1371:AL
1372:NS
rx_active_r <= rx_active;
Leaf_1371:AL
1374:AS
crc16_clr = rx_active & !rx_active_r;
1461:BS
next_state = IDLE;
Leaf_1403:AL
1357:AL
1288:AL
1291:AL
1294:AL
1398:AL
1253:AS
pid_ld_en = pid_le_sm & rx_active & rx_valid;
1335:AL
1366:AS
rx_data_done = data_done;
1376:AL
1358:BL
1360:IF
1359:IF
1361:IF
data_valid_d
data_valid_d
data_valid_d
1360:NS
d1 <= d0;
1359:NS
d0 <= rx_data;
1361:NS
d2 <= d1;
Leaf_1357:AL
1364:AS
rx_data_st = d2;
1289:IF
token_le_1
1289:NS
token0 <= rx_data;
1292:IF
token_le_2
1292:NS
token1 <= rx_data;
1295:NS
token_valid_r1 <= token_le_2;
Leaf_1294:AL
1297:AL
1400:IF
(!rst)
!((!rst))
1400:NS
state <= IDLE;
1401:NS
state <= next_state;
Leaf_1398:AL
1258:AL
1260:IF
!((!rst))
(!rst)
1262:IF
pid_ld_en
1262:NS
pid <= rx_data;
1260:NS
pid <= 8'hf0;
1337:IF
!((!rst))
(!rst)
1339:IF
!(data_valid_d)
data_valid_d
1337:NS
rxv1 <= 1'b0;
1341:IF
data_done
1339:NS
rxv1 <= 1'b1;
1341:NS
rxv1 <= 1'b0;
Leaf_1335:AL
1377:IF
crc16_clr
!(crc16_clr)
1379:IF
1377:NS
crc16_sum <= 16'hffff;
data_valid_d
1379:NS
crc16_sum <= crc16_out;
Leaf_1376:AL
1388:AS
crc16_err = data_done & (crc16_sum != 16'h800d);
Leaf_1288:AL
Leaf_1291:AL
1326:AS
token_fadr = token0[6:0];
1325:AS
frame_no = { token1[2:0], token0 };
1327:AS
token_endp = { token1[2:0], token0[7] };
1328:AS
token_crc5 = token1[7:3];
1298:NS
token_valid_str1 <= token_valid_r1 | got_pid_ack;
Leaf_1297:AL
1300:AS
token_valid = token_valid_str1;
1322:AS
crc5_out2 = ~{ crc5_out[0], crc5_out[1], crc5_out[2], crc5_out[3], crc5_out[4] };
1304:AS
crc5_err = token_valid & (crc5_out2 != token_crc5);
Leaf_1258:AL
1267:AS
pid_IN = pid[3:0] == 4'b1001;
1281:AS
pid_PING = pid[3:0] == 4'b0100;
1268:AS
pid_SOF = pid[3:0] == 4'b0101;
1266:AS
pid_OUT = pid[3:0] == 4'b0001;
1269:AS
pid_SETUP = pid[3:0] == 4'b1101;
1273:AS
pid_MDATA = pid[3:0] == 4'b1111;
1271:AS
pid_DATA1 = pid[3:0] == 4'b1011;
1270:AS
pid_DATA0 = pid[3:0] == 4'b0011;
1264:AS
pid_cks_err = pid[3:0] != ~pid[7:4];
1282:AS
pid_RES = pid[3:0] == 4'b0000;
1276:AS
pid_STALL = pid[3:0] == 4'b1110;
1279:AS
pid_ERR = pid[3:0] == 4'b1100;
1272:AS
pid_DATA2 = pid[3:0] == 4'b0111;
1275:AS
pid_NACK = pid[3:0] == 4'b1010;
1278:AS
pid_PRE = pid[3:0] == 4'b1100;
1280:AS
pid_SPLIT = pid[3:0] == 4'b1000;
1277:AS
pid_NYET = pid[3:0] == 4'b0110;
1274:AS
pid_ACK = pid[3:0] == 4'b0010;
1284:AS
pid_TOKEN = pid_OUT | pid_IN | pid_SOF | pid_SETUP | pid_PING;
1285:AS
pid_DATA = pid_DATA0 | pid_DATA1 | pid_DATA2 | pid_MDATA;
1346:AL
1348:IF
!((!rst))
(!rst)
1350:IF
(rxv1 && data_valid_d)
!((rxv1 && data_valid_d))
1348:NS
rxv2 <= 1'b0;
1352:IF
data_done
1350:NS
rxv2 <= 1'b1;
1352:NS
rxv2 <= 1'b0;
Leaf_1346:AL
1354:AL
1355:NS
data_valid0 <= rxv2 & data_valid_d;
Leaf_1354:AL
1365:AS
rx_data_valid = data_valid0;
(!rx_active)
1403:AL
1405:BL
next_state = state;
pid_le_sm = 1'b0;
token_le_1 = 1'b0;
token_le_2 = 1'b0;
data_valid_d = 1'b0;
data_done = 1'b0;
seq_err = 1'b0;
got_pid_ack = 1'b0;
1414:CS
state
state
state
state
1450:CA
1420:CA
1464:CA
1415:CA
1451:BL
1421:BL
1465:BL
1416:BL
pid_le_sm = 1'b1;
1452:IF
(rx_valid && rx_active && !rx_err)
!((rx_valid && rx_active && !rx_err))
1453:BL
token_le_2 = 1'b1;
next_state = IDLE;
1458:IF
(!rx_active || rx_err)
1459:BL
seq_err = !rx_err;
1461:IF
1423:IF
!((pid_ACK && !rx_err))
(pid_ACK && !rx_err)
1424:BL
got_pid_ack = 1'b1;
1466:IF
(rx_valid && rx_active && !rx_err)
1466:BS
data_valid_d = 1'b1;
1467:IF
1418:IF
(!rx_active || rx_err)
(rx_valid && rx_active)
1468:BL
data_done = 1'b1;
1418:BS
next_state = ACTIVE;
1430:IF
(pid_TOKEN && rx_valid && rx_active && !rx_err)
!((pid_TOKEN && rx_valid && rx_active && !rx_err))
1431:BL
token_le_1 = 1'b1;
next_state = TOKEN;
1437:IF
(pid_DATA && rx_valid && rx_active && !rx_err)
!((pid_DATA && rx_valid && rx_active && !rx_err))
1438:BL
data_valid_d = 1'b1;
next_state = DATA;
1426:IF
(!rx_active)
1426:BS
next_state = IDLE;
1470:IF
(!rx_active)
1470:BS
next_state = IDLE;
1443:IF
(!rx_active || rx_err || rx_valid && !(pid_TOKEN || pid_DATA))
1445:BL
seq_err = !rx_err;
1447:IF
(!rx_active)
1447:BS
next_state = IDLE;