

به نام خدا

پروژه درس طراحی سیستم های دیجیتال

علیرضا سلیمیان 400105036

علیرضا مصلی نژاد 400108944

امیرعلی سلیمی 400109384

مقدمه:

در ابتدا توضیحاتی درباره خود پروژه می دهیم. $LeNet - 5$ یک شبکه عصبی است که عکس ورودی می گیرد و کاراکترهایی که در آن عکس وجود دارد را نمایش می دهد. این شبکه عصبی از بخش های مختلفی تشکیل شده است که در ادامه به آنها می پردازیم.

شبکه ی عصبی $LeNet-5$ یکی از اولین شبکه های عصبی کانولوشنی ($Convolutional Neural Network$) بود که برای تشخیص تصاویر طراحی شده است. این شبکه در سال 1998 توسط یوشوا بنجیو ($Yann Lecun$) و همکارانش برای تشخیص ارقام دست نوشته شده در چک های بانکی طراحی شد.

شبکه $LeNet - 5$ از لایه های کانولوشنی، لایه های ادغام ($Pooling$) و لایه های کاملاً متصل ($Fully Connected$) تشکیل شده است. این شبکه با ورودی یک تصویر با ابعاد 32 در 32 پیکسل و با سه کانال رنگی (RGB) کار می کند و با استفاده از 7 لایه ی مختلف، تصویر ورودی را به یکی از 10 کلاس مختلف (اعداد 0 تا 9) تشخیص می دهد.

$LeNet - 5$ از تکنیک هایی مانند پیچش، ادغام، تابع فعال سازی سیگموئید ($Sigmoid$) و تابع خطی استفاده می کند. این شبکه در زمان طراحی خود به دلیل داشتن عملکرد خوب در تشخیص ارقام دست نوشته، مورد توجه بسیاری از پژوهشگران و محققان در زمینه ی یادگیری عمیق قرار گرفته است.

شبکه عصبی $LeNet-5$ شامل 7 لایه مختلف است که به طور خلاصه مراحل آن به صورت زیر است:

- 1 - لایه ورودی: در این لایه، تصویر ورودی با ابعاد 32 در 32 پیکسل و سه کانال رنگی (RGB) قرار می گیرد.
- 2 - لایه کانولوشنی اول: در این لایه، 6 فیلتر کانولوشن با ابعاد 5 در 5 و با کرنل یکسان برای تصویر ورودی اعمال می شود. سپس تابع فعال سازی سیگموئید برای خروجی این لایه استفاده می شود.
- 3 - لایه ادغام اول: در این لایه، پس از اعمال فیلترهای کانولوشن، تصویر حاصل به ابعاد 14 در 14 پیکسل با کاهش ابعاد و افزایش سرعت پردازش تبدیل می شود.
- 4 - لایه کانولوشنی دوم: در این لایه، 16 فیلتر کانولوشن با ابعاد 5 در 5 و با کرنل یکسان برای تصویر حاصل از لایه ادغام اول اعمال می شود و سپس تابع فعال سازی سیگموئید برای خروجی این لایه استفاده می شود.
- 5 - لایه ادغام دوم: در این لایه، پس از اعمال فیلترهای کانولوشن، تصویر حاصل به ابعاد 5 در 5 پیکسل با کاهش ابعاد و افزایش سرعت پردازش تبدیل می شود.
- 6 - لایه کاملاً متصل: در این لایه، وزن هایی که باید با داده های قبلی ضرب شوند، با استفاده از 120 نورون محاسبه می شوند. سپس تابع فعال سازی سیگموئید برای خروجی این لایه استفاده می شود.
- 7 - لایه کاملاً متصل خروجی: در این لایه، با استفاده از 10 نورون، احتمال تعلق هر تصویر ورودی به یکی از 10 کلاس مختلف (اعداد 0 تا 9) محاسبه می شود.

شبکه $LeNet - 5$ برای تشخیص تصاویر با ابعاد بزرگ تر مناسب نیست، زیرا این شبکه برای تشخیص ارقام دست نوشته بر روی تصاویر با ابعاد 32 در 32 پیکسل طراحی شده است و دارای لایه های ادغامی کوچک است که باعث کاهش ابعاد تصویر می شود.

با این حال، می توان از این شبکه برای پردازش تصاویر با ابعاد کوچکتر استفاده کرد یا با اعمال تغییراتی در شبکه، آن را برای تشخیص اشیاء در تصاویر با ابعاد بزرگتر نیز قابل استفاده کرد. به طور کلی، استفاده از شبکه های عصبی عمیق تر و پیچیده تر، مانند $ResNet$ و $Inception$ برای پردازش تصاویر با ابعاد بزرگتر توصیه می شود.

در این آزمایش قرار است که با استفاده از *testbench* داده شده، شبیه سازی را انجام دهیم. البته این کار آنچنان ساده نبود و در طی مراحل به مشکلات مختلفی برخورد کردیم. که در ادامه به توضیح آنها می پردازیم.

بخش اول:

در این بخش لازم است که شبیه سازی را انجام دهیم. مشکلی که در همین ابتدا به آن برخورد کردیم، این بود که ماژول هایی که نوشته شده است با پسوند *.v* ذخیره شده است و فایل هایی که با *vivado* باز میشوند، اگر پسوند *.v* داشته باشند به عنوان فایل *Verilog* در نظر گرفته میشوند. اما این ماژول ها به زبان *systemVerilog* نوشته شده اند، و کسی که کد ها را نوشته است، موقع کامپایل تمام *.v* ها را تبدیل به *sv* می کند.

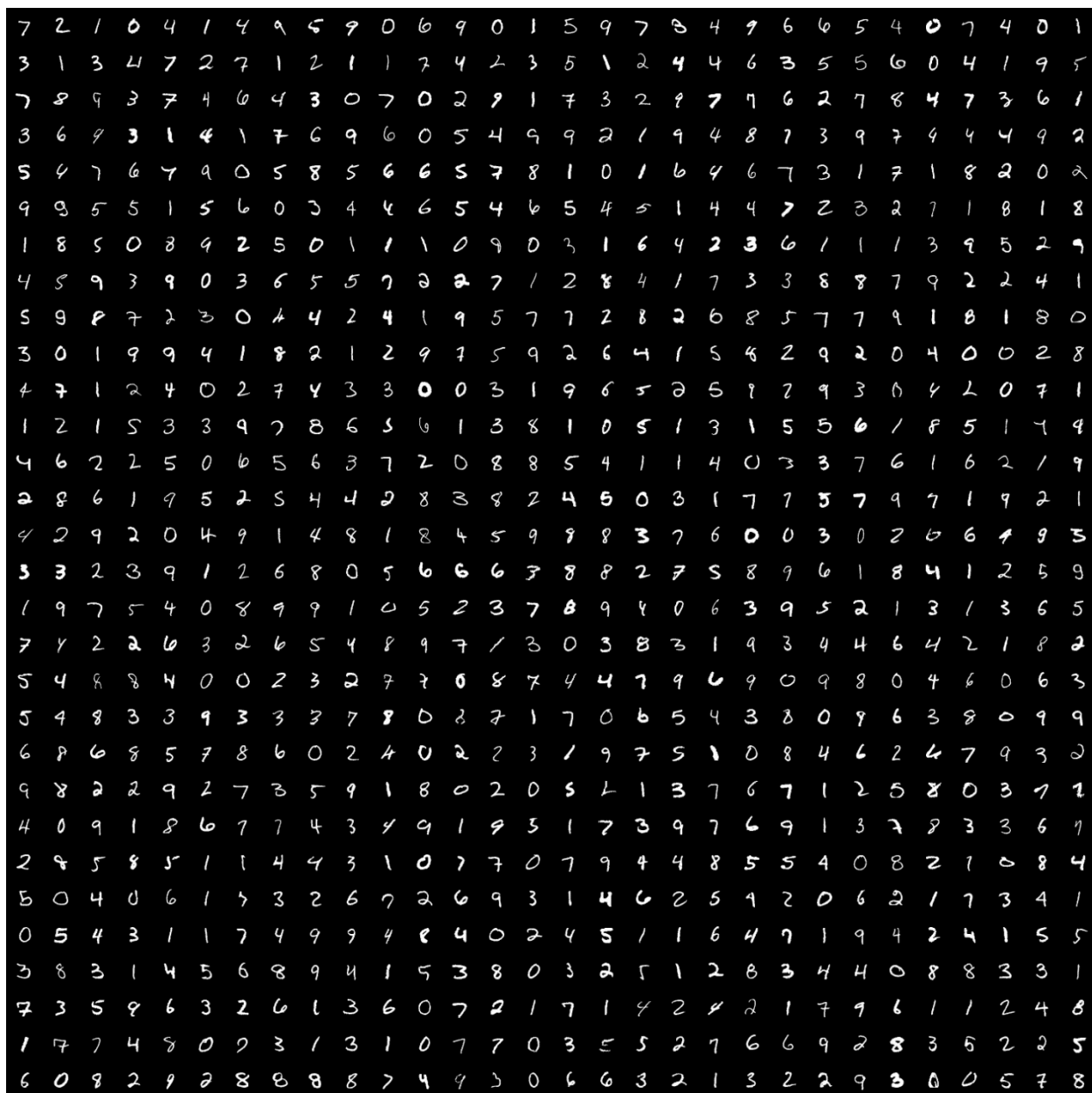
دومین چالشی که به آن برخورد کردیم این بود که نرم افزار *vivado*، فایل هایی که در پوشه های مختلف هستند را شناسایی نمی کند و ما مجبور شدیم که تمام فایل ها را یکی کنیم.

برای آدرس دهی فایل *test1000f* هم به مشکل خوردیم که ناشی از شناسایی نکردن پوشه ها توسط *vivado* بود.

