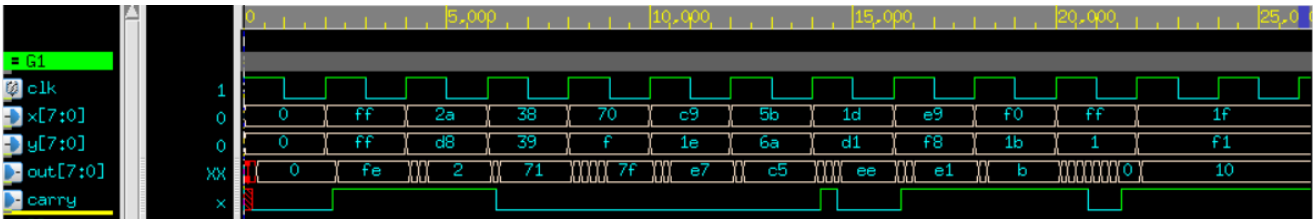
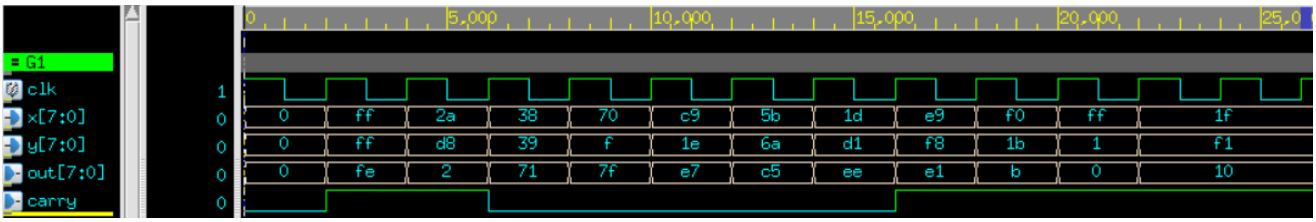
b07901152 張力元

I. 8-bit Carry Ripple Adder

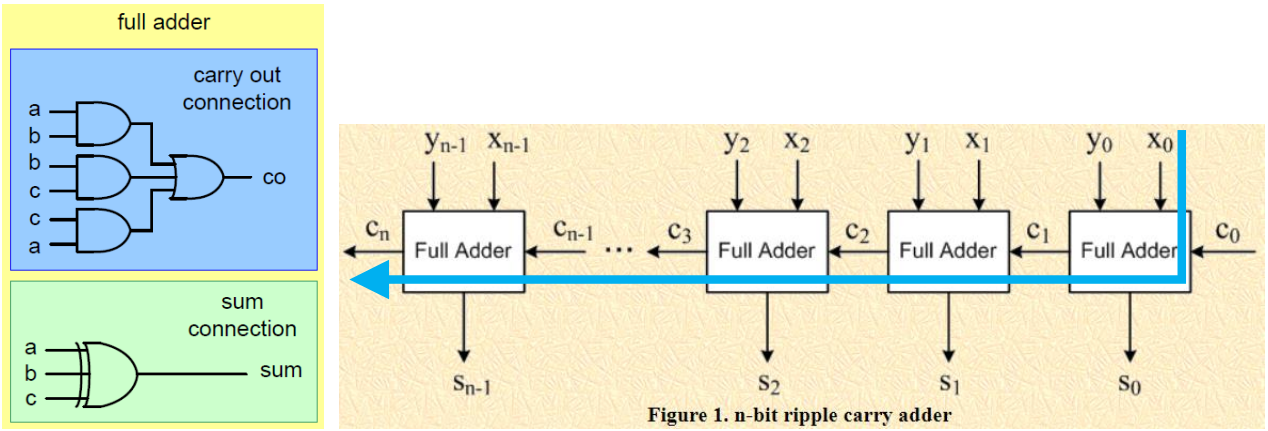
1. Waveform result of the gate-level design (clock cycle = 20 ns)



2. Waveform result of the RTL design (clock cycle = 20 ns)



3. Critical path analysis



The blue line above is the critical path of the 8-bit carry ripple adder.

