

EE lab3 report: mini-IDS

Name: Chenyang Zhang

GitHub Link: <https://github.com/zhangc74/ee533-lab3>

For this lab, I will create an intrusion detection system (mini-IDS) using schematics, IP Cores, and verilog.

## Schematic Capture

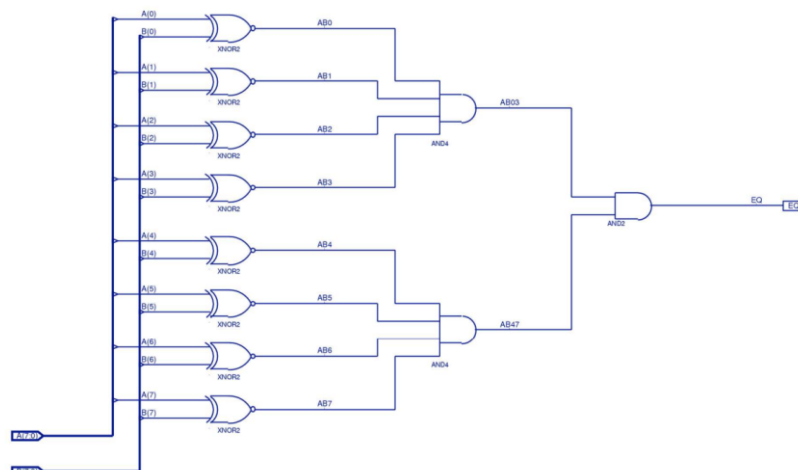
### 1. Create busmerge.v

Used to spell 48-bit + 64-bit into 112-bit, it is a "splicer" for the subsequent wordmatch

```
1 module busmerge(da, db, q);
2
3     input [47:0] da;
4     input [63:0] db;
5     output [111:0] q;
6     assign q = {da,db};
7
8 endmodule
9
```

### 2. Draw comp8.sch

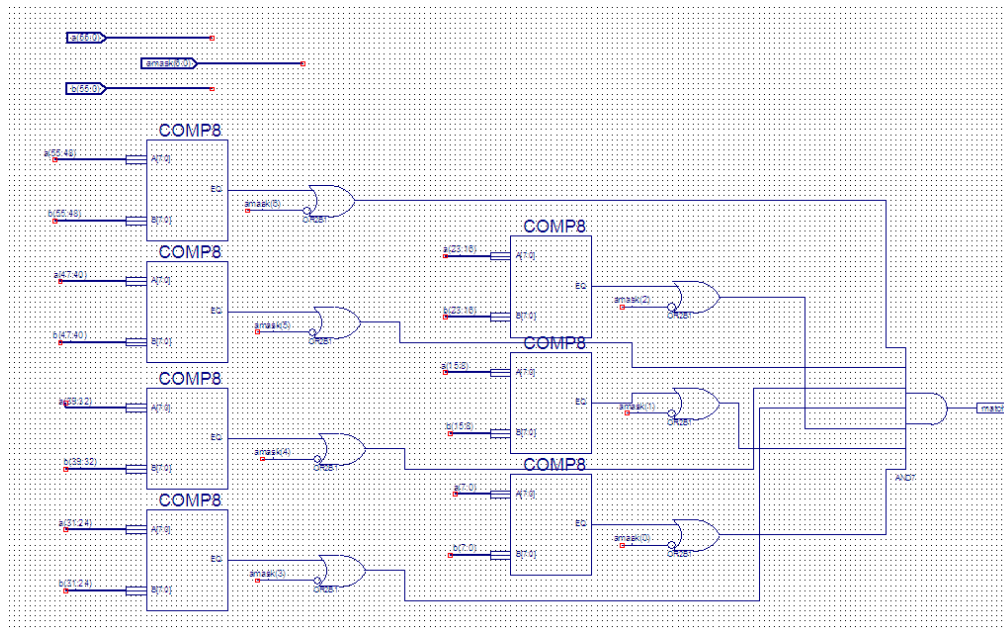
The 8-bit comparator is a built-in Xilinx component and is included for reference, so I do not need to re-implement it.      input: A[7:0], B[7:0]      output: EQ (equal = 1)



### 3. Draw comparator.sch

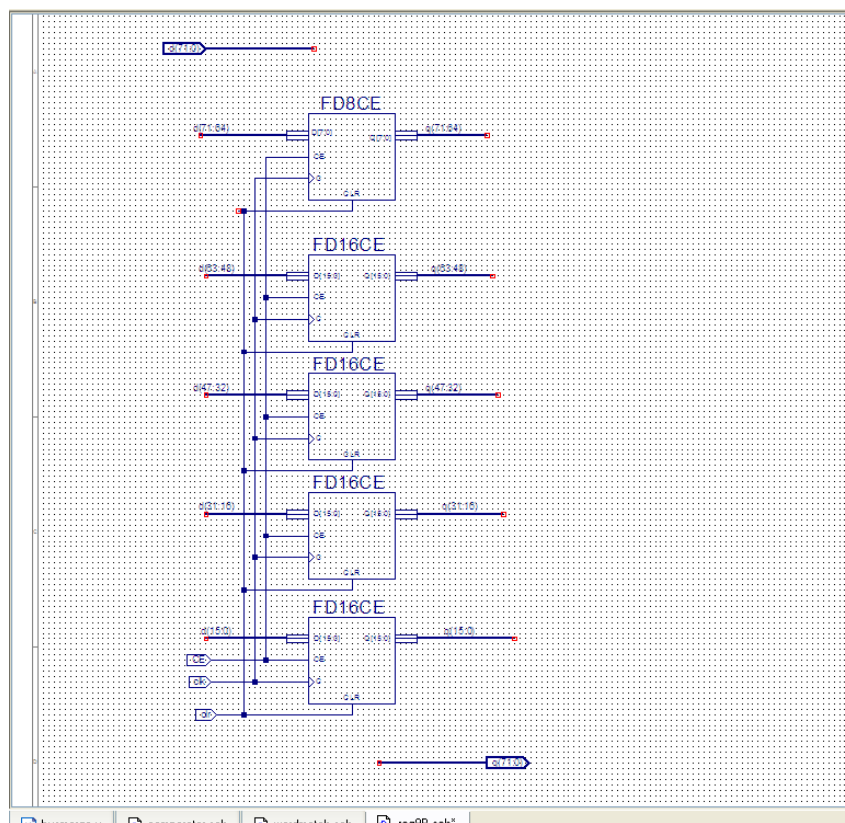
Used to compare a 56-bit input. It is built using seven 8-bit comparators, where each comparator checks one byte of the data. The AMASK[6:0] signal allows some bytes to be ignored during comparison. The output match becomes 1 only when the required bytes

match the pattern.



