Анализ разработанного модуля

Анализ критических путей в дизайне

Данный модуль является комбинационным, поэтому для анализа критических путей необходимо было добавить в проект файл, который будет содержать дополнительные параметры для его временного анализа. Был добавлен файл timing.xdc который содержал следующие строки:

- create_clock -name VCLK -period 8.0 -waveform {0 4.0} - создание виртуальных часов для подсчета временных задержек блока, с именем VCLK, пириодом в 8 нс, переход с единицы на ноль происходит в середине периода.
- set_input_delay 1.0 -clock [get_clocks VCLK] [all_inputs] - для всех входных портов новый сигнал поступает через 1нс после VCLK нарастания.
- set_output_delay 1.0 -clock [get_clocks VCLK] [all_outputs] - все выходные сигналы должны быть установлены за 1нс до VCLK нарастания.

При применении данных параметров я получил такой результат

Setup		**Hold**	
Worst Negative Slack (WNS):	0,255 ns	Worst Hold Slack (WHS):	3,606 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns

Setup

Intra-Clock Paths - VCLK - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 1	0.255	5	6	22	rst_n	m_last_o[0]	5.720	3.451	2.269	8.0	VCLK	VCLK		0.025
Path 2	0.704	4	5	5	m_ready_i[1]	m_last_o[1]	5.271	3.412	1.859	8.0	VCLK	VCLK		0.025
Path 3	2.100	2	3	3	m_ready_i[0]	m_valid_o[0]	3.875	3.292	0.584	8.0	VCLK	VCLK		0.025
Path 4	2.100	2	3	5	m_ready_i[1]	m_valid_o[1]	3.875	3.292	0.584	8.0	VCLK	VCLK		0.025
Path 5	2.100	2	3	3	s_valid_i[0]	s_ready_o[0]	3.875	3.292	0.584	8.0	VCLK	VCLK		0.025
Path 6	2.100	2	3	5	s_valid_i[1]	s_ready_o[1]	3.875	3.292	0.584	8.0	VCLK	VCLK		0.025

Hold

Intra-Clock Paths - VCLK - Hold

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock	Exception	Clock Uncertainty
Path 7	3.606	2	3	3	m_ready_i[0]	m_valid_o[0]	1.631	1.385	0.246	0.0	VCLK	VCLK		0.025
Path 8	3.606	2	3	5	m_ready_i[1]	m_valid_o[1]	1.631	1.385	0.246	0.0	VCLK	VCLK		0.025
Path 9	3.606	2	3	3	s_valid_i[0]	s_ready_o[0]	1.631	1.385	0.246	0.0	VCLK	VCLK		0.025
Path 10	3.606	2	3	5	s_valid_i[1]	s_ready_o[1]	1.631	1.385	0.246	0.0	VCLK	VCLK		0.025
Path 11	4.044	4	5	2	s_last_i[1]	m_last_o[1]	2.069	1.442	0.627	0.0	VCLK	VCLK		0.025
Path 12	4.046	4	5	4	s_dest_i[0]	m_last_o[0]	2.071	1.441	0.630	0.0	VCLK	VCLK		0.025

Стоит упомянуть, что данный синтез, был произведен для кроссбара из примера, а именно с двумя master, slave и разрядностью данных в 8 бит, соответственно для других реализация время будет различаться.

Получается при данной реализации тактовая частота может быть около 125 Мгц, что не сильно быстроё. Как видно по табличке первые два пути в setup являются критическими и сильно отличаются от следующих. Данные пути — это выходные сигналы m_last_o, они информируют приемник о том, что данный пакет является последним. Изменив путь для этих сигналов, мы сможем заметно ускорить модуль.