Assume all logic gates have the same propagation delay of 1 ns

$$\text{propagation delay } t_p = 8 \times t_{\text{and-or delay}} = 16 \text{ ns}$$

Verify the timing by decreasing the clock cycle:

Clock cycle = 16.1 ns

```
==========================================

======  Congratulation! You Pass!  =======

==========================================

Simulation complete via $finish(1) at time 209300 PS + 1
```

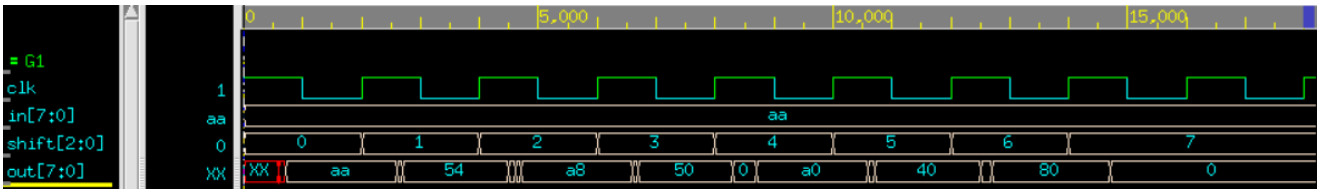
Clock cycle = 16.0 ns

```
An ERROR occurs at no.          11 pattern: {carry out, output} 000 != answer 100.

==============================

There are           1 errors.
Simulation complete via $finish(1) at time 208 NS + 1
```

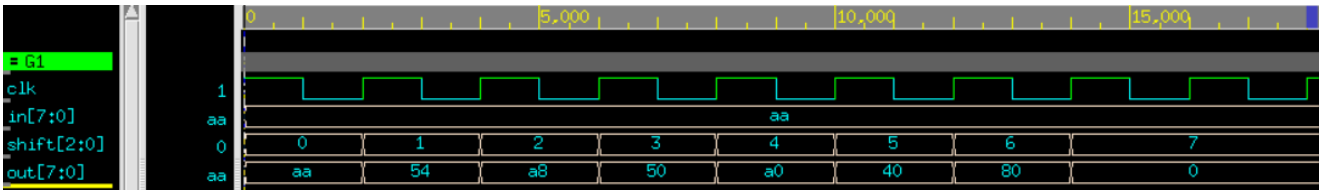
b07901152 張力元

II. 8-bit Barrel-shifter

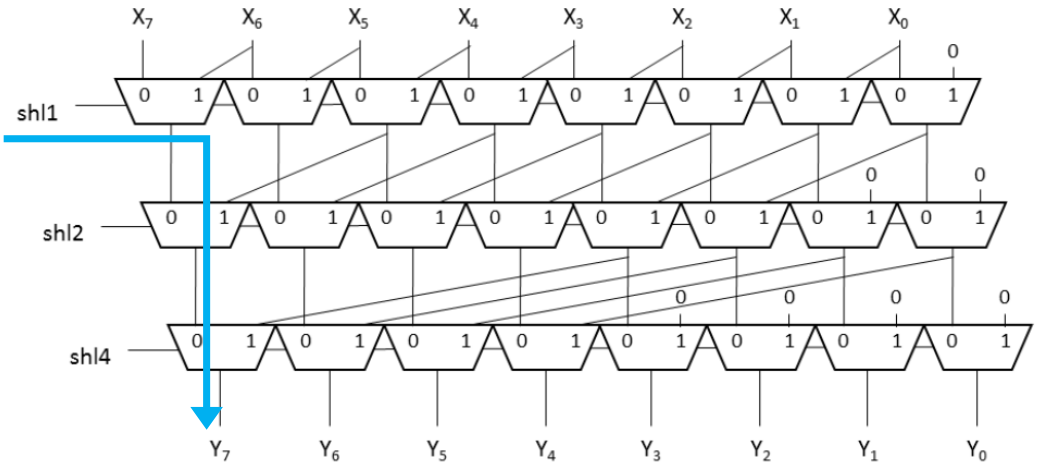
1. Waveform result of the gate-level design (clock cycle = 20 ns)



2. Waveform result of the RTL design (clock cycle = 20 ns)



3. Critical path analysis



The blue line above is the critical path of the 8-bit barrel-shifter.