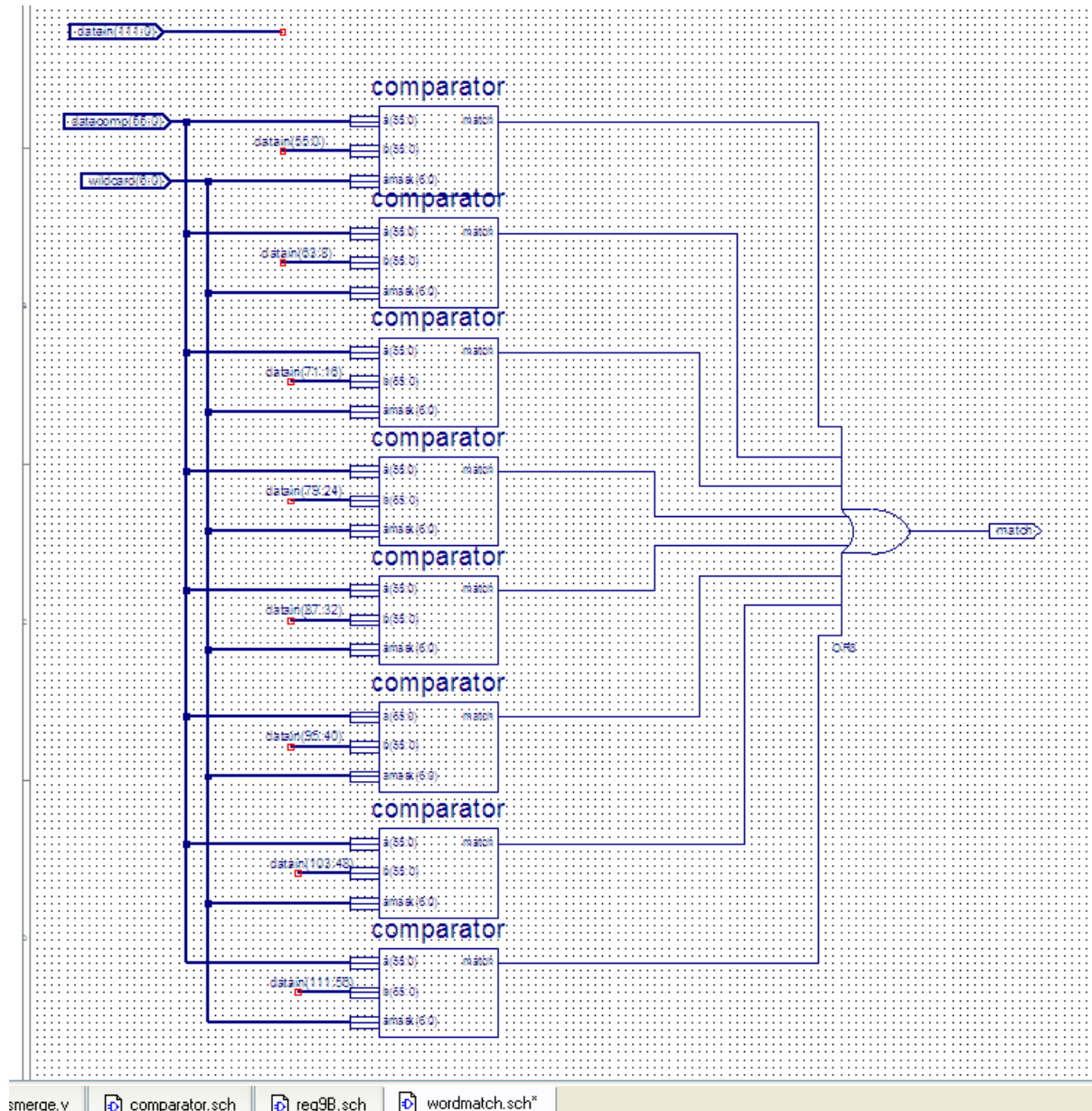
#### 4. Draw reg9B.sch

This is a 9-byte (72-bit) register used to store packet data. It is built using one 8-bit register and four 16-bit registers connected together. All registers share the same clock, enable, and clear signals so that the entire 72-bit value is updated at the same time. This module is used to temporarily hold data before it is processed by other modules.



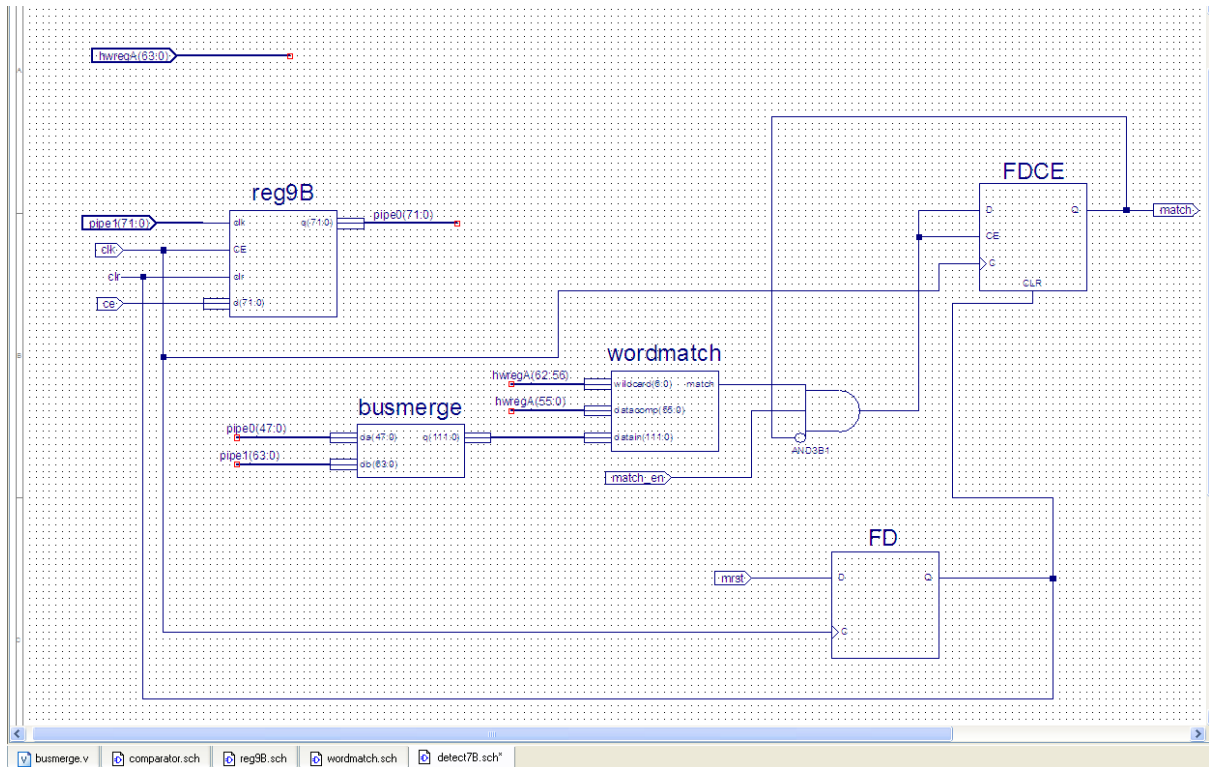
5. Draw wordmatch.sch

Used to scan incoming data for a matching pattern. It takes a 112-bit input and splits it into several overlapping 56-bit segments. Each segment is compared with the target pattern using the comparator module. The match results from all comparators are combined using OR logic, so the output match becomes 1 if any segment matches the pattern.



6. Draw detect7B.sch

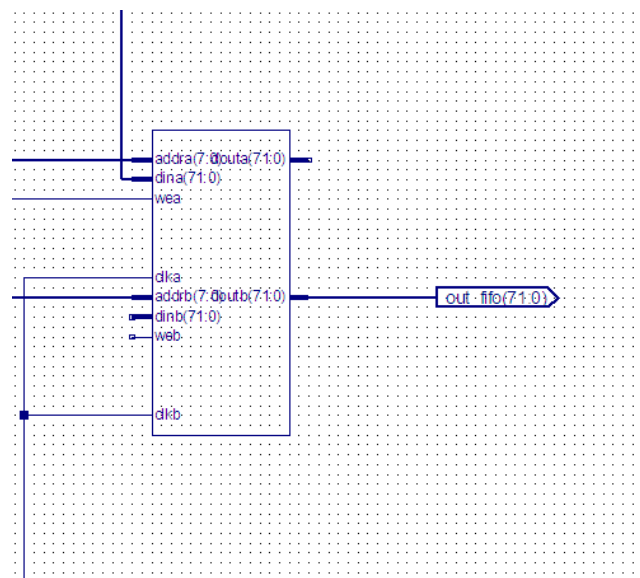
Used to connect the pattern matching logic with the data flow control. It takes packet data, combines different data fields using busmerge, and sends the result to the wordmatch module for comparison. When a match is detected, control signals are generated and stored using flip-flops to indicate that a suspicious pattern has been found. This module acts as the main detection unit of the mini intrusion detection system.

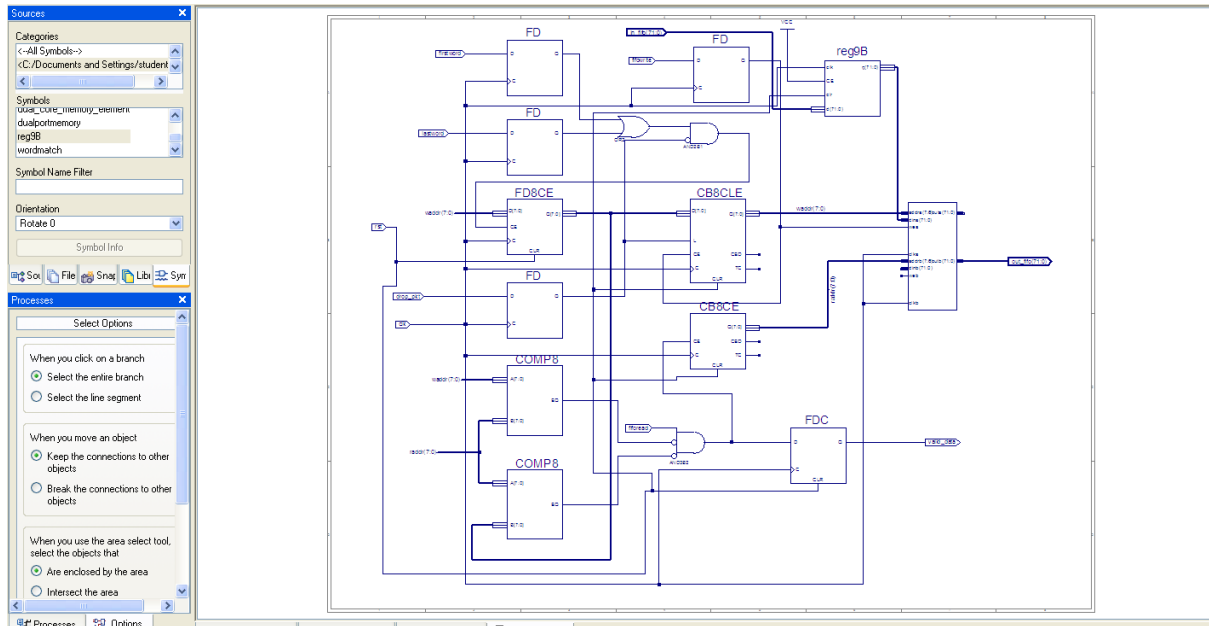


## 7. Draw dropfifo.sch

Used to control whether packets are forwarded or dropped. It uses a FIFO memory to buffer packet data and comparison results. When the detection logic indicates a match, the module prevents the corresponding packet from being forwarded. This module simulates how an intrusion detection system can drop suspicious packets in a real network environment.

