Computer System Design

陈浩东 Chen Haodong

232320029

2023.10.24

Variant number: 9

1 C code of the module:

The C code of this module is shown below, we used $\#pragma\ HLS\ PIPELINE$ to pipeline the design. $\#pragma\ HLS\ INTERFACE\ ap\_none\ port = x$ instructs the HLS tool to create an interface for the variable or port x that doesn't have a specific protocol. Typically, such an interface is just a simple connection without complex control or handshake signals. $\#pragma\ HLS\ INTERFACE\ ap\_vld\ port = ap\_return\ register$ instructs the HLS tool to create an interface for the ap_return port that comes with a valid signal. The ap_vld interface typically contains a valid signal which indicates when the output data is valid as data becomes available.

```
unsigned int ln(unsigned int x) {
 #pragma HLS INTERFACE ap_none port=x
 #pragma HLS INTERFACE ap_vld port=ap_return register

   unsigned int div2 = 128;
   unsigned int div3 = 85;
   unsigned int div4 = 64;
   unsigned int term0 = x;

   unsigned long pow2_32;
   unsigned int pow2;
   unsigned long pow3_32;
   unsigned int pow3;
   unsigned long term1_32;
   unsigned int term1;
   unsigned long pow4_32;
   unsigned long term2_32;
   unsigned int term2;
   unsigned int term0_minus_term1;
   unsigned long term3_32;
   unsigned int term3;
   unsigned int term0_minus_term1_plus_term2;
```

```
23.    unsigned int y;
24.    unsigned int pow4;
25.
26.    // Pipeline loop
27.    for (int stage = 0; stage < 5; stage++) {
28.        #pragma HLS pipeline
29.
30.        switch (stage) {
31.            case 0:
32.                // stage 0
33.                pow2_32 = x * x;
34.                pow2 = pow2_32 >> 8;
35.                break;
36.
37.            case 1:
38.                // stage 1
39.                pow3_32 = x * pow2;
40.                pow3 = pow3_32 >> 8;
41.
42.                term1_32 =  pow2 * div2;
43.                term1 = term1_32 >> 8;
44.                break;
45.
46.            case 2:
47.                // stage 2
48.                pow4_32 = (unsigned long)pow2 * (unsigned long)pow2;
49.                pow4 = pow4_32 >> 8;
50.
51.                term2_32 = pow3 * div3;
52.                term2 = term2_32 >> 8;
53.                term0_minus_term1 = term0 - term1;
54.                break;
55.
56.            case 3:
57.                // stage 3
58.                term3_32 = pow4 * div4;
59.                term3 = term3_32 >> 8;
60.                term0_minus_term1_plus_term2 = term0_minus_term1 + term2;
61.                break;
62.
63.            case 4:
64.                // stage 4
65.                y = term0_minus_term1_plus_term2 - term3;
66.                break;
```

```
67.         }
68.     }
69.
70.     return y;
71. }
```

2 Screenshot of obtained simulation waveforms:

Figure 1 proves that the results obtained from Vivado HLS is correct. In figure 2, after $x[31:0]$ is input at time 54595 ns which is also the start time of $stage\ 0$, the $y\_1\_fu\_70$, which represents the generated result, is changed at $stage\ 4$, time 54645 ns. According to the definition of latency, the time from input start to output is stable, we can conclude that the latency of this design is 50 ns, which is equal to 5 clock cycle since we set a clock cycle equal to 10 ns (100 Mhz). I suppose the period time of stage 5 is longer than a clock cycle is due to the setup and hold time of output, but I still confused about this phenomenon. Because of the pipeline design, the initiation interval of this design is a clock cycle (10 ns). The reason why I didn't give the figure of Co-simulation waveform is that a compile error:

```
1. Compiling apatb_ln.cpp
2. apatb_ln.cpp:16:27: fatal error: autopilot_cbe.h: No such file or directory
```

is met. After searching the internet, the error tells that a head file is missing, and this header file should have been installed at the same time when installing Vivado HLS. I tried reinstalling Vivado but still did not work.