در ادامه موفق شدیم که تست انجام دهیم و خروجی زیر را مشاهده کردیم:

T	186618==process a frame	0, digit	7 =====
T	371238==process a frame	1, digit	2 =====
T	555858==process a frame	2, digit	1 =====
T	740478==process a frame	3, digit	0 =====
T	925098==process a frame	4, digit	4 =====
T	1109718==process a frame	5, digit	1 =====
T	1294338==process a frame	6, digit	4 =====
T	1478958==process a frame	7, digit	9 =====
T	1663578==process a frame	8, digit	5 =====
T	1848198==process a frame	9, digit	9 =====
T	2032818==process a frame	10, digit	0 =====
T	2217438==process a frame	11, digit	6 =====
T	2402058==process a frame	12, digit	9 =====
T	2586678==process a frame	13, digit	0 =====
T	2771298==process a frame	14, digit	1 =====
T	2955918==process a frame	15, digit	5 =====
T	3140538==process a frame	16, digit	9 =====
T	3325158==process a frame	17, digit	7 =====
T	3509778==process a frame	18, digit	3 =====
T	3694398==process a frame	19, digit	4 =====

که همه جواب ها درست است.



قسمت دوم (بررسی کدبیس و واحدهای ضرب کننده و جمع کننده):

فایل testbench

برای شروع از testbench اصلی پروژه شروع می کنیم. یعنی فایل lenet_tb.v

```
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  cnn_open
  model
  src
  top
  capture.v
  cnn.v
  draw_rectangle.v
  global.v
  lenet.v
  lenet_roms.v
  misc.v
  osd.v
  tensorflow
  testbench
  conv1_tb.v
  lenet_tb.v
  vcs
  vivado
  README.md

70      .aa      (aa_src_rom),
71      .cena    (cena_src_rom),
72      .qa      (qa_src_rom)
73      );
74
75  lenet lenet
76      .clk      (clk),
77      .rstn     (rstn),
78      .go       (go),
79      .cena_src (cena_src_rom),
80      .aa_src   (aa_src_rom),
81      .qa_src   (qa_src_rom),
82      .digit    (digit),
83      .ready    (ready)
84      );
85
86  reg [31:0] digit_cnt;
87  always @(CLK_RST_EDGE)
88      if (!RST) digit_cnt <= 0;
89      else if (ready) begin
90          $display("T%d==process a frame %d, digit %d =====", $time, digit_cnt, digit);
91          digit_cnt <= digit_cnt + 1;
92      end
93
94
95
```

مهمترین قسمت این فایل ماژول lenet می باشد که شبکه ی عصبی تشخیص دهنده ی اعداد درون این ماژول قرار دارد.

```
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  cnn_open
  model
  src
  top
  capture.v
  cnn.v
  draw_rectangle.v
  global.v
  lenet.v
  lenet_roms.v
  misc.v
  osd.v
  tensorflow
  testbench
  conv1_tb.v
  lenet_tb.v
  vcs
  vivado
  README.md

70      .aa      (aa_src_rom),
71      .cena    (cena_src_rom),
72      .qa      (qa_src_rom)
73      );
74
75  lenet lenet
76      .clk      (clk),
77      .rstn     (rstn),
78      .go       (go),
79      .cena_src (cena_src_rom),
80      .aa_src   (aa_src_rom),
81      .qa_src   (qa_src_rom),
82      .digit    (digit),
83      .ready    (ready)
84      );
85
86  reg [31:0] digit_cnt;
87  always @(CLK_RST_EDGE)
88      if (!RST) digit_cnt <= 0;
89      else if (ready) begin
90          $display("T%d==process a frame %d, digit %d =====", $time, digit_cnt, digit);
91          digit_cnt <= digit_cnt + 1;
92      end
93
94
95
```

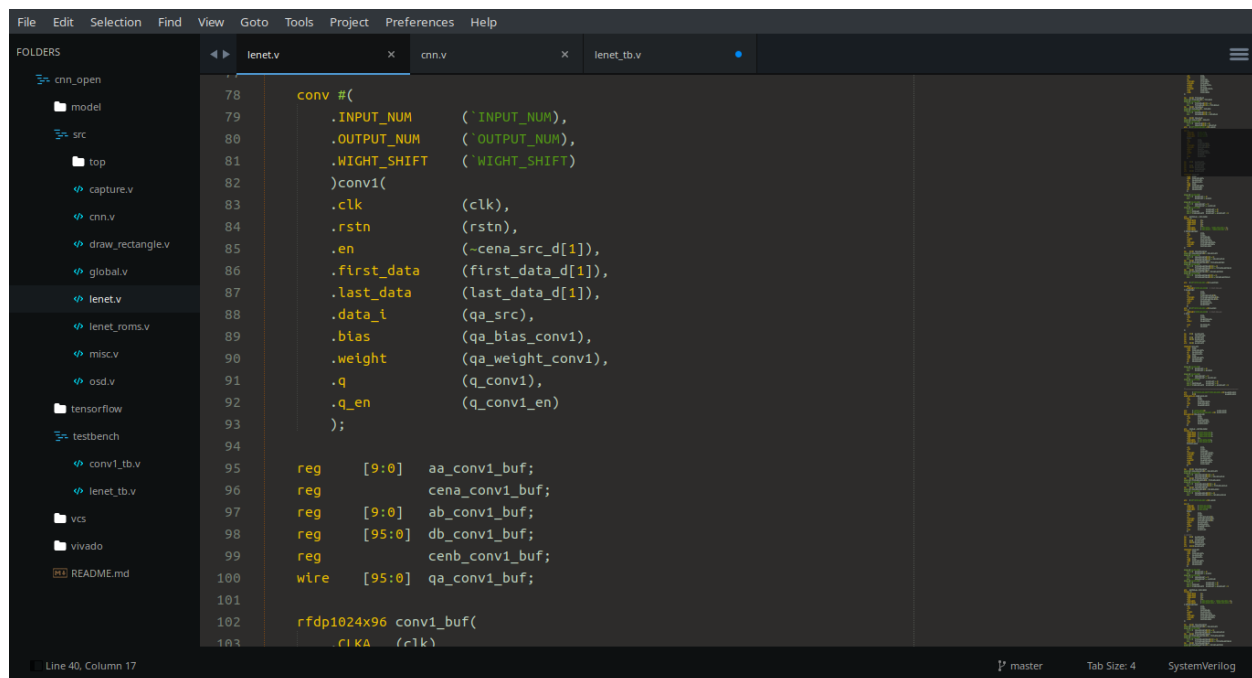
در این قسمت تست ها از روی فایل خوانده شده و به ترتیب وارد شبکه ی عصبی می شوند؛ هر تصویر که خوانده می شود روی mem.src_rom لود می شود. سپس سیگنال go فعال شده و ماژول شروع به تشخیص می کند. پس از بدست آمدن خروجی تصویری بعدی برای تست وارد می شود.

```
169 logic [ MAX_PATH*8-1:0 ] sequence_name = "./test_1000f.yuv";
170 go      <= 0;
171 @cb;
172 @cb;
173
174 fd = $fopen(sequence_name, "rb");
175 if (fd == 0) begin
176     errno = $ferror(fd, errinfo);
177     $display("sensor Failed to open file %0s for read.", sequence_name);
178     $display("errno: %0d", errno);
179     $display("reason: %0s", errinfo);
180     $finish();
181 end
182
183 for(int f = 0; f<nframe; f++ ) begin
184     @cb;
185     @cb;
186     for(int i = 0; i< 32*32; i++ ) begin
187         $root.tb.src_rom.mem[i] <= $fgetc(fd);
188     end
189     @cb;
190     @cb;
191     go      <= 1;
192     @cb;
193     go      <= 0;
194     @cb.ready;
```

ماژول lenet

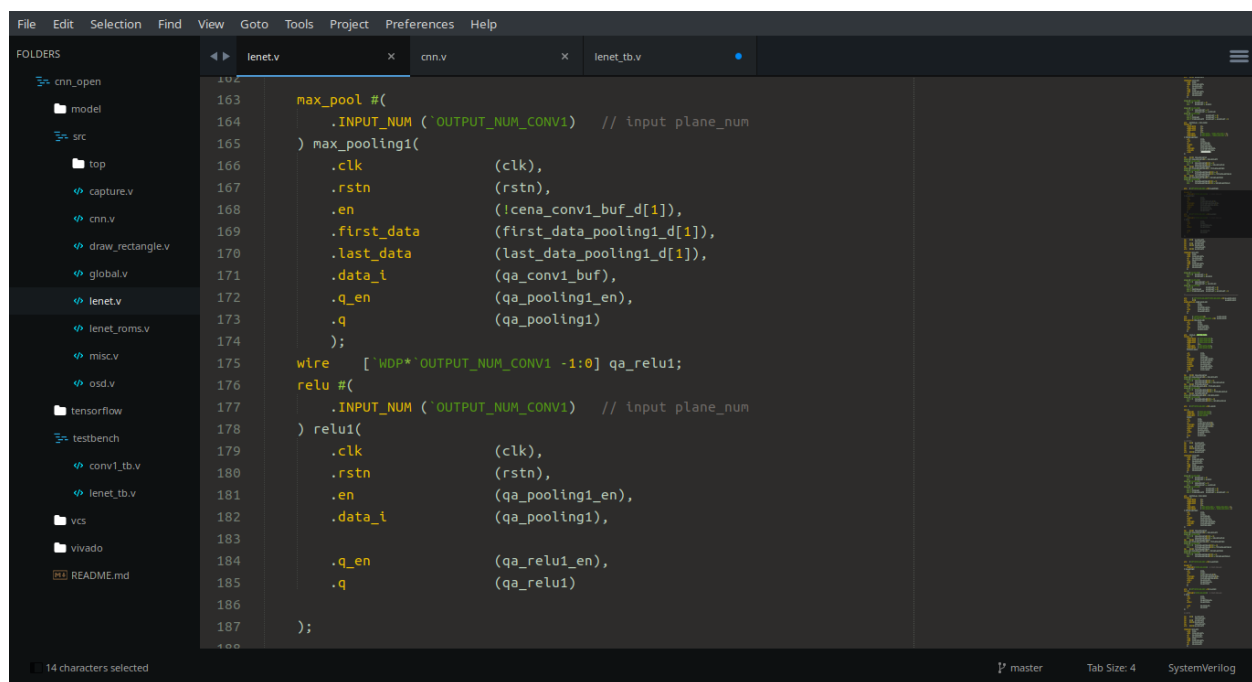
```
1 // Copyright (c) 2018 LulinChen, All Rights Reserved
2 // AUTHOR : LulinChen
3 // AUTHOR'S EMAIL : lulinch@aliyun.com
4 // Release history
5 // VERSION Date AUTHOR DESCRIPTION
6 `include "global.v"
7
8 module lenet(
9     input          clk,
10    input          rstn,
11    input          go,
12    output          cena_src,
13    output [9:0]    aa_src,
14    input  [ WD:0]  qa_src,
15    output reg [3:0] digit,
16    output          ready
17 );
18
19 wire [9:0] aa_weight_conv1;
20 wire [ WDP* OUTPUT_NUM_CONV1-1:0] qa_weight_conv1;
21
22 wire [ W_OUPUT_BATCH:0] aa_bias_conv1;
23 wire [ WDP_BIAS* OUTPUT_NUM_CONV1 -1:0] qa_bias_conv1;
24
25 wieght_conv1_rom wieght_conv1_rom(
26     .clk          (clk),
```