- ✓ Add a new IP Core component for a 9-byte (72 bit) wide synchronous dual port memory (1 read, 1 write port).





8. Convert all of my schematics to Verilog (ISE provides this feature).

- 1) Take a look at the created Verilog. Do they make sense? Which do you think easier: entering the schematics or writing Verilog? Why? In which cases might you do the other?  
I think the generated Verilog files make sense.

I think drawing schematics is easier because it is more visual and easier to understand how the system works.

However, for simple logic like a multiplexer, writing Verilog is faster and cleaner, especially when the data width is large, such as 64 bits.

Generated Verilog files are following:

### i) comparator.v

```

1 // comparator.v
2 // Copyright (c) 2006-2008 Xilinx, Inc. All rights reserved.
3 //
4 //
5 //
6 //
7 //
8 //
9 //
10 //
11 //
12 //
13 //
14 //
15 //
16 //
17 //
18 //
19 //
20 //
21 //
22 //
23 //
24 //
25 //
26 //
27 //
28 //
29 //
30 //
31 //
32 //
33 //
34 //
35 //
36 //
37 //
38 //
39 //
40 //
41 //
42 //
43 //
44 //
45 //
46 //
47 //
48 //
49 //
50 //
51 //
52 //
53 //
54 //
55 //
56 //
57 //
58 //
59 //
60 //
61 //
62 //
63 //
64 //
65 //
66 //
67 //
68 //
69 //
70 //
71 //
72 //
73 //
74 //
75 //
76 //
77 //
78 //
79 //
80 //
81 //
82 //
83 //
84 //
85 //
86 //
87 //
88 //
89 //
90 //
91 //
92 //
93 //
94 //
95 //
96 //
97 //
98 //
99 //
100 //
101 //
102 //
103 //
104 //
105 //
106 //
107 //
108 //
109 //
110 //
111 //
112 //
113 //
114 //
115 //
116 //
117 //
118 //
119 //
120 //
121 //
122 //
123 //
124 //
125 //
126 //
127 //
128 //
129 //
130 //
131 //
132 //
133 //
134 //
135 //
136 //
137 //
138 //
139 //
140 //
141 //
142 //
143 //
144 //
145 //
146 //
147 //
148 //
149 //
150 //
151 //
152 //
153 //
154 //
155 //
156 //
157 //
158 //
159 //
160 //
161 //
162 //
163 //
164 //
165 //
166 //
167 //
168 //
169 //
170 //
171 //
172 //
173 //
174 //
175 //
176 //
177 //
178 //
179 //
180 //
181 //
182 //
183 //
184 //
185 //
186 //
187 //
188 //
189 //
190 //
191 //
192 //
193 //
194 //
195 //
196 //
197 //
198 //
199 //
200 //
201 //
202 //
203 //
204 //
205 //
206 //
207 //
208 //
209 //
210 //
211 //
212 //
213 //
214 //
215 //
216 //
217 //
218 //
219 //
220 //
221 //
222 //
223 //
224 //
225 //
226 //
227 //
228 //
229 //
230 //
231 //
232 //
233 //
234 //
235 //
236 //
237 //
238 //
239 //
240 //
241 //
242 //
243 //
244 //
245 //
246 //
247 //
248 //
249 //
250 //
251 //
252 //
253 //
254 //
255 //
256 //
257 //
258 //
259 //
260 //
261 //
262 //
263 //
264 //
265 //
266 //
267 //
268 //
269 //
270 //
271 //
272 //
273 //
274 //
275 //
276 //
277 //
278 //
279 //
280 //
281 //
282 //
283 //
284 //
285 //
286 //
287 //
288 //
289 //
290 //
291 //
292 //
293 //
294 //
295 //
296 //
297 //
298 //
299 //
300 //
301 //
302 //
303 //
304 //
305 //
306 //
307 //
308 //
309 //
310 //
311 //
312 //
313 //
314 //
315 //
316 //
317 //
318 //
319 //
320 //
321 //
322 //
323 //
324 //
325 //
326 //
327 //
328 //
329 //
330 //
331 //
332 //
333 //
334 //
335 //
336 //
337 //
338 //
339 //
340 //
341 //
342 //
343 //
344 //
345 //
346 //
347 //
348 //
349 //
350 //
351 //
352 //
353 //
354 //
355 //
356 //
357 //
358 //
359 //
360 //
361 //
362 //
363 //
364 //
365 //
366 //
367 //
368 //
369 //
370 //
371 //
372 //
373 //
374 //
375 //
376 //
377 //
378 //
379 //
380 //
381 //
382 //
383 //
384 //
385 //
386 //
387 //
388 //
389 //
390 //
391 //
392 //
393 //
394 //
395 //
396 //
397 //
398 //
399 //
400 //
401 //
402 //
403 //
404 //
405 //
406 //
407 //
408 //
409 //
410 //
411 //
412 //
413 //
414 //
415 //
416 //
417 //
418 //
419 //
420 //
421 //
422 //
423 //
424 //
425 //
426 //
427 //
428 //
429 //
430 //
431 //
432 //
433 //
434 //
435 //
436 //
437 //
438 //
439 //
440 //
441 //
442 //
443 //
444 //
445 //
446 //
447 //
448 //
449 //
450 //
451 //
452 //
453 //
454 //
455 //
456 //
457 //
458 //
459 //
460 //
461 //
462 //
463 //
464 //
465 //
466 //
467 //
468 //
469 //
470 //
471 //
472 //
473 //
474 //
475 //
476 //
477 //
478 //
479 //
480 //
481 //
482 //
483 //
484 //
485 //
486 //
487 //
488 //
489 //
490 //
491 //
492 //
493 //
494 //
495 //
496 //
497 //
498 //
499 //
500 //
501 //
502 //
503 //
504 //
505 //
506 //
507 //
508 //
509 //
510 //
511 //
512 //
513 //
514 //
515 //
516 //
517 //
518 //
519 //
520 //
521 //
522 //
523 //
524 //
525 //
526 //
527 //
528 //
529 //
530 //
531 //
532 //
533 //
534 //
535 //
536 //
537 //
538 //
539 //
540 //
541 //
542 //
543 //
544 //
545 //
546 //
547 //
548 //
549 //
550 //
551 //
552 //
553 //
554 //
555 //
556 //
557 //
558 //
559 //
560 //
561 //
562 //
563 //
564 //
565 //
566 //
567 //
568 //
569 //
570 //
571 //
572 //
573 //
574 //
575 //
576 //
577 //
578 //
579 //
580 //
581 //
582 //
583 //
584 //
585 //
586 //
587 //
588 //
589 //
590 //
591 //
592 //
593 //
594 //
595 //
596 //
597 //
598 //
599 //
600 //
601 //
602 //
603 //
604 //
605 //
606 //
607 //
608 //
609 //
610 //
611 //
612 //
613 //
614 //
615 //
616 //
617 //
618 //
619 //
620 //
621 //
622 //
623 //
624 //
625 //
626 //
627 //
628 //
629 //
630 //
631 //
632 //
633 //
634 //
635 //
636 //
637 //
638 //
639 //
640 //
641 //
642 //
643 //
644 //
645 //
646 //
647 //
648 //
649 //
650 //
651 //
652 //
653 //
654 //
655 //
656 //
657 //
658 //
659 //
660 //
661 //
662 //
663 //
664 //
665 //
666 //
667 //
668 //
669 //
670 //
671 //
672 //
673 //
674 //
675 //
676 //
677 //
678 //
679 //
680 //
681 //
682 //
683 //
684 //
685 //
686 //
687 //
688 //
689 //
690 //
691 //
692 //
693 //
694 //
695 //
696 //
697 //
698 //
699 //
700 //
701 //
702 //
703 //
704 //
705 //
706 //
707 //
708 //
709 //
710 //
711 //
712 //
713 //
714 //
715 //
716 //
717 //
718 //
719 //
720 //
721 //
722 //
723 //
724 //
725 //
726 //
727 //
728 //
729 //
730 //
731 //
732 //
733 //
734 //
735 //
736 //
737 //
738 //
739 //
740 //
741 //
742 //
743 //
744 //
745 //
746 //
747 //
748 //
749 //
750 //
751 //
752 //
753 //
754 //
755 //
756 //
757 //
758 //
759 //
760 //
761 //
762 //
763 //
764 //
765 //
766 //
767 //
768 //
769 //
770 //
771 //
772 //
773 //
774 //
775 //
776 //
777 //
778 //
779 //
780 //
781 //
782 //
783 //
784 //
785 //
786 //
787 //
788 //
789 //
790 //
791 //
792 //
793 //
794 //
795 //
796 //
797 //
798 //
799 //
800 //
801 //
802 //
803 //
804 //
805 //
806 //
807 //
808 //
809 //
810 //
811 //
812 //
813 //
814 //
815 //
816 //
817 //
818 //
819 //
820 //
821 //
822 //
823 //
824 //
825 //
826 //
827 //
828 //
829 //
830 //
831 //
832 //
833 //
834 //
835 //
836 //
837 //
838 //
839 //
840 //
841 //
842 //
843 //
844 //
845 //
846 //
847 //
848 //
849 //
850 //
851 //
852 //
853 //
854 //
855 //
856 //
857 //
858 //
859 //
860 //
861 //
862 //
863 //
864 //
865 //
866 //
867 //
868 //
869 //
870 //
871 //
872 //
873 //
874 //
875 //
876 //
877 //
878 //
879 //
880 //
881 //
882 //
883 //
884 //
885 //
886 //
887 //
888 //
889 //
890 //
891 //
892 //
893 //
894 //
895 //
896 //
897 //
898 //
899 //
900 //
901 //
902 //
903 //
904 //
905 //
906 //
907 //
908 //
909 //
910 //
911 //
912 //
913 //
914 //
915 //
916 //
917 //
918 //
919 //
920 //
921 //
922 //
923 //
924 //
925 //
926 //
927 //
928 //
929 //
930 //
931 //
932 //
933 //
934 //
935 //
936 //
937 //
938 //
939 //
940 //
941 //
942 //
943 //
944 //
945 //
946 //
947 //
948 //
949 //
950 //
951 //
952 //
953 //
954 //
955 //
956 //
957 //
958 //
959 //
960 //
961 //
962 //
963 //
964 //
965 //
966 //
967 //
968 //
969 //
970 //
971 //
972 //
973 //
974 //
975 //
976 //
977 //
978 //
979 //
980 //
981 //
982 //
983 //
984 //
985 //
986 //
987 //
988 //
989 //
990 //
991 //
992 //
993 //
994 //
995 //
996 //
997 //
998 //
999 //
1000 //

