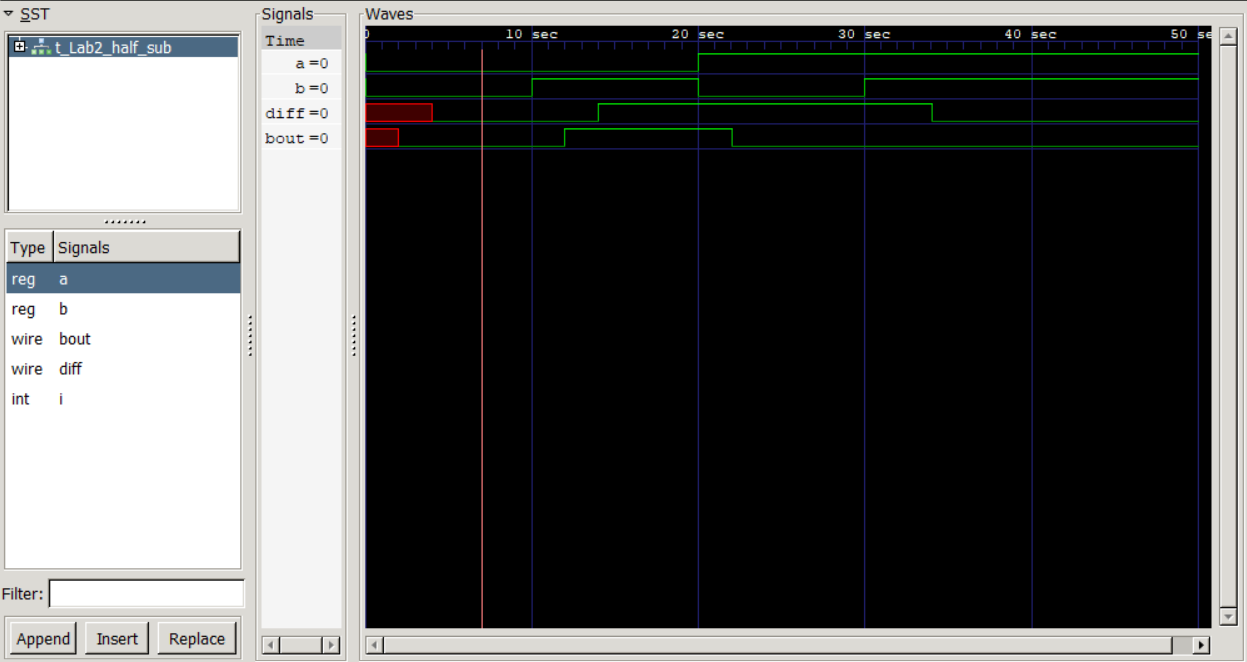
**(1) half subtractor**

一張含有 文字 的圖片

自動產生的描述

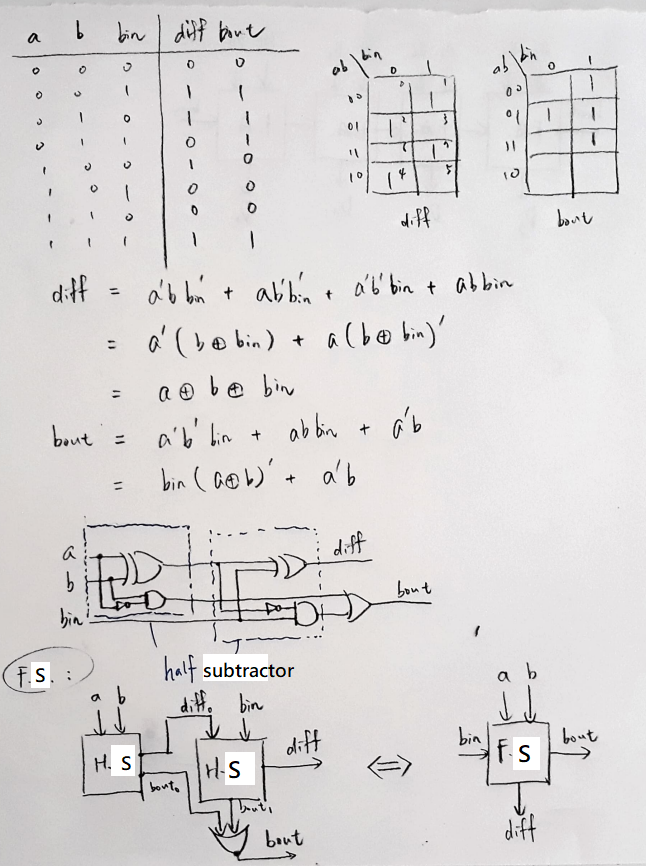
(truth table, Boolean expression, circuit diagram)