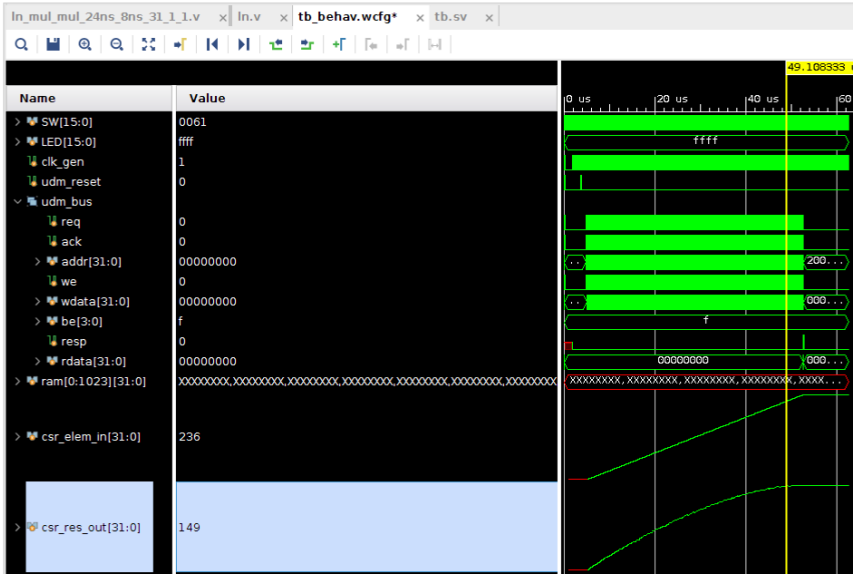


Figure 1: simulation waveform.