```

[illegible]

ii) reg9B.v

```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %% Copyright (c) 1996-2004 Xilinx, Inc. All rights reserved.
3  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4
5  //
6  // \ / \ /
7  // \ / \ /   Vendor: Xilinx
8  // \ / \ /   Version : 14.1
9  // \ / \ /   Application : subsetting
10 // \ / \ /   Filename : reg96.vf
11 // \ / \ /   Timestamp : 01/31/2006 01:07:45
12 // \ / \ /
13 // \ / \ /
14
15 //Command: C:\Xilinx\14.1\ISE\bin\ntuserapp\subsetting.exe -intstyle ise -family virtex2p - "C:\Documents and Settings\stefano\working\reg96.vf"
16 //Device: virtex2p
17 //Partname:
18
19 // This verilog netlist is translated from an ECL schematic. It can be
20 // synthesized and simulated, but it should not be modified.
21 //
22 "timescale 1ns / 1ps
23
24 module PRDCTL_MULLING_reg96(C,
25                             CLK,
26                             D0,
27                             Q0);
28
29     Input C;
30     Input CLK;
31     Input CLK0;
32     Input D0[0:3];
33     output Q0[0:3];
34
35
36     PRCTL_1_00 (C,CLK,
37                @0(CLOCK),
38                @0(CLK0),
39                @0(D0[0]),
40                @0(Q0[0]));
41
42     PRCTL_1_00 (C,CLK,
43                @0(CLOCK),
44                @0(CLK0),
45                @0(D0[1]),
46                @0(Q0[1]));
47
48     PRCTL_1_00 (C,CLK,
49                @0(CLOCK),
50                @0(CLK0),
51                @0(D0[2]),
52                @0(Q0[2]));
53
54     PRCTL_1_00 (C,CLK,
55                @0(CLOCK),
56                @0(CLK0),
57                @0(D0[3]),
58                @0(Q0[3]));
59
60     PRCTL_1_00 (C,CLK,
61                @0(CLOCK),
62                @0(CLK0),
63                @0(D0[0]),
64                @0(Q0[0]));
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

[illegible]

```

Z compiler@X: X regfile
in0 > X regfile
1 module FORCE_MULTIM_regfile_C,
2   FORCE_I_Q00 (.C0),
3
4   .CLK(C0),
5   .M0(M0),
6   .Q0(Q0[31]),
7   derivate_I_Q00_2M0T = 1'100,
8
9   FORCE_I_Q01 (.C0),
10  .C0(C0),
11  .CLK(C0),
12  .M0(M0),
13  .Q0(Q0[31]),
14  derivate_I_Q01_2M0T = 1'100,
15
16  FORCE_I_Q02 (.C0),
17  .C0(C0),
18  .CLK(C0),
19  .M0(M0),
20  .Q0(Q0[31]),
21  derivate_I_Q02_2M0T = 1'100,
22
23  FORCE_I_Q03 (.C0),
24  .C0(C0),
25  .CLK(C0),
26  .M0(M0),
27  .Q0(Q0[31]),
28  derivate_I_Q03_2M0T = 1'100,
29
30  FORCE_I_Q04 (.C0),
31  .C0(C0),
32  .CLK(C0),
33  .M0(M0),
34  .Q0(Q0[31]),
35  derivate_I_Q04_2M0T = 1'100,
36
37  FORCE_I_Q05 (.C0),
38  .C0(C0),
39  .CLK(C0),
40  .M0(M0),
41  .Q0(Q0[31]),
42  derivate_I_Q05_2M0T = 1'100,
43
44  endmodule
45
46 "Generate 1m / 3m
47
48 module FORCE_MULTIM_regfile_C,
49   C0,
50   C1,
51   C2,
52   C3,
53   C4,
54   C5,
55   C6,
56   C7,
57   C8,
58   C9,
59   C10,
60   C11,
61   C12,
62   C13,
63   C14,
64   C15,
65   C16,
66   C17,
67   C18,
68   C19,
69   C20,
70   C21,
71   C22,
72   C23,
73   C24,
74   C25,
75   C26,
76   C27,
77   C28,
78   C29,
79   C30,
80   C31,
81   C32,
82   C33,
83   C34,
84   C35,
85   C36,
86   C37,
87   C38,
88   C39,
89   C40,
90   C41,
91   C42,
92   C43,
93   C44,
94   C45,
95   C46,
96   C47,
97   C48,
98   C49,
99   C50,
100  C51,
101  C52,
102  C53,
103  C54,
104  C55,
105  C56,
106  C57,
107  C58,
108  C59,
109  C60,
110  C61,
111  C62,
112  C63,
113  C64,
114  C65,
115  C66,
116  C67,
117  C68,
118  C69,
119  C70,
120  C71,
121  C72,
122  C73,
123  C74,
124  C75,
125  C76,
126  C77,
127  C78,
128  C79,
129  C80,
130  C81,
131  C82,
132  C83,
133  C84,
134  C85,
135  C86,
136  C87,
137  C88,
138  C89,
139  C90,
140  C91,
141  C92,
142  C93,
143  C94,
144  C95,
145  C96,
146  C97,
147  C98,
148  C99,
149  C100,
150  C101,
151  C102,
152  C103,
153  C104,
154  C105,
155  C106,
156  C107,
157  C108,
158  C109,
159  C110,
160  C111,
161  C112,
162  C113,
163  C114,
164  C115,
165  C116,
166  C117,
167  C118,
168  C119,
169  C120,
170  C121,
171  C122,
172  C123,
173  C124,
174  C125,
175  C126,
176  C127,
177  C128,
178  C129,
179  C130,
180  C131,
181  C132,
182  C133,
183  C134,
184  C135,
185  C136,
186  C137,
187  C138,
188  C139,
189  C140,
190  C141,
191  C142,
192  C143,
193  C144,
194  C145,
195  C146,
196  C147,
197  C148,
198  C149,
199  C150,
200  C151,
201  C152,
202  C153,
203  C154,
204  C155,
205  C156,
206  C157,
207  C158,
208  C159,
209  C160,
210  C161,
211  C162,
212  C163,
213  C164,
214  C165,
215  C166,
216  C167,
217  C168,
218  C169,
219  C170,
220  C171,
221  C172,
222  C173,
223  C174,
224  C175,
225  C176,
226  C177,
227  C178,
228  C179,
229  C180,
230  C181,
231  C182,
232  C183,
233  C184,
234  C185,
235  C186,
236  C187,
237  C188,
238  C189,
239  C190,
240  C191,
241  C192,
242  C193,
243  C194,
244  C195,
245  C196,
246  C197,
247  C198,
248  C199,
249  C200,
250  C201,
251  C202,
252  C203,
253  C204,
254  C205,
255  C206,
256  C207,
257  C208,
258  C209,
259  C210,
260  C211,
261  C212,
262  C213,
263  C214,
264  C215,
265  C216,
266  C217,
267  C218,
268  C219,
269  C220,
270  C221,
271  C222,
272  C223,
273  C224,
274  C225,
275  C226,
276  C227,
277  C228,
278  C229,
279  C230,
280  C231,
281  C232,
282  C233,
283  C234,
284  C235,
285  C236,
286  C237,
287  C238,
288  C239,
289  C240,
290  C241,
291  C242,
292  C243,
293  C244,
294  C245,
295  C246,
296  C247,
297  C248,
298  C249,
299  C250,
300  C251,
301  C252,
302  C253,
303  C254,
304  C255,
305  C256,
306  C257,
307  C258,
308  C259,
309  C260,
310  C261,
311  C262,
312  C263,
313  C264,
314  C265,
315  C266,
316  C267,
317  C268,
318  C269,
319  C270,
320  C271,
321  C272,
322  C273,
323  C274,
324  C275,
325  C276,
326  C277,
327  C278,
328  C279,
329  C280,
330  C281,
331  C282,
332  C283,
333  C284,
334  C285,
335  C286,
336  C287,
337  C288,
338  C289,
339  C290,
340  C291,
341  C292,
342  C293,
343  C294,
344  C295,
345  C296,
346  C297,
347  C298,
348  C299,
349  C300,
350  C301,
351  C302,
352  C303,
353  C304,
354  C305,
355  C306,
356  C307,
357  C308,
358  C309,
359  C310,
360  C311,
361  C312,
362  C313,
363  C314,
364  C315,
365  C316,
366  C317,
367  C318,
368  C319,
369  C320,
370  C321,
371  C322,
372  C323,
373  C324,
374  C325,
375  C326,
376  C327,
377  C328,
378  C329,
379  C330,
380  C331,
381  C332,
382  C333,
383  C334,
384  C335,
385  C336,
386  C337,
387  C338,
388  C339,
389  C340,
390  C341,
391  C342,
392  C343,
393  C344,
394  C345,
395  C346,
396  C347,
397  C348,
398  C349,
399  C350,
400  C351,
401  C352,
402  C353,
403  C354,
404  C355,
405  C356,
406  C357,
407  C358,
408  C359,
409  C360,
410  C361,
411  C362,
412  C363,
413  C364,
414  C365,
415  C366,
416  C367,
417  C368,
418  C369,
419  C370,
420  C371,

```