وارد ماژول lenet می شویم. qa_weight_conv1 و qa_bias_conv1 به ترتیب وزن های اولین لایه ی شبکه هستند که در ادامه می آید:



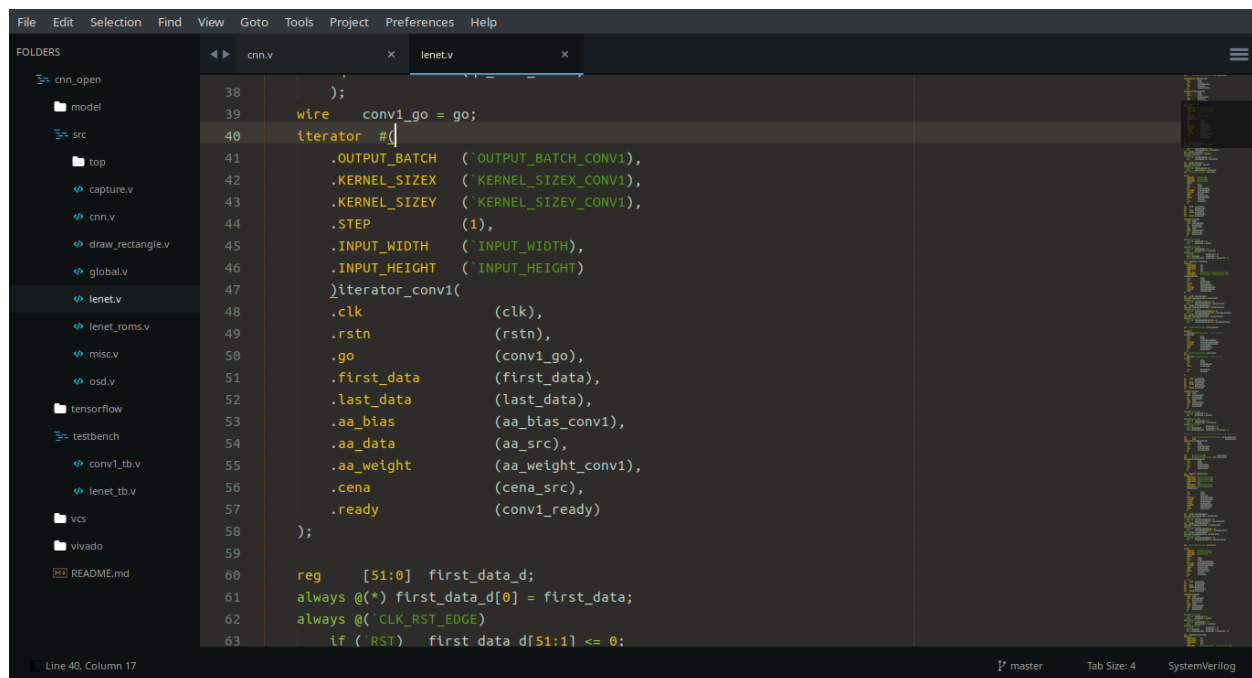
```
78     conv #(
79         .INPUT_NUM    ( `INPUT_NUM ),
80         .OUTPUT_NUM   ( `OUTPUT_NUM ),
81         .WIGHT_SHIFT  ( `WIGHT_SHIFT)
82     ) conv1(
83         .clk            (clk),
84         .rstn           (rstn),
85         .en             (~cena_src_d[1]),
86         .first_data     (first_data_d[1]),
87         .last_data      (last_data_d[1]),
88         .data_i         (qa_src),
89         .bias           (qa_bias_conv1),
90         .weight         (qa_weight_conv1),
91         .q              (q_conv1),
92         .q_en           (q_conv1_en)
93     );
94
95     reg    [9:0] aa_conv1_buf;
96     reg    [9:0] cena_conv1_buf;
97     reg    [9:0] ab_conv1_buf;
98     reg   [95:0] db_conv1_buf;
99     reg   [95:0] cenb_conv1_buf;
100    wire   [95:0] qa_conv1_buf;
101
102    rfdp1024x96 conv1_buf(
103        .CLKA      (clk)
```

اولین لایه‌ی convolutional network در این قسمت ایجاد شده.



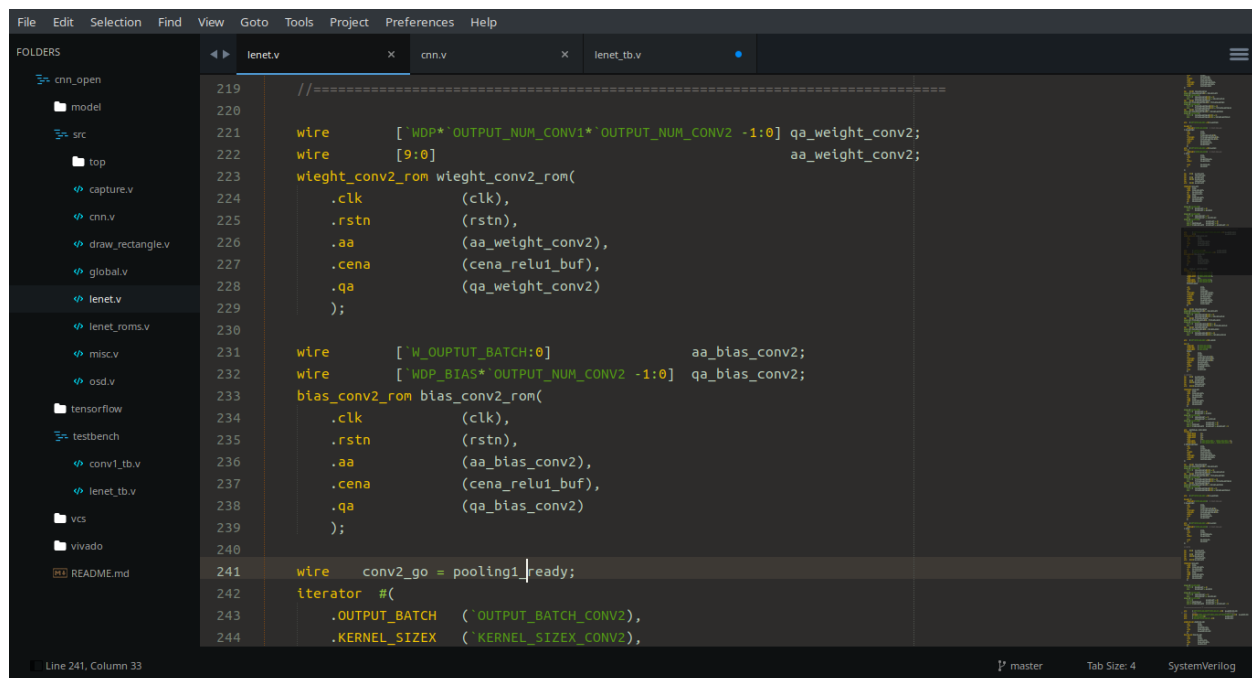
```
163    max_pool #(
164        .INPUT_NUM ( `OUTPUT_NUM_CONV1 ) // input plane_num
165    ) max_pooling1(
166        .clk            (clk),
167        .rstn           (rstn),
168        .en             (lcena_conv1_buf_d[1]),
169        .first_data     (first_data_pooling1_d[1]),
170        .last_data      (last_data_pooling1_d[1]),
171        .data_i         (qa_conv1_buf),
172        .q_en           (qa_pooling1_en),
173        .q              (qa_pooling1)
174    );
175    wire [ `WDP*`OUTPUT_NUM_CONV1 -1:0] qa_relu1;
176    relu #(
177        .INPUT_NUM ( `OUTPUT_NUM_CONV1 ) // input plane_num
178    ) relu1(
179        .clk            (clk),
180        .rstn           (rstn),
181        .en             (qa_pooling1_en),
182        .data_i         (qa_pooling1),
183
184        .q_en           (qa_relu1_en),
185        .q              (qa_relu1)
186
187    );
```

