This presentation addresses some comments that were submitted as individual comment, per the rule one comment per issue, however they are related since they touch the same STATES and exit conditions and their solution need to be integrated for completeness and easy and clear instructions for the editor if these comments will be accepted.

PART A: Comments i-250, i-251(Page 136 Line 20)

Executives Summary

1) In the current state machine, we will be stuck in ENTRY_SEC in the transition from ENTRY_SEC to START_DETECT_SEC due to:

1.1) failing detection and/or classification in the primary that will cause pwr_app_pri=FALSE when class_4PID_mult_event_sec=FALSE or

1.2) tdet_timer_pri signal stay done even if we did detection in the primary, continue to DETECT_EVAL_PRI and then to IDLE_PRI due to invalid signature which will keep pwr_app_pri=FALSE

2) The current state machine in the exit from IDLE_SEC to START_DETECT_SEC doesn't allow doing staggered detection when CC-DET-SEQ=3 when primary is not turn on yet and also prevents doing multiple cycles of detection+classification until host decides to power on the port.

As a result, we need to:

(1) Eliminate the possibility of being stuck in ENTRY_SEC in the transition from ENTRY_SEC to START_DETECT_SEC due to failing detection and/or classification in the primary or tdet_timer_pri signal stay done event if we did detection in the primary.

(2) Per the CC_DET_SEQ parameter options (0 to 3) to verify that detection and detection+classification cycles can be done in staggered or parallel manner per the CC_DET_SEQ definition options and the state machine. This work is focused in CC_DET_SEQ=3 (there other options look OK).

Comments Details:

Comment i-251 (Page 136 Line 20).

In the exit from ENTRY_SEC to START_DETECT_SEC, when selecting CC_DET_SEQ 0 or 1, and class_4PID_multi_event_sec = FALSE, the secondary state machine allows to move from ENTRY_SEC state to START_DETECT_SEC only if pwr_app_pri = TRUE per the existing condition:

sism * ((!class_4PID_mult_events_sec * **pwr_app_pri**) + class_4PID_mult_events_sec) * (CC_DET_SEQ=0 + CC_DET_SEQ=1)

1		*	1	*	1	+	**0**	*	1	+	1

Result: 1 ➔ Moving to START_DETECT_SEC ➔ OK

If Primary fails to powerup, the Primary state machine returns back to IDLE_PRI. As a result, pwr_app_pri variable will remain in FALSE, and the secondary state machine won't be able to exit from ENTRY_SEC i.e. will be stuck there.

sism * ((!class_4PID_mult_events_sec * **pwr_app_pri**) + class_4PID_mult_events_sec) * (CC_DET_SEQ=0 + CC_DET_SEQ=1)

1	*	**1**	*	0	+	**0**	*	1	+	1

Result: 0 ➔ stuck in ENTRY_SEC.

The straightforward way to handle this problem is to enable moving to START_DETECT_SEC from ENTRY_SEC, also if primary performed detection at least once and is now in IDLE_PRI state which prevents stuck at ENTRY_SEC. This solution requires the addition of new variable det_once_pri (the current draft has only det_once_sec) which is required also by other comments that all related to each other (e.g. to allow the possibility to do cycles of detection + class probe events on primary and secondary.

Proposed Remedy for comment i-251 only (See proposed baseline that addresses all related comments):

1) Add the following variable:

det_once_pri

This variable indicates if the PSE has probed the Primary Alternative at least once, when entering to DETECT_EVAL_PRI. Values:

FALSE: The PSE has not probed on the Primary Alternative since entering the Primary Alternative state diagram.

TRUE: The PSE has probed the Primary Alternative at least once since entering the Primary Alternative state diagram.

2) Change from:

"sism *((!class_4PID_mult_events_sec * pwr_app_pri) + class_4PID_mult_events_sec) * (CC_DET_SEQ=0 + CC_DET_SEQ=1)"

To:

sism * ((!class_4PID_mult_events_sec * (pwr_app_pri + det_once_pri * !det_start_pri)) + class_4PID_mult_events_sec) * (CC_DET_SEQ=0 + CC_DET_SEQ=1).

