

```
1  -- Copyright (C) 1991-2013 Altera Corporation
2  -- Your use of Altera Corporation's design tools, logic functions
3  -- and other software and tools, and its AMPP partner logic
4  -- functions, and any output files from any of the foregoing
5  -- (including device programming or simulation files), and any
6  -- associated documentation or information are expressly subject
7  -- to the terms and conditions of the Altera Program License
8  -- Subscription Agreement, Altera MegaCore Function License
9  -- Agreement, or other applicable license agreement, including,
10 -- without limitation, that your use is for the sole purpose of
11 -- programming logic devices manufactured by Altera and sold by
12 -- Altera or its authorized distributors. Please refer to the
13 -- applicable agreement for further details.
14
15 -- PROGRAM      "Quartus II 64-Bit"
16 -- VERSION      "Version 13.0.1 Build 232 06/12/2013 Service Pack 1 SJ Full Version"
17 -- CREATED      "Sat Mar 04 15:01:59 2017"
18
19 LIBRARY ieee;
20 USE ieee.std_logic_1164.all;
21
22 LIBRARY work;
23
24 ENTITY ELEC374 IS
25     PORT
26     (
27         ren0 : IN  STD_LOGIC;
28         ren1 : IN  STD_LOGIC;
29         ren2 : IN  STD_LOGIC;
30         ren3 : IN  STD_LOGIC;
31         ren4 : IN  STD_LOGIC;
32         ren5 : IN  STD_LOGIC;
33         ren6 : IN  STD_LOGIC;
34         ren7 : IN  STD_LOGIC;
35         ren8 : IN  STD_LOGIC;
36         ren9 : IN  STD_LOGIC;
37         renA : IN  STD_LOGIC;
38         renB : IN  STD_LOGIC;
39         renC : IN  STD_LOGIC;
40         renD : IN  STD_LOGIC;
41         renE : IN  STD_LOGIC;
42         renF : IN  STD_LOGIC;
43         clr  : IN  STD_LOGIC;
44         clk  : IN  STD_LOGIC;
45         R0out : IN  STD_LOGIC;
46         R1out : IN  STD_LOGIC;
47         R2out : IN  STD_LOGIC;
48         R3out : IN  STD_LOGIC;
49         R4out : IN  STD_LOGIC;
50         R5out : IN  STD_LOGIC;
51         R6out : IN  STD_LOGIC;
52         R7out : IN  STD_LOGIC;
53         R8out : IN  STD_LOGIC;
54         R9out : IN  STD_LOGIC;
55         R10out : IN  STD_LOGIC;
56         R11out : IN  STD_LOGIC;
57         R12out : IN  STD_LOGIC;
58         R13out : IN  STD_LOGIC;
59         R14out : IN  STD_LOGIC;
60         R15out : IN  STD_LOGIC;
61         Zhighout : IN  STD_LOGIC;
62         Zlowout : IN  STD_LOGIC;
```

```

63      InPortout : IN STD_LOGIC;
64      HIin : IN STD_LOGIC;
65      PCin : IN STD_LOGIC;
66      LOin : IN STD_LOGIC;
67      MDRin : IN STD_LOGIC;
68      HIoutEn : IN STD_LOGIC;
69      LOoutEn : IN STD_LOGIC;
70      PCoutEn : IN STD_LOGIC;
71      MDRoutEn : IN STD_LOGIC;
72      CoutEn : IN STD_LOGIC;
73      reny : IN STD_LOGIC;
74      renz : IN STD_LOGIC;
75      ReadIn : IN STD_LOGIC;
76      busmuxout : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
77      Empty : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
78      HIout : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
79      LOout : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
80      Mdatain : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
81      MDRout : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
82      operation : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
83      PCout : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
84      rout0 : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
85      rout1 : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
86      rout2 : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
87      rout3 : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
88      rout4 : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
89      rout5 : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
90      rout6 : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
91      rout7 : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
92      rout8 : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
93      rout9 : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
94      routa : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
95      routb : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
96      routc : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