بعد از آن یک لایه‌ی max_pool قرار گرفته که با پایان عملکرد لایه‌ی قبل شروع به کار می‌کند.



```
38     );
39     wire conv1_go = go;
40     iterator #(
41         .OUTPUT_BATCH ('OUTPUT_BATCH_CONV1),
42         .KERNEL_SIZEX ('KERNEL_SIZEX_CONV1),
43         .KERNEL_SIZEY ('KERNEL_SIZEY_CONV1),
44         .STEP          (1),
45         .INPUT_WIDTH   ('INPUT_WIDTH),
46         .INPUT_HEIGHT  ('INPUT_HEIGHT)
47     ) iterator_conv1(
48         .clk            (clk),
49         .rstn           (rstn),
50         .go             (conv1_go),
51         .first_data     (first_data),
52         .last_data      (last_data),
53         .aa_bias        (aa_bias_conv1),
54         .aa_data        (aa_src),
55         .aa_weight      (aa_weight_conv1),
56         .cena           (cena_src),
57         .ready          (conv1_ready)
58     );
59
60     reg [51:0] first_data_d;
61     always @(*) first_data_d[0] = first_data;
62     always @(CLK_RST_EDGE)
63         if (!RST) first_data_d[51:1] <= 0;
```

هر یک از لایه‌های شبکه شامل یک `iterator` است که وظیفه‌ی آن اجرای یک‌به‌یک سلول‌های درون آن لایه است تا عملیات آن لایه تکمیل گردد. تعریف درون ماژول `iterator` در فایل بعدی آمده است.



```
219 //=====
220
221 wire [WDP*OUTPUT_NUM_CONV1*OUTPUT_NUM_CONV2 -1:0] qa_weight_conv2;
222 wire [9:0] aa_weight_conv2;
223 wleight_conv2_rom wleight_conv2_rom(
224     .clk            (clk),
225     .rstn           (rstn),
226     .aa            (aa_weight_conv2),
227     .cena          (cena_relu1_buf),
228     .qa            (qa_weight_conv2)
229 );
230
231 wire [W_OUPTUT_BATCH:0] aa_bias_conv2;
232 wire [WDP_BIAS*OUTPUT_NUM_CONV2 -1:0] qa_bias_conv2;
233 bias_conv2_rom bias_conv2_rom(
234     .clk            (clk),
235     .rstn           (rstn),
236     .aa            (aa_bias_conv2),
237     .cena          (cena_relu1_buf),
238     .qa            (qa_bias_conv2)
239 );
240
241 wire conv2_go = pooling1_ready;
242 iterator #(
243     .OUTPUT_BATCH ('OUTPUT_BATCH_CONV2),
244     .KERNEL_SIZEX ('KERNEL_SIZEX_CONV2),
```

بعد از این دو لایه به سطح بعدی شبکه‌ی عصبی می‌رسیم که لایه‌ی دوم `conv` در شروع آن قرار گرفته:

The screenshot shows a Verilog code editor with the file 'lenet.v' open. The code defines a function 'conv' that takes several inputs: 'INPUT_NUM', 'OUTPUT_NUM', 'WIGHT_SHIFT', 'clk', 'rstn', 'en', 'first_data', 'last_data', 'data_i', 'bias', 'weight', 'q', and 'q_en'. The function performs a convolution operation and returns the result. The code is written in SystemVerilog and includes comments for the input and output dimensions.

```
281
282   conv #(
283     .INPUT_NUM    ('OUTPUT_NUM_CONV1),
284     .OUTPUT_NUM    ('OUTPUT_NUM_CONV2),
285     .WIGHT_SHIFT    ('WIGHT_SHIFT)
286   )conv2(
287     .clk            (clk),
288     .rstn           (rstn),
289     .en             (~cena_relu1_buf_d[1]),
290     .first_data      (first_data_conv2_d[1]),
291     .last_data       (last_data_conv2_d[1]),
292     .data_i          (qa_relu1_buf),
293     .bias            (qa_bias_conv2),
294     .weight          (qa_weight_conv2),
295     .q               (q_conv2),
296     .q_en            (q_conv2_en)
297   );
298
299   // 10*10*16
300   reg [6:0] aa_conv2_buf;
301   reg       cena_conv2_buf;
302   reg [6:0] ab_conv2_buf;
303   reg [255:0] db_conv2_buf;
304   reg       cenb_conv2_buf;
305   wire [255:0] qa_conv2_buf;
306
```

سپس لایه‌ی max_pool ...

The screenshot shows the same Verilog code editor with the file 'lenet.v' open. The code defines a function 'max_pool' that takes several inputs: 'INPUT_NUM', 'OUTPUT_NUM', 'clk', 'rstn', 'en', 'first_data', 'last_data', 'data_i', 'q', and 'q_en'. The function performs a max pooling operation and returns the result. The code is written in SystemVerilog and includes comments for the input and output dimensions.

```
366
367   max_pool #(
368     .INPUT_NUM    ('OUTPUT_NUM_CONV2) // input plane_num
369   ) max_pooling2(
370     .clk            (clk),
371     .rstn           (rstn),
372     .en             (lcena_conv2_buf_d[1]),
373     .first_data      (first_data_pooling2_d[1]),
374     .last_data       (last_data_pooling2_d[1]),
375     .data_i          (qa_conv2_buf),
376     .q_en           (qa_pooling2_en),
377     .q               (qa_pooling2)
378   );
379
380   wire [ 'WDW*'OUTPUT_NUM_CONV2 -1:0] qa_relu2;
381   relu #(
382     .INPUT_NUM    ('OUTPUT_NUM_CONV2) // input plane_num
383   ) relu2(
384     .clk            (clk),
385     .rstn           (rstn),
386     .en             (qa_pooling2_en),
387     .data_i         (qa_pooling2),
388
389     .q_en           (qa_relu2_en),
390     .q              (qa_relu2)
391   );
```

از سطح سوم قسمت fully_connected شروع می‌شود:

```
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  cnn_open
  model
  src
  top
  capture.v
  cnn.v
  draw_rectangle.v
  global.v
  lenet.v
  lenet_roms.v
  misc.v
  osd.v
  tensorflow
  testbench
  conv1_tb.v
  lenet_tb.v
  vcs
  vivado
  README.md

425
426 //===== FC =====
427
428 wire [ 'WDP*'OUTPUT_NUM_CONV2*'OUTPUT_NUM_FC1 -1:0] qa_weight_FC1_rom;
429 // wire [11:0] aa_weight_FC1;
430 wire [ $clog2('KERNEL_SIZE_FC1*'KERNEL_SIZE_FC1*'OUTPUT_BATCH_FC1)-1:0] aa_weight_FC1;
431 wire [ 'W_OUPUT_BATCH:0] aa_bias_FC1;
432 wire [ 'WDP_BIAS*'OUTPUT_NUM_FC1 -1:0] qa_bias_FC1;
433
434 wiewght_fc1_rom wiewght_fc1_rom(
435     .clk (clk),
436     .rstn (rstn),
437     .aa (aa_weight_FC1),
438     .cena (cena_relu2_buf),
439     .qa (qa_weight_FC1_rom)
440 );
441 bias_fc1_rom bias_fc1_rom(
442     .clk (clk),
443     .rstn (rstn),
444     .aa (aa_bias_FC1),
445     .cena (cena_relu2_buf),
446     .qa (qa_bias_FC1)
447 );
448
449
450 wire fc1_go = pooling2_ready;
```

در این سطح تنها یک لایه وجود دارد که به صورت fully_connected می باشد.

```
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  cnn_open
  model
  src
  top
  capture.v
  cnn.v
  draw_rectangle.v
  global.v
  lenet.v
  lenet_roms.v
  misc.v
  osd.v
  tensorflow
  testbench
  conv1_tb.v
  lenet_tb.v
  vcs
  vivado
  README.md

608
609 wire [ 'WDP*'OUTPUT_NUM_FC2 -1:0] q_fc2;
610 conv #(
611     .INPUT_NUM ('OUTPUT_NUM_FC1),
612     .OUTPUT_NUM ('OUTPUT_NUM_FC2),
613     .WIGHT_SHIFT ('WIGHT_SHIFT)
614 )conv_fc2(
615     .clk (clk),
616     .rstn (rstn),
617     .en (~cena_relu_fc1_buf_d[1]),
618     .first_data (first_data_fc2_d[1]),
619     .last_data (last_data_fc2_d[1]),
620     .data_i (qa_relu_fc1_buf),
621     .bias (qa_bias_fc2),
622     .weight (qa_weight_fc2_rom),
623     .q (q_fc2),
624     .q_en (q_fc2_en)
625 );
626
627 wire [ 'WDP*'OUTPUT_NUM_FC2 -1:0] qa_relu_fc2;
628 relu #(
629     .INPUT_NUM ('OUTPUT_NUM_FC2) // input plane_num
630 )relu_fc2(
631     .clk (clk),
632     .rstn (rstn),
633     .en (q_fc2_en),
```

