

2019 MSOC Assignment 1

Student: 傅子興

Student ID: B04901015

0. Environment Setup

0.1. Hello world result

```
louiefu@louiefu-AB350-Gaming-3:~/Desktop/MSOC2019/Lab1/0_hello$ ./ex0
SystemC 2.3.2-Accellera --- Mar 27 2019 20:56:48
Copyright (c) 1996-2017 by all Contributors,
ALL RIGHTS RESERVED
4 ns Hello world!
louiefu@louiefu-AB350-Gaming-3:~/Desktop/MSOC2019/Lab1/0_hello$
```

1. Counter

1.1. Counter result

```
louiefu@louiefu-AB350-Gaming-3:~/Desktop/MSOC2019/Lab1/1_c
SystemC 2.3.2-Accellera --- Mar 27 2019 20:56:48
Copyright (c) 1996-2017 by all Contributors,
ALL RIGHTS RESERVED
Updated to 1 at 30 ns
Updated to 2 at 40 ns
Updated to 3 at 50 ns
Updated to 4 at 60 ns
Updated to 5 at 70 ns
Updated to 6 at 80 ns
Updated to 7 at 90 ns
Updated to 8 at 100 ns
Updated to 9 at 110 ns
Updated to 10 at 120 ns
Updated to 11 at 130 ns
Updated to 12 at 140 ns
Updated to 13 at 150 ns
Updated to 14 at 160 ns
Updated to 15 at 170 ns
Updated to 16 at 180 ns
Updated to 17 at 190 ns
Updated to 18 at 200 ns
Updated to 19 at 210 ns
Updated to 20 at 220 ns
Updated to 19 at 230 ns
Updated to 18 at 240 ns
Updated to 17 at 250 ns
Updated to 16 at 260 ns
Updated to 15 at 270 ns
louiefu@louiefu-AB350-Gaming-3:~/Desktop/MSOC2019/Lab1/1_c
```

1.2. Discussion: The “else” branch is never executed, why?

Inside **Display.cpp**, we wrote *dont_initialize()* in the constructor, and during testing, the counter's value changed every cycle, so the condition is never triggered. Noted that this branch would be activated if we didn't write *dont_initialize()*, as the comparison started since Display object is constructed.

2. FIR

2.1. FIR result

```
SystemC 2.3.2-Accellera --- Mar 27 2019 20:56:48
Copyright (c) 1996-2017 by all Contributors,
ALL RIGHTS RESERVED
filter coefficients: [ 1 3 2 0 5 ]
[tap 0] 0 0
[tap 1] 0 0
[tap 2] 0 0
[tap 3] 0 0
[tap 4] 0 0
[tap 5] 0 0
[tap 6] 0 0
[tap 7] 0 0
[tap 8] 0 0
[tap 9] 0 0
[tap 10] 1 1
[tap 11] 0 3
[tap 12] 1 3
[tap 13] 0 3
[tap 14] 0 7
[tap 15] 0 0
[tap 16] 0 5
[tap 17] 0 0
[tap 18] 0 0
[tap 19] 0 0
louiefu@louiefu-AB350-Gaming-3:~/Desktop/MSOC2019/Lab1/2_f
```

3. FIFO

3.1. FIFO result

```
SystemC 2.3.2-Accellera --- Mar 27 2019 20:56:48
Copyright (c) 1996-2017 by all Contributors,
ALL RIGHTS RESERVED
Read: Nonblocking from 0 s to 0 s(H).
Read: Blocking from 100 ns to 1 us(H).
Read: Nonblocking from 1100 ns to 1100 ns(i).
Read: Nonblocking from 1200 ns to 1200 ns(,).
Read: Blocking from 1300 ns to 2 us(H).
Read: Nonblocking from 2100 ns to 2100 ns(i).
Read: Nonblocking from 2200 ns to 2200 ns(,).
Read: Nonblocking from 2300 ns to 2300 ns(_).
Read: Nonblocking from 2400 ns to 2400 ns(M).
Read: Nonblocking from 2500 ns to 2500 ns(o).
Read: Nonblocking from 2600 ns to 2600 ns(n).
Read: Nonblocking from 2700 ns to 2700 ns(i).
```

```
Fifo size is: 10
Average Fifo fill depth: 5.63768
Maximum Fifo fill depth: 9
Average transfer time per character: 143478 ps
Total characters transferred: 69
Total time: 9900 ns
```

3.2. Discussion: Changing **ALWAYS_BLOCK** flag to “true”

The result is shown as following. We can observe that except the “Blocking” and “nonblocking” word, the output word and time stamp is the same, i.e. the *in_->Empty()* branch is redundant. This is because “Read under empty” issue is already hold by *Read()* function; whenever reading an empty fifo, the fifo will wait until the *write_event_* is notified.

```
SystemC 2.3.2-Accellera --- Mar 27 2019 20:56:48
Copyright (c) 1996-2017 by all Contributors,
ALL RIGHTS RESERVED
Read: Blocking from 0 s to 0 s(H).
Read: Blocking from 100 ns to 1 us(H).
Read: Blocking from 1100 ns to 1100 ns(i).
Read: Blocking from 1200 ns to 1200 ns(,).
Read: Blocking from 1300 ns to 2 us(H).
Read: Blocking from 2100 ns to 2100 ns(i).
Read: Blocking from 2200 ns to 2200 ns(,).
Read: Blocking from 2300 ns to 2300 ns(_).
Read: Blocking from 2400 ns to 2400 ns(M).
Read: Blocking from 2500 ns to 2500 ns(o).
Read: Blocking from 2600 ns to 2600 ns(n).
Read: Blocking from 2700 ns to 2700 ns(i).

Fifo size is: 10
Average Fifo fill depth: 5.63768
Maximum Fifo fill depth: 9
Average transfer time per character: 143478 ps
Total characters transferred: 69
Total time: 9900 ns
```

4. Transactor

4.1. Transactor result

```
INFO: Simulating memory read/write (PCA)...
----- Original data in Memory:
[ 56 11 47 41 1 72 55 95 ]
[ 75 53 64 44 78 13 94 92 ]
[ 23 48 50 89 59 24 51 11 ]
[ 94 73 74 49 78 0 20 46 ]
[ 14 91 16 3 85 38 25 14 ]
[ 0 51 50 52 69 25 97 98 ]
[ 16 23 27 4 65 59 22 10 ]
[ 76 4 96 60 55 4 26 71 ]
INFO: iTransactor::read starting @ 0 s Addr (0,0)
INFO: iTransactor::read starting @ 10 ns Addr (1,0)
INFO: iTransactor::read starting @ 20 ns Addr (2,0)
INFO: iTransactor::write starting @ 1270 ns Addr (7,7)
INFO: Complete memory write.
----- New data in Memory:
[ 71 10 98 14 46 11 92 95 ]
[ 26 22 97 25 20 51 94 55 ]
[ 4 59 25 38 0 24 13 72 ]
[ 55 65 69 85 78 59 78 1 ]
[ 60 4 52 3 49 89 44 41 ]
[ 96 27 50 16 74 50 64 47 ]
[ 4 23 51 91 73 48 53 11 ]
[ 76 16 0 14 94 23 75 56 ]
```