Comment i-252 (Page 136 Line 21).

The exit from ENTRY_SEC to START_DETECT_SEC:

In the transition between ENTRY_SEC to START_DET_SEC we have the following condition:
sism * ((!class_4PID_mult_events_sec * pwr_app_pri) + class_4PID_mult_events_sec) * (CC_DET_SEQ=0 + CC_DET_SEQ=1).

When class_4PID_mult_events_sec=FALSE, and CC_DET_SEQ=0 is TRUE or CC_DET_SEQ=1 is TRUE, If START_DET_PRI exit to IDLE_PRI due to tdet_timer_pri_done, the pwr_app_pri will remain in FALSE which won't allow exiting from ENTRY_SEC to START_DETECT_SEC and the secondary state machine remain stuck in ENTRY_SEC.

Note: Even if we did detection before tdet_timer_pri is expired, we will get tdet_timer_pri_done anyway at some time. There is missing stop timer assignment.

Time	t1	t2	t3	t4	t5
Pri	START_DETECT_PRI	do_detect_Pri_done = FALSE tdet_timer_pri_done = TRUE	Exit to IDLE_PRI	alt_pwrd_sec=FALSE det_start_sec=FALSE	Exit to WAIT_PRI
Sec	ENTRY_SEC	*pwr_app_pri = FALSE*	***Stay at ENTRY_SEC***	***Stay at ENTRY_SEC***	***stay at ENTRY_SEC***

sism * ((!class_4PID_mult_events_sec * **pwr_app_pri**) + class_4PID_mult_events_sec) * (CC_DET_SEQ=0 + CC_DET_SEQ=1)

1	*	**1**	*	0	+	**0**	*	1	+	1

Result: 0 ➔ stuck in ENTRY_SEC.

Proposed Remedy for comment i-252 only (See proposed baseline that addresses all related comments):
Make the following changes to allow to move to START_DETECT_SEC in case that we move from START_DETECT_PRI to IDLE_PRI due to tdet_timer_pri expiration:

1. "sism *((!class_4PID_mult_events_sec * **(** pwr_app_pri + tdet_timer_pri_**done)**) + class_4PID_mult_events_sec) * (CC_DET_SEQ=0 + CC_DET_SEQ=1)"

2. Add stop_tdet_timer_pri in the DETECT_EVAL_PRI state in order to ensure that tdet_timer_pri_done signal stay FALSE when the primary state machine moves from START_DETECT_PRI to DETECT_EVAL_PRI and forward down the road.

Summary for comments (i-251, i-252):

Step by step changing the condition from ENTRY_SEC to START_DETECT_SEC

1. *The original condition*

sism * ((!class_4PID_mult_events_sec * pwr_app_pri) + class_4PID_mult_events_sec) * (CC_DET_SEQ=0 + CC_DET_SEQ=1)

2. ***(i-251): To resolve detection or classification failure that cause us to return to IDLE_PRI and stuck in*** *ENTRY_SEC due to pwr_app_pri=FALSE*

sism * ((!class_4PID_mult_events_sec * **(**pwr_app_pri + det_once_pri * !det_start_pri **)**) + class_4PID_mult_events_sec) * (CC_DET_SEQ=0 + CC_DET_SEQ=1).

3. *(i-252): To resolve tdet_timer_pri expiration that cause us to return to IDLE_PRI from START_DETECT_PRI and remain stuck in ENTRY_SEC due to pwr_app_pri=FALSE*

"sism *((!class_4PID_mult_events_sec * **(** pwr_app_pri + tdet_timer_pri_done **)**) + class_4PID_mult_events_sec) * (CC_DET_SEQ=0 + CC_DET_SEQ=1)"

4. *The combined solution for i-251 and i-252:*

sism * ((!class_4PID_mult_events_sec * **(** pwr_app_pri + det_once_pri * !det_start_pri + tdet_timer_pri_**done** **)**) + class_4PID_mult_events_sec) * (CC_DET_SEQ=0 + CC_DET_SEQ=1).