همچنین سطح بعدی نیز fully_connected می باشد:

```
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  cnn_open
  model
  src
  top
  capture.v
  cnn.v
  draw_rectangle.v
  global.v
  lenet.v
  lenet_roms.v
  misc.v
  osd.v
  tensorflow
  testbench
  conv1_tb.v
  lenet_tb.v
  vcs
  vivado
  README.md

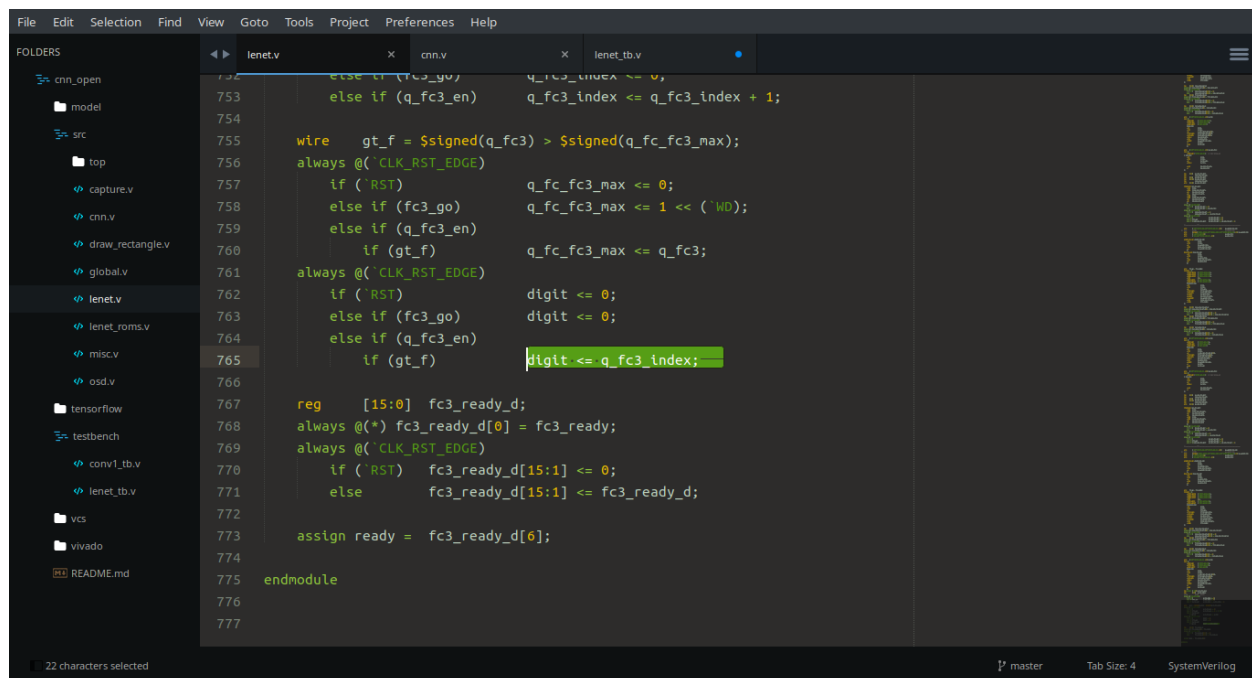
546
547 //=====FC2=====
548
549 wire [ 'WDP'*OUTPUT_NUM_FC1*OUTPUT_NUM_FC2 -1:0] qa_weight_fc2_rom;
550 // wire [11:0] aa_weight_fc2;
551 wire [$clog2('KERNEL_SIZEX_FC2*'KERNEL_SIZEY_FC2* OUTPUT_BATCH_FC2)-1:0] aa_weight_fc2;
552 wire [ 'W_OUPUT_BATCH:0] aa_bias_fc2;
553 wire [ 'WDP_BIAS* OUTPUT_NUM_FC2 -1:0] qa_bias_fc2;
554
555 wleight_fc2_rom wleight_fc2_rom(
556     .clk (clk),
557     .rstn (rstn),
558     .aa (aa_weight_fc2),
559     .cena (cena_relu_fc1_buf),
560     .qa (qa_weight_fc2_rom)
561 );
562 bias_fc2_rom bias_fc2_rom(
563     .clk (clk),
564     .rstn (rstn),
565     .aa (aa_bias_fc2),
566     .cena (cena_relu_fc1_buf),
567     .qa (qa_bias_fc2)
568 );
569
570
571 wire fc2_go = fc1_ready;
```

و سطح بعد:

```
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
  cnn_open
  model
  src
  top
  capture.v
  cnn.v
  draw_rectangle.v
  global.v
  lenet.v
  lenet_roms.v
  misc.v
  osd.v
  tensorflow
  testbench
  conv1_tb.v
  lenet_tb.v
  vcs
  vivado
  README.md

668 //=====FC3=====
669
670 wire [ 'WDP'*OUTPUT_NUM_FC2*OUTPUT_NUM_FC3 -1:0] qa_weight_fc3_rom;
671 // wire [11:0] aa_weight_fc3;
672 wire [$clog2('KERNEL_SIZEX_FC3*'KERNEL_SIZEY_FC3* OUTPUT_BATCH_FC3)-1:0] aa_weight_fc3;
673 wire [ 'W_OUPUT_BATCH:0] aa_bias_fc3;
674 wire [ 'WDP_BIAS* OUTPUT_NUM_FC3 -1:0] qa_bias_fc3;
675
676 wleight_fc3_rom wleight_fc3_rom(
677     .clk (clk),
678     .rstn (rstn),
679     .aa (aa_weight_fc3),
680     .cena (cena_relu_fc2_buf),
681     .qa (qa_weight_fc3_rom)
682 );
683 bias_fc3_rom bias_fc3_rom(
684     .clk (clk),
685     .rstn (rstn),
686     .aa (aa_bias_fc3),
687     .cena (cena_relu_fc2_buf),
688     .qa (qa_bias_fc3)
689 );
690
691
692 wire fc3_go = fc2_ready;
```

و در اینجا ساختار شبکه‌ی عصبی به پایان می‌رسد و نتیجه بعد از سه لایه fully_connected مشخص می‌گردد:

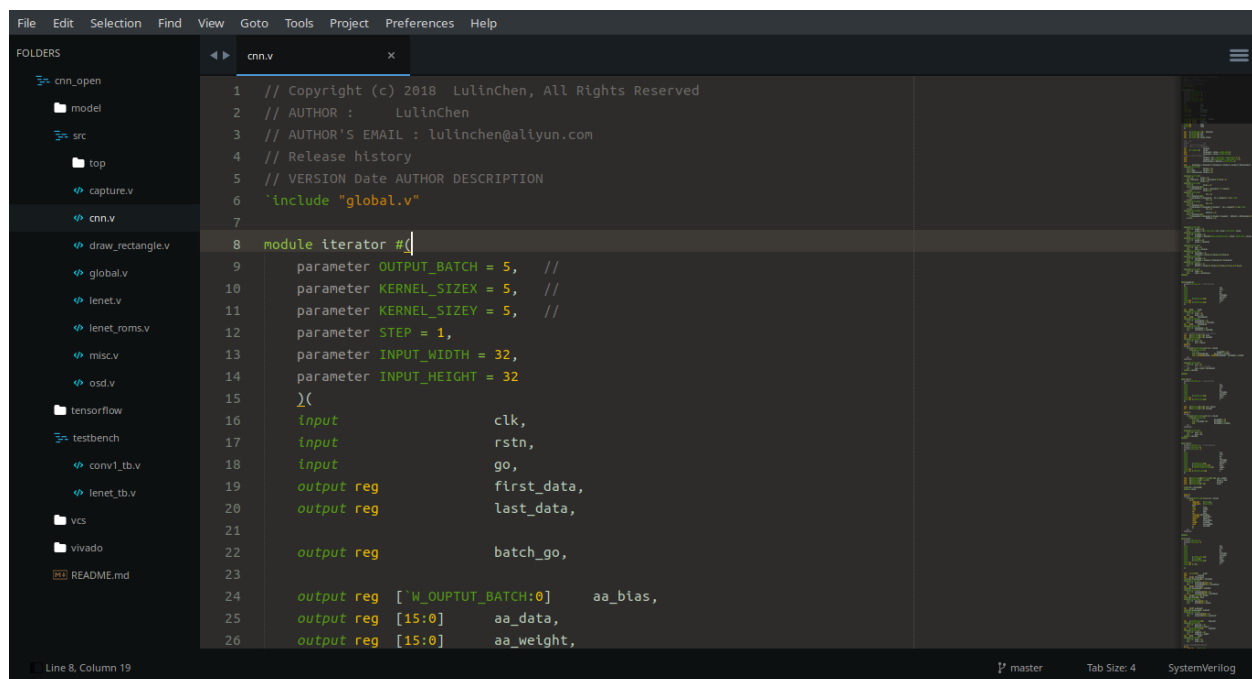


به این ترتیب معماری کلی شبکه‌ی عصبی lenet به این شکل است:

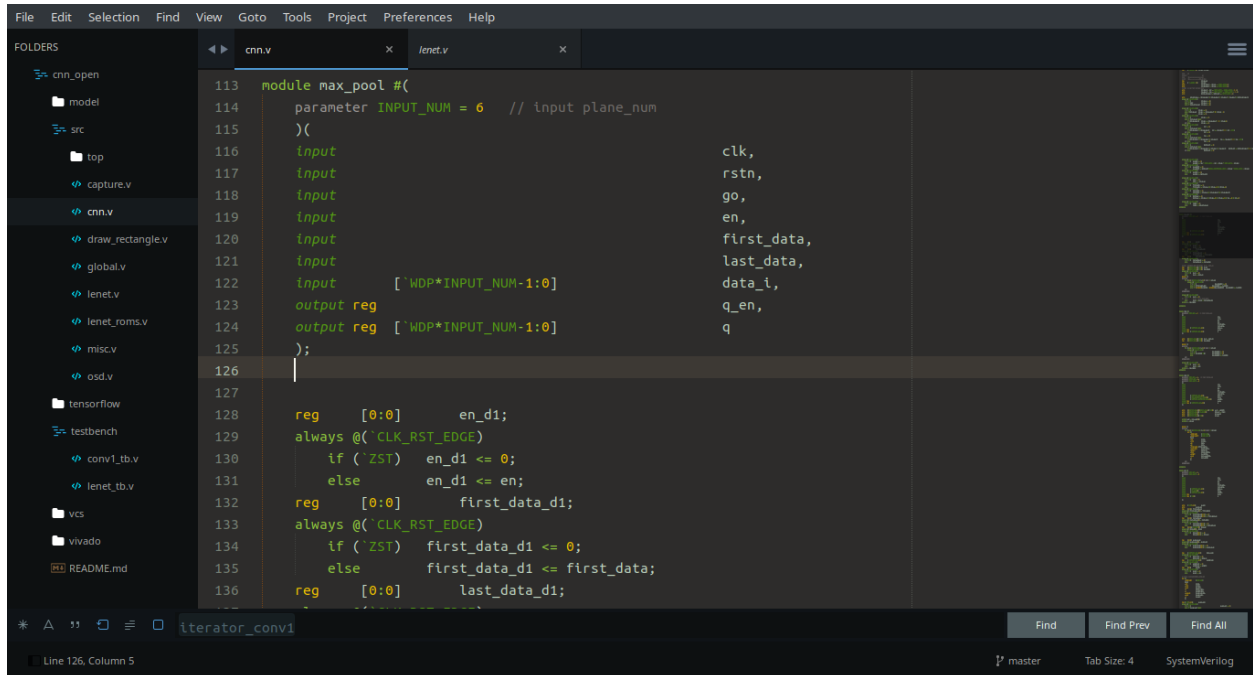
conv - max_pool - conv - max_pool - fully_connected – fully_connected – fully_connected

ماژول های cnn.v

ماژول های تشکیل دهنده‌ی لایه‌های شبکه‌ی عصبی در فایل cnn.v تعریف شده‌اند. اولین ماژول iterator است.

