

EE lab3 report: mini-IDS

Name: Chenyang Zhang

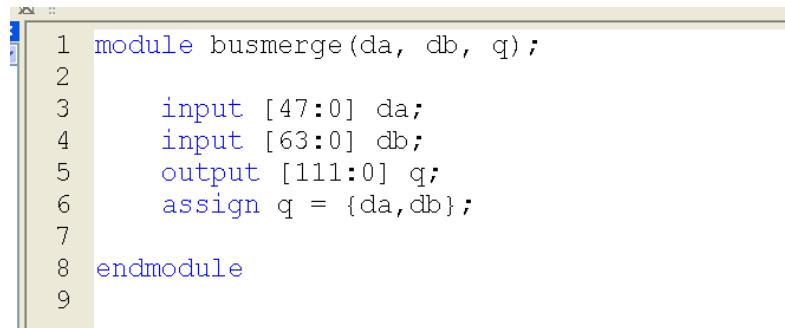
GitHub Link: <https://github.com/zhangc74/ee533-lab3>

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

### Schematic Capture

1. Create busmerge.v

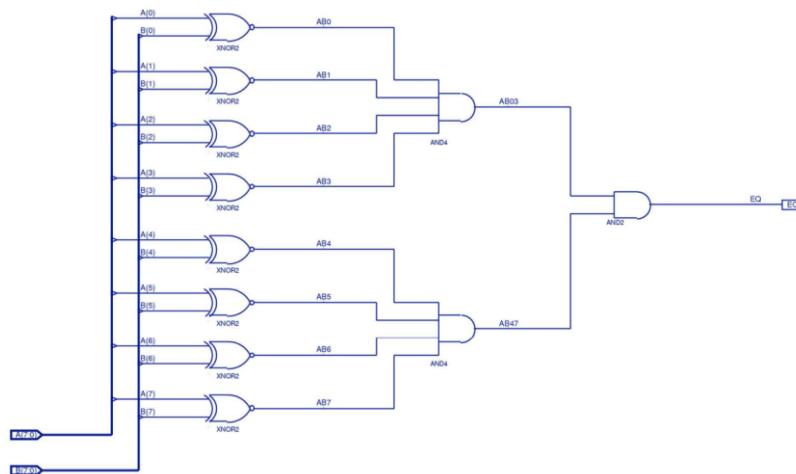
Used to spell 48-bit + 64-bit into 112-bit, it is a "splicer" for the subsequent wordmatch



```
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

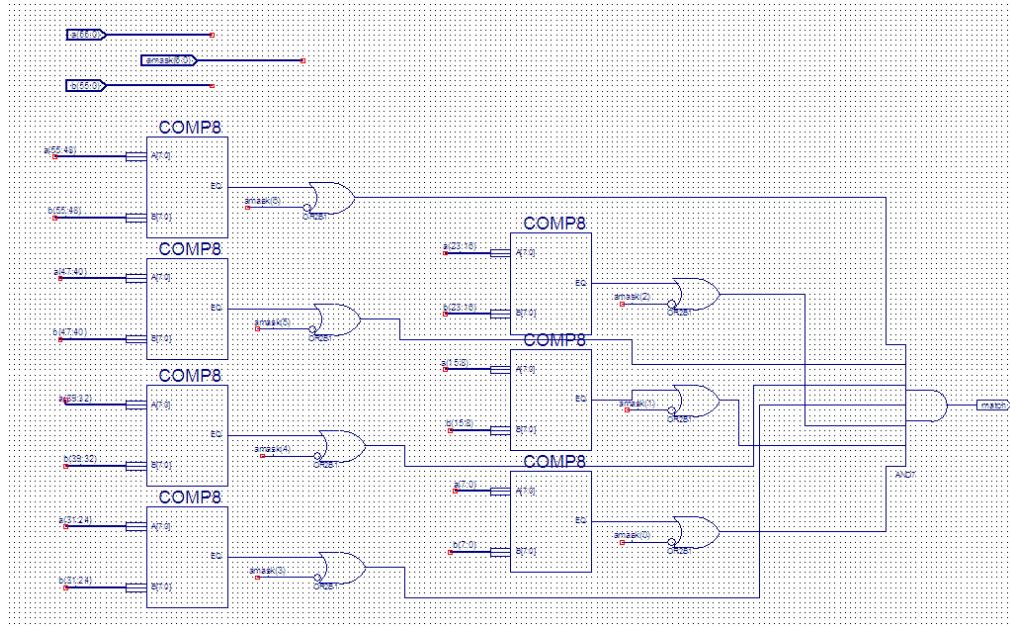
The 8-bit comparator is a built-in Xilinx component and is included for reference, so I do not need to re-implement it.      input: A[7:0], B[7:0]      output: EQ (equal = 1)



3. Draw comparator.sch

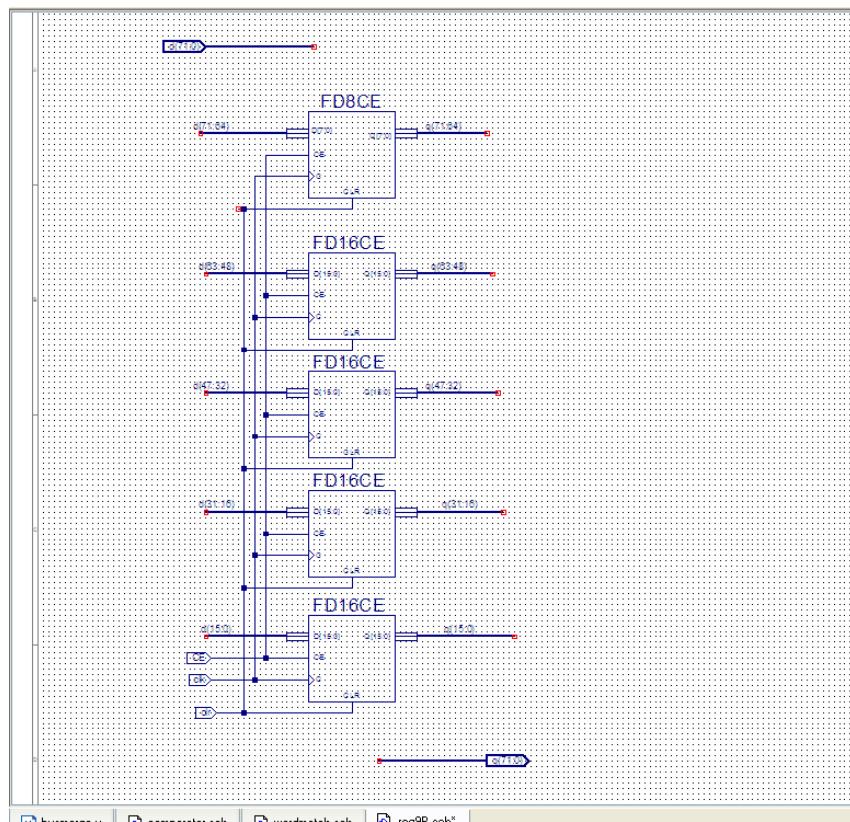
Used to compare a 56-bit input. It is built using seven 8-bit comparators, where each comparator checks one byte of the data. The AMASK[6:0] signal allows some bytes to be ignored during comparison. The output match becomes 1 only when the required bytes

match the pattern.



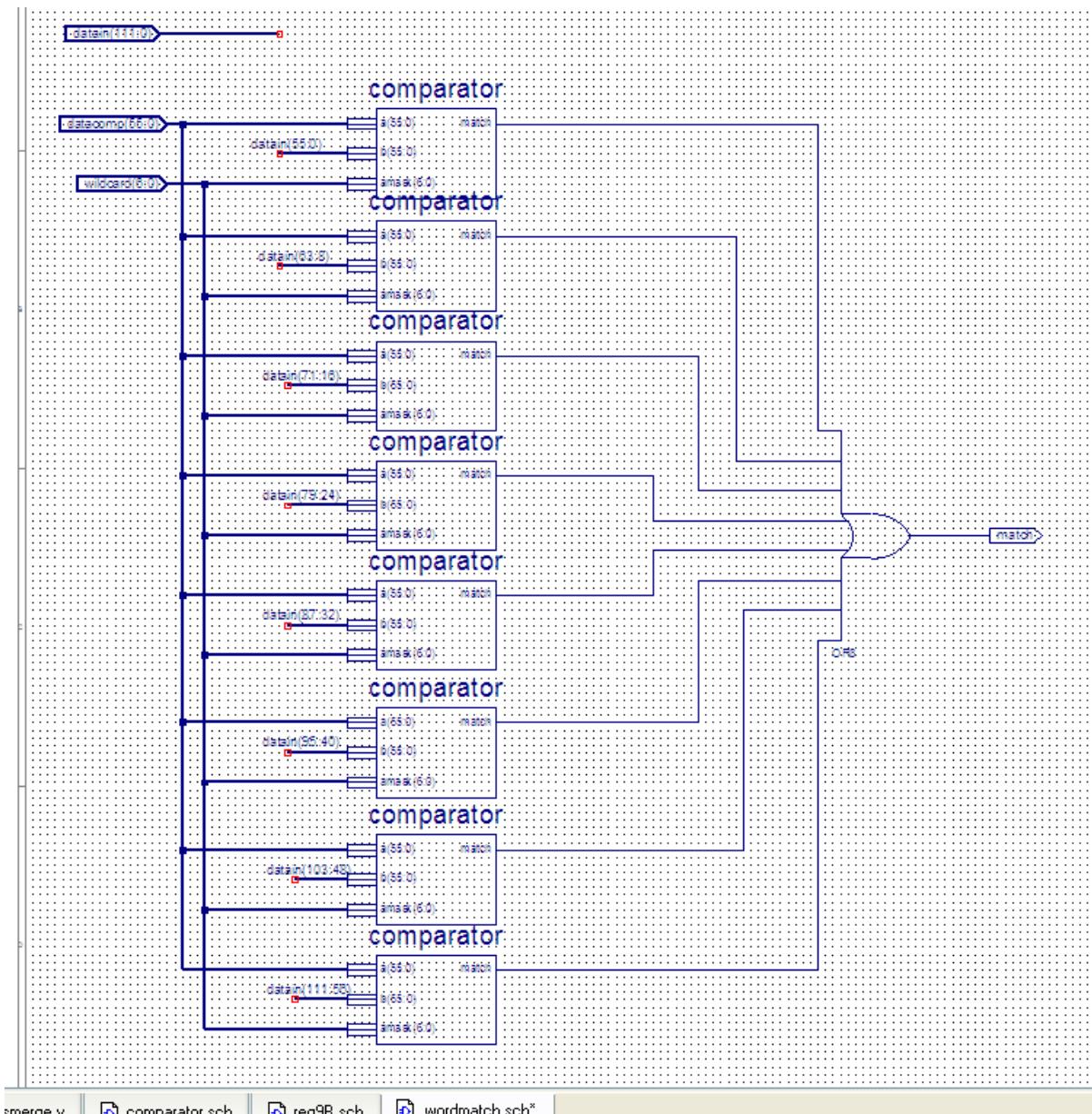
#### 4. Draw reg9B.sch

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.



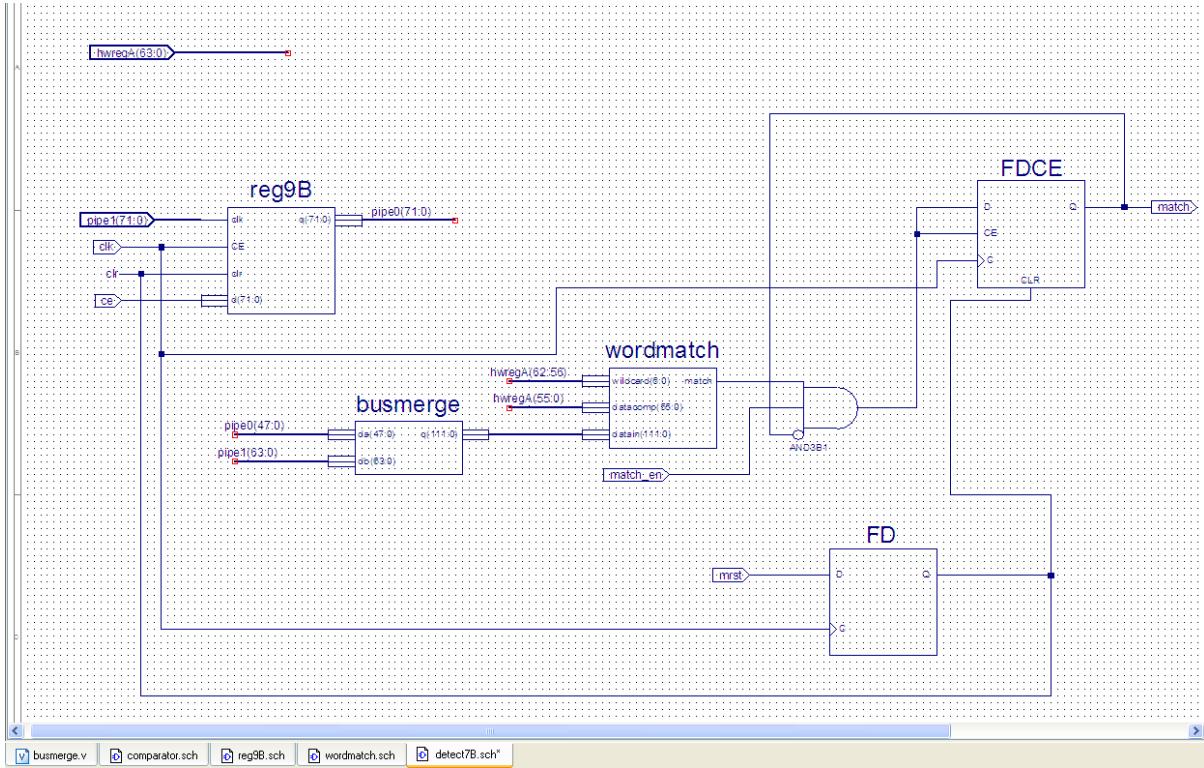
## 5. Draw wordmatch.sch

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

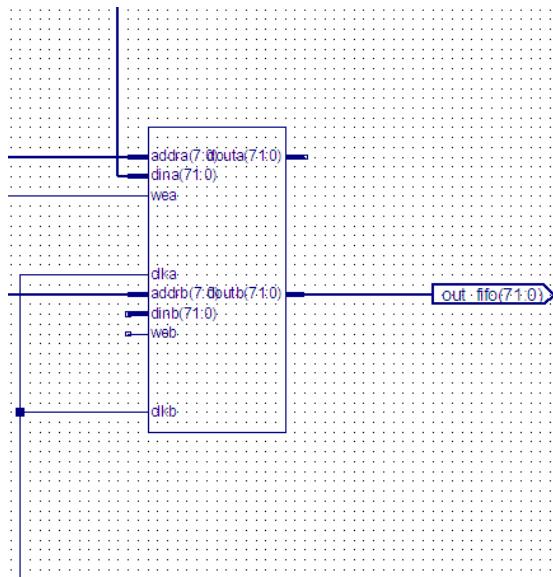
Used to connect the pattern matching logic with the data flow control. It takes packet data, combines different data fields using busmerge, and sends the result to the wordmatch module for comparison. When a match is detected, control signals are generated and stored using flip-flops to indicate that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

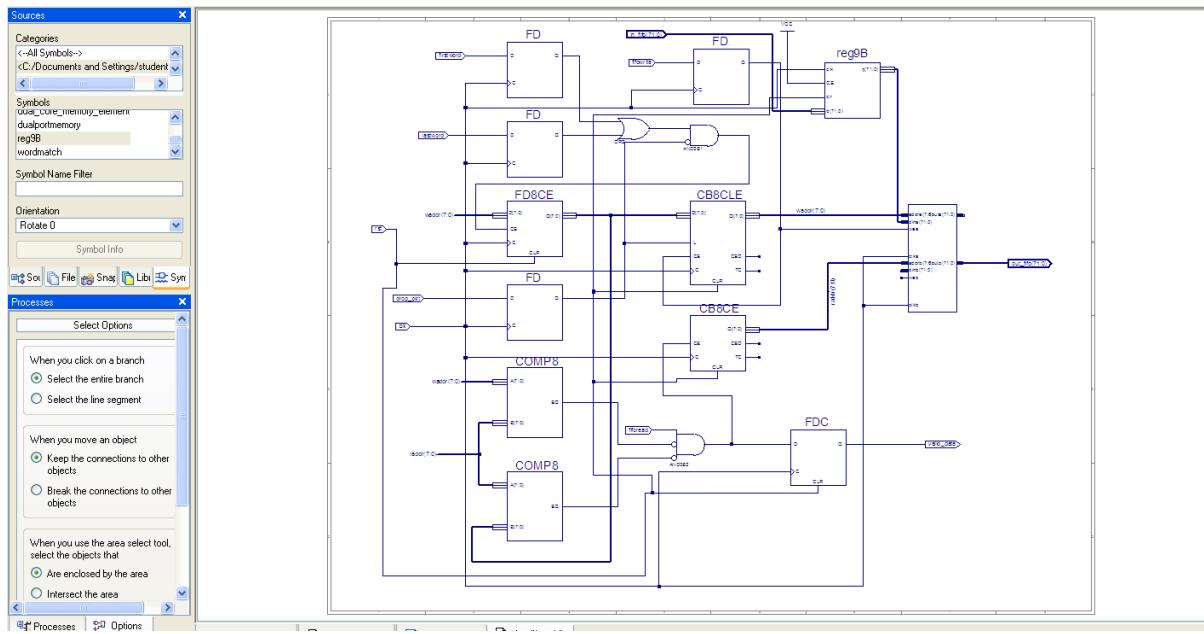


## 7. Draw dropfifo.sch

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

```
# comparator.v
`timescale 1ns / 1ns

module COMP9_8WLINK_comparator(A,
  B,
  C,
  D,
  E,
  F,
  G,
  H,
  I,
  J,
  K,
  L,
  M,
  N,
  O,
  P,
  Q,
  R,
  S,
  T,
  U,
  V,
  W,
  X,
  Y,
  Z);
  input [7:0] A;
  input [7:0] B;
  output [8:0] C;
  output [8:0] D;
  output [8:0] E;
  output [8:0] F;
  output [8:0] G;
  output [8:0] H;
  output [8:0] I;
  output [8:0] J;
  output [8:0] K;
  output [8:0] L;
  output [8:0] M;
  output [8:0] N;
  output [8:0] O;
  output [8:0] P;
  output [8:0] Q;
  output [8:0] R;
  output [8:0] S;
  output [8:0] T;
  output [8:0] U;
  output [8:0] V;
  output [8:0] W;
  output [8:0] X;
  output [8:0] Y;
  output [8:0] Z;
endmodule

// Copyright (c) 1995-2008 Xilinx, Inc. All rights reserved.
// Univerilog Verilog Model for Xilinx Device
// Version : 10.1
// Application : schwierig
// Date : 01/31/2006 02:07:41
// File : comparator.vf
// Timestamp : 01/31/2006 02:07:41
// Component : XILINX10M11USVB161011unwrappedischwierig.exe -intitle lse -family virtex2p -w "C:/Documents and Settings/student/lab1/comparator.vf" comparator.vf
// Design Name: comparator
// Device: virtex2p
// Processor: 
// This verilog netlist is translated from an ECS schematic. It can be
// synthesized and simulated, but it should not be modified.
// 
//timescale 1ns / 1ns

//module COMP9_8WLINK_comparator(A,
//  B,
//  C,
//  D,
//  E,
//  F,
//  G,
//  H,
//  I,
//  J,
//  K,
//  L,
//  M,
//  N,
//  O,
//  P,
//  Q,
//  R,
//  S,
//  T,
//  U,
//  V,
//  W,
//  X,
//  Y,
//  Z);
//  input [7:0] A;
//  input [7:0] B;
//  output [8:0] C;
//  output [8:0] D;
//  output [8:0] E;
//  output [8:0] F;
//  output [8:0] G;
//  output [8:0] H;
//  output [8:0] I;
//  output [8:0] J;
//  output [8:0] K;
//  output [8:0] L;
//  output [8:0] M;
//  output [8:0] N;
//  output [8:0] O;
//  output [8:0] P;
//  output [8:0] Q;
//  output [8:0] R;
//  output [8:0] S;
//  output [8:0] T;
//  output [8:0] U;
//  output [8:0] V;
//  output [8:0] W;
//  output [8:0] X;
//  output [8:0] Y;
//  output [8:0] Z;
//endmodule

// Copyright (c) 1995-2008 Xilinx, Inc. All rights reserved.
// Univerilog Verilog Model for Xilinx Device
// Version : 10.1
// Application : schwierig
// Date : 01/31/2006 02:07:41
// File : comparator.vf
// Timestamp : 01/31/2006 02:07:41
// Component : XILINX10M11USVB161011unwrappedischwierig.exe -intitle lse -family virtex2p -w "C:/Documents and Settings/student/lab1/comparator.vf" comparator.vf
// Design Name: comparator
// Device: virtex2p
// Processor: 
// This verilog netlist is translated from an ECS schematic. It can be
// synthesized and simulated, but it should not be modified.
// 
//timescale 1ns / 1ns

//module COMP9_8WLINK_comparator(A,
//  B,
//  C,
//  D,
//  E,
//  F,
//  G,
//  H,
//  I,
//  J,
//  K,
//  L,
//  M,
//  N,
//  O,
//  P,
//  Q,
//  R,
//  S,
//  T,
//  U,
//  V,
//  W,
//  X,
//  Y,
//  Z);
//  input [7:0] A;
//  input [7:0] B;
//  output [8:0] C;
//  output [8:0] D;
//  output [8:0] E;
//  output [8:0] F;
//  output [8:0] G;
//  output [8:0] H;
//  output [8:0] I;
//  output [8:0] J;
//  output [8:0] K;
//  output [8:0] L;
//  output [8:0] M;
//  output [8:0] N;
//  output [8:0] O;
//  output [8:0] P;
//  output [8:0] Q;
//  output [8:0] R;
//  output [8:0] S;
//  output [8:0] T;
//  output [8:0] U;
//  output [8:0] V;
//  output [8:0] W;
//  output [8:0] X;
//  output [8:0] Y;
//  output [8:0] Z;
//endmodule
```

```

# comparators
001 > # comparators
002   module C0P8_MKLINX_comparator(A,
003     .A0(A[36:40]),
004     .I1(I1),
005     .I2(I2),
006     .I3(I3),
007     .I4(I4),
008     .I5(I5),
009     .I6(I6),
010     .I7(I7),
011     .I8(I8),
012     .I9(I9),
013     .I10(I10),
014     .I11(I11),
015     .I12(I12),
016     .I13(I13),
017     .I14(I14),
018     .I15(I15),
019     .I16(I16),
020     .I17(I17),
021     .I18(I18),
022     .I19(I19),
023     .I20(I20),
024     .I21(I21),
025     .I22(I22),
026     .I23(I23),
027     .I24(I24),
028     .I25(I25),
029     .I26(I26),
030     .I27(I27),
031     .I28(I28),
032     .I29(I29),
033     .I30(I30),
034     .I31(I31),
035     .I32(I32),
036     .I33(I33),
037     .I34(I34),
038     .I35(I35),
039     .I36(I36),
040     .I37(I37),
041     .I38(I38),
042     .I39(I39),
043     .I40(I40),
044     .I41(I41),
045     .I42(I42),
046     .I43(I43),
047     .I44(I44),
048     .I45(I45),
049     .I46(I46),
050     .I47(I47),
051     .I48(I48),
052     .I49(I49),
053     .I50(I50),
054     .I51(I51),
055     .I52(I52),
056     .I53(I53),
057     .I54(I54),
058     .I55(I55),
059     .I56(I56),
060     .I57(I57),
061     .I58(I58),
062     .I59(I59),
063     .I60(I60),
064     .I61(I61),
065     .I62(I62),
066     .I63(I63),
067     .I64(I64),
068     .I65(I65),
069     .I66(I66),
070     .I67(I67),
071     .I68(I68),
072     .I69(I69),
073     .I70(I70),
074     .I71(I71),
075     .I72(I72),
076     .I73(I73),
077     .I74(I74),
078     .I75(I75),
079     .I76(I76),
080     .I77(I77),
081     .I78(I78),
082     .I79(I79),
083     .I80(I80),
084     .I81(I81),
085     .I82(I82),
086     .I83(I83),
087     .I84(I84),
088     .I85(I85),
089     .I86(I86),
090     .I87(I87),
091     .I88(I88),
092     .I89(I89),
093     .I90(I90),
094     .I91(I91),
095     .I92(I92),
096     .I93(I93),
097     .I94(I94),
098     .I95(I95),
099     .I96(I96),
100     .I97(I97),
101     .I98(I98),
102     .I99(I99),
103     .I100(I100),
104     .I101(I101),
105     .I102(I102),
106     .I103(I103),
107     .I104(I104),
108     .I105(I105),
109     .I106(I106),
110     .I107(I107),
111     .I108(I108),
112     .I109(I109),
113     .I110(I110),
114     .I111(I111),
115     .I112(I112),
116     .I113(I113),
117     .I114(I114),
118     .I115(I115),
119     .I116(I116),
120     .I117(I117),
121     .I118(I118),
122     .I119(I119),
123     .I120(I120),
124     .I121(I121),
125     .I122(I122),
126     .I123(I123),
127     .I124(I124),
128     .I125(I125),
129     .I126(I126),
130     .I127(I127),
131     .I128(I128),
132     .I129(I129),
133     .I130(I130),
134     .I131(I131),
135     .I132(I132),
136     .I133(I133),
137     .I134(I134),
138     .I135(I135),
139     .I136(I136),
140     .I137(I137),
141     .I138(I138),
142     .I139(I139),
143     .I140(I140),
144     .I141(I141),
145     .I142(I142),
146     .I143(I143),
147     .I144(I144),
148     .I145(I145),
149     .I146(I146),
150     .I147(I147),
151     .I148(I148),
152     .I149(I149),
153     .I150(I150),
154     .I151(I151),
155     .I152(I152),
156     .I153(I153),
157     .I154(I154),
158     .I155(I155),
159     .I156(I156),
160     .I157(I157),
161     .I158(I158),
162     .I159(I159),
163     .I160(I160),
164     .I161(I161),
165     .I162(I162),
166     .I163(I163),
167     .I164(I164),
168     .I165(I165),
169     .I166(I166),
170     .I167(I167),
171     .I168(I168),
172     .I169(I169),
173     .I170(I170),
174     .I171(I171),
175     .I172(I172),
176     .I173(I173),
177     .I174(I174),
178     .I175(I175),
179     .I176(I176),
180     .I177(I177),
181     .I178(I178),
182     .I179(I179),
183     .I180(I180),
184     .I181(I181),
185     .I182(I182),
186     .I183(I183),
187     .I184(I184),
188     .I185(I185),
189     .I186(I186),
190     .I187(I187),
191     .I188(I188),
192     .I189(I189),
193     .I190(I190),
194     .I191(I191),
195     .I192(I192),
196     .I193(I193),
197     .I194(I194),
198     .I195(I195),
199     .I196(I196),
200     .I197(I197),
201     .I198(I198),
202     .I199(I199),
203     .I200(I200),
204     .I201(I201),
205     .I202(I202),
206     .I203(I203),
207     .I204(I204),
208     .I205(I205),
209     .I206(I206),
210     .I207(I207),
211     .I208(I208),
212     .I209(I209),
213     .I210(I210),
214     .I211(I211),
215     .I212(I212),
216     .I213(I213),
217     .I214(I214),
218     .I215(I215),
219     .I216(I216),
220     .I217(I217),
221     .I218(I218),
222     .I219(I219),
223     .I220(I220),
224     .I221(I221),
225     .I222(I222),
226     .I223(I223),
227     .I224(I224),
228     .I225(I225),
229     .I226(I226),
230     .I227(I227),
231     .I228(I228),
232     .I229(I229),
233     .I230(I230),
234     .I231(I231),
235     .I232(I232),
236     .I233(I233),
237     .I234(I234),
238     .I235(I235),
239     .I236(I236),
240     .I237(I237),
241     .I238(I238),
242     .I239(I239),
243     .I240(I240),
244     .I241(I241),
245     .I242(I242),
246     .I243(I243),
247     .I244(I244),
248     .I245(I245),
249     .I246(I246),
250     .I247(I247),
251     .I248(I248),
252     .I249(I249),
253     .I250(I250),
254     .I251(I251),
255     .I252(I252),
256     .I253(I253),
257     .I254(I254),
258     .I255(I255),
259     .I256(I256),
260     .I257(I257),
261     .I258(I258),
262     .I259(I259),
263     .I260(I260),
264     .I261(I261),
265     .I262(I262),
266     .I263(I263),
267     .I264(I264),
268     .I265(I265),
269     .I266(I266),
270     .I267(I267),
271     .I268(I268),
272     .I269(I269),
273     .I270(I270),
274     .I271(I271),
275     .I272(I272),
276     .I273(I273),
277     .I274(I274),
278     .I275(I275),
279     .I276(I276),
280     .I277(I277),
281     .I278(I278),
282     .I279(I279),
283     .I280(I280),
284     .I281(I281),
285     .I282(I282),
286     .I283(I283),
287     .I284(I284),
288     .I285(I285),
289     .I286(I286),
290     .I287(I287),
291     .I288(I288),
292     .I289(I289),
293     .I290(I290),
294     .I291(I291),
295     .I292(I292),
296     .I293(I293),
297     .I294(I294),
298     .I295(I295),
299     .I296(I296),
300     .I297(I297),
301     .I298(I298),
302     .I299(I299),
303     .I300(I300),
304     .I301(I301),
305     .I302(I302),
306     .I303(I303),
307     .I304(I304),
308     .I305(I305),
309     .I306(I306),
310     .I307(I307),
311     .I308(I308),
312     .I309(I309),
313     .I310(I310),
314     .I311(I311),
315     .I312(I312),
316     .I313(I313),
317     .I314(I314),
318     .I315(I315),
319     .I316(I316),
320     .I317(I317),
321     .I318(I318),
322     .I319(I319),
323     .I320(I320),
324     .I321(I321),
325     .I322(I322),
326     .I323(I323),
327     .I324(I324),
328     .I325(I325),
329     .I326(I326),
330     .I327(I327),
331     .I328(I328),
332     .I329(I329),
333     .I330(I330),
334     .I331(I331),
335     .I332(I332),
336     .I333(I333),
337     .I334(I334),
338     .I335(I335),
339     .I336(I336),
340     .I337(I337),
341     .I338(I338),
342     .I339(I339),
343     .I340(I340),
344     .I341(I341),
345     .I342(I342),
346     .I343(I343),
347     .I344(I344),
348     .I345(I345),
349     .I346(I346),
350     .I347(I347),
351     .I348(I348),
352     .I349(I349),
353     .I350(I350),
354     .I351(I351),
355     .I352(I352),
356     .I353(I353),
357     .I354(I354),
358     .I355(I355),
359     .I356(I356),
360     .I357(I357),
361     .I358(I358),
362     .I359(I359),
363     .I360(I360),
364     .I361(I361),
365     .I362(I362),
366     .I363(I363),
367     .I364(I364),
368     .I365(I365),
369     .I366(I366),
370     .I367(I367),
371     .I368(I368),
372     .I369(I369),
373     .I370(I370),
374     .I371(I371),
375     .I372(I372),
376     .I373(I373),
377     .I374(I374),
378     .I375(I375),
379     .I376(I376),
380     .I377(I377),
381     .I378(I378),
382     .I379(I379),
383     .I380(I380),
384     .I381(I381),
385     .I382(I382),
386     .I383(I383),
387     .I384(I384),
388     .I385(I385),
389     .I386(I386),
390     .I387(I387),
391     .I388(I388),
392     .I389(I389),
393     .I390(I390),
394     .I391(I391),
395     .I392(I392),
396     .I393(I393),
397     .I394(I394),
398     .I395(I395),
399     .I396(I396),
400     .I397(I397),
401     .I398(I398),
402     .I399(I399),
403     .I400(I400),
404     .I401(I401),
405     .I402(I402),
406     .I403(I403),
407     .I404(I404),
408     .I405(I405),
409     .I406(I406),
410     .I407(I407),
411     .I408(I408),
412     .I409(I409),
413     .I410(I410),
414     .I411(I411),
415     .I412(I412),
416     .I413(I413),
417     .I414(I414),
418     .I415(I415),
419     .I416(I416),
420     .I417(I417),
421     .I418(I418),
422     .I419(I419),
423     .I420(I420),
424     .I421(I421),
425     .I422(I422),
426     .I423(I423),
427     .I424(I424),
428     .I425(I425),
429     .I426(I426),
430     .I427(I427),
431     .I428(I428),
432     .I429(I429),
433     .I430(I430),
434     .I431(I431),
435     .I432(I432),
436     .I433(I433),
437     .I434(I434),
438     .I435(I435),
439     .I436(I436),
440     .I437(I437),
441     .I438(I438),
442     .I439(I439),
443     .I440(I440),
444     .I441(I441),
445     .I442(I442),
446     .I443(I443),
447     .I444(I444),
448     .I445(I445),
449     .I446(I446),
450     .I447(I447),
451     .I448(I448),
452     .I449(I449),
453     .I450(I450),
454     .I451(I451),
455     .I452(I452),
456     .I453(I453),
457     .I454(I454),
458     .I455(I455),
459     .I456(I456),
460     .I457(I457),
461     .I458(I458),
462     .I459(I459),
463     .I460(I460),
464     .I461(I461),
465     .I462(I462),
466     .I463(I463),
467     .I464(I464),
468     .I465(I465),
469     .I466(I466),
470     .I467(I467),
471     .I468(I468),
472     .I469(I469),
473     .I470(I470),
474     .I471(I471),
475     .I472(I472),
476     .I473(I473),
477     .I474(I474),
478     .I475(I475),
479     .I476(I476),
480     .I477(I477),
481     .I478(I478),
482     .I479(I479),
483     .I480(I480),
484     .I481(I481),
485     .I482(I482),
486     .I483(I483),
487     .I484(I484),
488     .I485(I485),
489     .I486(I486),
490     .I487(I487),
491     .I488(I488),
492     .I489(I489),
493     .I490(I490),
494     .I491(I491),
495     .I492(I492),
496     .I493(I493),
497     .I494(I494),
498     .I495(I495),
499     .I496(I496),
499     .I497(I497),
499     .I498(I498),
499     .I499(I499),
499     .I500(I500),
499     .I501(I501),
499     .I502(I502),
499     .I503(I503),
499     .I504(I504),
499     .I505(I505),
499     .I506(I506),
499     .I507(I507),
499     .I508(I508),
499     .I509(I509),
499     .I510(I510),
499     .I511(I511),
499     .I512(I512),
499     .I513(I513),
499     .I514(I514),
499     .I515(I515),
499     .I516(I516),
499     .I517(I517),
499     .I518(I518),
499     .I519(I519),
499     .I520(I520),
499     .I521(I521),
499     .I522(I522),
499     .I523(I523),
499     .I524(I524),
499     .I525(I525),
499     .I526(I526),
499     .I527(I527),
499     .I528(I528),
499     .I529(I529),
499     .I530(I530),
499     .I531(I531),
499     .I532(I532),
499     .I533(I533),
499     .I534(I534),
499     .I535(I535),
499     .I536(I536),
499     .I537(I537),
499     .I538(I538),
499     .I539(I539),
499     .I540(I540),
499     .I541(I541),
499     .I542(I542),
499     .I543(I543),
499     .I544(I544),
499     .I545(I545),
499     .I546(I546),
499     .I547(I547),
499     .I548(I548),
499     .I549(I549),
499     .I550(I550),
499     .I551(I551),
499     .I552(I552),
499     .I553(I553),
499     .I554(I554),
499     .I555(I555),
499     .I556(I556),
499     .I557(I557),
499     .I558(I558),
499     .I559(I559),
499     .I560(I560),
499     .I561(I561),
499     .I562(I562),
499     .I563(I563),
499     .I564(I564),
499     .I565(I565),
499     .I566(I566),
499     .I567(I567),
499     .I568(I568),
499     .I569(I569),
499     .I570(I570),
499     .I571(I571),
499     .I572(I572),
499     .I573(I573),
499     .I574(I574),
499     .I575(I575),
499     .I576(I576),
499     .I577(I577),
499     .I578(I578),
499     .I579(I579),
499     .I580(I580),
499     .I581(I581),
499     .I582(I582),
499     .I583(I583),
499     .I584(I584),
499     .I585(I585),
499     .I586(I586),
499     .I587(I587),
499     .I588(I588),
499     .I589(I589),
499     .I590(I590),
499     .I591(I591),
499     .I592(I592),
499     .I593(I593),
499     .I594(I594),
499     .I595(I595),
499     .I596(I596),
499     .I597(I597),
499     .I598(I598),
499     .I599(I599),
499     .I600(I600),
499     .I601(I601),
499     .I602(I602),
499     .I603(I603),
499     .I604(I604),
499     .I605(I605),
499     .I606(I606),
499     .I607(I607),
499     .I608(I608),
499     .I609(I609),
499     .I610(I610),
499     .I611(I611),
499     .I612(I612),
499     .I613(I613),
499     .I614(I614),
499     .I615(I615),
499     .I616(I616),
499     .I617(I617),
499     .I618(I618),
499     .I619(I619),
499     .I620(I620),
499     .I621(I621),
499     .I622(I622),
499     .I623(I623),
499     .I624(I624),
499     .I625(I625),
499     .I626(I626),
499     .I627(I627),
499     .I628(I628),
499     .I629(I629),
499     .I630(I630),
499     .I631(I631),
499     .I632(I632),
499     .I633(I633),
499     .I634(I634),
499     .I635(I635),
499     .I636(I636),
499     .I637(I637),
499     .I638(I638),
499     .I639(I639),
499     .I640(I640),
499     .I641(I641),
499     .I642(I642),
499     .I643(I643),
499     .I644(I644),
499     .I645(I645),
499     .I646(I646),
499     .I647(I647),
499     .I648(I648),
499     .I649(I649),
499     .I650(I650),
499     .I651(I651),
499     .I652(I652),
499     .I653(I653),
499     .I654(I654),
499     .I655(I655),
499     .I656(I656),
499     .I657(I657),
499     .I658(I658),
499     .I659(I659),
499     .I660(I660),
499     .I661(I661),
499     .I662(I662),
499     .I663(I663),
499     .I664(I664),
499     .I665(I665),
499     .I666(I666),
499     .I667(I667),
499     .I668(I668),
499     .I669(I669),
499     .I670(I670),
499     .I671(I671),
499     .I672(I672),
499     .I673(I673),
499     .I674(I674),
499     .I675(I675),
499     .I676(I676),
499     .I677(I677),
499     .I678(I678),
499     .I679(I679),
499     .I680(I680),
499     .I681(I681),
499     .I682(I682),
499     .I683(I683),
499     .I684(I684),
499     .I685(I685),
499     .I686(I686),
499     .I687(I687),
499     .I688(I688),
499     .I689(I689),
499     .I690(I690),
499     .I691(I691),
499     .I692(I692),
499     .I693(I693),
499     .I694(I694),
499     .I695(I695),
499     .I696(I696),
499     .I697(I697),
499     .I698(I698),
499     .I699(I699),
499     .I700(I700),
499     .I701(I701),
499     .I702(I702),
499     .I703(I703),
499     .I704(I704),
499     .I705(I705),
499     .I706(I706),
499     .I707(I707),
499     .I708(I708),
499     .I709(I709),
499     .I710(I710),
499     .I711(I711),
499     .I712(I712),
499     .I713(I713),
499     .I714(I714),
499     .I715(I715),
499     .I716(I716),
499     .I717(I717),
499     .I718(I718),
499     .I719(I719),
499     .I720(I720),
499     .I721(I721),
499     .I722(I722),
499     .I723(I723),
499     .I724(I724),
499     .I725(I725),
499     .I726(I726),
499     .I727(I727),
499     .I728(I728),
499     .I729(I729),
499     .I730(I730),
499     .I731(I731),
499     .I732(I732),
499     .I733(I733),
499     .I734(I734),
499     .I735(I735),
499     .I736(I736),
499     .I737(I737),
499     .I738(I738),
499     .I739(I739),
499     .I740(I740),
499     .I741(I741),
499     .I742(I742),
499     .I743(I743),
499     .I744(I744),
499     .I745(I745),
499     .I746(I746),
499     .I747(I747),
499     .I748(I748),
499     .I749(I749),
499     .I750(I750),
499     .I751(I751),
499     .I752(I752),
499     .I753(I753),
499     .I754(I754),
499     .I755(I755),
499     .I756(I756),
499     .I757(I757),
499     .I758(I758),
499     .I759(I759),
499     .I760(I760),
499     .I761(I761),
499     .I762(I762),
499     .I763(I763),
499     .I764(I764),
499     .I765(I765),
499     .I766(I766),
499     .I767(I767),
499     .I768(I768),
499     .I769(I769),
499     .I770(I770),
499     .I771(I771),
499     .I772(I772),
499     .I773(I773),
499     .I774(I774),
499     .I775(I775),
499     .I776(I776),
499     .I777(I777),
499     .I778(I778),
499     .I779(I779),
499     .I780(I780),
499     .I781(I781),
499     .I782(I782),
499     .I783(I783),
499     .I784(I784),
499     .I785(I785),
499     .I786(I786),
499     .I787(I787),
499     .I788(I788),
499     .I789(I789),
499     .I790(I790),
499     .I791(I791),
499     .I792(I792),
499     .I793(I793),
499     .I794(I794),
499     .I795(I795),
499     .I796(I796),
499     .I797(I797),
499     .I798(I798),
499     .I799(I799),
499     .I800(I800),
499     .I801(I801),
499     .I802(I802),
499     .I803(I803),
499     .I804(I804),
499     .I805(I805),
499     .I806(I806),
499     .I807(I807),
499     .I808(I808),
499     .I809(I8
```

ii) reg9B.v

```
lab1 > E nqf0 ->
 0000: module PDKCF_MULLINK_regW();
 0001:   FORCE L_Q001 .(C(0));
 0002:   .(CE(C));
 0003:   .(CLK(CLK));
 0004:   .(D(D));
 0005:   .(Q(Q[10]));
 0006:   defparam L_Q001.NEXT = 1'b0;
 0007:   FORCE L_Q001 .(C(1));
 0008:   .(CE(C));
 0009:   .(CLK(CLK));
 0010:   .(D(D[1]));
 0011:   .(Q(Q[1]));
 0012:   defparam L_Q001.NEXT = 1'b0;
 0013:   FORCE L_Q012 .(C(0));
 0014:   .(CE(C));
 0015:   .(CLK(CLK));
 0016:   .(D(D[12]));
 0017:   .(Q(Q[12]));
 0018:   defparam L_Q012.NEXT = 1'b0;
 0019:   FORCE L_Q013 .(C(0));
 0020:   .(CE(C));
 0021:   .(CLK(CLK));
 0022:   .(D(D[13]));
 0023:   .(Q(Q[13]));
 0024:   defparam L_Q013.NEXT = 1'b0;
 0025:   FORCE L_Q014 .(C(0));
 0026:   .(CE(C));
 0027:   .(CLK(CLK));
 0028:   .(D(D[14]));
 0029:   .(Q(Q[14]));
 0030:   defparam L_Q014.NEXT = 1'b0;
 0031:   FORCE L_Q015 .(C(0));
 0032:   .(CE(C));
 0033:   .(CLK(CLK));
 0034:   .(D(D[15]));
 0035:   .(Q(Q[15]));
 0036:   defparam L_Q015.NEXT = 1'b0;
 0037: endmodule
 0038: /*timescale 1ns / 1ps
 0039: module PDKCF_MULLINK_regB();
 0040:   input [1:0] CL;
 0041:   input [1:0] CLB;
 0042:   input [1:0] D;
 0043:   output [7:0] Q;
 0044:   output [7:0] Qb;
 0045:
 0046:   logic G;
 0047:   logic C;
 0048:   input CLK;
 0049:   output [7:0] S;
 0050:   output [7:0] Q;
 0051:
 0052:   FORCE L_Q0 .(C(0));
 0053:   .(CE(C));
 0054:   .(CLK(CLK));
 0055:   .(D(D[0]));
 0056:   .(Q(Q[0]));
 0057:   defparam L_Q0.NEXT = 1'b0;
 0058:   FORCE L_Q1 .(C(0));
 0059:   .(CE(C));
 0060:   .(CLK(CLK));
 0061:   .(D(D[1]));
 0062:   .(Q(Q[1]));
 0063:   defparam L_Q1.NEXT = 1'b0;
 0064:   FORCE L_Q2 .(C(0));
 0065:   .(CE(C));
 0066:   .(CLK(CLK));
 0067:   .(D(D[2]));
 0068:   .(Q(Q[2]));
 0069:   defparam L_Q2.NEXT = 1'b0;
 0070:   FORCE L_Q3 .(C(0));
 0071:   .(CE(C));
 0072:   .(CLK(CLK));
 0073:   .(D(D[3]));
 0074:   .(Q(Q[3]));
 0075:   defparam L_Q3.NEXT = 1'b0;
 0076:   FORCE L_Q4 .(C(0));
 0077:   .(CE(C));
 0078:   .(CLK(CLK));
 0079:   .(D(D[4]));
 0080:   .(Q(Q[4]));
 0081:   defparam L_Q4.NEXT = 1'b0;
 0082:   FORCE L_Q5 .(C(0));
 0083:   .(CE(C));
 0084:   .(CLK(CLK));
 0085:   .(D(D[5]));
 0086:   .(Q(Q[5]));
 0087:   defparam L_Q5.NEXT = 1'b0;
 0088:   FORCE L_Q6 .(C(0));
 0089:   .(CE(C));
 0090:   .(CLK(CLK));
 0091:   .(D(D[6]));
 0092:   .(Q(Q[6]));
 0093:   defparam L_Q6.NEXT = 1'b0;
 0094:   FORCE L_Q7 .(C(0));
 0095:   .(CE(C));
 0096:   .(CLK(CLK));
 0097:   .(D(D[7]));
 0098:   .(Q(Q[7]));
 0099:   defparam L_Q7.NEXT = 1'b0;
 0100: endmodule
```

```
135 module FDBCE_MXILINX_reg9B(
136   FDCE_I_Q0 (.C(.C),
137   .D(0[0]),
138   .Q(0[0]));
139 
140 defparam I_Q0.INIT = 1'b0;
141 FDCE_I_Q1 (.C(.C),
142   .CE(.CE),
143   .CLR(CLK),
144   .D(0[1]),
145   .Q(0[1]));
146 
147 defparam I_Q1.INIT = 1'b0;
148 FDCE_I_Q2 (.C(.C),
149   .CE(.CE),
150   .CLR(CLK),
151   .D(0[2]),
152   .Q(0[2]));
153 
154 defparam I_Q2.INIT = 1'b0;
155 FDCE_I_Q3 (.C(.C),
156   .CE(.CE),
157   .CLR(CLK),
158   .D(0[4]),
159   .Q(0[4]));
160 
161 defparam I_Q3.INIT = 1'b0;
162 FDCE_I_Q4 (.C(.C),
163   .CE(.CE),
164   .CLR(CLK),
165   .D(0[4]),
166   .Q(0[4]));
167 
168 defparam I_Q4.INIT = 1'b0;
169 FDCE_I_Q5 (.C(.C),
170   .CE(.CE),
171   .CLR(CLK),
172   .D(0[5]),
173   .Q(0[5]));
174 
175 defparam I_Q5.INIT = 1'b0;
176 FDCE_I_Q6 (.C(.C),
177   .CE(.CE),
178   .CLR(CLK),
179   .D(0[6]),
180   .Q(0[6]));
181 
182 defparam I_Q6.INIT = 1'b0;
183 FDCE_I_Q7 (.C(.C),
184   .CE(.CE),
185   .CLR(CLK),
186   .D(0[7]),
187   .Q(0[7]));
188 
189 defparam I_Q7.INIT = 1'b0;
190 endmodule
191 
192 `timescale 1ns / 1ps
193 
194 module reg9B(CE,
195   input CE;
196   input clk;
197   input clr;
198   input [7:0] d;
199   output [7:0] q;
200 
201   q);
202 
203   input CE;
204   input clk;
205   input clr;
206   input [7:0] d;
207   output [7:0] q;
208 
209   q);
210 
211   q);
212 
213 FDBCE_MXILINX_reg9B XLI1_1 (.C(.clk),
214   .CE(.CE),
215   .CLR(clr),
216   .D(d[7:64]),
217   .Q(q[7:64]));
218 
219 // synthesis attribute MU_SET of XLI1_1 is "XLI1_1_0"
220 FD16CE_MXILINX_reg9B XLI1_2 (.C(.clk),
221   .CE(.CE),
222   .CLR(clr),
223   .D(d[65:48]),
224   .Q(q[65:48]));
225 
226 // synthesis attribute MU_SET of XLI1_2 is "XLI1_2_1"
227 FD16CE_MXILINX_reg9B XLI1_3 (.C(.clk),
228   .CE(.CE),
229   .CLR(clr),
230   .D(d[47:32]),
231   .Q(q[47:32]));
232 
233 // synthesis attribute MU_SET of XLI1_3 is "XLI1_3_2"
234 FD16CE_MXILINX_reg9B XLI1_4 (.C(.clk),
235   .CE(.CE),
236   .CLR(clr),
237   .D(d[31:16]),
238   .Q(q[31:16]));
239 
240 // synthesis attribute MU_SET of XLI1_4 is "XLI1_4_3"
241 FD16CE_MXILINX_reg9B XLI1_5 (.C(.clk),
242   .CE(.CE),
243   .CLR(clr),
244   .D(d[15:8]),
245   .Q(q[15:8]));
246 
247 // synthesis attribute MU_SET of XLI1_5 is "XLI1_5_4"
248 endmodule
```

### iii) wordmatch.v

```

// synthesis attribute RLOC of I_36_91 is "X0Y1"
OR2 I_36_94 (.IO(50),
              .I1(10),
              .O(O_DUMMY));
OR4 I_36_95 (.IO(10),
              .I1(15),
              .I2(16),
              .I3(17),
              .O(S1));
OR4 I_36_112 (.IO(10),
              .I1(11),
              .I2(12),
              .I3(13),
              .O(S0));
FMAP I_36_116 (.I1(10),
                 .I2(11),
                 .I3(12),
                 .I4(13),
                 .O(S0));
// synthesis attribute RLOC of I_36_116 is "X0Y0"
FMAP I_36_117 (.I1(14),
                 .I2(15),
                 .I3(16),
                 .I4(17),
                 .O(S1));
// synthesis attribute RLOC of I_36_117 is "X0Y0"
endmodule
`timescale 1ns / 1ps

module wordmatch(datacomp,
                  datain,
                  wildcard,
                  match);

  input [55:0] datacomp;
  input [111:0] datain;
  input [6:0] wildcard;
  output match;

  wire XLXN_1;
  wire XLXN_2;
  wire XLXN_3;
  wire XLXN_4;
  wire XLXN_5;
  wire XLXN_6;
  wire XLXN_7;
  wire XLXN_8;

  comparator XLI_1 (.a(datacomp[55:0]),
                     .amask(wildcard[6:0]),
                     .b(datain[55:0]),
                     .match(XLXN_1));
  comparator XLI_2 (.a(datacomp[55:0]),
                     .amask(wildcard[6:0]),
                     .b(datain[63:8]),
                     .match(XLXN_2));

```

```

160           .match(XLXN_2));
161     comparator XLXI_3 (.a(datacomp[55:0]),
162                         .amask(wildcard[6:0]),
163                         .b(datain[71:16]),
164                         .match(XLXN_3));
165     comparator XLXI_4 (.a(datacomp[55:0]),
166                         .amask(wildcard[6:0]),
167                         .b(datain[79:24]),
168                         .match(XLXN_4));
169     comparator XLXI_5 (.a(datacomp[55:0]),
170                         .amask(wildcard[6:0]),
171                         .b(datain[87:32]),
172                         .match(XLXN_5));
173     comparator XLXI_6 (.a(datacomp[55:0]),
174                         .amask(wildcard[6:0]),
175                         .b(datain[95:40]),
176                         .match(XLXN_6));
177     comparator XLXI_7 (.a(datacomp[55:0]),
178                         .amask(wildcard[6:0]),
179                         .b(datain[103:48]),
180                         .match(XLXN_7));
181     comparator XLXI_8 (.a(datacomp[55:0]),
182                         .amask(wildcard[6:0]),
183                         .b(datain[111:56]),
184                         .match(XLXN_8));
185   ORB_XKILINX_wordmatch XLXI_9 (.I0(XLXN_8),
186                                 .I1(XLXN_7),
187                                 .I2(XLXN_6),
188                                 .I3(XLXN_5),
189                                 .I4(XLXN_4),
190                                 .I5(XLXN_3),
191                                 .I6(XLXN_2),
192                                 .I7(XLXN_1),
193                                 .O(match));
194   // synthesis attribute HU_SET of XLXI_9 is "XLXI_9_0"
195 endmodule

```

#### iv) detect7B.v

```

` detect7B.v ×
lab3 > E detect7Bv
16 // Device: virtex2p
17 // Purpose:
18 //   This verilog netlist is translated from an ECS 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 module detect7B(ce,
27                   clk,
28                   hregA,
29                   match_en,
30                   mrst,
31                   pipe1,
32                   match);
33
34   input ce;
35   input clk;
36   input [63:0] hregA;
37   input match_en;
38   input mrst;
39   input [71:0] pipe1;
40   output match;
41
42   wire clr;
43   wire [71:0] pipe0;
44   wire [111:0] XLXN_1;
45   wire XLXN_18;
46   wire XLXN_23;
47   wire match_DUMMY;
48
49   assign match = match_DUMMY;
50
51   busmerge XLXI_1 (.da(pipe0[47:0]),
52                     .db(pipe1[63:0]),
53                     .q(XLXN_1[111:0]));
54   wordmatch XLXI_2 (.datacomp(hregA[55:0]),
55                     .datain(XLXN_1[111:0]),
56                     .wildcard(hregA[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 (.T(clk),
63                 .CE(XLXN_23),
64                 .CLR(clr),
65                 .DO(XLXN_23),
66                 .O(match_DUMMY));
67   defparam XLXI_5.INIT = 1'b0;
68   ANDSB1_XLXI_6 (.I0(match_DUMMY),
69                 .I1(match_en),
70                 .I2(XLXN_18),
71                 .O(XLXN_23));
72   reg8B 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



```

F detectTB.v      dropIf0.v x
lab1 > E dropIf0.v
52 module C88CE_MKLINX_dropIf0(C,
53   AND2_I_36_26 (.I0(Q_DUMMV[2]),
54     .I1(Q_DUMMV[1]),
55     .O(T3));
56   AND4_I_36_28 (.I0(Q_DUMMV[4]),
57     .I1(Q_DUMMV[3]),
58     .I2(Q_DUMMV[5]),
59     .I3(Q_DUMMV[4]),
60     .I4(T4),
61     .O(T7));
62   AND2_I_36_31 (.I0(A),
63     .I1(TC_DUMMV),
64     .O(CEO));
65 endmodule
66 timescale 1ns / 1ps
67
68 module COMP8_MKLINX_dropIf0(A,
69   B,
70   EQ);
71   input [7:0] A;
72   input [7:0] B;
73   output EQ;
74
75   wire AB0;
76   wire AB1;
77   wire AB2;
78   wire AB3;
79   wire AB4;
80   wire AB5;
81   wire AB6;
82   wire AB7;
83   wire AB8;
84   wire AB9;
85   wire AB10;
86
87   AND4_I_36_32 (.I0(AB7),
88     .I1(AB6),
89     .I2(AB5),
90     .I3(AB4),
91     .O(AB7));
92   XNOR2_I_36_33 (.I0(B[6]),
93     .I1(B[5]),
94     .O(AB6));
95   XNOR2_I_36_34 (.I0(A[7]),
96     .I1(A[7]),
97     .O(AB5));
98   XNOR2_I_36_35 (.I0(B[5]),
99     .I1(B[5]),
100    .O(AB5));
101  XNOR2_I_36_36 (.I0(B[4]),
102    .I1(A[4]),
103    .O(AB4));
104  AND4_I_36_41 (.I0(AB3),
105    .I1(AB2),
106    .I2(AB1),
107    .I3(AB0),
108    .O(AB0));
109  XNOR2_I_36_42 (.I0(A[2]),
110    .I1(A[2]),
111    .O(AB3));
112  XNOR2_I_36_43 (.I0(A[1]),
113    .I1(A[1]),
114    .O(AB2));
115  XNOR2_I_36_44 (.I0(A[0]),
116    .I1(A[0]),
117    .O(AB1));
118  XNOR2_I_36_45 (.I0(B[0]),
119    .I1(B[0]),
120    .O(AB0));
121  AND2_I_36_50 (.I0(AB4),
122    .I1(AB3),
123    .O(EQ));
124 endmodule
125
126
E dropIf0.v
lab1 > E dropIf0.v x
130 module COMP8_MKLINX_dropIf0(A,
131   B,
132   EQ);
133   input [7:0] A;
134   input [7:0] B;
135   output EQ;
136
137   wire M0;
138   wire M1;
139
140   AND2B1_I_36_7 (.I0(S0),
141     .I1(W0),
142     .O(W0));
143   OR2_I_36_8 (.I0(M0),
144     .I1(W0),
145     .O(W0));
146   AND2_I_36_9 (.I0(W0),
147     .I1(W0),
148     .O(M0));
149 endmodule
150
151 timescale 1ns / 1ps
152
153 module FTCLEx_MKLINX_dropIf0(C,
154   CE,
155   CLR,
156   D,
157   L,
158   T,
159   Q);
160
161   input C;
162   input CE;
163   input CLR;
164   input D;
165   input L;
166   input T;
167   output Q;
168
169   wire M0;
170   wire TQ;
171   wire Q_DUMMV;
172
173   assign Q = Q_DUMMV;
174   M2_1_MKLINX_dropIf0 T_36_30 (.D0(TQ),
175     .S0(W0),
176     .S1(L),
177     .O(W0));
178
179 // synthesis attribute HU_SET of T_36_30 is "T_36_30_8"
180   XNOR2_I_36_30 (.I0(T),
181     .I1(W0),
182     .O(TQ));
183   FDCE_I_36_35 (.C(C),
184     .CE(CE),
185     .CLR(CL),
186     .D(D),
187     .Q(Q_DUMMV));
188
189 // synthesis attribute HU_SET of T_36_35 is "T_36_35_8"
190   dropIf0_I_36_35(INIT = 1'b0);
191 endmodule
192
193 timescale 1ns / 1ps
194
195 module C88CE_MKLINX_dropIf0(C,
196   CE,
197   CLR,
198   D,
199

```

```

If detect?8v u  dropfifov u
lab3 > dropfifov u
288 module CIRCLE_MKLINX_dropFifo,
289   input C;
290   input CE;
291   input CLK;
292   input [7:0] D;
293   input [7:0] CEO;
294   output [7:0] Q;
295   output TC;
296
297   wire OR_CE_L;
298   wire T2;
299   wire T3;
300   wire T4;
301   wire T5;
302   wire T6;
303   wire T7;
304   wire XLMN_1;
305   wire XLMN_2;
306   wire TC_DUMMY;
307
308   assign Q[0] = Q_DUMMY[7:0];
309   assign TC = TC_DUMMY;
310
311   FTCLEx_MKLINX_dropFifo I_Q0 (.C(C),
312     .CE(OR_CE_L),
313     .CLK(CLK),
314     .D(D));
315   assign Q[1] = Q_DUMMY[6:0];
316   assign TC = TC_DUMMY;
317
318   FTCLEx_MKLINX_dropFifo I_Q1 (.C(C),
319     .CE(OR_CE_L),
320     .CLK(CLK),
321     .D(D));
322   assign Q[2] = Q_DUMMY[5:0];
323   assign TC = TC_DUMMY;
324
325   FTCLEx_MKLINX_dropFifo I_Q2 (.C(C),
326     .CE(OR_CE_L),
327     .CLK(CLK),
328     .D(D));
329   assign Q[3] = Q_DUMMY[4:0];
330   assign TC = TC_DUMMY;
331
332   FTCLEx_MKLINX_dropFifo I_Q3 (.C(C),
333     .CE(OR_CE_L),
334     .CLK(CLK),
335     .D(D));
336   assign Q[4] = Q_DUMMY[3:0];
337   assign TC = TC_DUMMY;
338
339   FTCLEx_MKLINX_dropFifo I_Q4 (.C(C),
340     .CE(OR_CE_L),
341     .CLK(CLK),
342     .D(D));
343   assign Q[5] = Q_DUMMY[2:0];
344   assign TC = TC_DUMMY;
345
346   FTCLEx_MKLINX_dropFifo I_Q5 (.C(C),
347     .CE(OR_CE_L),
348     .CLK(CLK),
349     .D(D));
350   assign Q[6] = Q_DUMMY[1:0];
351   assign TC = TC_DUMMY;
352
353   FTCLEx_MKLINX_dropFifo I_Q6 (.C(C),
354     .CE(OR_CE_L),
355     .CLK(CLK),
356     .D(D));
357   assign Q[7] = Q_DUMMY[0:0];
358   assign TC = TC_DUMMY;
359
360   // synthesis attribute MU_SET of I_Q0 is "I_QL_9"
361   FTCLEx_MKLINX_dropFifo I_Q7 (.C(C),
362     .CE(OR_CE_L),
363     .CLK(CLK),
364     .D(D));
365   assign Q[8] = Q_DUMMY[3];
366   assign TC = TC_DUMMY;
367
368   // synthesis attribute MU_SET of I_Q1 is "I_QL_10"
369   FTCLEx_MKLINX_dropFifo I_Q8 (.C(C),
370     .CE(OR_CE_L),
371     .CLK(CLK),
372     .D(D));
373   assign Q[9] = Q_DUMMY[2];
374   assign TC = TC_DUMMY;
375
376   // synthesis attribute MU_SET of I_Q2 is "I_QL_11"
377   FTCLEx_MKLINX_dropFifo I_Q9 (.C(C),
378     .CE(OR_CE_L),
379     .CLK(CLK),
380     .D(D));
381   assign Q[10] = Q_DUMMY[1];
382   assign TC = TC_DUMMY;
383
384   // synthesis attribute MU_SET of I_Q3 is "I_QL_12"
385   FTCLEx_MKLINX_dropFifo I_Q10 (.C(C),
386     .CE(OR_CE_L),
387     .CLK(CLK),
388     .D(D));
389   assign Q[11] = Q_DUMMY[0];
390   assign TC = TC_DUMMY;
391
392   // synthesis attribute MU_SET of I_Q4 is "I_QL_13"
393   FTCLEx_MKLINX_dropFifo I_Q11 (.C(C),
394     .CE(OR_CE_L),
395     .CLK(CLK),
396     .D(D));
397   assign Q[12] = Q_DUMMY[4];
398   assign TC = TC_DUMMY;
399
400   // synthesis attribute MU_SET of I_Q5 is "I_QL_14"
401   FTCLEx_MKLINX_dropFifo I_Q12 (.C(C),
402     .CE(OR_CE_L),
403     .CLK(CLK),
404     .D(D));
405   assign Q[13] = Q_DUMMY[5];
406   assign TC = TC_DUMMY;
407
408   // synthesis attribute MU_SET of I_Q6 is "I_QL_15"
409   FTCLEx_MKLINX_dropFifo I_Q13 (.C(C),
410     .CE(OR_CE_L),
411     .CLK(CLK),
412     .D(D));
413   assign Q[14] = Q_DUMMY[6];
414   assign TC = TC_DUMMY;
415
416   // synthesis attribute MU_SET of I_Q7 is "I_QL_16"
417   AND1 I_36_8 .IB(Q_DUMMY[5]);
418   AND1 I_36_9 .IB(Q_DUMMY[4]);
419   AND1 I_36_10 .IB(Q_DUMMY[3]);
420   AND1 I_36_11 .IB(Q_DUMMY[2]);
421   AND1 I_36_12 .IB(Q_DUMMY[1]);
422   AND1 I_36_13 .IB(Q_DUMMY[0]);
423
424   AND2 I_36_14 .IB(Q_DUMMY[4]);
425   AND2 I_36_15 .IB(Q_DUMMY[3]);
426   AND2 I_36_16 .IB(Q_DUMMY[2]);
427   AND2 I_36_17 .IB(Q_DUMMY[1]);
428   AND2 I_36_18 .IB(Q_DUMMY[0]);
429
430   VCC I_36_19 (.P(XLMN_1));
431   AND2 I_36_20 .IB(Q_DUMMY[1]);
432   AND2 I_36_21 .IB(Q_DUMMY[0]);
433
434   AND1 I_36_22 .IB(Q_DUMMY[1]);
435   AND1 I_36_23 .IB(Q_DUMMY[0]);
436
437   ANDA I_36_24 .IB(Q_DUMMY[1]);
438   ANDA I_36_25 .IB(Q_DUMMY[0]);
439
440   AND1 I_36_26 .IB(Q_DUMMY[1]);
441   AND1 I_36_27 .IB(Q_DUMMY[0]);
442
443   ANDA I_36_28 .IB(Q_DUMMY[1]);
444   ANDA I_36_29 .IB(Q_DUMMY[0]);
445
446   ANDS I_36_30 .IB(Q_DUMMY[1]);
447   ANDS I_36_31 .IB(Q_DUMMY[0]);
448
449   AND2 I_36_32 .IB(Q_DUMMY[1]);
450   AND2 I_36_33 .IB(Q_DUMMY[0]);
451
452   OR2 I_36_34 (.IN(CE),
453     .IN(TC),
454     .OUT(Q));
455
456   endmodule
457
458   'timescale 1ns / 1ps
459
460   module F0FCE_MKLINX_dropFifo,
461     input C;
462     input CE;
463     input CLK;
464     input D;
465     output Q;
466
467     input C;
468     input CE;
469     input CLK;
470     input D;
471     output Q;
472
473   endmodule
474
475   module F0FCE_MKLINX_dropFifo,
476     input C;
477     input CE;
478     input CLK;
479     input D;
480     output Q;
481
482   endmodule

```

```

If detect?8v u  dropfifov u
lab3 > dropfifov u
288 module CIRCLE_MKLINX_dropFifo,
289   input C;
290   input CE;
291   input CLK;
292   input D;
293   output Q;
294
295   wire OR_CE_L;
296   wire T2;
297   wire T3;
298   wire T4;
299   wire T5;
300   wire T6;
301   wire T7;
302   wire XLMN_1;
303   wire XLMN_2;
304   wire TC_DUMMY;
305
306   assign Q[0] = Q_DUMMY[7:0];
307   assign TC = TC_DUMMY;
308
309   FTCLEx_MKLINX_dropFifo I_Q0 (.C(C),
310     .CE(OR_CE_L),
311     .CLK(CLK),
312     .D(D));
313   assign Q[1] = Q_DUMMY[6:0];
314   assign TC = TC_DUMMY;
315
316   FTCLEx_MKLINX_dropFifo I_Q1 (.C(C),
317     .CE(OR_CE_L),
318     .CLK(CLK),
319     .D(D));
320   assign Q[2] = Q_DUMMY[5:0];
321   assign TC = TC_DUMMY;
322
323   FTCLEx_MKLINX_dropFifo I_Q2 (.C(C),
324     .CE(OR_CE_L),
325     .CLK(CLK),
326     .D(D));
327   assign Q[3] = Q_DUMMY[4:0];
328   assign TC = TC_DUMMY;
329
330   FTCLEx_MKLINX_dropFifo I_Q3 (.C(C),
331     .CE(OR_CE_L),
332     .CLK(CLK),
333     .D(D));
334   assign Q[4] = Q_DUMMY[3:0];
335   assign TC = TC_DUMMY;
336
337   FTCLEx_MKLINX_dropFifo I_Q4 (.C(C),
338     .CE(OR_CE_L),
339     .CLK(CLK),
340     .D(D));
341   assign Q[5] = Q_DUMMY[2:0];
342   assign TC = TC_DUMMY;
343
344   FTCLEx_MKLINX_dropFifo I_Q5 (.C(C),
345     .CE(OR_CE_L),
346     .CLK(CLK),
347     .D(D));
348   assign Q[6] = Q_DUMMY[1:0];
349   assign TC = TC_DUMMY;
350
351   FTCLEx_MKLINX_dropFifo I_Q6 (.C(C),
352     .CE(OR_CE_L),
353     .CLK(CLK),
354     .D(D));
355   assign Q[7] = Q_DUMMY[0:0];
356   assign TC = TC_DUMMY;
357
358   // synthesis attribute MU_SET of I_Q0 is "I_QL_9"
359   FTCLEx_MKLINX_dropFifo I_Q7 (.C(C),
360     .CE(OR_CE_L),
361     .CLK(CLK),
362     .D(D));
363   assign Q[8] = Q_DUMMY[3];
364   assign TC = TC_DUMMY;
365
366   // synthesis attribute MU_SET of I_Q1 is "I_QL_10"
367   FTCLEx_MKLINX_dropFifo I_Q8 (.C(C),
368     .CE(OR_CE_L),
369     .CLK(CLK),
370     .D(D));
371   assign Q[9] = Q_DUMMY[2];
372   assign TC = TC_DUMMY;
373
374   // synthesis attribute MU_SET of I_Q2 is "I_QL_11"
375   FTCLEx_MKLINX_dropFifo I_Q9 (.C(C),
376     .CE(OR_CE_L),
377     .CLK(CLK),
378     .D(D));
379   assign Q[10] = Q_DUMMY[1];
380   assign TC = TC_DUMMY;
381
382   // synthesis attribute MU_SET of I_Q3 is "I_QL_12"
383   FTCLEx_MKLINX_dropFifo I_Q10 (.C(C),
384     .CE(OR_CE_L),
385     .CLK(CLK),
386     .D(D));
387   assign Q[11] = Q_DUMMY[0];
388   assign TC = TC_DUMMY;
389
390   // synthesis attribute MU_SET of I_Q4 is "I_QL_13"
391   FTCLEx_MKLINX_dropFifo I_Q11 (.C(C),
392     .CE(OR_CE_L),
393     .CLK(CLK),
394     .D(D));
395   assign Q[12] = Q_DUMMY[4];
396   assign TC = TC_DUMMY;
397
398   // synthesis attribute MU_SET of I_Q5 is "I_QL_14"
399   FTCLEx_MKLINX_dropFifo I_Q12 (.C(C),
400     .CE(OR_CE_L),
401     .CLK(CLK),
402     .D(D));
403   assign Q[13] = Q_DUMMY[5];
404   assign TC = TC_DUMMY;
405
406   // synthesis attribute MU_SET of I_Q6 is "I_QL_15"
407   FTCLEx_MKLINX_dropFifo I_Q13 (.C(C),
408     .CE(OR_CE_L),
409     .CLK(CLK),
410     .D(D));
411   assign Q[14] = Q_DUMMY[6];
412   assign TC = TC_DUMMY;
413
414   // synthesis attribute MU_SET of I_Q7 is "I_QL_16"
415   AND1 I_36_8 .IB(Q_DUMMY[5]);
416   AND1 I_36_9 .IB(Q_DUMMY[4]);
417   AND1 I_36_10 .IB(Q_DUMMY[3]);
418   AND1 I_36_11 .IB(Q_DUMMY[2]);
419   AND1 I_36_12 .IB(Q_DUMMY[1]);
420   AND1 I_36_13 .IB(Q_DUMMY[0]);
421
422   AND2 I_36_14 .IB(Q_DUMMY[4]);
423   AND2 I_36_15 .IB(Q_DUMMY[3]);
424   AND2 I_36_16 .IB(Q_DUMMY[2]);
425   AND2 I_36_17 .IB(Q_DUMMY[1]);
426   AND2 I_36_18 .IB(Q_DUMMY[0]);
427
428   VCC I_36_19 (.P(XLMN_1));
429   AND2 I_36_20 .IB(Q_DUMMY[1]);
430   AND2 I_36_21 .IB(Q_DUMMY[0]);
431
432   AND1 I_36_22 .IB(Q_DUMMY[1]);
433   AND1 I_36_23 .IB(Q_DUMMY[0]);
434
435   ANDA I_36_24 .IB(Q_DUMMY[1]);
436   ANDA I_36_25 .IB(Q_DUMMY[0]);
437
438   AND1 I_36_26 .IB(Q_DUMMY[1]);
439   AND1 I_36_27 .IB(Q_DUMMY[0]);
440
441   ANDA I_36_28 .IB(Q_DUMMY[1]);
442   ANDA I_36_29 .IB(Q_DUMMY[0]);
443
444   ANDS I_36_30 .IB(Q_DUMMY[1]);
445   ANDS I_36_31 .IB(Q_DUMMY[0]);
446
447   AND2 I_36_32 .IB(Q_DUMMY[1]);
448   AND2 I_36_33 .IB(Q_DUMMY[0]);
449
450   OR2 I_36_34 (.IN(CE),
451     .IN(TC),
452     .OUT(Q));
453
454   endmodule
455
456   'timescale 1ns / 1ps
457
458   module F0FCE_MKLINX_dropFifo,
459     input C;
460     input CE;
461     input CLK;
462     input D;
463     output Q;
464
465   endmodule
466
467   module F0FCE_MKLINX_dropFifo,
468     input C;
469     input CE;
470     input CLK;
471     input D;
472     output Q;
473
474   endmodule