Assume all logic gates and inverters have the same propagation delay of 1 ns

$$\text{propagation delay } t_p = t_{\text{inv-and-or delay}} + 2 \times t_{\text{and-or delay}} = 7 \text{ ns}$$

Verify the timing by decreasing the clock cycle:

Clock cycle = 7.0 ns

```
==========================================

======  Congratulation! You Pass!  =======

==========================================

Simulation complete via $finish(1) at time 63 NS + 1
```

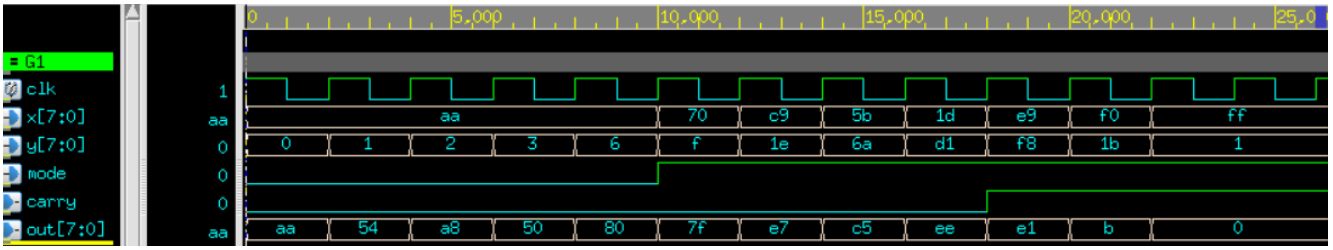
Clock cycle = 6.9 ns

```
==============================

There are           8 errors.
==============================

Simulation complete via $finish(1) at time 62100 PS + 1
```

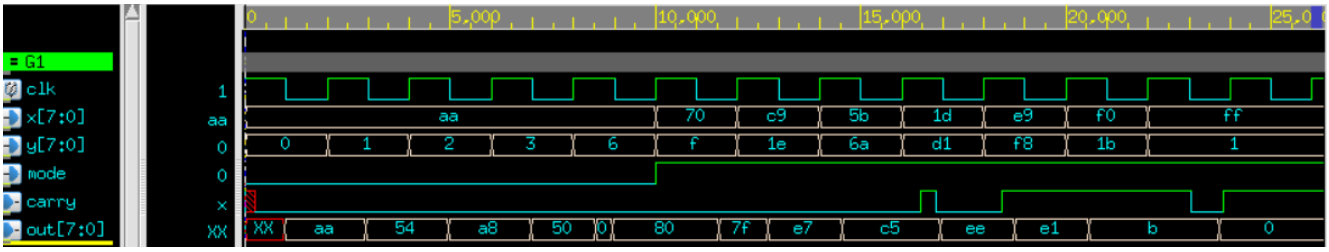
b07901152 張力元

III. Adder-Shifter Unit

1. Waveform result of the RTL design (clock cycle = 20 ns)



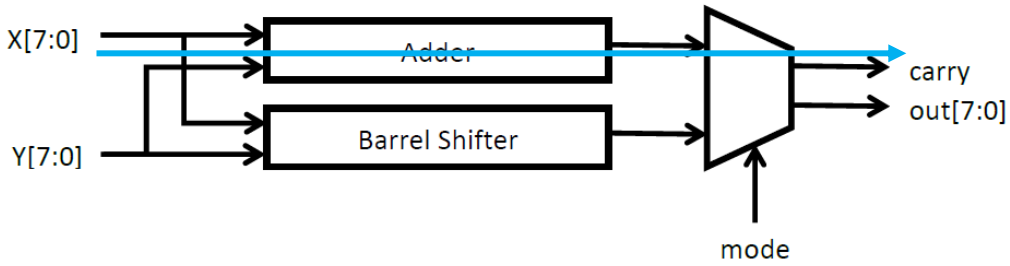
2. Waveform result of the gate-level design (clock cycle = 20 ns)



3. Critical path analysis

propagation delay of adder = 16 ns

propagation delay of barrel-shifter = 7 ns



The blue line above is the critical path of the adder-shifter unit.