```

lab3 > reg98.v
135 module FDBCE_MXILINX_reg98(C,
148 FDBCE_I_Q0 (.C(C),
151 .D(0(0)),
152 .Q(0(0)));
153 defparam I_Q0.INIT = 1'b0;
154 FDBCE_I_Q1 (.C(C),
155 .CE(CE),
156 .CLR(CLR),
157 .D(0(1)),
158 .Q(0(1)));
159 defparam I_Q1.INIT = 1'b0;
160 FDBCE_I_Q2 (.C(C),
161 .CE(CE),
162 .CLR(CLR),
163 .D(0(2)),
164 .Q(0(2)));
165 defparam I_Q2.INIT = 1'b0;
166 FDBCE_I_Q3 (.C(C),
167 .CE(CE),
168 .CLR(CLR),
169 .D(0(3)),
170 .Q(0(3)));
171 defparam I_Q3.INIT = 1'b0;
172 FDBCE_I_Q4 (.C(C),
173 .CE(CE),
174 .CLR(CLR),
175 .D(0(4)),
176 .Q(0(4)));
177 defparam I_Q4.INIT = 1'b0;
178 FDBCE_I_Q5 (.C(C),
179 .CE(CE),
180 .CLR(CLR),
181 .D(0(5)),
182 .Q(0(5)));
183 defparam I_Q5.INIT = 1'b0;
184 FDBCE_I_Q6 (.C(C),
185 .CE(CE),
186 .CLR(CLR),
187 .D(0(6)),
188 .Q(0(6)));
189 defparam I_Q6.INIT = 1'b0;
190 FDBCE_I_Q7 (.C(C),
191 .CE(CE),
192 .CLR(CLR),
193 .D(0(7)),
194 .Q(0(7)));
195 defparam I_Q7.INIT = 1'b0;
196 endmodule
197 `timescale 1ns / 1ps
198
199 module reg98(CE,
200 clk,
201 clr,
202 d,
203 q);
204
205 input CE;
206 input clk;
207 input clr;
208 input [71:0] d;
209 output [71:0] q;
210

```

```

211 q);
212
213 FDBCE_MXILINX_reg98 XLXI_1 (.C(clk),
214 .CE(CE),
215 .CLR(clr),
216 .D(d[71:64]),
217 .Q(q[71:64]));
218 // synthesis attribute HJ_SET of XLXI_1 is "XLXI_1_0"
219 FDBCE_MXILINX_reg98 XLXI_2 (.C(clk),
220 .CE(CE),
221 .CLR(clr),
222 .D(d[63:48]),
223 .Q(q[63:48]));
224 // synthesis attribute HJ_SET of XLXI_2 is "XLXI_2_1"
225 FDBCE_MXILINX_reg98 XLXI_3 (.C(clk),
226 .CE(CE),
227 .CLR(clr),
228 .D(d[47:32]),
229 .Q(q[47:32]));
230 // synthesis attribute HJ_SET of XLXI_3 is "XLXI_3_2"
231 FDBCE_MXILINX_reg98 XLXI_4 (.C(clk),
232 .CE(CE),
233 .CLR(clr),
234 .D(d[31:16]),
235 .Q(q[31:16]));
236 // synthesis attribute HJ_SET of XLXI_4 is "XLXI_4_3"
237 FDBCE_MXILINX_reg98 XLXI_5 (.C(clk),
238 .CE(CE),
239 .CLR(clr),
240 .D(d[15:0]),
241 .Q(q[15:0]));
242 // synthesis attribute HJ_SET of XLXI_5 is "XLXI_5_4"
243 endmodule
244

```

iii) wordmatch.v





```

109         .match(XLXN_2));
110     comparator XLXI_3 (.a(datacomp[55:0]),
111         .amask(wildcard[6:0]),
112         .b(datain[71:16]),
113         .match(XLXN_3));
114     comparator XLXI_4 (.a(datacomp[55:0]),
115         .amask(wildcard[6:0]),
116         .b(datain[79:24]),
117         .match(XLXN_4));
118     comparator XLXI_5 (.a(datacomp[55:0]),
119         .amask(wildcard[6:0]),
120         .b(datain[87:32]),
121         .match(XLXN_5));
122     comparator XLXI_6 (.a(datacomp[55:0]),
123         .amask(wildcard[6:0]),
124         .b(datain[95:48]),
125         .match(XLXN_6));
126     comparator XLXI_7 (.a(datacomp[55:0]),
127         .amask(wildcard[6:0]),
128         .b(datain[103:48]),
129         .match(XLXN_7));
130     comparator XLXI_8 (.a(datacomp[55:0]),
131         .amask(wildcard[6:0]),
132         .b(datain[111:56]),
133         .match(XLXN_8));
134     OR8_PXILINX_wordmatch XLXI_9 (.i0(XLXN_8),
135         .i1(XLXN_7),
136         .i2(XLXN_6),
137         .i3(XLXN_5),
138         .i4(XLXN_4),
139         .i5(XLXN_3),
140         .i6(XLXN_2),
141         .i7(XLXN_1),
142         .O(match));
143     // synthesis attribute HU_SET of XLXI_9 is "XLXI_9_0"
144 endmodule
145

```

#### iv) detect7B.v

```

E detect7B.v u X
lab3 > E detect7B.v
16 //Device: virtex2p
17 //Purpose:
18 // This verilog netlist is translated from an ECU schematic. It can be
19 // synthesized and simulated, but it should not be modified.
20 //
21 `timescale 1ns / 1ps
22 `include "busmerge.v"
23 `include "wordmatch.v"
24
25
26
27 module detect7B(ce,
28     clk,
29     hwregA,
30     match_en,
31     mrst,
32     pipel,
33     match);
34
35     input ce;
36     input clk;
37     input [63:0] hwregA;
38     input match_en;
39     input mrst;
40     input [71:0] pipel;
41     output match;
42
43     wire clr;
44     wire [71:0] pipe0;
45     wire [111:0] XLXN_1;
46     wire XLXN_18;
47     wire XLXN_23;
48     wire match_DUMMY;
49
50     assign match = match_DUMMY;
51     busmerge XLXI_1 (.da(pipe0[47:0]),
52         .db(pipe1[63:0]),
53         .q(XLXN_1[111:0]));
54     wordmatch XLXI_2 (.datacomp(hwregA[55:0]),
55         .datain(XLXN_1[111:0]),
56         .wildcard(hwregA[62:56]),
57         .match(XLXN_18));
58     FD XLXI_4 (.C(clk),
59         .D(mrst),
60         .Q(clr));
61     defparam XLXI_4.INIT = 1'b0;
62     FDCE XLXI_5 (.C(clk),
63         .ce(XLXN_23),
64         .clr(clr),
65         .D(XLXN_23),
66         .Q(match_DUMMY));
67     defparam XLXI_5.INIT = 1'b0;
68     AND3B1 XLXI_6 (.i0(match_DUMMY),
69         .i1(match_en),
70         .i2(XLXN_18),
71         .O(XLXN_23));
72     reg9B XLXI_7 (.CE(ce),
73         .clk(clk),
74         .clr(clr),
75         .d(pipe1[71:0]),
76         .q(pipe0[71:0]));
77 endmodule

```

#### v) dropfifo.v

