Embedded Systems (digital design)

Verilog and SystemVerilog

- basics on files -

Prof. Davide Zoni (davide.zoni@polimi.it)

2022

File Input/Output

SystemVerilog files

Verilog 2001 support C-like operations on files

- read/write formatted lines
- read/write chars/lines
- dump/load memory content (2D-arrays)

Main scenarios with IO files:

- Feed input vector to the DUT
- Dump output vectors from DUT
- Debug purposes, if combined with hierarchical names

Subset of effective I/O primitives for files

- `fid = $fopen("filename", "mode");`
  - mode ={w,r,a, wb,rb,ab, r+ w+..}
- `$fclose(fid);`
- `$fscanf(), $fgets(), $fgetc(), $fstrobe()`
- `$fread(), $fwrite(), $fdisplay(), $fmonitor()`
- There are more … (check-out the specification manual)

The critical aspect: when is a File IO system task processed (within the time step)?

Write on files

- `$fwrite, $fdisplay, $fmonitor` -

File IO: $fwrite, $fdisplay, $fmonitor, and $fstrobe

```
1  `timescale 1ns / 1ps
2  module tb_fmonitor_fdisplay_fwrite;
3      reg [1:0] q;
4      integer fid_fmonitor, fid_fdisplay, fid_fwrite, fid_fstrobe;
5      initial begin
6          fid_fmonitor  = $fopen("outfile_fmonitor.txt");
7          fid_fdisplay  = $fopen("outfile_fdisplay.txt");
8          fid_fwrite    = $fopen("outfile_fwrite.txt");
9          fid_fstrobe   = $fopen("outfile_fstrobe.txt");
10         $fmonitor(fid_fmonitor, "@%0d, q=%b\n", $time, q);
11         q='0;
12         #1 q=2'b10;
13            $fdisplay(fid_fdisplay, "@%0d, q=%b\n", $time, q);
14            $fwrite(fid_fwrite, "@%0d, q=%b\n", $time, q);
15            $fstrobe(fid_fstrobe,"@%0d, q=%b\n", $time, q);
16         #2 q=2'b01; q=2'b10;
17            $fdisplay(fid_fdisplay, "@%0d, q=%b\n", $time, q);
18            $fwrite(fid_fwrite, "@%0d, q=%b\n", $time, q);
19            $fstrobe(fid_fstrobe,"@%0d, q=%b\n", $time, q);
20         #1 q=2'b01;
21         #1 $fclose(fid_fmonitor);
22            $fclose(fid_fdisplay);
23            $fclose(fid_fwrite);
24            $fclose(fid_fstrobe);
25         #1000
26             $finish;
27     end
28 endmodule
```

Note

- Line 4 – file handles for **`$fmonitor, $fdisplay, $fwrite, $fstrobe`**
- Lines 11,12,16,20 – generate values for **q**
- Line 14 – close all files
- Lines 13-15, 17-19 – monitor/write to files

Questions

- **(1)** What is written on files?
- **(2)** Does the output changes if we change the assignment to **q**

```
#1 q=2'b11; q=2'b00;
#1 q=2'b11; q<=2'b00;
```

Hint: Check the time the system tasks are executed during the time step

Output: $fwrite, $fdisplay, $fmonitor

```
1  `timescale 1ns / 1ps
2  module tb_fmonitor_fdisplay_fwrite;
3      reg [1:0] q;
4      integer fid_fmonitor, fid_fdisplay, fid_fwrite, fid_fstrobe;
5      initial begin
6          fid_fmonitor  = $fopen("outfile_fmonitor.txt");
7          fid_fdisplay  = $fopen("outfile_fdisplay.txt");
8          fid_fwrite    = $fopen("outfile_fwrite.txt");
9          fid_fstrobe   = $fopen("outfile_fstrobe.txt");
10         $fmonitor(fid_fmonitor, "@%0d, q=%b\n", $time, q);
11         q='0;
12         #1 q=2'b10;
13            $fdisplay(fid_fdisplay, "@%0d, q=%b\n", $time, q);
14            $fwrite(fid_fwrite, "@%0d, q=%b\n", $time, q);
15            $fstrobe(fid_fstrobe,"@%0d, q=%b\n", $time, q);
16         #2 q=2'b01; q=2'b10;
17            $fdisplay(fid_fdisplay, "@%0d, q=%b\n", $time, q);
18            $fwrite(fid_fwrite, "@%0d, q=%b\n", $time, q);
19            $fstrobe(fid_fstrobe,"@%0d, q=%b\n", $time, q);
20         #1 q=2'b01;
21         #1 $fclose(fid_fmonitor);
22            $fclose(fid_fdisplay);
23            $fclose(fid_fwrite);
24            $fclose(fid_fstrobe);
25         #1000
26             $finish;
27     end
28 endmodule
```

Note

- Line 4 – file handles for **`$fmonitor, $fdisplay, $fwrite, $fstrobe`**
- **$fmonitor** samples all **q** changes postponed region
- **$fdisplay, $fwrite** active region, when called
- **$fstrobe** post-poned region, when called

Output

$fwrite

```
1  @1, q=10
2  @3, q=10
```

$fdisplay

```
1  @1, q=10
2
3  @3, q=10
```

$fmonitor

```
1  @0, q=00
2
3  @1, q=10
4
5  @4, q=01
```

$fstrobe

```
1  @1, q=10
2
3  @3, q=10
```

Hint: Check the time the system tasks are executed during the time step

File IO: use NBAs

```
1  `timescale 1ns / 1ps
2  module tb_fmonitor_fdisplay_fwrite;
3      reg [1:0] q;
4      integer fid_fmonitor, fid_fdisplay, fid_fwrite, fid_fstrobe;
5      initial begin
6          fid_fmonitor  = $fopen("outfile_fmonitor.txt");
7          fid_fdisplay  = $fopen("outfile_fdisplay.txt");
8          fid_fwrite    = $fopen("outfile_fwrite.txt");
9          fid_fstrobe   = $fopen("outfile_fstrobe.txt");
10         $fmonitor(fid_fmonitor, "@%0d, q=%b\n", $time, q);
11         q='0;
12         #1 q=2'b10;
13            $fdisplay(fid_fdisplay, "@%0d, q=%b\n", $time, q);
14            $fwrite(fid_fwrite, "@%0d, q=%b\n", $time, q);
15            $fstrobe(fid_fstrobe,"@%0d, q=%b\n", $time, q);
16         #2 q=2'b01; q<=2'b10;
17            $fdisplay(fid_fdisplay, "@%0d, q=%b\n", $time, q);
18            $fwrite(fid_fwrite, "@%0d, q=%b\n", $time, q);
19            $fstrobe(fid_fstrobe,"@%0d, q=%b\n", $time, q);
20         #1 q=2'b01;
21         #1 $fclose(fid_fmonitor);
22            $fclose(fid_fdisplay);
23            $fclose(fid_fwrite);
24            $fclose(fid_fstrobe);
25         #1000
26             $finish;
27     end
28 endmodule
```

Note

- Line 4 – file handles for **`$fmonitor, $fdisplay, $fwrite, $fstrobe`**
- Lines 11,12,16,20 – generate values for **`q`**
- Line 16 – assign q using both blocking and NB assignments
- Lines 13-15, 17-19 – monitor/write to files

What is the output written to files?

Hint: Check the time the system tasks are executed during the time step

Output: using NBAs

```
1  `timescale 1ns / 1ps
2  module tb_fmonitor_fdisplay_fwrite;
3      reg [1:0] q;
4      integer fid_fmonitor, fid_fdisplay, fid_fwrite, fid_fstrobe;
5      initial begin
6          fid_fmonitor  = $fopen("outfile_fmonitor.txt");
7          fid_fdisplay  = $fopen("outfile_fdisplay.txt");
8          fid_fwrite    = $fopen("outfile_fwrite.txt");
9          fid_fstrobe   = $fopen("outfile_fstrobe.txt");
10         $fmonitor(fid_fmonitor, "@%0d, q=%b\n", $time, q);
11         q='0;
12         #1 q=2'b10;
13            $fdisplay(fid_fdisplay, "@%0d, q=%b\n", $time, q);
14            $fwrite(fid_fwrite, "@%0d, q=%b\n", $time, q);
15            $fstrobe(fid_fstrobe,"@%0d, q=%b\n", $time, q);
16         #2 q=2'b01; q<=2'b10;
17            $fdisplay(fid_fdisplay, "@%0d, q=%b\n", $time, q);
18            $fwrite(fid_fwrite, "@%0d, q=%b\n", $time, q);
19            $fstrobe(fid_fstrobe,"@%0d, q=%b\n", $time, q);
20         #1 q=2'b01;
21         #1 $fclose(fid_fmonitor);
22            $fclose(fid_fdisplay);
23            $fclose(fid_fwrite);
24            $fclose(fid_fstrobe);
25         #1000
26             $finish;
27     end
28 endmodule
```