Assume that the propagation delay of the mode multiplexer is equal to 2.5 ns

$$\text{propagation delay } t_p = t_{\text{adder delay}} + t_{\text{mode\_mux delay}} = 18.5 \text{ ns}$$

Verify the timing by decreasing the clock cycle:

Clock cycle = 18.5 ns

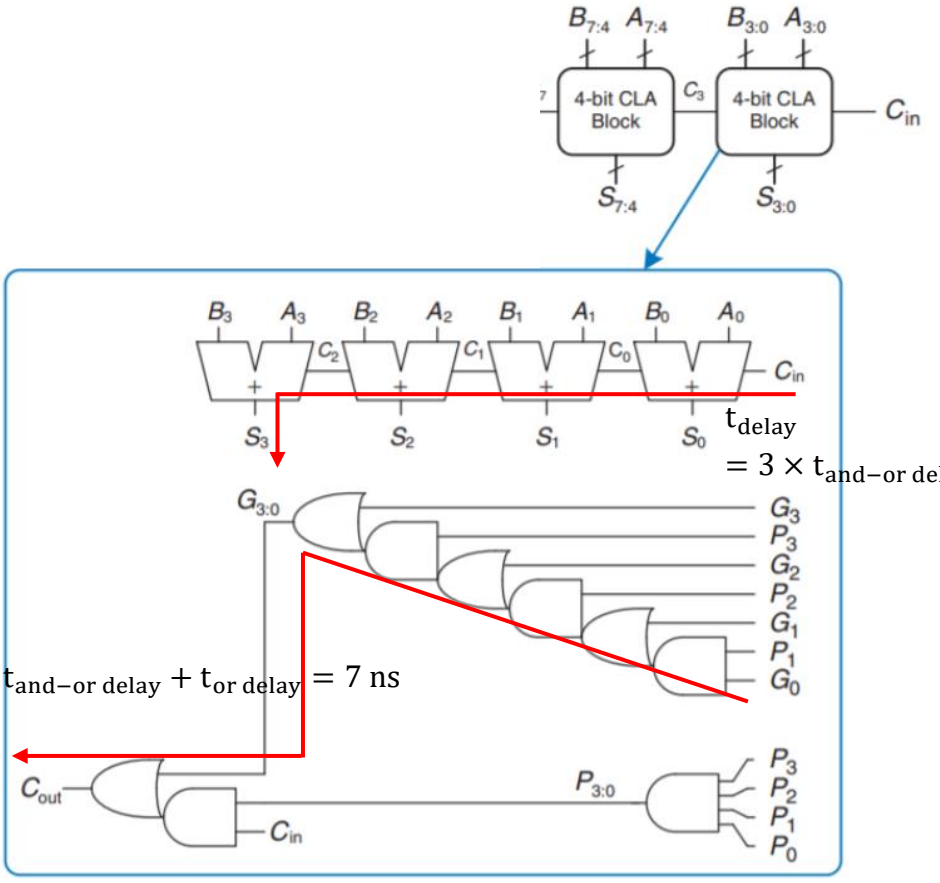
```
============================================

======  Congratulation! You Pass!  =======

============================================

Simulation complete via $finish(1) at time 240500 PS + 1
```

Clock cycle = 18.4 ns

```
============================

There are           1 errors.
============================

Simulation complete via $finish(1) at time 239200 PS + 1
```

4. Optimize the gate-level design

The critical path is on the adder followed by the mode multiplexer, so we can use carry-lookahead adder instead of carry-ripple adder to decrease the propagation delay of the adder-shifter unit, and thus reduce the minimum clock cycle.