```

# descTb.u      E dropifo.u X
lab3 > E dropifo
1 // Copyright (c) 1995-2008 Xilinx, Inc. All rights reserved.
2 //
3 //
4 //
5 // \ / \ / Vendor: Xilinx
6 // \ \ \ / Version : 10.1
7 // \ \ \ / Application : sch2verilog
8 // \ \ \ / Filename : dropifo.vf
9 // \ \ \ / Timestamp : 03/11/2016 04:27:08
10 // \ \ \ /
11 // \ \ \ /
12 // \ \ \ /
13 // \ \ \ /
14 // Command: C:\Xilinx\10.1\ISE\bin\nt\unwrapped\sch2verilog.exe -intstyle ise -family virtex2p -w "C:/Documents and Settings/student/lab3/dropifo.sch" dropifo.vf
15 // Design Name: dropifo
16 // Device: virtex2p
17 // Purpose:
18 // This verilog netlist is translated from an EES schematic. It can be
19 // synthesized and simulated, but it should not be modified.
20 //
21 timescale 1ns / 1ps
22
23 module FICE_MKILINX_dropifo(C,
24                             CE,
25                             CLA,
26                             T,
27                             Q);
28
29     input C;
30     input CE;
31     input CLA;
32     input T;
33     output Q;
34
35     wire TQ;
36     wire Q_SUMWV;
37
38     assign Q = Q_SUMWV;
39     XOR2 I_36_32 (.in(T),
40                  .Y1(Q_SUMWV),
41                  .O(TQ));
42     FDCE I_36_35 (.C(C),
43                  .CE(CE),
44                  .CLK(CLA),
45                  .D(TQ),
46                  .Q(Q_SUMWV));
47     // synthesis attribute HLOC of I_36_35 is "X0V0"
48     defparam I_36_35.INIT = 1'b0;
49 endmodule
50 timescale 1ns / 1ps
51
52 module CBCEI_MKILINX_dropifo(C,
53                             CE,
54                             CLA,
55                             CEO,
56                             Q,
57                             TC);
58
59     input C;
60     input CE;
61     input CLA;
62     output CEO;
63     output [7:0] Q;
64     output TC;
65
66     wire T2;
67     wire T3;
68     wire T4;
69     wire T5;
70     wire T6;
71     wire T7;
72     wire XLN_1;
73     wire [7:0] Q_SUMWV;
74
75     assign Q[7:0] = Q_SUMWV[7:0];
76     assign TC = TC_SUMWV;
77     FICE_MKILINX_dropifo I_Q0 (.C(C),
78                              .CE(CE),
79                              .CLA(CLA),
80                              .T(XLN_1),
81                              .Q(Q_SUMWV[0]));
82     // synthesis attribute HU_SET of I_Q0 is "I_Q0_0"
83     FICE_MKILINX_dropifo I_Q1 (.C(C),
84                              .CE(CE),
85                              .CLA(CLA),
86                              .T(Q_SUMWV[0]),
87                              .Q(Q_SUMWV[1]));
88     // synthesis attribute HU_SET of I_Q1 is "I_Q1_1"
89     FICE_MKILINX_dropifo I_Q2 (.C(C),
90                              .CE(CE),
91                              .CLA(CLA),
92                              .T(T2),
93                              .Q(Q_SUMWV[2]));
94     // synthesis attribute HU_SET of I_Q2 is "I_Q2_3"
95     FICE_MKILINX_dropifo I_Q3 (.C(C),
96                              .CE(CE),
97                              .CLA(CLA),
98                              .T(T3),
99                              .Q(Q_SUMWV[3]));
100    // synthesis attribute HU_SET of I_Q3 is "I_Q3_4"
101    FICE_MKILINX_dropifo I_Q4 (.C(C),
102                             .CE(CE),
103                             .CLA(CLA),
104                             .T(T4),
105                             .Q(Q_SUMWV[4]));
106    // synthesis attribute HU_SET of I_Q4 is "I_Q4_5"
107    FICE_MKILINX_dropifo I_Q5 (.C(C),
108                             .CE(CE),
109                             .CLA(CLA),
110                             .T(T5),
111                             .Q(Q_SUMWV[5]));
112    // synthesis attribute HU_SET of I_Q5 is "I_Q5_2"
113    FICE_MKILINX_dropifo I_Q6 (.C(C),
114                             .CE(CE),
115                             .CLA(CLA),
116                             .T(T6),
117                             .Q(Q_SUMWV[6]));
118    // synthesis attribute HU_SET of I_Q6 is "I_Q6_1"
119    FICE_MKILINX_dropifo I_Q7 (.C(C),
120                             .CE(CE),
121                             .CLA(CLA),
122                             .T(T7),
123                             .Q(Q_SUMWV[7]));
124    // synthesis attribute HU_SET of I_Q7 is "I_Q7_8"
125    AND2 I_36_1 (.I0(Q_SUMWV[7]),
126                .I1(Q_SUMWV[4]),
127                .Y1(Q_SUMWV[8]),
128                .Y2(Q_SUMWV[9]),
129                .Y3(Q_SUMWV[10]),
130                .Y4(T4));
131    AND2 I_36_2 (.I0(Q_SUMWV[4]),
132                .I1(T4),
133                .O(T5));
134    AND3 I_36_11 (.I0(Q_SUMWV[5]),
135                 .I1(Q_SUMWV[4]),
136                 .I2(T4),
137                 .O(T6));
138    AND4 I_36_15 (.I0(Q_SUMWV[3]),
139                 .I1(Q_SUMWV[2]),
140                 .I2(Q_SUMWV[1]),
141                 .I3(Q_SUMWV[0]),
142                 .O(T4));
143    ORC I_36_16 (.p0(XLN_1));

```

```

# descTb.u      E dropifo.u X
lab3 > E dropifo
71     wire T7;
72     wire XLN_1;
73     wire [7:0] Q_SUMWV;
74     wire TC_SUMWV;
75
76     assign Q[7:0] = Q_SUMWV[7:0];
77     assign TC = TC_SUMWV;
78     FICE_MKILINX_dropifo I_Q0 (.C(C),
79                              .CE(CE),
80                              .CLA(CLA),
81                              .T(XLN_1),
82                              .Q(Q_SUMWV[0]));
83     // synthesis attribute HU_SET of I_Q0 is "I_Q0_0"
84     FICE_MKILINX_dropifo I_Q1 (.C(C),
85                              .CE(CE),
86                              .CLA(CLA),
87                              .T(Q_SUMWV[0]),
88                              .Q(Q_SUMWV[1]));
89     // synthesis attribute HU_SET of I_Q1 is "I_Q1_1"
90     FICE_MKILINX_dropifo I_Q2 (.C(C),
91                              .CE(CE),
92                              .CLA(CLA),
93                              .T(T2),
94                              .Q(Q_SUMWV[2]));
95     // synthesis attribute HU_SET of I_Q2 is "I_Q2_3"
96     FICE_MKILINX_dropifo I_Q3 (.C(C),
97                              .CE(CE),
98                              .CLA(CLA),
99                              .T(T3),
100                             .Q(Q_SUMWV[3]));
101    // synthesis attribute HU_SET of I_Q3 is "I_Q3_4"
102    FICE_MKILINX_dropifo I_Q4 (.C(C),
103                             .CE(CE),
104                             .CLA(CLA),
105                             .T(T4),
106                             .Q(Q_SUMWV[4]));
107    // synthesis attribute HU_SET of I_Q4 is "I_Q4_5"
108    FICE_MKILINX_dropifo I_Q5 (.C(C),
109                             .CE(CE),
110                             .CLA(CLA),
111                             .T(T5),
112                             .Q(Q_SUMWV[5]));
113    // synthesis attribute HU_SET of I_Q5 is "I_Q5_2"
114    FICE_MKILINX_dropifo I_Q6 (.C(C),
115                             .CE(CE),
116                             .CLA(CLA),
117                             .T(T6),
118                             .Q(Q_SUMWV[6]));
119    // synthesis attribute HU_SET of I_Q6 is "I_Q6_1"
120    FICE_MKILINX_dropifo I_Q7 (.C(C),
121                             .CE(CE),
122                             .CLA(CLA),
123                             .T(T7),
124                             .Q(Q_SUMWV[7]));
125    // synthesis attribute HU_SET of I_Q7 is "I_Q7_8"
126    AND2 I_36_1 (.I0(Q_SUMWV[7]),
127                .I1(Q_SUMWV[4]),
128                .Y1(Q_SUMWV[8]),
129                .Y2(Q_SUMWV[9]),
130                .Y3(Q_SUMWV[10]),
131                .Y4(T4));
132    AND2 I_36_2 (.I0(Q_SUMWV[4]),
133                .I1(T4),
134                .O(T5));
135    AND3 I_36_11 (.I0(Q_SUMWV[5]),
136                 .I1(Q_SUMWV[4]),
137                 .I2(T4),
138                 .O(T6));
139    AND4 I_36_15 (.I0(Q_SUMWV[3]),
140                 .I1(Q_SUMWV[2]),
141                 .I2(Q_SUMWV[1]),
142                 .I3(Q_SUMWV[0]),
143                 .O(T4));
144    ORC I_36_16 (.p0(XLN_1));

```

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

Verilog