Note

- Line 4 – file handles for **`$fmonitor, $fdisplay, $fwrite, $fstrobe`**
- **$fmonitor** samples all **q** changes postponed region
- **$fdisplay, $fwrite** active region, when called
  - Line 16 q=2'b01 is written
- **$fstrobe** post-poned region, when called
  - Line 16 q=2'b10 is written to file

Output

$fwrite

```
1  @1, q=10
2  @3, q=01
```

$fdisplay

```
1  @1, q=10
2
3  @3, q=01
```

$fmonitor

```
1  @0, q=00
2
3  @1, q=10
4
5  @4, q=01
```

$fstrobe

```
1  @1, q=10
2
3  @3, q=10
```

Hint: Check the time the system tasks are executed during the time step

Read from files to drive the DUT

– `$fscanf` –

File IO: $fscanf, $display and $strobe

```
1  timeunit 1ns; timeprecision 100ps;
2  module tb_fscanf_display_strobe;
3      logic [7:0] q_temp, q;
4      integer fin; integer n;
5
6      initial begin
7        fin    = $fopen("ifile.txt","r");
8        q      <='0;
9        q_temp<='0;
10       repeat(5) begin
11         #1 n=$fscanf(fin,"%2h\n",q_temp);
12         q   <=  q_temp;
13         $display("@%0t - $display - q_temp=%h q=%h (n=%0d)",
14                          $time, q_temp, q, n);
15         $strobe( "@%0t - $strobe  - q_temp=%h q=%h (n=%0d)",
16                          $time, q_temp, q, n);
17       end
18       #1     $fclose(fin);
19       #1000 $finish;
20     end
21 endmodule
```

Note

- **$fscanf** – read everything as a *sequence of char*
- **$display** – is processed in the active region
- **$strobe** – is processed in the postponed region
- Use the following description to generate driving values for the DUT (q is fed to the DUT)

```
n=$fscanf(fin,"%2h\n",q_temp);

q<=q_temp;
```

Output

```
1  @1000 - $display - q_temp=ab q=00 (n=1)
2  @1000 - $strobe  - q_temp=ab q=ab (n=1)
3  @2000 - $display - q_temp=cd q=ab (n=1)
4  @2000 - $strobe  - q_temp=cd q=cd (n=1)
5  @3000 - $display - q_temp=11 q=cd (n=1)
6  @3000 - $strobe  - q_temp=11 q=11 (n=1)
7  @4000 - $display - q_temp=ff q=11 (n=1)
8  @4000 - $strobe  - q_temp=ff q=ff (n=1)
9  @5000 - $display - q_temp=f1 q=ff (n=1)
10 @5000 - $strobe  - q_temp=f1 q=f1 (n=1)
```

ifile.txt

```
AB
CD
11
FF
F1
```

File IO: $fread (read stream of data)

```
1  timeunit 1ns; timeprecision 100ps;
2
3  module tb_fscanf_display_strobe;
4  reg [7:0] q_temp, q;
5  // open write and close files.
6  integer fin; integer n;
7  initial begin
8    fin   = $fopen("ifile.txt","r");
9    q='0;  q_temp='0;
10   repeat(5) begin
11     #1 n=$fread(q_temp,fin); q<=q_temp;
12     $display("@%0t - $display - q_temp=%h q=%h (n=%0d)",
13                          $time,q_temp,q,n);
14     $strobe( "@%0t - $strobe  - q_temp=%h q=%h (n=%0d)",
15                          $time,q_temp,q,n);
16   end
17   $fclose(fin);
18   #1000
19   $finish;
20 end
21 endmodule
```

- **$fread –** reads unstructured data from file using a binary format.
- **In contrast, $fscanf** – reads data from file following a fixed format and considers everything as a char

Output

```
1  @1000 - $display - q_temp=41 q=00 (n=1)
2  @1000 - $strobe  - q_temp=41 q=41 (n=1)
3  @2000 - $display - q_temp=42 q=41 (n=1)
4  @2000 - $strobe  - q_temp=42 q=42 (n=1)
5  @3000 - $display - q_temp=0a q=42 (n=1)
6  @3000 - $strobe  - q_temp=0a q=0a (n=1)
7  @4000 - $display - q_temp=43 q=0a (n=1)
8  @4000 - $strobe  - q_temp=43 q=43 (n=1)
9  @5000 - $display - q_temp=44 q=43 (n=1)
10 @5000 - $strobe  - q_temp=44 q=44 (n=1)
```

ifile.txt

```
AB
CD
11
FF
F1
```