97      routd : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
98      route : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
99      routf : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
100     routy : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
101     Zhigh : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0);
102     Zlow : INOUT STD_LOGIC_VECTOR (31 DOWNTO 0)
103 );
104 END ELEC374;
105
106 ARCHITECTURE bdf_type OF ELEC374 IS
107
108 COMPONENT alu
109     PORT (clk : IN STD_LOGIC;
110          a : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
111          b : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
112          op : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
113          y : OUT STD_LOGIC_VECTOR (63 DOWNTO 0)
114     );
115 END COMPONENT;
116
117 COMPONENT reg
118     PORT (clr : IN STD_LOGIC;
119          clk : IN STD_LOGIC;
120          ren : IN STD_LOGIC;
121          rin : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
122          rout : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
123     );
124 END COMPONENT;

```

```

125
126 COMPONENT good_mux
127     PORT (data0x : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
128           data10x : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
129           data11x : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
130           data12x : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
131           data13x : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
132           data14x : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
133           data15x : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
134           data16x : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
135           data17x : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
136           data18x : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
137           data19x : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
138           data1x  : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
139           data20x : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
140           data21x : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
141           data22x : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
142           data23x : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
143           data24x : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
144           data25x : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
145           data2x  : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
146           data3x  : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
147           data4x  : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
148           data5x  : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
149           data6x  : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
150           data7x  : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
151           data8x  : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
152           data9x  : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
153           sel     : IN STD_LOGIC_VECTOR (4 DOWNTO 0);
154           result  : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
155     );
156 END COMPONENT;
157
158 COMPONENT encoder32to5
159     PORT (R0out : IN STD_LOGIC;
160           R1out : IN STD_LOGIC;
161           R2out : IN STD_LOGIC;
162           R3out : IN STD_LOGIC;
163           R4out : IN STD_LOGIC;
164           R5out : IN STD_LOGIC;
165           R6out : IN STD_LOGIC;
166           R7out : IN STD_LOGIC;
167           R8out : IN STD_LOGIC;
168           R9out : IN STD_LOGIC;
169           R10out : IN STD_LOGIC;
170           R11out : IN STD_LOGIC;
171           R12out : IN STD_LOGIC;
172           R13out : IN STD_LOGIC;
173           R14out : IN STD_LOGIC;
174           R15out : IN STD_LOGIC;
175           HIout  : IN STD_LOGIC;
176           LOout  : IN STD_LOGIC;
177           Zhighout : IN STD_LOGIC;
178           Zlowout  : IN STD_LOGIC;
179           PCout   : IN STD_LOGIC;
180           MDRout  : IN STD_LOGIC;
181           InPortout : IN STD_LOGIC;
182           Cout    : IN STD_LOGIC;
183           Sin     : OUT STD_LOGIC_VECTOR (4 DOWNTO 0)
184     );
185 END COMPONENT;
186

```

```

187 COMPONENT mdmux
188     PORT (ReadIn : IN STD_LOGIC;
189           BusMuxOut : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
190           Mdatain : IN STD_LOGIC_VECTOR (31 DOWNTO 0);
191           MDMuxOut : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
192     );
193 END COMPONENT;
194
195 COMPONENT reg64
196     PORT (clr : IN STD_LOGIC;
197           clk : IN STD_LOGIC;
198           ren : IN STD_LOGIC;
199           rin : IN STD_LOGIC_VECTOR (63 DOWNTO 0);
200           rh : OUT STD_LOGIC_VECTOR (31 DOWNTO 0);
201           rlow : OUT STD_LOGIC_VECTOR (31 DOWNTO 0)
202     );
203 END COMPONENT;
204
205 SIGNAL Cout : STD_LOGIC_VECTOR (31 DOWNTO 0);
206 SIGNAL InPortout0 : STD_LOGIC;
207 SIGNAL InPortout1 : STD_LOGIC;
208 SIGNAL InPortout10 : STD_LOGIC;
209 SIGNAL InPortout11 : STD_LOGIC;
210 SIGNAL InPortout12 : STD_LOGIC;
211 SIGNAL InPortout13 : STD_LOGIC;
212 SIGNAL InPortout14 : STD_LOGIC;
213 SIGNAL InPortout15 : STD_LOGIC;
214 SIGNAL InPortout16 : STD_LOGIC;
215 SIGNAL InPortout17 : STD_LOGIC;
216 SIGNAL InPortout18 : STD_LOGIC;
217 SIGNAL InPortout19 : STD_LOGIC;
218 SIGNAL InPortout2 : STD_LOGIC;
219 SIGNAL InPortout20 : STD_LOGIC;
220 SIGNAL InPortout21 : STD_LOGIC;
221 SIGNAL InPortout22 : STD_LOGIC;
222 SIGNAL InPortout23 : STD_LOGIC;
223 SIGNAL InPortout24 : STD_LOGIC;
224 SIGNAL InPortout25 : STD_LOGIC;
225 SIGNAL InPortout26 : STD_LOGIC;
226 SIGNAL InPortout27 : STD_LOGIC;
227 SIGNAL InPortout28 : STD_LOGIC;
228 SIGNAL InPortout29 : STD_LOGIC;
229 SIGNAL InPortout3 : STD_LOGIC;
230 SIGNAL InPortout30 : STD_LOGIC;
231 SIGNAL InPortout31 : STD_LOGIC;
232 SIGNAL InPortout4 : STD_LOGIC;
233 SIGNAL InPortout5 : STD_LOGIC;
234 SIGNAL InPortout6 : STD_LOGIC;
235 SIGNAL InPortout7 : STD_LOGIC;
236 SIGNAL InPortout8 : STD_LOGIC;
237 SIGNAL InPortout9 : STD_LOGIC;
238 SIGNAL SYNTHESIZED_WIRE_0 : STD_LOGIC_VECTOR (4 DOWNTO 0);
239 SIGNAL SYNTHESIZED_WIRE_1 : STD_LOGIC_VECTOR (31 DOWNTO 0);
240 SIGNAL SYNTHESIZED_WIRE_2 : STD_LOGIC_VECTOR (63 DOWNTO 0);
241
242 SIGNAL GDFX_TEMP_SIGNAL_0 : STD_LOGIC_VECTOR (31 DOWNTO 0);
243
244 BEGIN
245
246 GDFX_TEMP_SIGNAL_0 <= (InPortout31 & InPortout30 & InPortout29 & InPortout28 & InPortout27 &
& InPortout26 & InPortout25 & InPortout24 & InPortout23 & InPortout22 & InPortout21 &
& InPortout20 & InPortout19 & InPortout18 & InPortout17 & InPortout16 & InPortout15 &

```

```
InPortout14 & InPortout13 & InPortout12 & InPortout11 & InPortout10 & InPortout9 &
InPortout8 & InPortout7 & InPortout6 & InPortout5 & InPortout4 & InPortout3 & InPortout2 &
InPortout1 & InPortout0);
```

```
247
248
249 b2v_ALU : alu
250 PORT MAP(clk => clk,
251          a => routy,
252          b => busmuxout,
253          op => operation,
254          y => SYNTHESIZED_WIRE_2);
255
256
257 b2v_H : reg
258 PORT MAP(clr => clr,
259          clk => clk,
260          ren => HIin,
261          rin => busmuxout,
262          rout => HIout);
263
264
265 b2v_inst : good_mux
266 PORT MAP(data0x => Empty,
267          data10x => rout9,
268          data11x => routa,
269          data12x => routb,
270          data13x => routc,
271          data14x => routd,
272          data15x => route,
273          data16x => routf,
274          data17x => HIout,
275          data18x => LOout,
276          data19x => Zhigh,
277          data1x  => rout0,
278          data20x => Zlow,
279          data21x => PCout,
280          data22x => MDRout,
281          data23x => GDFX_TEMP_SIGNAL_0,
282          data24x => Cout,
283          data25x => Empty,
284          data2x  => rout1,
285          data3x  => rout2,
286          data4x  => rout3,
287          data5x  => rout4,
288          data6x  => rout5,
289          data7x  => rout6,
290          data8x  => rout7,
291          data9x  => rout8,
292          sel  => SYNTHESIZED_WIRE_0,
293          result => busmuxout);
294
295
296 b2v_inst1 : encoder32to5
297 PORT MAP(R0out => R0out,
298          R1out => R1out,
299          R2out => R2out,
300          R3out => R3out,
301          R4out => R4out,
302          R5out => R5out,
303          R6out => R6out,
304          R7out => R7out,
305          R8out => R8out,
```

```
306         R9out => R9out,
307         R10out => R10out,
308         R11out => R11out,
309         R12out => R12out,
310         R13out => R13out,
311         R14out => R14out,
312         R15out => R15out,
313         HIout => HIoutEn,
314         LOout => LOoutEn,
315         Zhighout => Zhighout,
316         Zlowout => Zlowout,
317         PCout => PCoutEn,
318         MDRout => MDRoutEn,
319         InPortout => InPortout,
320         Cout => CoutEn,
321         Sin => SYNTHESIZED_WIRE_0 );
322
323
324 b2v_inst2 : mdmux
325 PORT MAP (ReadIn => ReadIn,
326           BusMuxOut => busmuxout,
327           Mdatain => Mdatain,
328           MDMuxOut => SYNTHESIZED_WIRE_1 );
329
330
331 b2v_LO : reg
332 PORT MAP (clr => clr,
333           clk => clk,
334           ren => LOin,
335           rin => busmuxout,
336           rout => LOout);
337
338
339 b2v_MDR : reg
340 PORT MAP (clr => clr,
341           clk => clk,
342           ren => MDRin,
343           rin => SYNTHESIZED_WIRE_1,
344           rout => MDRout);
345
346
347 b2v_PC : reg
348 PORT MAP (clr => clr,
349           clk => clk,
350           ren => PCin,
351           rin => busmuxout,
352           rout => PCout);
353
354
355 b2v_R0 : reg
356 PORT MAP (clr => clr,
357           clk => clk,
358           ren => ren0,
359           rin => busmuxout,
360           rout => rout0);
361
362
363 b2v_R1 : reg
364 PORT MAP (clr => clr,
365           clk => clk,
366           ren => ren1,
367           rin => busmuxout,
```

```
368         rout => rout1);
369
370
371 b2v_R2 : reg
372 PORT MAP(clr => clr,
373         clk => clk,
374         ren => ren2,
375         rin => busmuxout,
376         rout => rout2);
377
378
379 b2v_R3 : reg
380 PORT MAP(clr => clr,
381         clk => clk,
382         ren => ren3,
383         rin => busmuxout,
384         rout => rout3);
385
386
387 b2v_R4 : reg
388 PORT MAP(clr => clr,
389         clk => clk,
390         ren => ren4,
391         rin => busmuxout,
392         rout => rout4);
393
394
395 b2v_R5 : reg
396 PORT MAP(clr => clr,
397         clk => clk,
398         ren => ren5,
399         rin => busmuxout,
400         rout => rout5);
401
402
403 b2v_R6 : reg
404 PORT MAP(clr => clr,
405         clk => clk,
406         ren => ren6,
407         rin => busmuxout,
408         rout => rout6);
409
410
411 b2v_R7 : reg
412 PORT MAP(clr => clr,
413         clk => clk,
414         ren => ren7,
415         rin => busmuxout,
416         rout => rout7);
417
418
419 b2v_R8 : reg
420 PORT MAP(clr => clr,
421         clk => clk,
422         ren => ren8,
423         rin => busmuxout,
424         rout => rout8);
425
426
427 b2v_R9 : reg
428 PORT MAP(clr => clr,
429         clk => clk,
```

```
430         ren => ren9,
431         rin => busmuxout,
432         rout => rout9);
433
434
435 b2v_Ra : reg
436 PORT MAP(clr => clr,
437          clk => clk,
438          ren => renA,
439          rin => busmuxout,
440          rout => routa);
441
442
443 b2v_Rb : reg
444 PORT MAP(clr => clr,
445          clk => clk,
446          ren => renB,
447          rin => busmuxout,
448          rout => routb);
449
450
451 b2v_Rc : reg
452 PORT MAP(clr => clr,
453          clk => clk,
454          ren => renC,
455          rin => busmuxout,
456          rout => routc);
457
458
459 b2v_Rd : reg
460 PORT MAP(clr => clr,
461          clk => clk,
462          ren => renD,
463          rin => busmuxout,
464          rout => routd);
465
466
467 b2v_Re : reg
468 PORT MAP(clr => clr,
469          clk => clk,
470          ren => renE,
471          rin => busmuxout,
472          rout => route);
473
474
475 b2v_regz : reg64
476 PORT MAP(clr => clr,
477          clk => clk,
478          ren => renz,
479          rin => SYNTHESIZED_WIRE_2,
480          rh => Zhigh,
481          rlow => Zlow);
482
483
484 b2v_Rf : reg
485 PORT MAP(clr => clr,
486          clk => clk,
487          ren => renF,
488          rin => busmuxout,
489          rout => routf);
490
491
```



```
492     b2v_RY : reg
493     PORT MAP(clr => clr,
494             clk => clk,
495             ren => reny,
496             rin => busmuxout,
497             rout => routy);
498
499
500     END bdf_type;
```