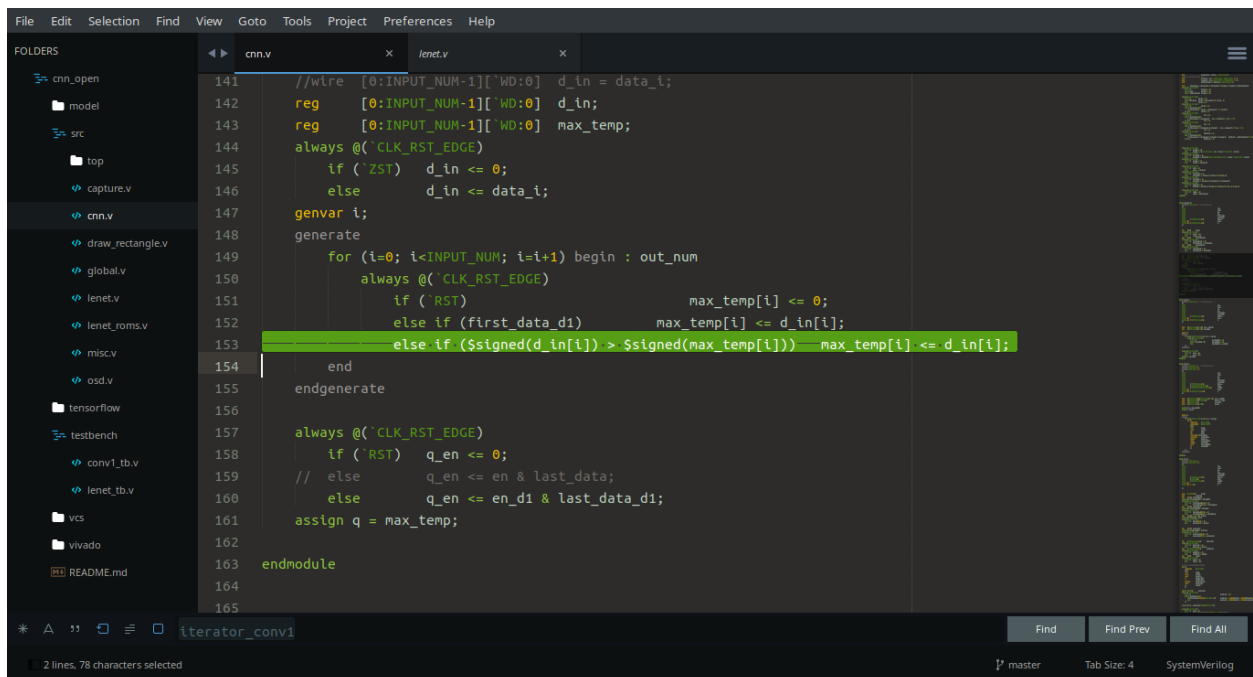


ماژول `iterator` همانگونه که اشاره شد وظیفه‌ی انجام یک‌به‌یک عملیات‌های مربوط به هر لایه را دارد و در همه‌ی لایه‌ها وجود دارد. این ماژول با توجه به سایز ورودی و اندازه‌ی `kernel` که به صورت پارامتر به آن داده شده، به صورت سطر به سطر روی آنها حرکت می‌کند.



```
113 module max_pool #(
114     parameter INPUT_NUM = 6 // input plane_num
115 )
116     input          clk,
117     input          rstn,
118     input          go,
119     input          en,
120     input          first_data,
121     input          last_data,
122     input          [WDP*INPUT_NUM-1:0] data_i,
123     output reg     q_en,
124     output reg     [WDP*INPUT_NUM-1:0] q
125 );
126
127
128     reg [0:0]      en_d1;
129     always @(CLK_RST_EDGE)
130     if ('ZST) en_d1 <= 0;
131     else en_d1 <= en;
132     reg [0:0]      first_data_d1;
133     always @(CLK_RST_EDGE)
134     if ('ZST) first_data_d1 <= 0;
135     else first_data_d1 <= first_data;
136     reg [0:0]      last_data_d1;
```

ماژول بعد `max_pool` است که عملکرد آن `max` گرفتن از ناحیه‌ی ورودی و انتقال این مقدار بیشینه به لایه‌ی بعد است.



```
141 //wire [0:INPUT_NUM-1][WDP:0] d_in = data_i;
142 reg [0:INPUT_NUM-1][WDP:0] d_in;
143 reg [0:INPUT_NUM-1][WDP:0] max_temp;
144 always @(CLK_RST_EDGE)
145 if ('ZST) d_in <= 0;
146 else d_in <= data_i;
147 genvar i;
148 generate
149     for (i=0; i<INPUT_NUM; i=i+1) begin : out_num
150         always @(CLK_RST_EDGE)
151             if ('RST) max_temp[i] <= 0;
152             else if (first_data_d1) max_temp[i] <= d_in[i];
153             else if ($signed(d_in[i]) > $signed(max_temp[i])) max_temp[i] <= d_in[i];
154         end
155     endgenerate
156
157     always @(CLK_RST_EDGE)
158     if ('RST) q_en <= 0;
159     // else q_en <= en & last_data;
160     else q_en <= en_d1 & last_data_d1;
161     assign q = max_temp;
162
163 endmodule
164
165
```

در این قسمت و مخصوصا در خط مشخص شده شیوه‌ی عملکرد `max_pool` دیده می‌شود.

```

165
166 module relu #(
167     parameter INPUT_NUM = 6 // input plane_num
168 )
169     input          clk,
170     input          rstn,
171     input          go,
172     input          en,
173     input          first_data,
174     input          last_data,
175     input          data_i,
176     output reg     q_en,
177     output reg     q [WDP*INPUT_NUM-1:0]
178 );
179
180
181 wire [0:INPUT_NUM-1][WD:0] d_in = data_i;
182 reg [0:INPUT_NUM-1][WD:0] max_temp;
183
184 genvar i;
185 generate
186     for (i=0; i<INPUT_NUM; i=i+1) begin : out_num
187         always @(`CLK_RST_EDGE)
188             if (`RST) max_temp[i] <= 0;
189             else if (d_in[i][`WD]) max_temp[i] <= 0;
190     end
191 endgenerate

```

ماژول بعد ماژول relu است که بعد از لایه‌های pooling و fully_connected در شبکه‌ی عصبی قرار می‌گیرد و با توجه به مقادیر ورودی بعضی از آنها را صفر می‌کند که این مسئله با خارج کردن تابع سلول‌ها از حالت صفر خطی در یادگیری شبکه مؤثر است.

```

200
201 module conv #(
202     parameter INPUT_NUM = 1, // input plane_num
203     parameter OUTPUT_NUM = 6,
204     parameter WIGHT_SHIFT = 8
205 )
206     input          clk,
207     input          rstn,
208     input          go,
209     input          en,
210     input          first_data,
211     input          last_data,
212     input          data_i,
213     input          bias,
214     input          weight,
215     output reg     q_en,
216     output reg     q [WDP*OUTPUT_NUM-1:0]
217 );
218
219
220 wire [0:OUTPUT_NUM-1][0:INPUT_NUM-1][WD:0] w_in = weight;
221 wire [0:OUTPUT_NUM-1][WD_BIAS:0] bias_in = bias;
222 wire [0:OUTPUT_NUM-1] acc_q_en;
223 wire [0:OUTPUT_NUM-1][WD:0] acc_q;
224

```

ماژول بعدی ماژول conv می‌باشد که هر لایه‌ی آن شامل مقادیر weight و bias می‌باشد که توسط ورودی‌ها این مقادیر learn می‌شود و همین مقادیر نقش اصلی را در قابلیت تشخیص اعداد توسط شبکه ایفا می‌کنند. مقادیر weight در ورودی‌ها ضرب می‌شوند و سپس با مقادیر bias جمع می‌شوند (لذا این قسمت یک تابع خطی را تشکیل می‌دهد که با اضافه کردن relu تبدیل به یک تابع غیرخطی می‌شود).

```

227
228
229     genvar i;
230     generate
231         for (i=0; i<OUTPUT_NUM; i=i+1) begin : out_num
232             acc #(
233                 .INPUT_NUM      (INPUT_NUM),
234                 .WIGHT_SHIFT    (WIGHT_SHIFT)
235             )acc(
236                 .clk             (clk),
237                 .rstn            (rstn),
238                 .go              (go),
239                 .en              (en),
240                 .first_data      (first_data),
241                 .last_data       (last_data),
242                 .data_i          (data_i),
243                 .bias            (bias_in[i]),
244                 .weight          (w_in[i]),
245                 .q_en            (acc_q_en[i]),
246                 .q               (acc_q[i])
247             );
248         end
249     endgenerate
250
251 endmodule
252

