static bool init = true;

    if (init) {
        feedback.write(0); // preload
        init = false;
    }
    block0(data_i, coef_i, feedback, data, dump_fb1);
    block1(data, data_o, feedback, dump_fb2);
}
```

preload = 2

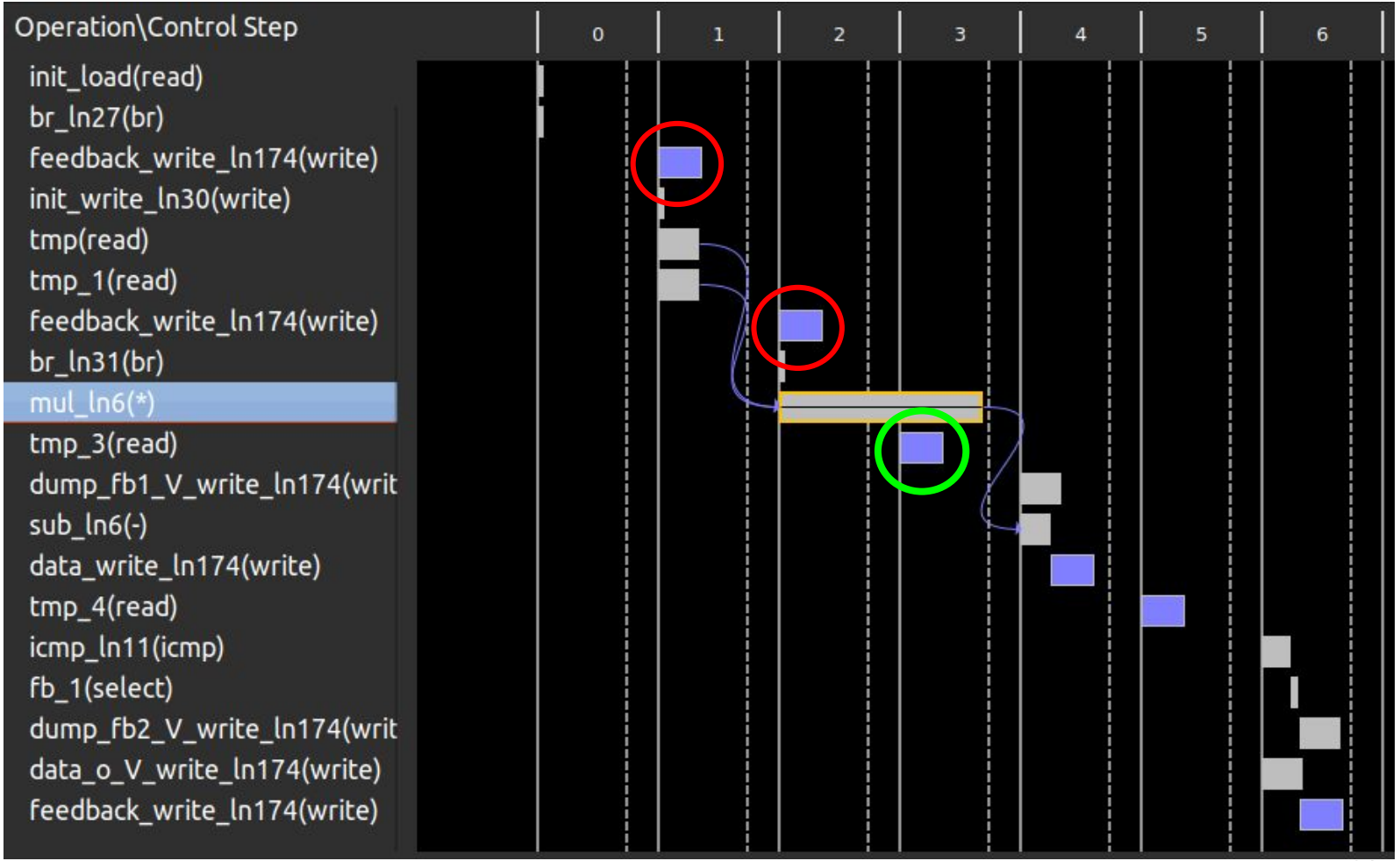
```
void top (stream_t &data_i, stream_t &coef_i, stream_t &data_o, stream_t &dump_fb1, stream_t &dump_fb2) {
#pragma HLS PIPELINE
#pragma HLS INTERFACE ap_fifo depth=20 port=dump_fb1
#pragma HLS INTERFACE ap_fifo depth=20 port=dump_fb2

    static stream_t data ("intermediate_data");
    static stream_t feedback ("feedback");
    static bool init = true;

    if (init) {
        feedback.write(0); // preload
        feedback.write(0); // preload
        init = false;
    }
    block0(data_i, coef_i, feedback, data, dump_fb1);
    block1(data, data_o, feedback, dump_fb2);
}
```

```
typedef hls::stream<int> stream_t;
```

preload = 1


Operation\Control Step
0
1
2
3
4
5
init_load(read)
br_ln27(br)
tmp(read)
tmp_1(read)
feedback_write_ln174(write)
init_write_ln29(write)
br_ln30(br)
mul_ln6(*)
tmp_3(read)
dump_fb1_V_write_ln174(writ
sub_ln6(-)
data_write_ln174(write)
tmp_4(read)
icmp_ln11(icmp)
fb_1(select)
dump_fb2_V_write_ln174(writ
data_o_V_write_ln174(write)
feedback_write_ln174(write)

preload = 2


Operation\Control Step
0
1
2
3
4
5
6
init_load(read)
br_ln27(br)
feedback_write_ln174(write)
init_write_ln30(write)
tmp(read)
tmp_1(read)
feedback_write_ln174(write)
br_ln31(br)
mul_ln6(*)
tmp_3(read)
dump_fb1_V_write_ln174(writ
sub_ln6(-)
data_write_ln174(write)
tmp_4(read)
icmp_ln11(icmp)
fb_1(select)
dump_fb2_V_write_ln174(writ
data_o_V_write_ln174(write)
feedback_write_ln174(write)

Final comments

- Coding Style
  - Compiler optimization?
  - Try it out!

- About Deadlock
  - add “PIPELINE” directive if needed
  - don’t implement preload by yourself as in previous examples
  - use **non-blocking** mode to avoid deadlock

Thanks for listening!

You're welcome!