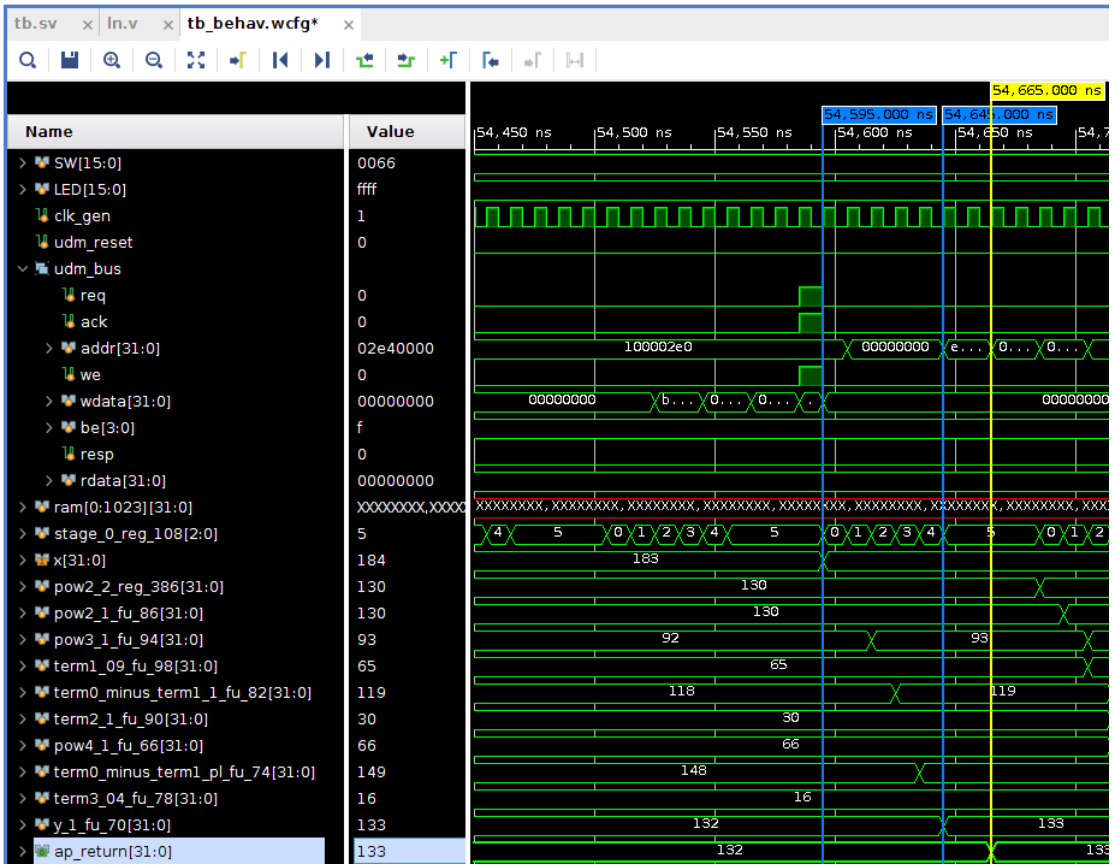


Figure 2: prove of latency.

3 Report on module characteristics:

3.1 Timing:

3.1.1 TNS: 0 ns

3.1.2 WNS: 5.302 ns

3.2 Module's performance:

3.2.1 Clock frequency: 10 ns (100 MHz)

3.2.2 Initiation Interval: 1 clock cycle; 10ns

3.2.3 Throughput: 1 op / cycle; 100Mop / second

3.2.4 Latency: 5 clock cycles; 50 ns

3.3 HW resources

3.3.1 LUTs: 203

3.3.2 FFs(registers): 253

The timing closure is successful.

s

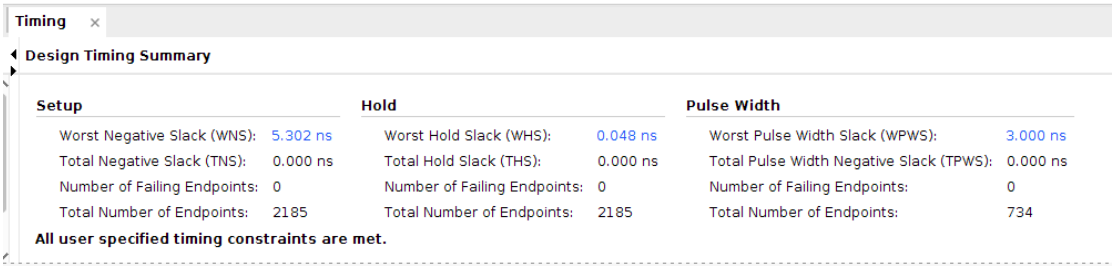
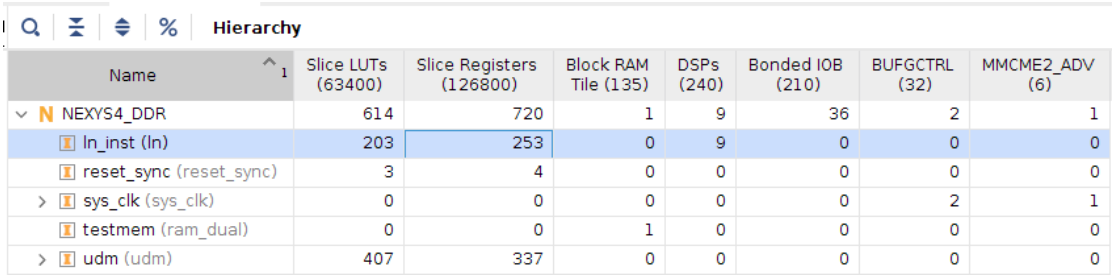


Figure 3: WNS and TNS



Name	Slice LUTs (63400)	Slice Registers (126800)	Block RAM Tile (135)	DSPs (240)	Bonded IOB (210)	BUFGCTRL (32)	MMCME2_ADV (6)
NEXYS4_DDR	614	720	1	9	36	2	1
In_inst (In)	203	253	0	9	0	0	0
reset_sync (reset_sync)	3	4	0	0	0	0	0
sys_clk (sys_clk)	0	0	0	0	0	2	1
testmem (ram_dual)	0	0	1	0	0	0	0
udm (udm)	407	337	0	0	0	0	0

Figure 4: LUTs and FFs