```

همانطور که دیده می‌شود واحد اصلی تشکیل‌دهندهی CONV تعدادی ماژول acc می‌باشد که هر یک وظیفه‌ی تولید یکی از خروجی‌های این لایه را بر عهده دارند.

```

253 module acc #(
254     parameter INPUT_NUM = 1,
255     parameter WIGHT_SHIFT = 8
256 ) (
257     input          clk,
258     input          rstn,
259     input          go,
260     input          en,
261     input          first_data,
262     input          last_data,
263     input          [ WDP*INPUT_NUM-1:0 ] data_i,
264     input          [ 'WD_BIAS:0 ] bias,
265     input          [ WDP*INPUT_NUM-1:0 ] weight,
266     output reg     q_en,
267     output reg     [ WD:0 ] q
268 );
269
270
271
272 wire [ 'WDP*2-1:0 ] q_mac;
273 wire q_mac_en;
274 reg [15:0] first_data_d;
275 always @(*) first_data_d[0] = first_data;
276 always @( 'CLK_RST_EDGE )
277     if ( 'RST ) first_data_d[15:1] <= 0;
278     else first_data_d[15:1] <= first_data_d[15:1];

```

ماژول acc بلافاصله پس از conv قرار گرفته که ساختار درونی آن را می‌بینیم:

```
File Edit Selection Find View Goto Tools Project Preferences Help
cnn.v x lenet.v x
309     else en_d1 <= en;
310
311     // delay 1+$clog2(KERNEL_SIZE_SQ)
312     mac #(
313         .INPUT_NUM      (INPUT_NUM)
314     )mac(
315         .clk             (clk),
316         .rstn            (rstn),
317         .d_en            (en_d1),
318         .d               (data_i_d1),
319         .w               (weight_d1),
320         .q_en_b1         (q_mac_en_b1),
321         .q_en            (q_mac_en),
322         .q               (q_mac)
323     );
324
325     reg [`WDP*2-1:0] q_mac_acc;
326     always @( `CLK_RST_EDGE)
327     if ( `ZST) q_mac_acc <= 0;
328     else if (q_mac_en) begin
329         if (first_data_d[1+$clog2(INPUT_NUM) + 1]) q_mac_acc <= $signed(q_mac) + $signed(bias_d[1+$clog2(INPUT_NUM)+1]);
330         else q_mac_acc <= $signed(q_mac) + $signed(q_mac_acc);
331     end
332
333     always @(*) q = q_mac_acc[ `WDP*2-1:WIGHT_SHIFT];
334
69 characters selected
master Tab Size: 4 SystemVerilog
```

همانطور که در تصویر مشخص است ماژول acc خود شامل یک ماژول mac می باشد (که تعریف آن در ادامه آمده) و عملکرد خود آن دریافت خروجی های mac و اضافه کردن مقدار bias مربوط به این واحد به آن است.

```
File Edit Selection Find View Goto Tools Project Preferences Help
cnn.v x lenet.v x
345 module mac#(
346     parameter INPUT_NUM = 1 // input plane_num
347 )(
348     input clk,
349     input rstn,
350     input d_en,
351     input [ `WDP*INPUT_NUM-1:0] d,
352     input [ `WDP*INPUT_NUM-1:0] w,
353     input q_en,
354     input q_en_b1,
355     output [ `WDP*2-1:0] q
356 );
357
358 parameter INPUT_NUM_PADDING = 2**$clog2(INPUT_NUM);
359
360 wire [0:INPUT_NUM-1][ `WD:0] d_in = d;
361 wire [0:INPUT_NUM-1][ `WD:0] w_in = w;
362 reg [0:INPUT_NUM_PADDING-1][ `WDP*2-1:0] mul = 0;
363
364 reg [15:0] d_en_d;
365 always @(*) d_en_d[0] = d_en;
366 always @( `CLK_RST_EDGE)
367 if ( `RST) d_en_d[15:1] <= 0;
368 else d_en_d[15:1] <= d_en_d;
369
370
Line 337, Column 48
master Tab Size: 4 SystemVerilog
```

ماژول mac مقادیر ورودی لایه را می گیرد و وزن های مربوط به سلول را هم می گیرد و در هم ضرب می کند:


```
File Edit Selection Find View Goto Tools Project Preferences Help
cnn.v x lenet.v x
356 );
357
358 parameter INPUT_NUM_PADDING = 2**$clog2(INPUT_NUM);
359
360 wire [0:INPUT_NUM-1][`WD:0] d_in = d;
361 wire [0:INPUT_NUM-1][`WD:0] w_in = w;
362 reg [0:INPUT_NUM_PADDING-1][`WDP*2-1:0] mul = 0;
363
364 reg [15:0] d_en_d;
365 always @(*) d_en_d[0] = d_en;
366 always @( `CLK_RST_EDGE)
367 if ( `RST) d_en_d[15:1] <= 0;
368 else d_en_d[15:1] <= d_en_d;
369
370
371
372 genvar i;
373 genvar j;
374 generate
375 for (i=0; i<INPUT_NUM; i=i+1) begin : multiply
376 always @( `CLK_RST_EDGE)
377 if ( `RST) mul[i] <= 0;
378 else mul[i] <= $signed(d_in[i]) * $signed(w_in[i]);
379 end
380 endgenerate
381
```

لذا ضرب وزن‌ها و ورودی‌ها در mac انجام شده و توسط acc جمع می‌شود، سپس با مقدار bias جمع می‌شود و این حاصل به عنوان خروجی یکی از سلول‌های یک لایه‌ی conv خواهد بود.

پس دو ماژول mac و acc به عنوان واحدهای ضرب‌کننده و جمع‌کننده‌ی شبکه‌ی عصبی مورد توجه ما هستند.

تعداد بیت‌های مقادیر واحدهای ضرب‌کننده و جمع‌کننده

```
File Edit Selection Find View Goto Tools Project Preferences Help
cnn.v x lenet.v x
338
339 endmodule
340
341
342 // INPUT_NUM 4 3 clks
343 // INPUT_NUM 8 4 clks
344 // INPUT_NUM 16 5 clks
345 module mac#(
346 parameter INPUT_NUM = 1 // input plane_num
347 )(
348 input clk,
349 input rstn,
350 input d_en,
351 input [`WDP*INPUT_NUM-1:0] d,
352 input [`WDP*INPUT_NUM-1:0] w,
353 input q_en,
354 input q_en_b1,
355 output [`WDP*2-1:0] q
356 );
357
358 parameter INPUT_NUM_PADDING = 2**$clog2(INPUT_NUM);
359
360 wire [0:INPUT_NUM-1][`WD:0] d_in = d;
361 wire [0:INPUT_NUM-1][`WD:0] w_in = w;
362 reg [0:INPUT_NUM_PADDING-1][`WDP*2-1:0] mul = 0;
363
```

در این قسمت مشاهده می‌شود که تعداد بیت‌های وزن‌ها که در ضرب‌کننده به کار می‌روند متغیر WDP می‌باشد.

```

198 endmodule
199
200
201 module conv #(
202     parameter INPUT_NUM = 1, // input plane_num
203     parameter OUTPUT_NUM = 6,
204     parameter WIGHT_SHIFT = 8
205 ) (
206     input clk,
207     input rstn,
208     input go,
209     input en,
210     input first_data,
211     input last_data,
212     input [WDP*INPUT_NUM-1:0] data_i,
213     input [WDP_BIAS*OUTPUT_NUM-1:0] bias,
214     input [WDP*INPUT_NUM*OUTPUT_NUM-1:0] weight,
215     output reg [WDP*OUTPUT_NUM-1:0] q_en,
216     output reg [WDP*OUTPUT_NUM-1:0] q
217 );
218
219
220 wire [0:OUTPUT_NUM-1][0:INPUT_NUM-1][WDP:0] w_in = weight;
221 wire [0:OUTPUT_NUM-1][WDP_BIAS:0] bias_in = bias;
222 wire [0:OUTPUT_NUM-1] acc_en;
223 wire [0:OUTPUT_NUM-1][WDP:0] acc_q;

```

همچنین در قسمت conv دیده می‌شود که طول بیت‌های bias برابر WDP_BIAS می‌باشد.

```

1
2 `ifndef GLOBAL_INC
3 `define GLOBAL_INC
4
5 `ifdef TIMESCALE_PS
6     `timescale 1ps/1ps
7     `define TIME_COEFF 1000
8 `else
9     `timescale 1ns/1ps
10    `define TIME_COEFF 1
11 `endif
12
13
14 `define WDP_BIAS 23
15 `define WDP_BIAS 24
16 `define WDP 15
17 `define WDP 16
18
19 `define WIGHT_SHIFT 15
20
21 `define INPUT_NUM 1
22 `define OUTPUT_NUM 6
23
24
25 `define W_PLANEW 4
26 `define W_PLANEH 4

```

این مقادیر در فایل global.v تعریف شده‌اند و برابر 24 و 16 می‌باشند.

به این ترتیب با بررسی این فایل‌های پروژه و ماژول‌ها توانستیم ساختار و معماری کلی شبکه‌ی عصبی lenet و همچنین ساختار درونی واحدهای تشکیل‌دهنده‌ی آن را کشف کنیم و همچنین سپس از این طریق تعداد بیت‌های اعداد مورد استفاده در واحدهای ضرب‌کننده و جمع‌کننده و همچنین محل تعریف آنها را برای تغییر پیدا کردیم.

قسمت سوم:

خیر، دارای اشکال خواهد بود چرا که در دقت آن پایین آمده است. در ادامه تعداد بیت های WD و WDP را به ۸ و ۷ تغییر دادیم و خروجی را مشاهده کردیم.

