An Experiment Design for Measuring Processing Sequence of FPGA Statements

Huanyin Zhou\textsuperscript{1,2}, Gaokui He\textsuperscript{2}, Mei Xu\textsuperscript{1}, Yanhui Xie\textsuperscript{1,2}, Jiewei Hu\textsuperscript{1}, Qilin Wen\textsuperscript{1} and Ziyong Lv\textsuperscript{1}

\textsuperscript{1}College of NBC Defense, PLA, Yangfang Town, Changping, Beijing, China
\textsuperscript{2}China Institute of Atomic Energy, Xinzhen Town, Fangshan, Beijing, China

*Corresponding author e-mail: eva_xiexie@126.com

Abstract. The mechanism of statement processing in FPGA is different from that in micro-processor. Statements are usually executed in parallel in FPGA, but they are done in series in processor. Moreover, “signal” and “variable” in FPGA are also executed differently. Because of conventional thought of series processing sequence in computer, mistakes are often made in analyzing processing sequence of FPGA statements. In this paper, some typical experiment cases are designed ingeniously to analyze the processing sequence of FPGA statements. Through experiment comparisons, some unexpected results are acquired. These experiment cases can demonstrate the sequence very clearly. A conclusion can be drawn that different processing sequences may lead to different results. So the processing sequence of FPGA statements is extremely important to signal processing. Therefore, more importance should be attached to the processing sequence of FPGA statements in FPGA digital system.

1. Introduction

Statements in FPGA(Field-Programmable Gate Array) are always executed in parallel, but they are usually done in series in micro-processor. The mechanism of statement processing in FPGA is quite different from that in micro-processor. So statements can be executed with higher speed in FPGA, and signal can be processed more quickly\textsuperscript{[1]}. Because different units of FPGA can run in parallel, editing sequence of many statements or processes is always not so important under some conditions. Different statements or processes can usually interchange their position. Some textbooks pointed out that statements are executed in serial when they are in a process, but they are run in parallel when they are out of process. “Signal” is executed in a different way from “variable” too. Processing sequence of FPGA statements is often a difficult problem to large numbers of FPGA beginners. If this question could not be solved, some mistakes would be made. To illuminate the processing sequence of FPGA statements, some very typical and convictive VHDL program cases are designed in this paper. Many kinds of process statements are designed and tested. Some results of these experiments are very unexpected. Through comparisons of these typical cases, the processing sequences of FPGA statements are demonstrated very clearly.

2. Experiments design for measuring processing sequence of VHDL statements

VHDL(Very-High-Speed-Integrated-Circuit Hardware Description Language) is an important describing language of FPGA statements. Some VHDL statements in FPGA are executed in parallel,
but the others are done in series. In the following paragraph, some typical experiments are designed to analyze the processing sequence of the statements. In these programs, x, y and z are defined as “variable”; xx, yy and zz are defined as “signal” respectively. The default initial values of these signals and variables are all zero in binary format.

2.1. Statements out of process and statements in different processes

Statements out of process and statements in different processes are executed in parallel. Different signals can be evaluated at the same time, but one signal can not be evaluated twice or more. If one signal is evaluated twice or more, the synthesize tool of FPGA developing system will give error messages. These kinds of instances have been explained very clearly in some textbook[2], so they will not be analyzed any more here.

2.2. Different signals evaluated in a process

In the following program, signals xx and yy are evaluated respectively.

```vhdl
process(cp, xx, yy)
begin
if falling_edge(cp) then
   xx <= xx + "0001"
   yy <= xx + "0001"
end if;
end process;
```

| Initial value | 1cp | 2cp | 3cp |
|---------------|-----|-----|-----|
| xx            | 0000| 0001| 0010| 0011|
| yy            | 0000| 0001| 0010| 0011|

If the statements are processed in series as they are done in processor, that is to say, the statements are executed one by one, yy would be more by 1 than xx after every clock pulse. If they are executed in parallel, yy is equal to xx. When this program is executed, and clock pulses are input from cp, xx and yy are shown in Table 1. Experiment shows that yy is equal to xx. The result proves that two statements are executed in parallel.

If the editing sequence of the statements is inversed, the program is shown as follows, and the result is shown in Table 2. It is just the same as Table 1. The result proves that two statements are executed in parallel once more.

```vhdl
if falling_edge(cp) then
   yy <= xx + "0001"
   xx <= xx + "0001"
end if;
```

Table 2. Statements sequence inversed and signals evaluated.

| Initial value | 1cp | 2cp | 3cp |
|---------------|-----|-----|-----|
| xx            | 0000| 0001| 0010| 0011|
| yy            | 0000| 0001| 0010| 0011|

2.3. One signal evaluated twice or more in a process

According to the preceding analysis, one signal can not be evaluated twice or more when it is in different processes or out of process. But in the following program, the signal yy can be evaluated twice or more in one process. Table 3 shows the result of the program.

```vhdl
if falling_edge(cp) then
   yy <= yy + "0001"
   yy <= yy + "0010"
end if;
```

Table 3. Signal evaluated twice in a process.

| Initial value | 1cp | 2cp | 3cp | 4cp |
|---------------|-----|-----|-----|-----|
| yy            | 0000| 0010| 0100| 0110|
If the sequence of the statements were executed one by one, the former statement is executed first, and then the next one goes on. After a clock pulse, yy will be “0011”. If they are executed in parallel, yy may be “0001” or “0010”. Experiment shows that yy is “0010” when a clock pulse is inputted. The result illuminates that two statements are executed in parallel, and only the lattermost statement is available. Signal is evaluated at the end of the process operating.

If we inverse the editing sequence of the statements, the result is shown in Table 4. The result testifies the before-mentioned conclusion further.

| Initial value | 1cp | 2cp | 3cp |
|---------------|-----|-----|-----|
| yy            | 0000| 0001| 0010| 0011|

Table 4. Statements sequence inversed and one signal evaluated twice.

2.4. Two different variables evaluated in a process
In the following program, variables x and y are evaluated respectively.

```cpp
if falling_edge(cp) then
    y := y + "0001";
    y := x + "0001";
end if;
```

If the statements are executed one by one, after a clock pulse, x will be “0001”, and y will be “0010”. If they are executed in parallel, the two statements are evaluated at the same time, both x and y will be “0001” after a clock pulse. The result is shown as Table.5.

| Initial value | 1cp | 2cp | 3cp |
|---------------|-----|-----|-----|
| x             | 0000| 0001| 0010| 0011|
| y             | 0000| 0001| 0010| 0011|

Table 5. Variables evaluated in series.

Experiment shows that x is “0001”, and y is “0010”. The result proves that two variables are evaluated in series. Variable x is evaluated at first, then y is evaluated. Table 5 is not same as Table 1. This experiment shows that evaluating of variable is different from that of signals.

When the editing sequence of the statements is inversed, the experiment result is shown in Table 6. Table 6 testifies that variables are evaluated in series too.

```cpp
if falling_edge(cp) then
    y := x + "0001";
    x := x + "0001";
end if;
```

| Initial value | 1cp | 2cp | 3cp |
|---------------|-----|-----|-----|
| x             | 0000| 0001| 0010| 0011|
| y             | 0000| 0001| 0010| 0011|

Table 6. Statements sequence inversed and variables evaluated in series.

2.5. One variable evaluated twice or more in a process
In the following program, the variable y is evaluated twice in the process. The result is shown in Table.7.

```cpp
if falling_edge(cp) then
    y := y + "0001";
    y := y + "0010";
end if;
```
Table 7. Result of variables y being evaluated twice.

|     | Initial | 1cp | 2cp | 3cp |
|-----|---------|-----|-----|-----|
| y   | 0000    | 0011| 0110| 1001|

Experiment shows that y is “0011” after one clock pulse. The result is also different from Table 4. The result proves that the variable is evaluated twice in series. The two statements are executed in turn. This program demonstrates the difference between variable and signal further.

2.6. Signal and variable are evaluated compositely in a process

In the following program, variable x and signal yy are evaluated respectively in a process. Table 8 shows the result of the program operating.

```plaintext
if falling_edge(cp) then
    x := x + "0001";
    yy := x + "0001";
end if;
```

Table 8. Variables x and signal yy are evaluated in a process.

|     | Initial | 1cp | 2cp | 3cp |
|-----|---------|-----|-----|-----|
| x   | 0000    | 0001| 0010| 0011|
| yy  | 0000    | 0010| 0011| 0100|

If the statements are executed one by one, after a pulse, x will be “0001”, and yy will be “0010”. If they are executed in parallel, both x and y will be “0001”. Experiment shows that, after a cp pulse, x is “0001”, and yy is “0010”. The result proves that the statements are executed in series.

If we inversed the editing sequence of the statements as following process, the experiment result is shown in Table 9. From this table, we can see that yy and x are evaluated in series. Further more, according to Table 9, in this program, the latter statement can not be executed before the former one.

```plaintext
if falling_edge(cp) then
    yy := x + "0001";
    x := x + "0001";
end if;
```

Table 9. Signal yy and variables x are evaluated in a process.

|     | Initial | 1cp | 2cp | 3cp |
|-----|---------|-----|-----|-----|
| x   | 0000    | 0001| 0010| 0011|
| yy  | 0000    | 0001| 0010| 0011|

If a signal xx and a variable y are evaluated as follows, the result is shown in Table 10. This result testifies the former conclusion once more.

```plaintext
if falling_edge(cp) then
    xx := xx + "0001";
    y := xx + "0001";
end if;
```

Table 10. Signal yy and variables x are evaluated in a process.

|     | Initial | 1cp | 2cp | 3cp |
|-----|---------|-----|-----|-----|
| x   | 0000    | 0001| 0010| 0011|
| yy  | 0000    | 0001| 0010| 0011|

3. Conclusions

According to the above experiment comparisons, processing sequence of statements in FPGA is illuminated very clearly. Some conclusions can be drawn. The statements in one process start operating in order, and then they operate in parallel. The signal evaluating needs response time, but variable evaluating has no delay. Variable can be evaluated immediately, but signal can not. Different signals are evaluated in parallel; so all signals are evaluated at the same time. If one signal is evaluated twice or more in a process at the same time, the last evaluating statement is executed, the former
evaluating is invalid. However, variables can be evaluated more than once, and they will be evaluated in order. That is similar to the operating in processor. From theses experiments, we can conclude that, different statements have different processing sequence, and different sequences will cause different results. The processing sequence has a close relationship with editing sequence of statements in some process. So we must attach more importance to the processing sequence of FPGA statements in FPGA digital system.

References
[1] Zhou Huanyin. Design of Mini DMCA Based on FPGA. Beijing: College of Chemical Defence, 2013
[2] Zhao Junchao. VHDL Course for IC Design. Beijing Hope Electronic Press, 2002