PART B: Comment i-254 (Page 136 Line 11).

The content of comment i-254 addressing the exit from IDLE_SEC to START_DETECT_SEC was changed here since the issue described there is no longer valid after understanding parallel detection and staggered detection definitions.

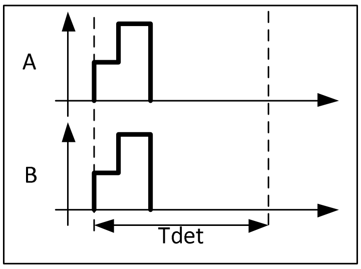
In the exit from IDLE_SEC to START_DETECT_SEC we have the following condition:
(!pwr_app_sec * pwr_app_pri) + ((CC_DET_SEQ=3) * option_probe_alt_sec * !det_start_pri * !det_once_sec.

- The yellow part of the condition means that detection on the secondary starts only after primary is already in power on state.
- The light blue part is allowed only with CC_DET_SEQ=3.

The first issue is that "parallel detection" and "staggered detection" are not clearly defined in D3.0, however we know what we wanted to do.

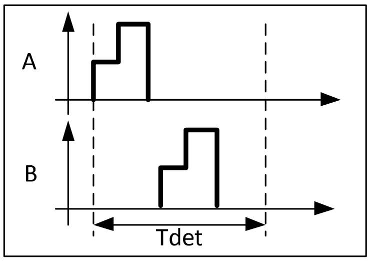
- *In "parallel detection" the intent was to do detection on both pairsets within Tdet window.*

Option A



OR

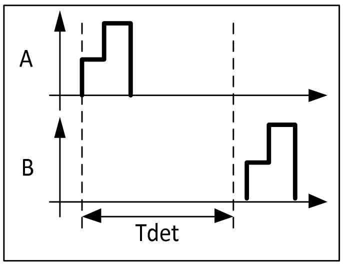
Option B



OR quasi-simultaneous

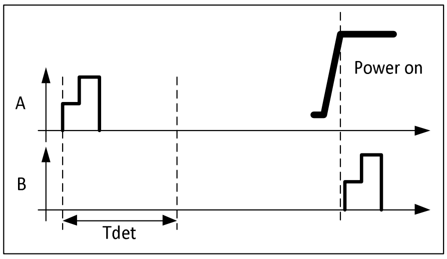
- In staggered detection, the detection on both pairsets is done in different Tdet cycles.
  As a result, staggered detection could be any of the following:

Option 1



OR

option 2



CC_DET_SEQ=3 for dual-signature PD is currently used in the state machine by the condition: (!pwr_app_sec * pwr_app_pri) which means that we need clarify what is staggered detection for single-signature and dual-signature.

The 2nd issue is the stuck in ENTRY_SEC as explained above.

Baseline starts here

Proposed Remedy

1. Add the following variables to 145.2.5.4

det_once_pri

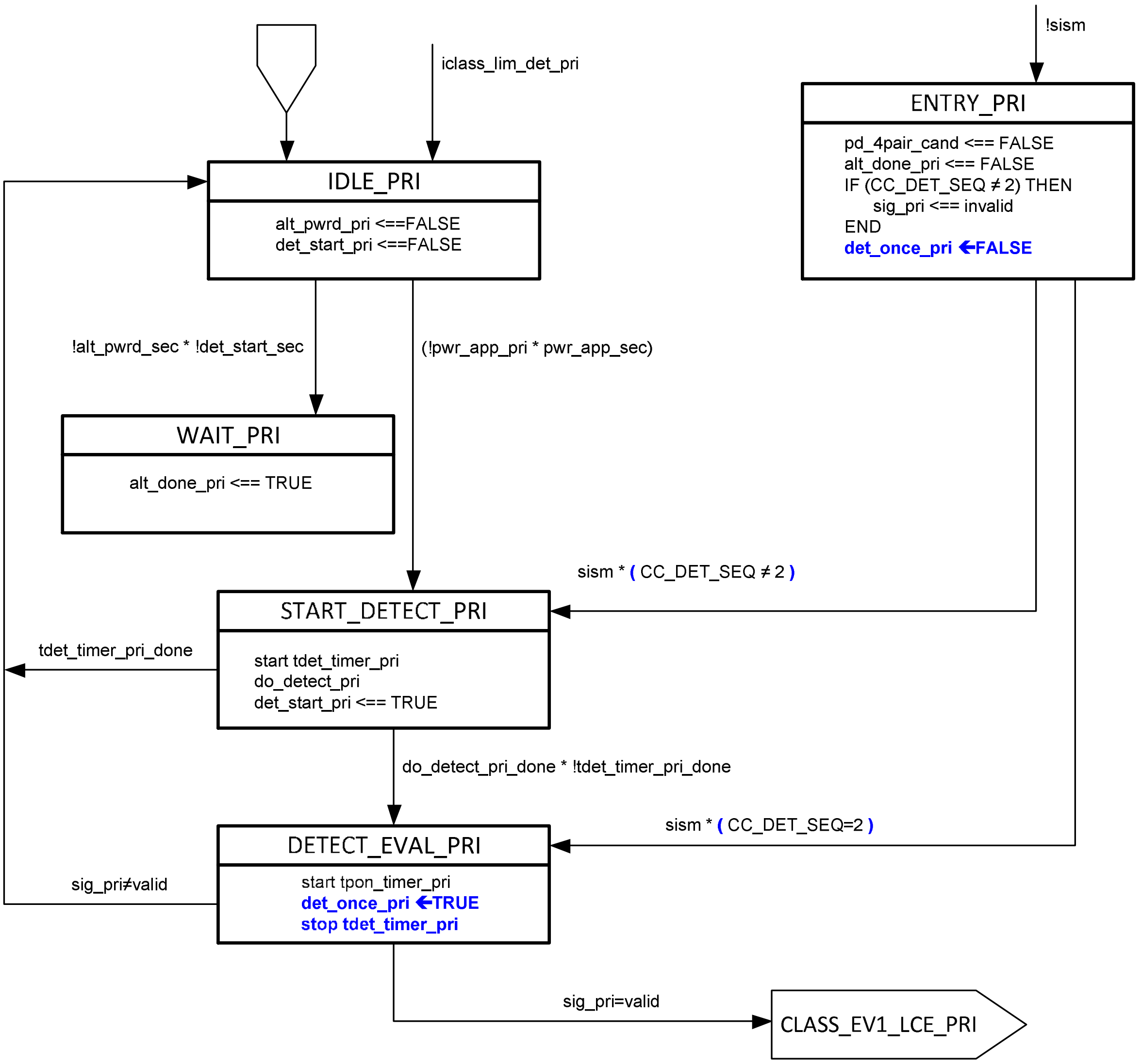
This variable indicates if the PSE has probed the Primary Alternative at least once, when entering to DETECT_EVAL_PRI.

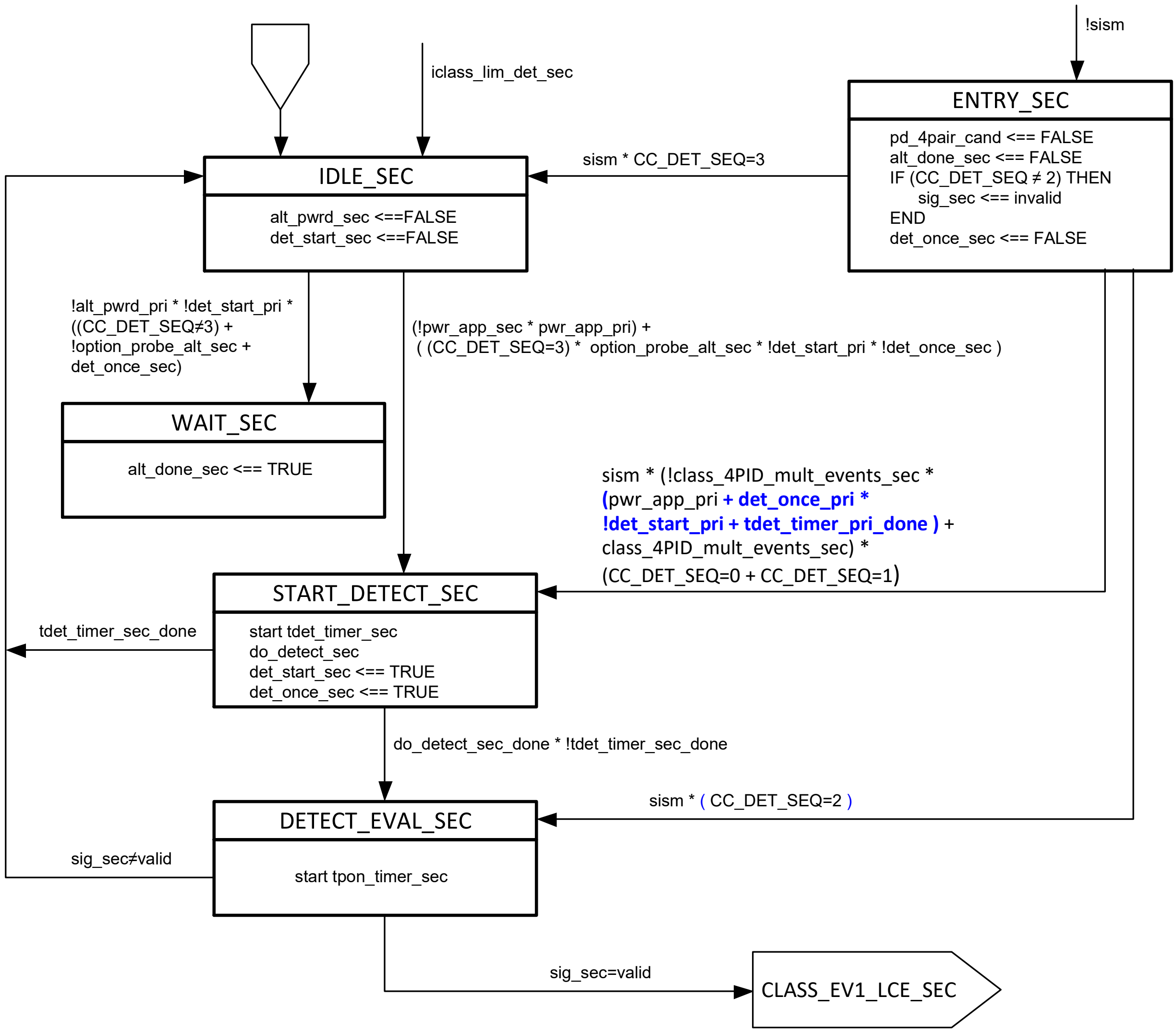
Values:

FALSE: The PSE has not probed on the Primary Alternative since entering the Primary Alternative state diagram.

TRUE: The PSE has probed the Primary Alternative at least once since entering the Primary Alternative state diagram.

2. Make the following changes to the state machine:





3. Properly defined what is "parallel" detection and what is "staggered" detection in CC_DET_SEQ definition on page 109 after line 49. Make the following changes:

CC_DET_SEQ
A constant indicating the sequence in which the PSE performs connection check and detection. See Annex 145B for timing diagrams.
Values: 0: Connection Check is followed by staggered detection for a single-signature PD and parallel detection for a dual-signature PD.
1: Detection on a pairset is followed by connection check and then detection on the other pairset for a single-signature PD and parallel or staggered (starting with first pairset) detection for a dual-signature PD.
2: Connection check and detection on both pairsets are performed within a single T_{det} window.
3: Connection check is followed staggered detection.

For single-signature PD, parallel detection means that detection on both pairsets are done within Tdet time period.
For dual-signature PD, parallel detection means that detection on both pairsets are done within the same Tdet time period.
For single-signature PD, staggered detection means detection on both pairsets are done in different Tdet cycles.
For dual-signature PD, staggered detection means detection on both pairsets are done in different Tdet cycles.

End of Baseline