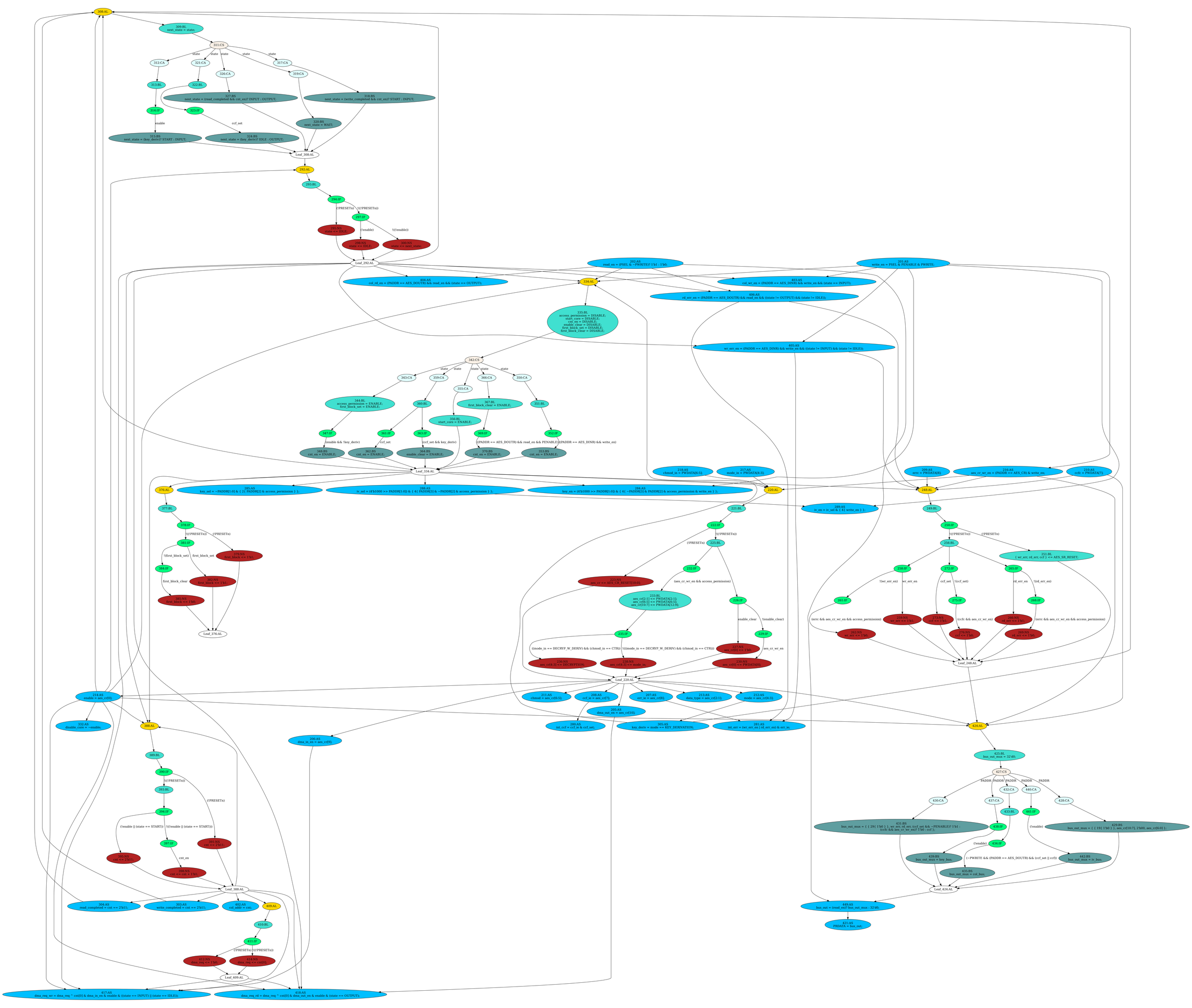
308:AL
309:BL
next_state = state;
311:CS
state
312:CA
321:CA
326:CA
317:CA
319:CA
313:BL
322:BL
327:BS
next_state = (read_completed && cnt_en)? !INPUT : OUTPUT;
318:BS
next_state = (write_completed && cnt_en)? START : INPUT;
314:IF
323:IF
320:BS
next_state = WAIT;
enable
ccf_set
315:BS
next_state = (key_deriv)? START : INPUT;
324:BS
next_state = (key_deriv)? IDLE : OUTPUT;
Leaf_308:AL
292:AL
293:BL
294:IF
!(!PRESETn)
297:IF
!(!enable)
!((enable))
290:NS
state <= IDLE;
298:NS
state <= IDLE;
300:NS
state <= next_state;
Leaf_292:AL
202:AS
read_en = (PSEL & ~PWRITE)? 1'b1 : 1'b0;
201:AS
write_en = PSEL & PENABLE & PWRITE;
404:AS
col_rd_en = (PADDR == AES_DOUTR) && read_en && (state == OUTPUT);
334:AL
403:AS
col_wr_en = (PADDR == AES_DINR) && write_en && (state == INPUT);
406:AS
rd_err_en = (PADDR == AES_DOUTR) && read_en && (state != OUTPUT) && (state != IDLE));
335:BL
access_permission = DISABLE;
start_core = DISABLE;
cnt_en = DISABLE;
enable_clear = DISABLE;
first_block_set = DISABLE;
first_block_clear = DISABLE;
405:AS
wr_err_en = (PADDR == AES_DINR) && write_en && (state != INPUT) && (state != IDLE));
342:CS
state
343:CA
359:CA
355:CA
366:CA
350:CA
344:BL
access_permission = ENABLE;
first_block_set = ENABLE;
360:BL
367:BL
first_block_clear = ENABLE;
351:BL
356:BL
start_core = ENABLE;
347:IF
361:IF
363:IF
369:IF
352:IF
(enable && !key_deriv)
ccf_set
(ccf_set && key_deriv)
((PADDR == AES_DOUTR) && read_en && PENABLE))||(PADDR == AES_DINR) && write_en)
348:BS
cnt_en = ENABLE;
362:BS
cnt_en = ENABLE;
364:BS
enable_clear = ENABLE;
370:BS
cnt_en = ENABLE;
353:BS
cnt_en = ENABLE;
Leaf_334:AL
210:AS
chmod_in = PWDATA[6:5];
217:AS
mode_in = PWDATA[4:3];
209:AS
errc = PWDATA[8];
216:AS
aes_cr_wr_en = (PADDR == AES_CR) & write_en;
210:AS
cccfc = PWDATA[7];
376:AL
285:AS
key_sel = ~PADDR[1:0] & { 2{ PADDR[2] & access_permission } };
288:AS
iv_sel = (4'b1000 >> PADDR[1:0]) & { 4{ PADDR[3] & ~PADDR[2] & access_permission } };
284:AS
key_en = (4'b1000 >> PADDR[1:0]) & { 4{ ~PADDR[3] & PADDR[2] & access_permission & write_en } };
220:AL
240:AL
377:BL
289:AS
iv_en = iv_sel & { 4{ write_en } };
221:BL
249:BL
378:IF
!((!PRESETn))
!(PRESETn)
381:IF
!(first_block_set)
first_block_set
379:NS
first_block <= 1'b1;
384:IF
first_block_clear
382:NS
first_block <= 1'b1;
385:NS
first_block <= 1'b0;
Leaf_376:AL
222:IF
!((!PRESETn))
225:BL
232:IF
223:NS
aes_cr <= AES_CR_RESET[10:0];
(aes_cr_wr_en && access_permission)
233:BL
aes_cr[2:1] <= PWDATA[2:1];
aes_cr[6:5] <= PWDATA[6:5];
aes_cr[10:7] <= PWDATA[12:9];
226:IF
!(enable_clear)
229:IF
aes_cr_wr_en
235:IF
((mode_in == DECRYP_W_DERIV) && (chmod_in == CTR))
!((((mode_in == DECRYP_W_DERIV) && (chmod_in == CTR))
227:NS
aes_cr[0] <= 1'b0;
236:NS
aes_cr[4:3] <= DECRYPTION;
238:NS
aes_cr[4:3] <= mode_in;
230:NS
aes_cr[0] <= PWDATA[0];
Leaf_220:AL
256:BL
251:BL
{ wr_err, rd_err, ccf } <= AES_SR_RESET;
258:IF
272:IF
265:IF
!(wr_err_en)
wr_err_en
ccf_set
!(ccf_set)
rd_err_en
!(rd_err_en)
261:IF
275:IF
268:IF
(errc && aes_cr_wr_en && access_permission)
259:NS
wr_err <= 1'b1;
273:NS
ccf <= 1'b1;
(cccfc && aes_cr_wr_en)
266:NS
rd_err <= 1'b1;
(errc && aes_cr_wr_en && access_permission)
262:NS
wr_err <= 1'b0;
276:NS
ccf <= 1'b0;
269:NS
rd_err <= 1'b0;
Leaf_248:AL
211:AS
chmod = aes_cr[6:5];
208:AS
ccf_ie = aes_cr[7];
207:AS
err_ie = aes_cr[8];
213:AS
data_type = aes_cr[2:1];
212:AS
mode = aes_cr[4:3];
205:AS
dma_out_en = aes_cr[10];
280:AS
int_ccf = ccf_ie & ccf_set;
305:AS
key_deriv = mode == KEY_DERIVATION;
281:AS
int_err = (wr_err_en | rd_err_en) & err_ie;
214:AS
enable = aes_cr[0];
388:AL
206:AS
dma_in_en = aes_cr[9];
424:AL
332:AS
disable_core = ~enable;
389:BL
390:IF
!((!PRESETn))
393:BL
!(PRESETn)
394:IF
(!enable || (state == START))
!((!enable || (state == START)))
397:IF
391:NS
cnt <= 2'b11;
395:NS
cnt <= 2'b11;
cnt_en
398:NS
cnt <= cnt + 1'b1;
Leaf_388:AL
304:AS
read_completed = cnt == 2'b11;
303:AS
write_completed = cnt == 2'b11;
402:AS
col_addr = cnt;
409:AL
410:BL
411:IF
!(PRESETn)
!((!PRESETn))
412:NS
dma_req <= 1'b0;
414:NS
dma_req <= cnt[0];
Leaf_409:AL
417:AS
dma_req_wr = dma_req ^ cnt[0] & dma_in_en & enable & ((state == INPUT) || (state == IDLE));
418:AS
dma_req_rd = dma_req ^ cnt[0] & dma_out_en & enable & (state == OUTPUT);
425:BL
bus_out_mux = 32'd0;
427:CS
PADDR
430:CA
432:CA
440:CA
437:CA
428:CA
431:BS
bus_out_mux = { { 29{ 1'b0 } }, wr_err, rd_err, (ccf_set && ~PENABLE)? 1'b1 :
(cccfc && aes_cr_wr_en)? 1'b0 : ccf };
433:BL
441:IF
438:IF
429:BS
bus_out_mux = { { 19{ 1'b0 } }, aes_cr[10:7], 2'b00, aes_cr[6:0] };
(!enable)
434:IF
439:BS
bus_out_mux = key_bus;
(~PWRITE && (PADDR == AES_DOUTR) && (ccf_set || ccf))
442:BS
bus_out_mux = iv_bus;
435:BS
bus_out_mux = col_bus;
Leaf_424:AL
449:AS
bus_out = (read_en)? bus_out_mux : 32'd0;
421:AS
PRDATA = bus_out;