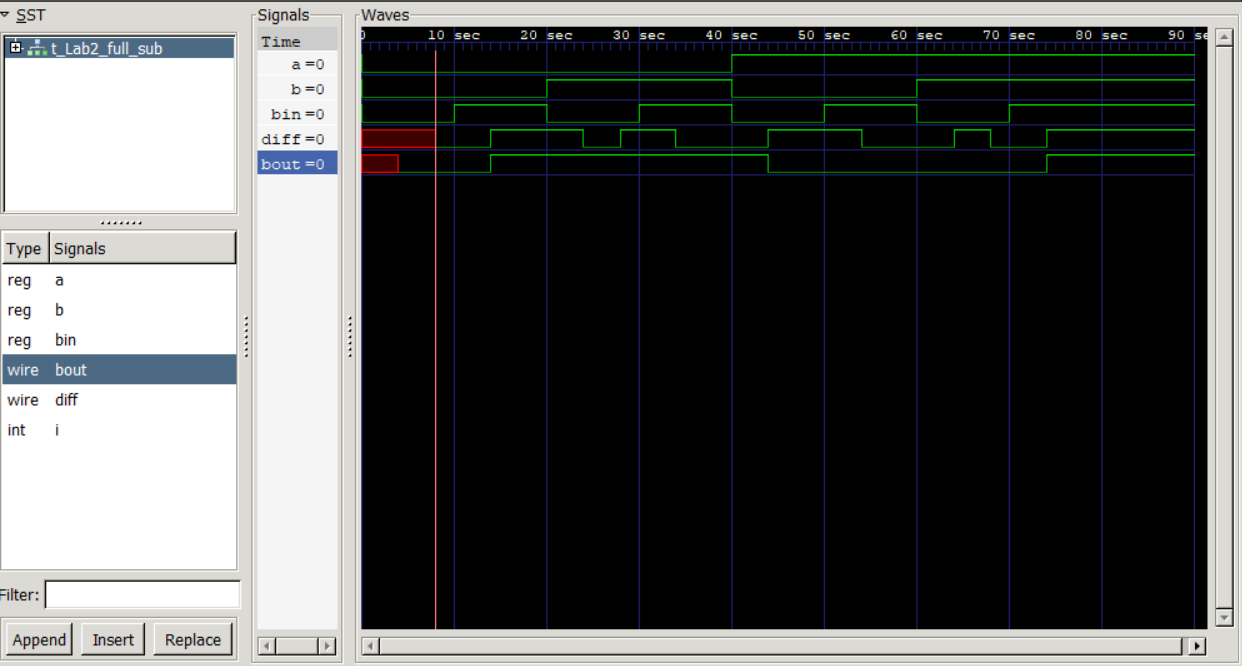
(waveform of half subtractor)

The delay of *diff* results from a XOR gate, which is 4 time units, and that of *bout* results from an AND gate, which is 2 time units; the delay shown in the waveform of *diff* and *bout* are identical to the estimations. Hence, the waveform is correct for it also complies with the truth table. The propagation delay of a half subtractor is *max(delay\_diff, delay\_bout) = 4* time units.

**(2) full subtractor**



(truth table, Boolean expression, circuit diagram, block diagram)

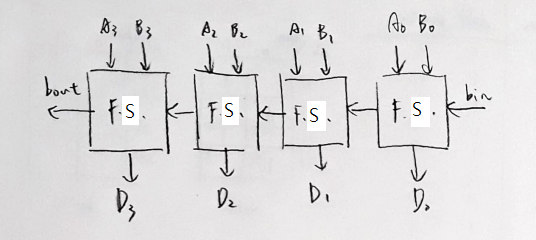


(waveform of full subtractor)