T	186618==process a frame	0, digit 1	=====
T	371238==process a frame	1, digit 1	=====
T	555858==process a frame	2, digit 1	=====
T	740478==process a frame	3, digit 1	=====
T	925098==process a frame	4, digit 1	=====
T	1109718==process a frame	5, digit 1	=====
T	1294338==process a frame	6, digit 1	=====
T	1478958==process a frame	7, digit 1	=====
T	1663578==process a frame	8, digit 1	=====
T	1848198==process a frame	9, digit 1	=====
T	2032818==process a frame	10, digit 1	=====
T	2217438==process a frame	11, digit 1	=====
T	2402058==process a frame	12, digit 1	=====
T	2586678==process a frame	13, digit 1	=====
T	2771298==process a frame	14, digit 1	=====
T	2955918==process a frame	15, digit 1	=====
T	3140538==process a frame	16, digit 1	=====
T	3325158==process a frame	17, digit 1	=====
T	3509778==process a frame	18, digit 1	=====
T	3694398==process a frame	19, digit 1	=====

که مشاهده می کنید تمام جواب ها غلط است. البته با افزایش WDP به ۱۲ و ۱۱ نتایج زیر را مشاهده کردیم:

T	186618==process a frame	0, digit 9	=====
T	371238==process a frame	1, digit 1	=====
T	555858==process a frame	2, digit 6	=====
T	740478==process a frame	3, digit 9	=====
T	925098==process a frame	4, digit 3	=====
T	1109718==process a frame	5, digit 8	=====
T	1294338==process a frame	6, digit 8	=====
T	1478958==process a frame	7, digit 7	=====
T	1663578==process a frame	8, digit 3	=====
T	1848198==process a frame	9, digit 2	=====
T	2032818==process a frame	10, digit 3	=====
T	2217438==process a frame	11, digit 3	=====
T	2402058==process a frame	12, digit 2	=====
T	2586678==process a frame	13, digit 5	=====
T	2771298==process a frame	14, digit 5	=====
T	2955918==process a frame	15, digit 6	=====
T	3140538==process a frame	16, digit 2	=====
T	3325158==process a frame	17, digit 7	=====
T	3509778==process a frame	18, digit 3	=====
T	3694398==process a frame	19, digit 3	=====

که تنها دو جواب درست دارد. که نشان می دهد دقت به شدت کاهش پیدا کرده است.

بخش چهارم:

در این قسمت با استفاده از ابزار سنتز *vivado*، باید مدار را سنتز کنیم. برای این کار روی گزینه زیر کلیک می‌کنیم:

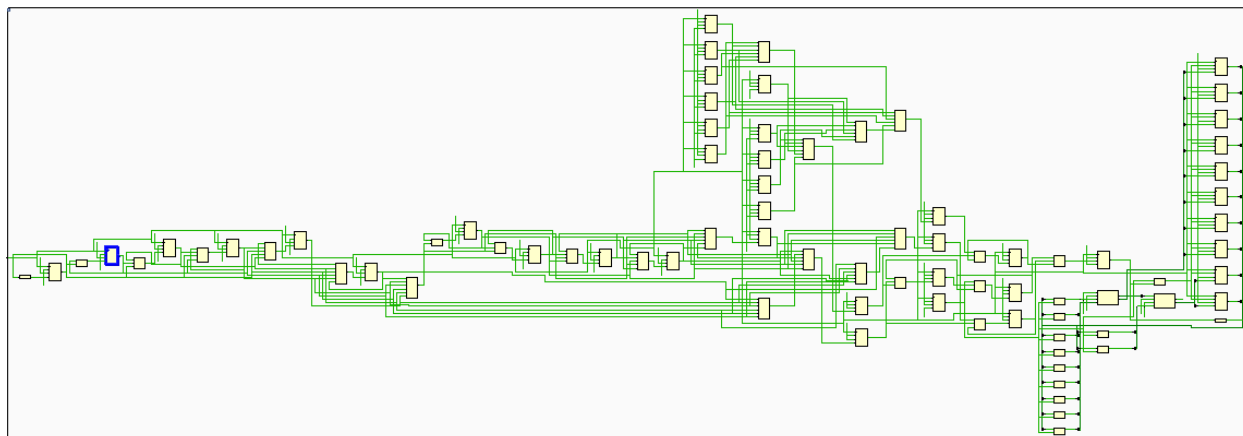
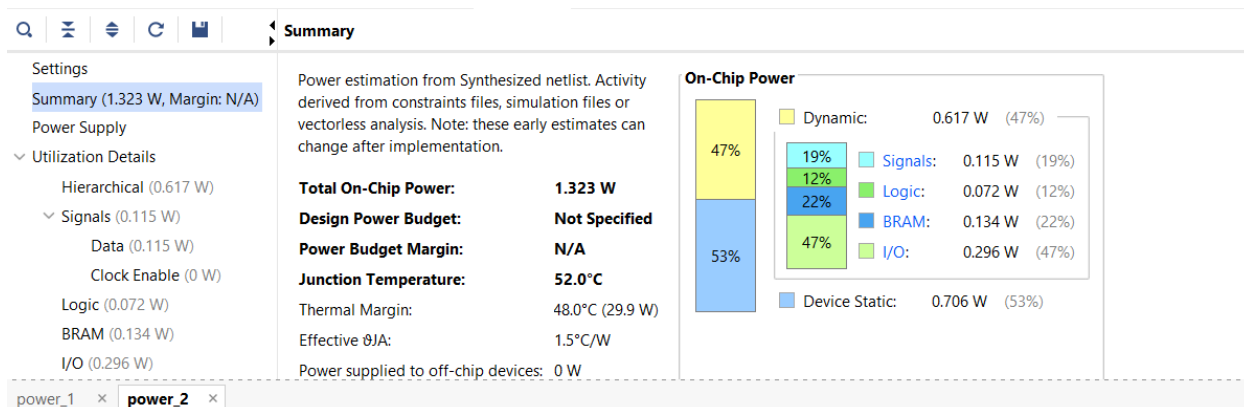
SYNTHESIS

▶ Run Synthesis

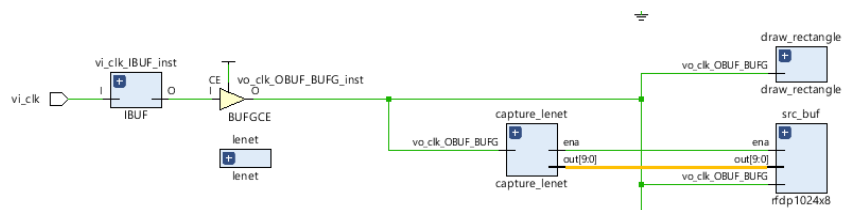
بعد از انجام این کار، از ما تعداد **core** هایی که برای سنتز استفاده شود را سوال می‌کند و بعد از مشخص کردن آن فرایند سنتز شروع می‌شود:

Running synth_design Cancel

در انتهای صفحه سوم گفته شده است عبارت *defineFPGA* را در ابتدای فایل *lenet_top.v* قرار دهیم؛ در ابتدا ما این کار را انجام ندادیم و نتایج زیر را مشاهده کردیم:



مدار داخل *lene* – *tcapture*



Settings

Summary (5.404 W, Margin: N/A)

Power Supply

Utilization Details

- Hierarchical (4.605 W)
 - Signals (1.754 W)
 - Data (1.301 W)
 - Clock Enable (0.44 W)
 - Set/Reset (0.014 W)
 - Logic (0.583 W)
 - BRAM (0.425 W)
 - Clock Manager (1.116 W)

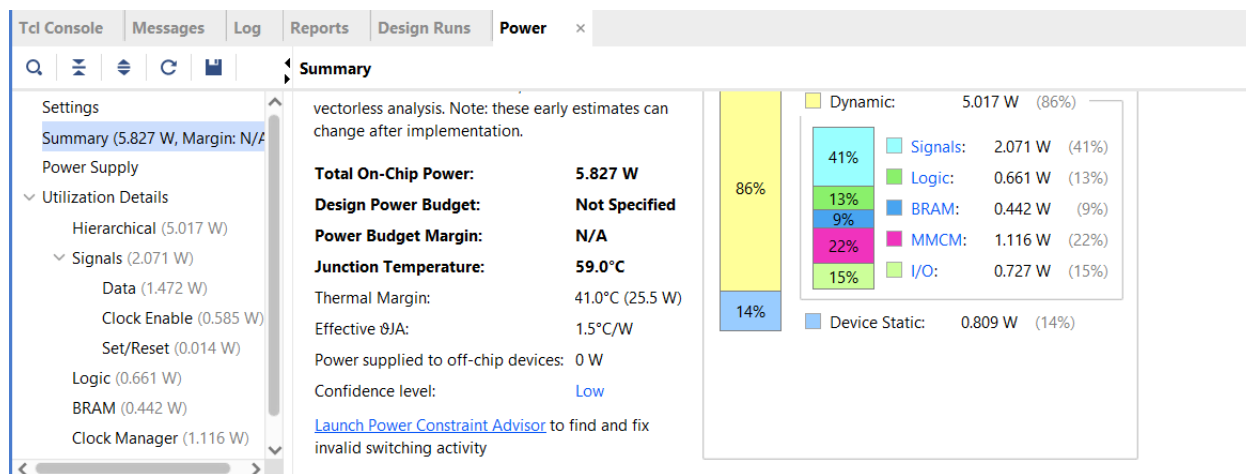
Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	5.404 W
Design Power Budget:	Not Specified
Power Budget Margin:	N/A
Junction Temperature:	58.4°C
Thermal Margin:	41.6°C (25.9 W)
Effective θ_{JA} :	1.5°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

On-Chip Power

■ Dynamic:	4.605 W (85%)
■ Signals:	1.754 W (38%)
■ Logic:	0.583 W (13%)
■ BRAM:	0.425 W (9%)
■ MMCM:	1.116 W (24%)
■ I/O:	0.727 W (16%)
■ Device Static:	0.799 W (15%)

در زیر می‌توانیم توان مصرفی با بیت‌های 16 و 15 را مشاهده کنیم.



همانطور که مشاهده می کنید توان مصرفی در حالتی که WD، ۱۶ است بیشتر از حالتی است که ۱۲ است.

لینک ویدیو آپارات:

<https://aparat.com/v/0T4GL>

لینک گیتهاب: