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:

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:

2-1) failing detection and/or classification in the primary that will cause pwr_app_pri=FALSE when class_4PID_mult_event_sec=FALSE or

2-2) 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

2-3) In addition, there is redundant parenthesis in:
"sism *((!class_4PID_mult_events_sec * pwr_app_pri) + class_4PID_mult_events_sec) * (CC_DET_SEQ=0 + CC_DET_SEQ=1)"

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-250 (Page 136 Line 20).

In the exit from ENTRY_SEC to START_DETECT_SEC:

There is redundant parenthesis in the exit from ENTRY_SEC to START_DETECT_SEC:
"sism *((!class_4PID_mult_events_sec * pwr_app_pri) + class_4PID_mult_events_sec) * (CC_DET_SEQ=0 + CC_DET_SEQ=1)"
in the part: (!class_4PID_mult_events_sec * pwr_app_pri).

Proposed Remedy:
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 + class_4PID_mult_events_sec) * (CC_DET_SEQ=0 + CC_DET_SEQ=1)"

See darshan_04_0917.pdf for additional changes proposed to this condition due to other comments.

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 with the option to go to IDLE_PRI/SEC and WAIT_PRI/SEC and others in this presentation).

Proposed Remedy <u>for comment i-251 only</u> (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.

tdet_timer_pri_done	➔	FALSE	➔	pwr_app_pri remains in FALSE

START_DET_PRI exit to IDLE_PRI ➔

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 proposed solution for this problem is:

1) To add stop_tdet_timer_pri in the DETECT_EVAL_PRI state.

(This action ensures that tdet_timer_pri_done will remain FALSE when moving from START_DETECT_PRI to DETECT_EVAL_PRI. This modification is required since even if we did detection before tdet_timer_pri is expired, we will get tdet_timer_pri_done anyway. This action will enable the usage of tdet_timer_pri_done in the secondary state machine at the exit from ENTRY_SEC to START_DETECT_SEC when we will add this variable in (2).)
2. To add "tdet_timer_pri_done to the condition of the exit from ENTRY_SEC to START_DETECT_SEC as follows:
"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)" . This change will 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.

Proposed Remedy for comment i-252 only (See proposed baseline that addresses all related comments):
1. Add "stop_tdet_timer_pri" to the DETECT_EVAL_PRI state.
2. Add "tdet_timer_pri_done to the condition of the exit from ENTRY_SEC to START_DETECT_SEC by performing the following change:
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 **+ tdet_timer_pri_done)**) + class_4PID_mult_events_sec) * (CC_DET_SEQ=0 + CC_DET_SEQ=1)"

Due to the fact that item 2 need additional changes due to other comments, and in order to meet the requirement that we need single independent comment for each issue which I did here but may cause editor confusion of how to apply the remedies of other comments, see darshan_13_0917.pdf.

Summary for comments (i-250, 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:***

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).

5. ***(i-250): Removing redundant parenthesis from the original condition:***

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).

6. ***Final condition:***

   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 exit from IDLE_SEC to START_DETECT_SEC.

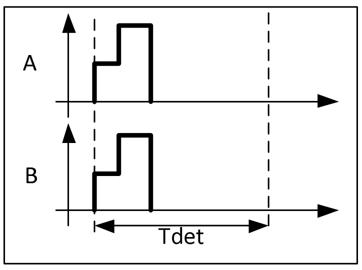
In the exit from IDLE_SEC to START_DETECT_SEC we have the following condition:
(!pwr_app_sec * pwr_app_pri) + ((CC_DET_SEQ=3) * option_probe_alt_sec * !det_start_pri * !det_once_sec.
Based on the description in page 109 lines 37-38 for CC_DET_SEQ and specifically, CC_DET_SEQ=3 for dual-signature means:
Connection check is followed by staggered detection
(The analysis and simulations results for other sequences 0, 1 and 2 are covered by other comments and most of them are OK).

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 per the following presentations and discussions:

a) page 15 of http://www.ieee802.org/3/bt/public/nov14/darshan_11_1114_rev_07.pdf, we introduce the practical definitions of it as part of the way how to find PD configuration (one implementation of connection check).
b) page 7 in http://www.ieee802.org/3/bt/public/jun15/abramson_01bt_0615.pdf

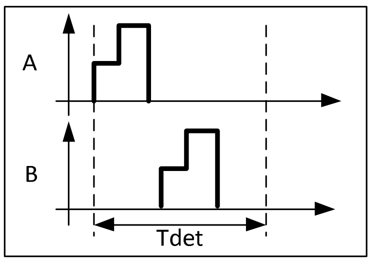
In "parallel detection" the intent was to do simultaneous detection on both pairsets in order to check if detection signature is corrupted when single-signature PD is connected. Now, in D3.0, it it is not clear if it is " simultaneous detection on both pairsets" or detecting both alternative in one detection cycle (Tdet) or both.

Option A



OR

Option B

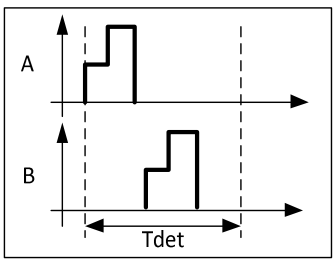


OR BOTH?

We need to define which definition to adopt?

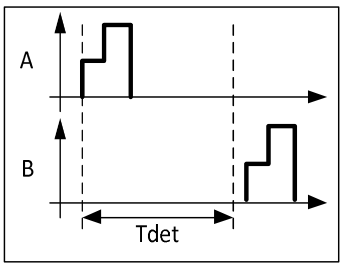
In staggered detection, it should be any non-overlapping detections, with no limits on tdet2ted.
As a result, staggered detection could be any of the following:

Option 1



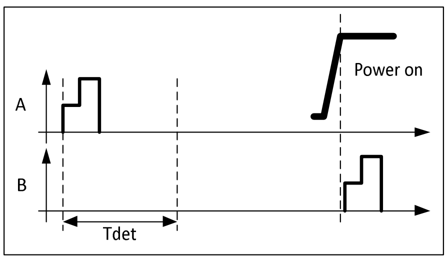
OR

option 2



OR

Option 3



OR ALL?

The last two options must be covered (2 and 3). The first option (1) depends on what is the definition for "parallel" (See above).

Staggered detection range may occur with starting the secondary detection after doing the primary detection (option 1 and 2) up to doing the secondary detection when the primary is on (option 3). This covers the full range of possibilities for staggered detection. Option 1 and 2 in CC_DET_SEQ=3 is normally used when class_4PID_mult_events_sec=TRUE. This currently is not covered by the state machine for CC_DET_SEQ=3.

Option 3 in CC_DET_SEQ=3 is normally used when class_4PID_mult_events_sec=FALSE and it is covered in the 1st part of the condition: (!pwr_app_sec * pwr_app_pri).
Option 3 covers ONLY the case that the primary return to IDLE_PRI due to various reasons and the secondary didn't detect even once: ((CC_DET_SEQ=3) * option_probe_alt_sec * !det_start_pri * !det_once_sec).

The state diagram should allow staggered detection:
-before Primary power up,
-after primary power up,
- and during power up in case that class_4PID_mult_events_sec is set to FALSE.

The proposed changes in the state diagram will allow staggered detection after Primary finished its 1st detection without affecting the previous functionality and flow, by OR-ing the additional missing possibility (option 1 and 2).
The proposed changes do not affect:
a) The behavior of other "CC_DET_SEQ NE 3" flows.
b) Previous state diagram possibilities.

In addition, the proposed changes also required to cover multiple cycles of detection+classification until host decides to power on the port (which is covered by darshan_04_0917.pdf).

The additional missing possibility is covered by adding the following part:
+ (class_4PID_mult_events_sec*(CC_DET_SEQ=3) * !det_once_sec * det_once_pri)
In order to implement the addition, we need to add the following variable for the primary side (similar variable is already existing for the secondary):

"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."

In the above proposed change, det_once_pri is used as a condition for starting detection in the secondary any time, after primary was detected at least once.
det_once_pri is set to FALSE when sism = FALSE at ENTRY_PRI.
det_once_pri is set to TRUE when Primary state diagram reaches to “DETECT_EVAL_PRI”, to clearly indicate that detection on primary has ended before tdet_timer_pri expired.

Proposed Remedy for ***Comment i-254 only***:
1. Change from:
"(!pwr_app_sec *pwr_app_pri) + ((CC_DET_SEQ=3) * option_probe_alt_sec * !det_start_pri * !det_once_sec)"
To:
"(!pwr_app_sec *pwr_app_pri) + ((CC_DET_SEQ=3) * option_probe_alt_sec * !det_start_pri * !det_once_sec) + (class_4PID_mult_events_sec*(CC_DET_SEQ=3) * !det_once_sec * det_once_pri)
2. Add the following variable to the variable list:
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.

Baseline starts here

Proposed Remedy

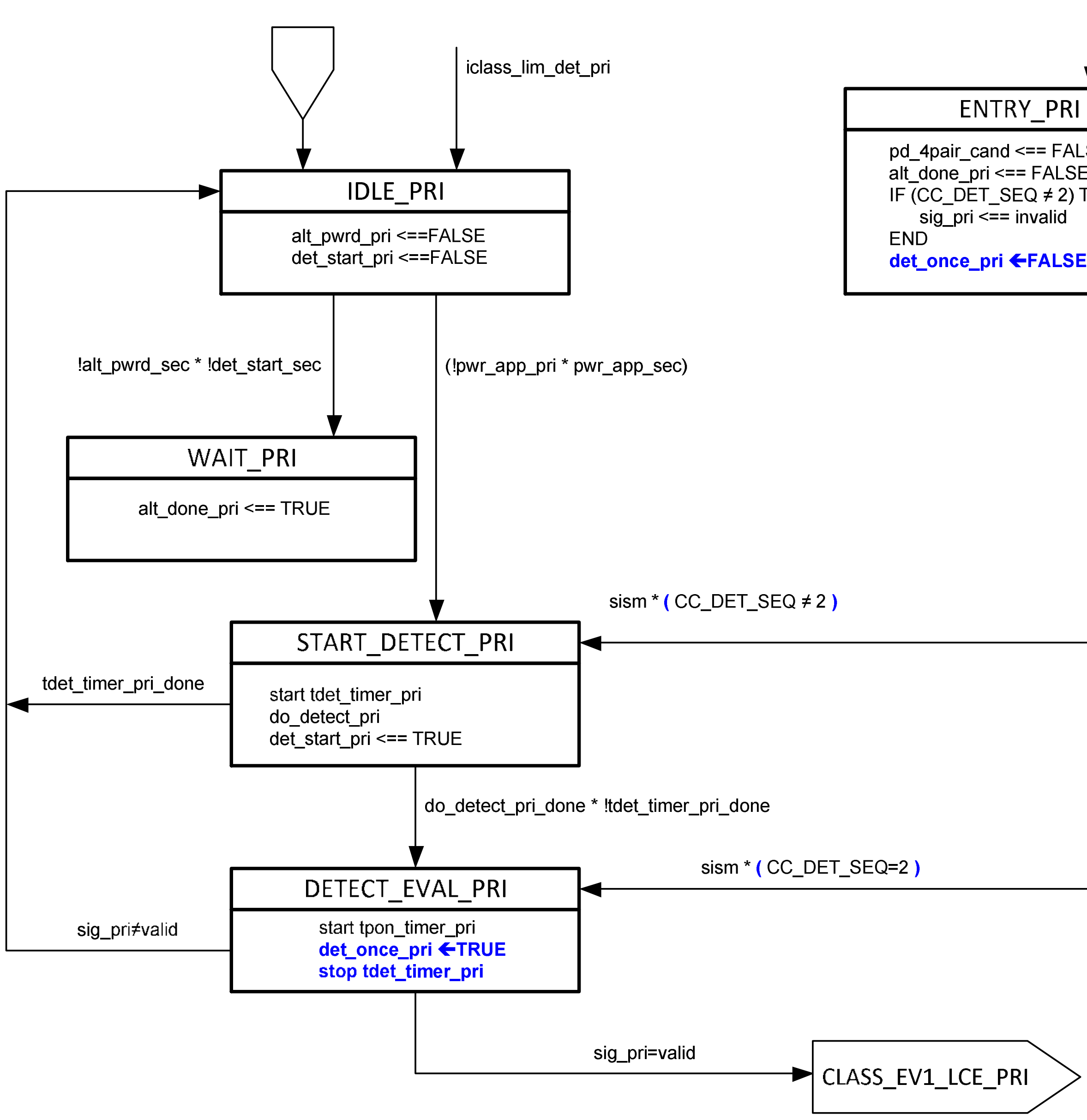
1. Add the following variables to 145.2.5.4

det_once_pri

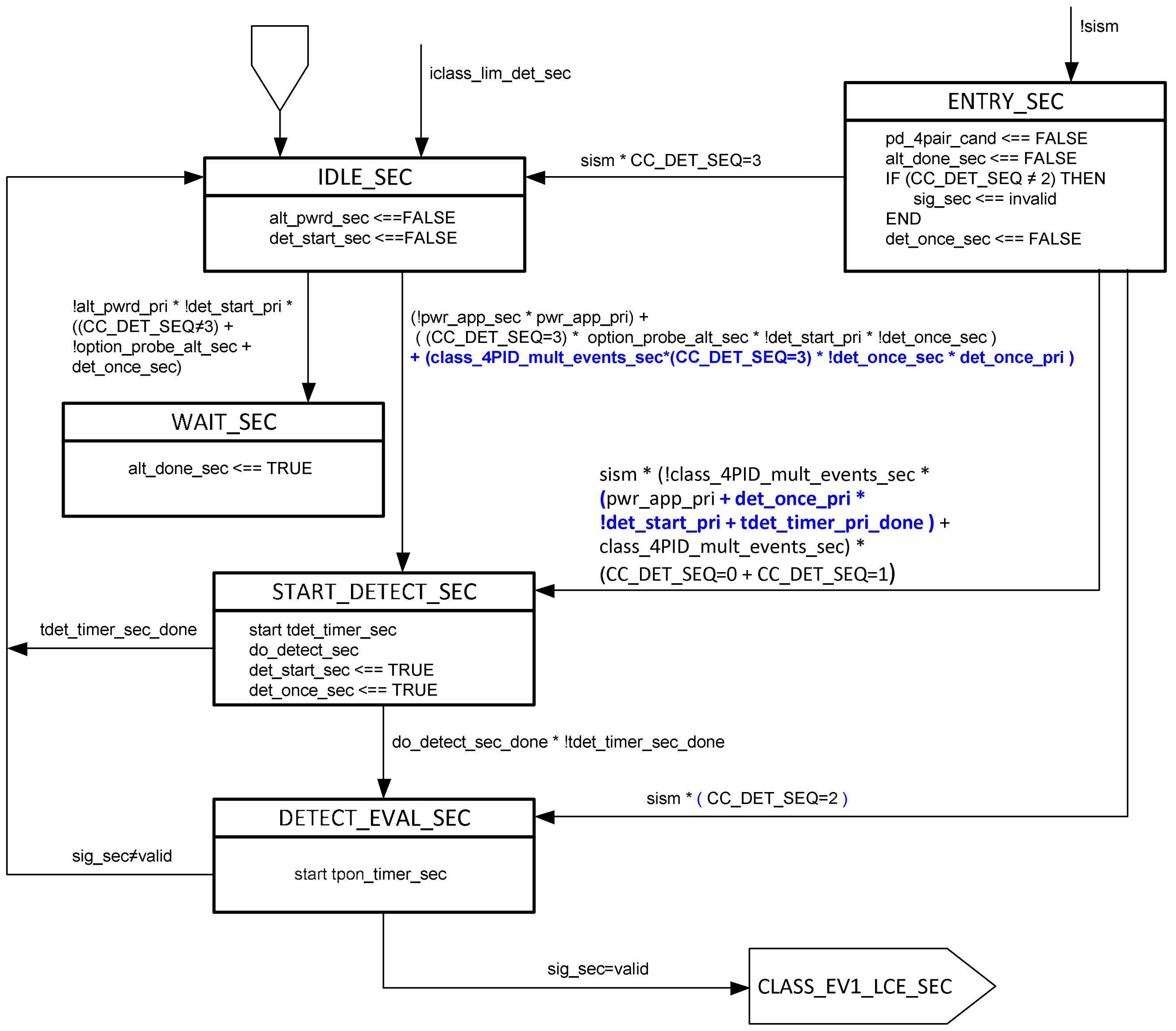
This variable indicates if the PSE has probed the Primary Alternative at least once, when entering to DETECT_EVAL_PRI. Values:
FALSE: The PSE has not probed on the Primary Alternative since entering the Primary Alternative state diagram.
TRUE: The PSE has probed the Primary Alternative at least once since entering the Primary Alternative state diagram.

TRUE: The PSE probes for the PD requested Class.

2. Make the following changes to the state machine:



3. Make the following changes to the state machine:



!sism
iclass_lim_det_sec
ENTRY_SEC
pd_4pair_cand <== FALSE
alt_done_sec <== FALSE
IF (CC_DET_SEQ ≠ 2) THEN
sig_sec <== invalid
END
det_once_sec <== FALSE
sism * CC_DET_SEQ=3
IDLE_SEC
alt_pwrd_sec <==FALSE
det_start_sec <==FALSE
!alt_pwrd_pri * !det_start_pri * ((CC_DET_SEQ≠3) + !option_probe_alt_sec + det_once_sec)
(!pwr_app_sec * pwr_app_pri) + ((CC_DET_SEQ=3) * option_probe_alt_sec * !det_start_pri * !det_once_sec)
+ (class_4PID_mult_events_sec*(CC_DET_SEQ=3) * !det_once_sec * det_once_pri)
WAIT_SEC
alt_done_sec <== TRUE
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)
START_DETECT_SEC
start tdet_timer_sec
do_detect_sec
det_start_sec <== TRUE
det_once_sec <== TRUE
tdet_timer_sec_done
do_detect_sec_done * !tdet_timer_sec_done
sism * (CC_DET_SEQ=2)
DETECT_EVAL_SEC
start tpon_timer_sec
sig_sec≠valid
sig_sec=valid
CLASS_EV1_LCE_SEC

End of Baseline