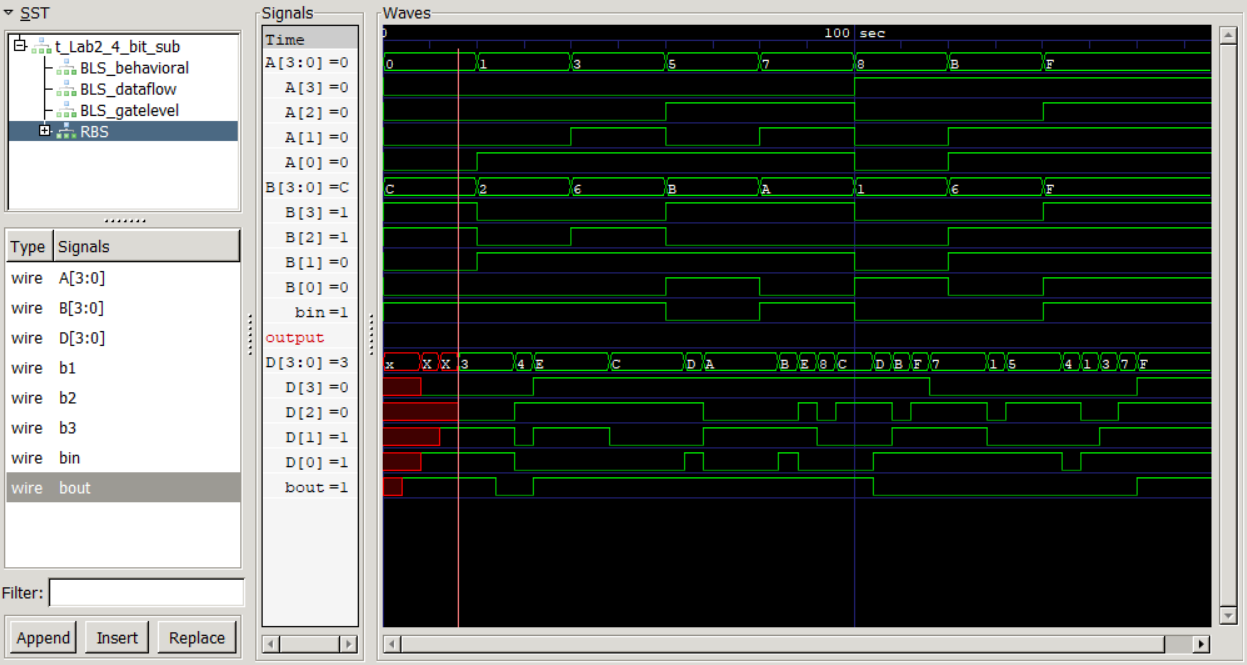
The longest path to generate *diff* is: XOR -> XOR, which takes 8 time units, and that to generate *bout* is XOR -> AND -> OR, which takes 8 time units as well. Hence, the propagation delay of a full subtractor is *max(delay\_diff, delay\_bout) = 8* time units.

However, since **x (unknown)** cannot be determined whether its value is 0 or 1, the output of **x AND/OR/XOR {any input}** is unknown as well. That’s why the red section (delay) in the waveform of *bout* only occupies time unit **0 to 4**, not conforming to the estimation (0 to 8). Nonetheless, if we only focus on the output signals at time *8+10n* (propagation time of *diff* and *bout* are 8 time units, and new inputs come every 10 time units; *n* is a non-negative integer), the waveform is then correct.

**(3) RBS**



(block diagram)