$t_{delay} = 3 \times t_{and-or\ delay} + t_{xor\ delay} = 7\ ns$

$t_{delay} = 3 \times t_{and-or\ delay} + t_{or\ delay} = 7\ ns$

The red lines above are both the critical path of a 4-bit carry-lookahead adder.

$$t_{adder\ delay} = 2 \times 7 = 14\ ns$$

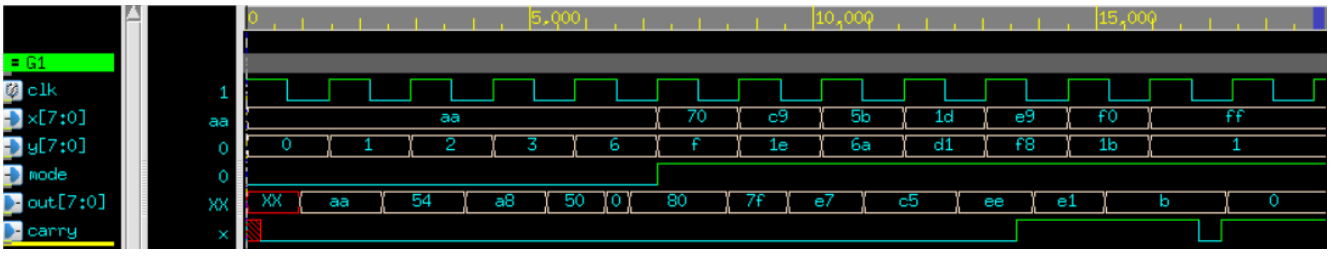
If we decompose the 8-bit adder into four 2-bit carry-lookahead adders, then the propagation delay of the adder can be reduced further.

$$t_{adder\ delay} = 4 \times 3 = 12\ ns$$

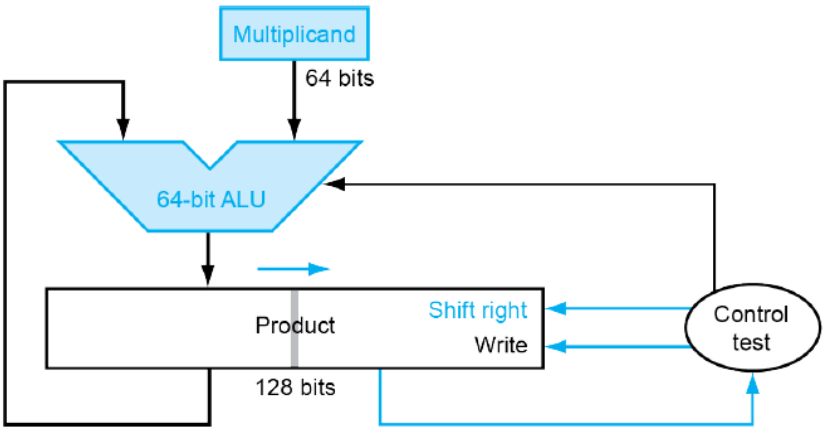
Assume that the propagation delay of the mode multiplexer is equal to 2.5 ns

$$\text{propagation delay } t_p = t_{adder\ delay} + t_{mode\_mux\ delay} = 14.5\ ns$$

Waveform result when the clock cycle = 14.5 ns:



5. Calculate unsigned multiplication with the adder-shifter unit



Step	
0	Initialize *Product* = {64'b0, *Multiplier*}
1	If the rightmost bit of *Product* is 0, then no operation is performed. If the rightmost bit of *Product* is 1, then use the adder-shifter unit to perform addition.
2	Use the adder-shifter unit to shift the *Product* right 1 bit.
3	Loop over the step 1 and step 2 until all the digits of the *Multiplier* are shifted outside the *Product* register.