```

```
if detectTB.vu
  if dropfifo.vu
    module FDCE_MQMLINX_dropfifo(C,
    421   .CE(C),
    422   .D(D),
    423   .Q(Q));
    424
    425   input C;
    426   input CE;
    427   input D;
    428   input CLK;
    429   input [7:0] D0;
    430   output [7:0] Q;
    431
    432
    433   FDCE T_Q0 (.C(CC),
    434     .CE(CE),
    435     .CLR(CLK),
    436     .D(D[0]),
    437     .Q(Q[0]));
    438
    439   defparam T_Q0.INIT = 1'b0;
    440
    441   FDCE T_Q1 (.C(CC),
    442     .CE(CE),
    443     .CLR(CLK),
    444     .D(D[1]),
    445     .Q(Q[1]));
    446
    447   defparam T_Q1.INIT = 1'b0;
    448
    449   FDCE T_Q2 (.C(CC),
    450     .CE(CE),
    451     .CLR(CLK),
    452     .D(D[2]),
    453     .Q(Q[2]));
    454
    455   defparam T_Q2.INIT = 1'b0;
    456
    457   FDCE T_Q3 (.C(CC),
    458     .CE(CE),
    459     .CLR(CLK),
    460     .D(D[3]),
    461     .Q(Q[3]));
    462
    463   defparam T_Q3.INIT = 1'b0;
    464
    465   FDCE T_Q4 (.C(CC),
    466     .CE(CE),
    467     .CLR(CLK),
    468     .D(D[4]),
    469     .Q(Q[4]));
    470
    471   defparam T_Q4.INIT = 1'b0;
    472
    473   FDCE T_Q5 (.C(CC),
    474     .CE(CE),
    475     .CLR(CLK),
    476     .D(D[5]),
    477     .Q(Q[5]));
    478
    479   defparam T_Q5.INIT = 1'b0;
    480
    481   FDCE T_Q6 (.C(CC),
    482     .CE(CE),
    483     .CLR(CLK),
    484     .D(D[6]),
    485     .Q(Q[6]));
    486
    487   defparam T_Q6.INIT = 1'b0;
    488
    489   FDCE T_Q7 (.C(CC),
    490     .CE(CE),
    491     .CLR(CLK),
    492     .D(D[7]),
    493     .Q(Q[7]));
    494
    495   defparam T_Q7.INIT = 1'b0;
    496
    497 endmodule
    498
    499
    500 *timescale 1ns / 1ps
    501
    502
    503 module dropfifo(clk,
    504   drop_pkt,
    505   #ioread,
    506   #iowrite,
    507   #firstword,
    508   #lastword,
    509   in_fifo,
    510   out_fifo,
    511   valid_data);
    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
```

```
 1 detectRx.v      dropRx.v x
 2
 3
 4 module dropRx(clk,
 5
 6   wire [7:0] address;
 7   wire [7:0] raddr;
 8   wire [31:0] add;
 9   wire [31:0] data;
10   wire [31:0] XLMN_23;
11   wire [31:0] XLMN_24;
12   wire [31:0] XLMN_25;
13   wire [31:0] XLMN_33;
14   wire [7:0] XLMN_34;
15   wire [31:0] XLMN_40;
16   wire [31:0] XLMN_41;
17   wire [31:0] XLMN_44;
18   wire [31:0] XLMN_47;
19   wire [71:0] XLMN_48;
20   wire [31:0] XLMN_49;
21
22   dual_core_memory_element XLMN_1 (.address(waddr[7:0]),
23   .addr(raddr[7:0]),
24   .data(data),
25   .clk(clk),
26   .dina(XLMN_47[71:0]),
27   .dinb(),
28   .weq(XLMN_47),
29   .web(),
30   .dout(),
31   .douts(met_fifo[71:0]));
32
33   FD_XLMN_2 (.C(c1),
34   .D(firstword),
35   .Q(XLMN_23));
36   defparam XLMN_2_INCT = 1'b0;
37
38   FD_XLMN_3 (.C(c2),
39   .D(lastword),
40   .Q(XLMN_24));
41   defparam XLMN_3_INCT = 1'b0;
42
43   FD_XLMN_4 (.C(c3),
44   .D(fifowrite),
45   .Q(XLMN_47));
46   defparam XLMN_4_INCT = 1'b0;
47
48   OR2_XLMN_5 (.A(XLMN_23),
49   .B(XLMN_24),
50   .I1(XLMN_23),
51   .O(XLMN_26));
52
53   FORCE_XMLINN_dropFifo XLMN_6 (.C(c4),
54   .CE(XLMN_27),
55   .CL(XLMN_27),
56   .CLR(rst),
57   .D(waddr[7:0]),
58   .Q(XLMN_34[7:0]));
59
60 // synthesis attribute MU_SET of XLMN_5 is "XLMN_6_17"
61 AND2R_XLMN_7 (.I0(XLMN_31),
62   .I1(XLMN_26),
63   .O(XLMN_27));
64
65   FD_XLMN_8 (.C(c5),
66   .D(drop_pk),
67   .Q(XLMN_31));
68   defparam XLMN_8_INCT = 1'b0;
69
70   CBRCLE_XMLINN_dropFifo XLMN_9 (.C(c6),
71   .CE(XLMN_47),
72   .CL(rst),
73   .D(XLMN_34[7:0]),
74   .I1(XLMN_31),
75   .CEO(),
76   .Q(waddr[7:0]),
77   .TC());
78
79 // synthesis attribute MU_SET of XLMN_8 is "XLMN_9_18"
80 COMP_XMLINN_dropFifo XLMN_10 (.A(waddr[7:0]),
81   .B(raddr[7:0]),
82   .Q(XLMN_48));
83
84 // synthesis attribute MU_SET of XLMN_9 is "XLMN_10_19"
85 COMP_XMLINN_dropFifo XLMN_11 (.A(raddr[7:0]),
86   .B(XLMN_34[7:0]),
87   .Q(XLMN_49));
88
89 // synthesis attribute MU_SET of XLMN_10 is "XLMN_11_20"
90 CBRCLE_XMLINN_dropFifo XLMN_12 (.C(c7),
91   .CE(XLMN_44),
92   .CL(rst),
93   .D(XLMN_49));
```

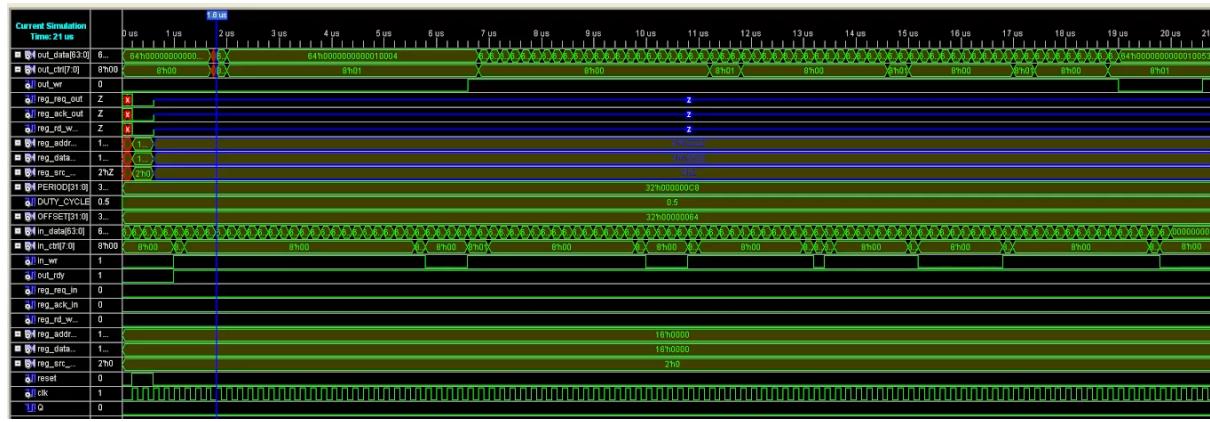


```

17 // synthesis attribute HU_SET of XLXI_11 is "XLXI_11_2B"
17 v CBCE_PXTL_10_dropIfNo XLXI_12 (.C(c),
17 .CE(XLXN_44),
17 .CLK(rst),
17 .CR(),
17 .Q(XLXN_44),
17 .TC());
17
18 // synthesis attribute HU_SET of XLXI_32 is "XLXI_32_2B"
18 v AND382 XLXI_33 (.C(c),
18 .CE(XLXN_45),
18 .CLK(rst),
18 .I2(fForasd),
18 .O(KLXN_44));
18
19 v FDC XLXI_34 (.C(c),
19 .CE(XLXN_45),
19 .CLK(rst),
19 .D(KLXN_44),
19 .Q(valid_data));
19
20 defparam XLXI_34_INIT = 1'b0;
20 VCC XLXI_36 (.P(1'b1));
20 v reg#(XLXI_17) XLXI_37 (.CE(XLXN_49),
20 .regData(XLXN_49),
20 .clk(clk),
20 .clr(rst),
20 .q(XLXN_48[71:0]),
20 .q(XLXN_48[71:0]));
20
20 endmodule

```

- 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.



The testbench sends packets into the mini-IDS and watches how the outputs change. It provides the clock and reset signals and then drives input data into the system, just like packets arriving from a network.

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.

- 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.

- Include the answers to your lab in this report.

- 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.