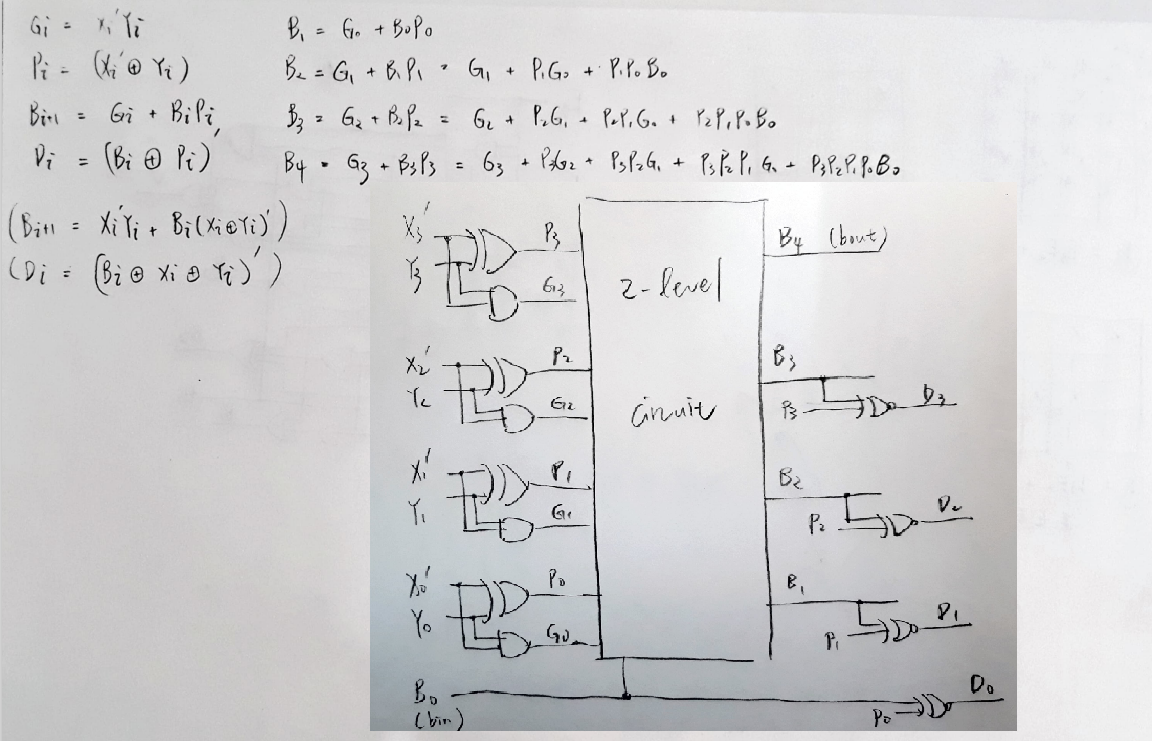


(waveform of RBS)

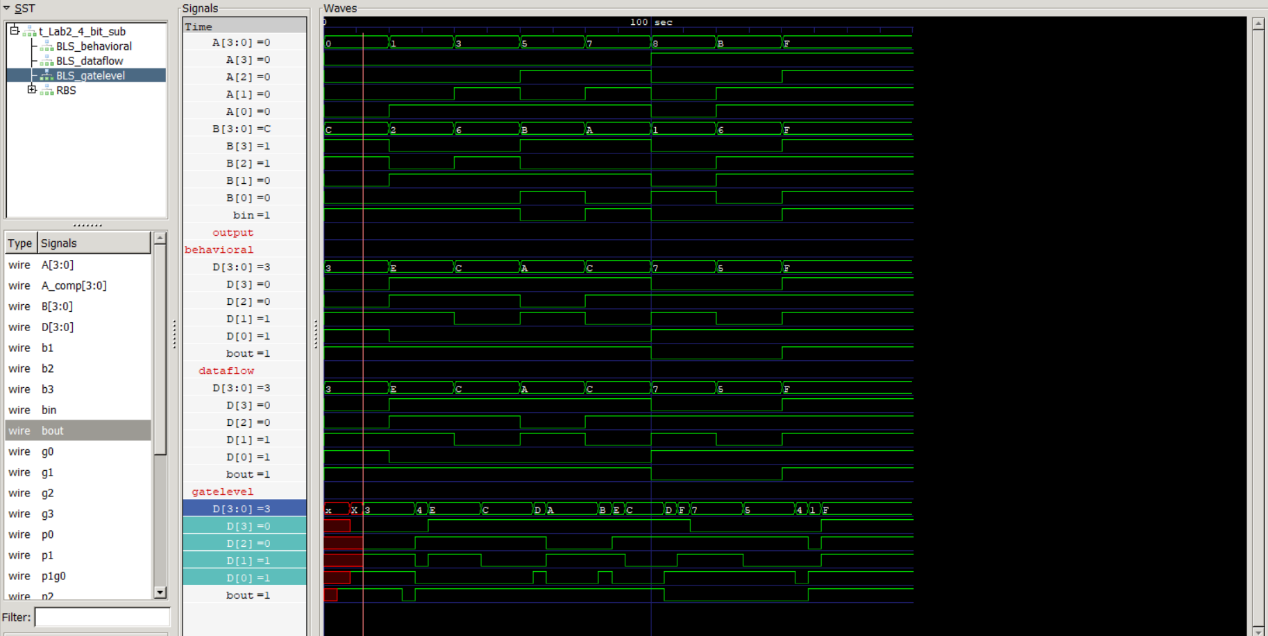
The delay of *D[i]*(0 <= i <= 3) is 8 time units, as *diff* is in a full subtractor; the delay of *bout* is: *4+2\*2\*4 = 20* time units (XOR -> AND -> OR -> AND -> OR ->…). Hence, the propagation delay of a 4-bit RBS is *max(delay\_D[i], delay\_bout) = 20* time units.