```

lab3 > E dropflow
E dropflow.v
module C88C1E_MXILINK_dropifto(C,
    input C;
    input CE;
    input CLR;
    input [7:0] D;
    input L;
    output CEG;
    output [7:0] Q;
    output TC;

    wire OR_CE_L;
    wire T2;
    wire T3;
    wire T4;
    wire T5;
    wire T6;
    wire T7;
    wire XL_ML_1;
    wire [7:8] Q_DUMMY;
    wire TC_DUMMY;

    assign Q[7:8] = Q_DUMMY[7:8];
    assign TC = TC_DUMMY;

    FTCLEX_MXILINK_dropifto I_Q0 (.C(C),
        .CE(OR_CE_L),
        .CLR(CLR),
        .D(D[0]),
        .L(L),
        .T(TX_ML_1),
        .Q(Q_DUMMY[0]));
    // synthesis attribute HU_SEY of I_Q0 is "I_Q0_9"
    FTCLEX_MXILINK_dropifto I_Q1 (.C(C),
        .CE(OR_CE_L),
        .CLR(CLR),
        .D(D[1]),
        .L(L),
        .T(Q_DUMMY[0]),
        .Q(Q_DUMMY[1]));
    // synthesis attribute HU_SEY of I_Q1 is "I_Q1_10"
    FTCLEX_MXILINK_dropifto I_Q2 (.C(C),
        .CE(OR_CE_L),
        .CLR(CLR),
        .D(D[2]),
        .L(L),
        .T(T2),
        .Q(Q_DUMMY[2]));
    // synthesis attribute HU_SEY of I_Q2 is "I_Q2_11"
    FTCLEX_MXILINK_dropifto I_Q3 (.C(C),
        .CE(OR_CE_L),
        .CLR(CLR),
        .D(D[3]),
        .L(L),
        .T(T3),
        .Q(Q_DUMMY[3]));
    // synthesis attribute HU_SEY of I_Q3 is "I_Q3_12"
    FTCLEX_MXILINK_dropifto I_Q4 (.C(C),
        .CE(OR_CE_L),
        .CLR(CLR),
        .D(D[4]),
        .L(L),
        .T(T4),
        .Q(Q_DUMMY[4]));
    // synthesis attribute HU_SEY of I_Q4 is "I_Q4_13"
    FTCLEX_MXILINK_dropifto I_Q5 (.C(C),
        .CE(OR_CE_L),
        .CLR(CLR),
        .D(D[5]),
        .L(L),
        .T(T5),
        .Q(Q_DUMMY[5]));
    // synthesis attribute HU_SEY of I_Q5 is "I_Q5_14"
    FTCLEX_MXILINK_dropifto I_Q6 (.C(C),
        .CE(OR_CE_L),
        .CLR(CLR),
        .D(D[6]),
        .L(L),
        .T(T6),
        .Q(Q_DUMMY[6]));
    // synthesis attribute HU_SEY of I_Q6 is "I_Q6_15"
    FTCLEX_MXILINK_dropifto I_Q7 (.C(C),
        .CE(OR_CE_L),
        .CLR(CLR),
        .D(D[7]),
        .L(L),
        .T(T7),
        .Q(Q_DUMMY[7]));
    // synthesis attribute HU_SEY of I_Q7 is "I_Q7_16"
    AND3 I_36_8 (.I0(Q_DUMMY[5]),
        .I1(Q_DUMMY[4]),
        .I2(T4),
        .O(T6));
    AND2 I_36_11 (.I0(Q_DUMMY[4]),
        .I1(T4),
        .O(T5));
    VCC I_36_12 (.P(XL_ML_1));
    AND2 I_36_19 (.I0(Q_DUMMY[1]),
        .I1(Q_DUMMY[0]),
        .O(T2));
    AND3 I_36_21 (.I0(Q_DUMMY[2]),
        .I1(Q_DUMMY[1]),
        .I2(Q_DUMMY[0]),
        .O(T3));
    AND4 I_36_23 (.I0(Q_DUMMY[3]),
        .I1(Q_DUMMY[2]),
        .I2(Q_DUMMY[1]),
        .I3(Q_DUMMY[0]),
        .O(T4));
    AND4 I_36_25 (.I0(Q_DUMMY[6]),
        .I1(Q_DUMMY[5]),
        .I2(Q_DUMMY[4]),
        .I3(T4),
        .O(T7));
    AND5 I_36_29 (.I0(Q_DUMMY[7]),
        .I1(Q_DUMMY[6]),
        .I2(Q_DUMMY[5]),
        .I3(Q_DUMMY[4]),
        .I4(T4),
        .O(T7));
    AND2 I_36_33 (.I0(CE),
        .I1(TC_DUMMY),
        .O(CE0));
    OR2 I_36_40 (.I0(CE),
        .I1(L),
        .O(OR_CE_L));
endmodule

timescale 1ns / 1ps

module FORCE_MXILINK_dropifto(C,
    CE,
    CLR,
    D,
    Q;

    input C;
    input CE;
    input CLR;
    input [7:8] D;
    output [7:8] Q;
endmodule

```

```

lab3 > E dropflow
E dropflow.v
module C88C1E_MXILINK_dropifto(C,
    input C;
    input CE;
    input CLR;
    input [7:8] D;
    input L;
    output CEG;
    output [7:0] Q;
    output TC;

    wire OR_CE_L;
    wire T2;
    wire T3;
    wire T4;
    wire T5;
    wire T6;
    wire T7;
    wire XL_ML_1;
    wire [7:8] Q_DUMMY;
    wire TC_DUMMY;

    assign Q[7:8] = Q_DUMMY[7:8];
    assign TC = TC_DUMMY;

    FTCLEX_MXILINK_dropifto I_Q0 (.C(C),
        .CE(OR_CE_L),
        .CLR(CLR),
        .D(D[0]),
        .L(L),
        .T(TX_ML_1),
        .Q(Q_DUMMY[0]));
    // synthesis attribute HU_SEY of I_Q0 is "I_Q0_9"
    FTCLEX_MXILINK_dropifto I_Q1 (.C(C),
        .CE(OR_CE_L),
        .CLR(CLR),
        .D(D[1]),
        .L(L),
        .T(Q_DUMMY[0]),
        .Q(Q_DUMMY[1]));
    // synthesis attribute HU_SEY of I_Q1 is "I_Q1_10"
    FTCLEX_MXILINK_dropifto I_Q2 (.C(C),
        .CE(OR_CE_L),
        .CLR(CLR),
        .D(D[2]),
        .L(L),
        .T(T2),
        .Q(Q_DUMMY[2]));
    // synthesis attribute HU_SEY of I_Q2 is "I_Q2_11"
    FTCLEX_MXILINK_dropifto I_Q3 (.C(C),
        .CE(OR_CE_L),
        .CLR(CLR),
        .D(D[3]),
        .L(L),
        .T(T3),
        .Q(Q_DUMMY[3]));
    // synthesis attribute HU_SEY of I_Q3 is "I_Q3_12"
    FTCLEX_MXILINK_dropifto I_Q4 (.C(C),
        .CE(OR_CE_L),
        .CLR(CLR),
        .D(D[4]),
        .L(L),
        .T(T4),
        .Q(Q_DUMMY[4]));
    // synthesis attribute HU_SEY of I_Q4 is "I_Q4_13"
    FTCLEX_MXILINK_dropifto I_Q5 (.C(C),
        .CE(OR_CE_L),
        .CLR(CLR),
        .D(D[5]),
        .L(L),
        .T(T5),
        .Q(Q_DUMMY[5]));
    // synthesis attribute HU_SEY of I_Q5 is "I_Q5_14"
    FTCLEX_MXILINK_dropifto I_Q6 (.C(C),
        .CE(OR_CE_L),
        .CLR(CLR),
        .D(D[6]),
        .L(L),
        .T(T6),
        .Q(Q_DUMMY[6]));
    // synthesis attribute HU_SEY of I_Q6 is "I_Q6_15"
    FTCLEX_MXILINK_dropifto I_Q7 (.C(C),
        .CE(OR_CE_L),
        .CLR(CLR),
        .D(D[7]),
        .L(L),
        .T(T7),
        .Q(Q_DUMMY[7]));
    // synthesis attribute HU_SEY of I_Q7 is "I_Q7_16"
    AND3 I_36_8 (.I0(Q_DUMMY[5]),
        .I1(Q_DUMMY[4]),
        .I2(T4),
        .O(T6));
    AND2 I_36_11 (.I0(Q_DUMMY[4]),
        .I1(T4),
        .O(T5));
    VCC I_36_12 (.P(XL_ML_1));
    AND2 I_36_19 (.I0(Q_DUMMY[1]),
        .I1(Q_DUMMY[0]),
        .O(T2));
    AND3 I_36_21 (.I0(Q_DUMMY[2]),
        .I1(Q_DUMMY[1]),
        .I2(Q_DUMMY[0]),
        .O(T3));
    AND4 I_36_23 (.I0(Q_DUMMY[3]),
        .I1(Q_DUMMY[2]),
        .I2(Q_DUMMY[1]),
        .I3(Q_DUMMY[0]),
        .O(T4));
    AND4 I_36_25 (.I0(Q_DUMMY[6]),
        .I1(Q_DUMMY[5]),
        .I2(Q_DUMMY[4]),
        .I3(T4),
        .O(T7));
    AND5 I_36_29 (.I0(Q_DUMMY[7]),
        .I1(Q_DUMMY[6]),
        .I2(Q_DUMMY[5]),
        .I3(Q_DUMMY[4]),
        .I4(T4),
        .O(T7));
    AND2 I_36_33 (.I0(CE),
        .I1(TC_DUMMY),
        .O(CE0));
    OR2 I_36_40 (.I0(CE),
        .I1(L),
        .O(OR_CE_L));
endmodule

timescale 1ns / 1ps

module FORCE_MXILINK_dropifto(C,
    CE,
    CLR,
    D,
    Q;

    input C;
    input CE;
    input CLR;
    input [7:8] D;
    output [7:8] Q;
endmodule

```