По логике работы данные сигналы формируется в модуле round-robin. После полного опроса арбитражем всех запросов, модуль начинает вычислять значение сигнала m_last_o, хотя это совсем не обязательно. Данный сигнал может быть вычислен сразу после нахождения следующего отправителя. Так, пришлось изменить модуль:

```
if(m_last_o) begin
    if(state == S_DATA_COUNT - 1) begin
        state = 0;
    end else begin
        state = state + 1;
    end

end

for (i = 0; i < S_DATA_COUNT; i = i + 1) begin
    if (request_mask_i[(state + i) % S_DATA_COUNT] && !first_one) begin
        grant_o[(state + i) % S_DATA_COUNT] = 1'b1;
        first_one = 1;
        state = (state + i) % S_DATA_COUNT;
    end else begin
        grant_o[(state + i) % S_DATA_COUNT] = 1'b0;
    end
end

if (s_last_i[state]) begin
    m_last_o = 1;
end else m_last_o = 0;
```

```
if(m_last_o) begin
    m_last_o = 0;
    if(state == S_DATA_COUNT - 1) begin
        state = 0;
    end else begin
        state = state + 1;
    end

end

for (i = 0; i < S_DATA_COUNT; i = i + 1) begin
    if (request_mask_i[(state + i) % S_DATA_COUNT] && !first_one) begin
        grant_o[(state + i) % S_DATA_COUNT] = 1'b1;
        first_one = 1;
        if (s_last_i[(state + i) % S_DATA_COUNT]) m_last_o = 1;
        state = (state + i) % S_DATA_COUNT;
    end else begin
        grant_o[(state + i) % S_DATA_COUNT] = 1'b0;
    end
end
```

Изменив путь для этого сигнала, я смог повысить тактовую частоту до 166 Мгц, при учете input, output delay = 1нс. Если считать частоту без учета данных задержек, то финальная частота модуля равна 250 Мгц.

Setup		**Hold**	
Worst Negative Slack (WNS):	0,100 ns	Worst Hold Slack (WHS):	3,606 ns
Total Negative Slack (TNS):	0,000 ns	Total Hold Slack (THS):	0,000 ns

Setup

Intra-Clock Paths - VCLK - Setup

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock
Path 1	0.100	2	3	3	m_ready_i[0]	m_valid_o[0]	3.875	3.292	0.584	6.0	VCLK	VCLK
Path 2	0.100	2	3	3	m_ready_i[1]	m_valid_o[1]	3.875	3.292	0.584	6.0	VCLK	VCLK
Path 3	0.100	2	3	3	s_valid_i[0]	s_ready_o[0]	3.875	3.292	0.584	6.0	VCLK	VCLK
Path 4	0.100	2	3	3	s_valid_i[1]	s_ready_o[1]	3.875	3.292	0.584	6.0	VCLK	VCLK

Hold

Intra-Clock Paths - VCLK - Hold

Name	Slack	Levels	Routes	High Fanout	From	To	Total Delay	Logic Delay	Net Delay	Requirement	Source Clock	Destination Clock
Path 5	3.606	2	3	3	m_ready_i[0]	m_valid_o[0]	1.631	1.385	0.246	0.0	VCLK	VCLK
Path 6	3.606	2	3	3	m_ready_i[1]	m_valid_o[1]	1.631	1.385	0.246	0.0	VCLK	VCLK
Path 7	3.606	2	3	3	s_valid_i[0]	s_ready_o[0]	1.631	1.385	0.246	0.0	VCLK	VCLK
Path 8	3.606	2	3	3	s_valid_i[1]	s_ready_o[1]	1.631	1.385	0.246	0.0	VCLK	VCLK

Рекомендации по изменению дизайна, которые могут повысить частоту и/или пропускную способность

Кроме уже предложенного ранее решения для повышения частоты данного модуля можно применить технологию конвейера.

Благодаря конвейеризации (временной параллелизм) можно увеличивает скорость работы схемы без сильного увеличения аппаратных затрат. Разделив данную схему на n участков, при условии, что временная задержка на всех участках одинаковая, мы сможем сохранить прежнюю латентность системы при серьезном повышении пропускной способности.