The problem that a full subtractor encounters is a problem of a RBS as well. As a result, valid output signals only appears at time *20+20n*, for this time new inputs come every 20 time units.

**(4) BLS**



(Boolean expression, circuit diagram)



(waveform of BLS: gatelevel, dataflow, behavioral)

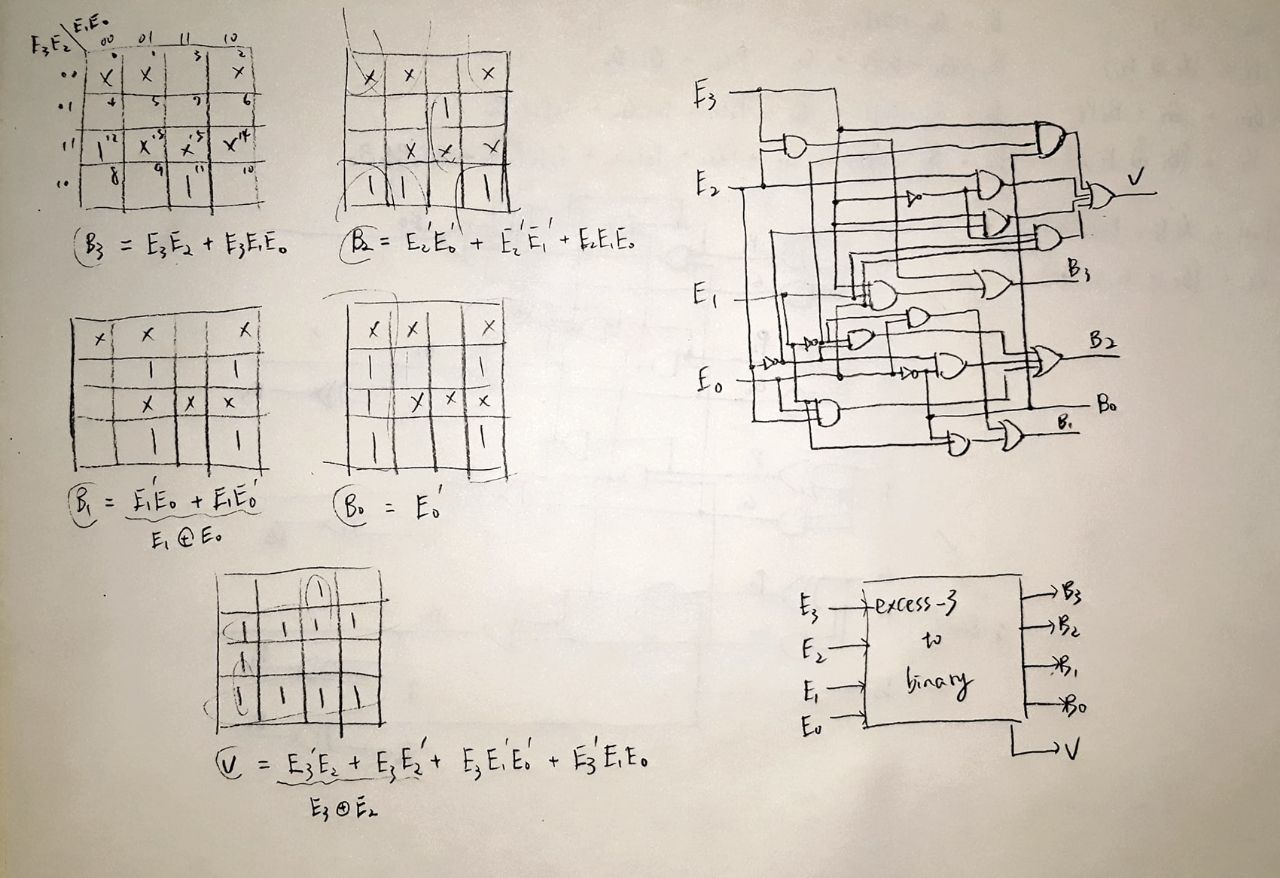
For gate-level modeling, the longest path to generate *bout* is: XOR -> AND -> OR, which takes 8 time units; the longest path to generate *D[i]* is: XOR -> AND -> OR -> XNOR, which takes 12 time units. Hence, the propagation delay of a 4-bit BLS is:

*max(delay\_D[i], delay\_bout) = 12;*

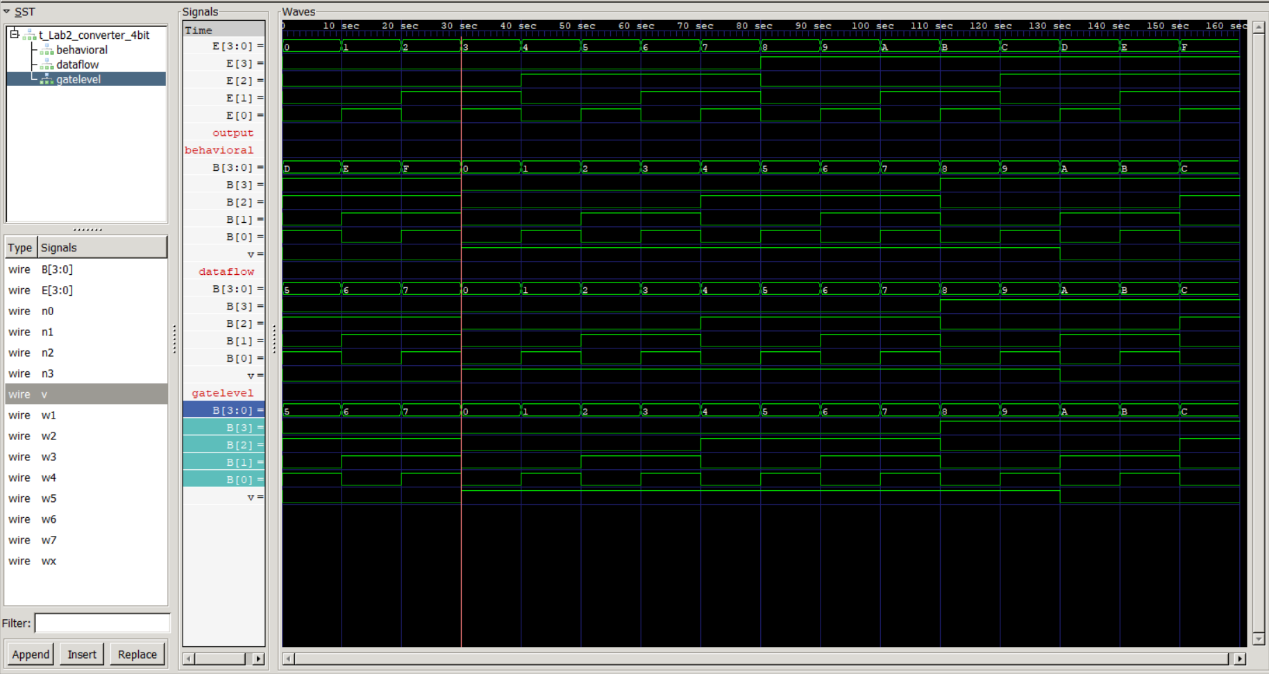
Gate-level modeling BLS has the mentioned delay problem as well. So, valid output signals only appears at time *12+20n* (new inputs come every 20 time units).

From the waveform, we can see that all three types of modeling generate the same output (gate-level modeling BLS generates the same output at time *12+20n)*, and they should all be correct. (the results are identical with the result of RBS)

**(5) Excess-3-to-binary Converter**



(K-map, Boolean expression, circuit diagram, block diagram)



(waveform of excess-3-to-binary converter)

As the waveform shows, as long as *v=1*, excess-3 inputs are correctly converted to binary numbers, for all three types of modeling.

**(6) Reflection & Difficulties**

The main difficulty is to handle delay. Whenever gate delay exists, the output signals become “bizarre” for not all wanted signals are generated simultaneously. Hence, it’s hard to interpret and evaluate the correctness of the output signals. I think to resolve this, the state of high impendence (Hi-z) or enable input (EN) should be introduced to these circuits.

By the way, when writing 4-bit BLS, I found that behavioral modeling is far more convenient than gate-level modeling; writing in behavioral modeling would save me a lot of time.