```

E: detect78.v u      E: dropfifo.v u X
lab3 > E: dropfifo.v
651 module FORCE_MXLINK_dropfifo(C,
652     D,
653     Q);
654
655     input C;
656     input CE;
657     input CLR;
658     input [7:0] D;
659     output [7:0] Q;
660
661
662     FORCE_1_Q0 (.C(C),
663         .CE(CE),
664         .CLK(CLK),
665         .D(Q[0]));
666     defparam 1_Q0.INIT = 1'b0;
667     FORCE_1_Q1 (.C(C),
668         .CE(CE),
669         .CLK(CLK),
670         .D(Q[1]));
671     defparam 1_Q1.INIT = 1'b0;
672     FORCE_1_Q2 (.C(C),
673         .CE(CE),
674         .CLK(CLK),
675         .D(Q[2]));
676     defparam 1_Q2.INIT = 1'b0;
677     FORCE_1_Q3 (.C(C),
678         .CE(CE),
679         .CLK(CLK),
680         .D(Q[3]));
681     defparam 1_Q3.INIT = 1'b0;
682     FORCE_1_Q4 (.C(C),
683         .CE(CE),
684         .CLK(CLK),
685         .D(Q[4]));
686     defparam 1_Q4.INIT = 1'b0;
687     FORCE_1_Q5 (.C(C),
688         .CE(CE),
689         .CLK(CLK),
690         .D(Q[5]));
691     defparam 1_Q5.INIT = 1'b0;
692     FORCE_1_Q6 (.C(C),
693         .CE(CE),
694         .CLK(CLK),
695         .D(Q[6]));
696     defparam 1_Q6.INIT = 1'b0;
697     FORCE_1_Q7 (.C(C),
698         .CE(CE),
699         .CLK(CLK),
700         .D(Q[7]));
701     defparam 1_Q7.INIT = 1'b0;
702
703 endmodule
704
705 timescale 1ns / 1ps
706
707 module dropfifo(clk,
708     drop_pkt,
709     fiforead,
710     fifowrite,
711     firstword,
712     in_fifo,
713     lastword,
714     rst,
715     out_fifo,
716     valid_data);
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999

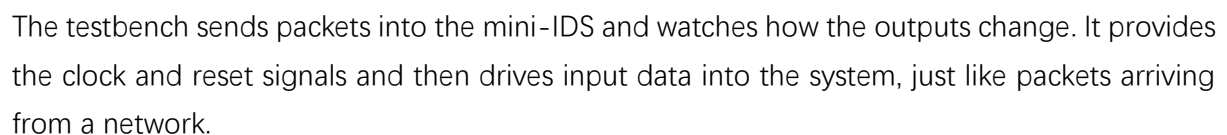
```

```

E: detect78.v u      E: dropfifo.v u X
lab3 > E: dropfifo.v
965 module dropfifo(clk,
966
967     wire [7:0] raddr;
968     wire [7:0] waddr;
969     wire XLNM_3;
970     wire XLNM_24;
971     wire XLNM_25;
972     wire XLNM_27;
973     wire XLNM_31;
974     wire [7:0] XLNM_34;
975     wire XLNM_40;
976     wire XLNM_41;
977     wire XLNM_44;
978     wire XLNM_47;
979     wire [7:0] XLNM_48;
980     wire XLNM_49;
981
982     dual_core_memory_element XLXI_1 (.addr(waddr[7:0]),
983         .addr(raddr[7:0]),
984         .clk(clk),
985         .clkb(clk),
986         .dina(xlNM_48[7:0]),
987         .diwb(),
988         .wen(xlNM_47),
989         .wdb(),
990         .douta(),
991         .doutb(out_fifo[7:0]));
992
993     FD XLXI_2 (.C(clk),
994         .D(fiforead),
995         .Q(XLNM_23));
996     defparam XLXI_2.INIT = 1'b0;
997     FD XLXI_3 (.C(clk),
998         .D(lastword),
999         .Q(XLNM_24));
1000     defparam XLXI_3.INIT = 1'b0;
1001     FD XLXI_4 (.C(clk),
1002         .D(fifowrite),
1003         .Q(XLNM_47));
1004     defparam XLXI_4.INIT = 1'b0;
1005     OR2 XLXI_5 (.B(XLNM_24),
1006         .I1(XLNM_23),
1007         .O(XLNM_26));
1008     FORCE_MXLINK_dropfifo XLXI_6 (.C(clk),
1009         .CE(XLNM_27),
1010         .CLK(rst),
1011         .D(waddr[7:0]),
1012         .Q(XLNM_24[7:0]));
1013     // synthesis attribute HU_SET of XLXI_6 is "XLXI_6_13"
1014     AND2B1 XLXI_7 (.B(XLNM_31),
1015         .I1(XLNM_26),
1016         .O(XLNM_27));
1017     FD XLXI_8 (.C(clk),
1018         .D(drop_pkt),
1019         .Q(XLNM_31));
1020     defparam XLXI_8.INIT = 1'b0;
1021     CHCIE_MXLINK_dropfifo XLXI_9 (.C(clk),
1022         .CE(XLNM_47),
1023         .CLK(rst),
1024         .D(XLNM_34[7:0]),
1025         .I(XLNM_31),
1026         .CFD(),
1027         .Q(waddr[7:0]),
1028         .TC());
1029     // synthesis attribute HU_SET of XLXI_9 is "XLXI_9_18"
1030     COMP_MXLINK_dropfifo XLXI_10 (.A(waddr[7:0]),
1031         .B(raddr[7:0]),
1032         .EQ(XLNM_41));
1033     // synthesis attribute HU_SET of XLXI_10 is "XLXI_10_19"
1034     COMP_MXLINK_dropfifo XLXI_11 (.A(raddr[7:0]),
1035         .B(XLNM_34[7:0]),
1036         .EQ(XLNM_40));
1037     // synthesis attribute HU_SET of XLXI_11 is "XLXI_11_20"
1038     CHCIE_MXLINK_dropfifo XLXI_12 (.C(clk),
1039         .CE(XLNM_44),
1040         .CLK(rst),
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999

```

9. Simulate the mini-IDS using the `ids_tb.tbw` testbench. Run the testbench and take a screen shot. Describe what the testbench does and how it shows that the mini-IDS is functioning.



Here I show the simulation waveform. We can see that when input data is written and accepted, the IDS processes it correctly. When no attack pattern is present, the output data is forwarded normally. When a match signal is triggered, the corresponding packet is dropped, which confirms that the IDS logic is working as intended.

10. Explain the pattern matching algorithm in the report

The mini-IDS checks incoming data to see if a specific pattern appears. Data arrives continuously, so it is first stored and then checked in small sections. A 7-byte window moves across the data, and each window is compared with the pattern. If any window matches, the system reports a detection. If an attack pattern is detected, the entire packet is dropped. Otherwise, the packet is forwarded normally.

11. Include the answers to your lab in this report.

- i) What is the purpose of AMASK[6:0]?

AMASK [6:0] is used to decide which bytes need to be compared. Each bit in AMASK controls one byte. If a bit is set, that byte is ignored during comparison. This makes the pattern matching more flexible.

ii) What exactly does busmerge.v do?

The busmerge module joins two data buses into one larger bus. It combines old data and new data so they can be checked together. This helps the system compare longer data sequences.

iii) What do the comp8 modules do in this schematic?

The comp8 modules compare two 8-bit values. Each comp8 checks whether one byte of input data matches the pattern. Several comp8 modules are used together to compare multiple bytes.

iv) What is the purpose of dual9Bmem in dropfifo.sch?

The dual9Bmem module is a 9-byte memory used as a FIFO buffer. It temporarily stores packet data while allowing read and write at the same time. When a match is detected, the system can stop the packet from being forwarded.