smaller signal/ variable names

s_c

after_a_var_rst_nFIP_ON_fd_ON_rston_ON

- smaller signal/ variable names
- Remove Synplify warnings
- Remove Synplify specific attributes
- Remove comments for entity/ architecture declarations, libraries?
- Better signal names: rstpon_i, rst_i; either LGTH or LENGTH!
- Solve messss with generics and constants; g_ for all generics, c_ for all constants
- Coherent architecture names (behavioral, rtl)
- Signal declaration in a logic way! signal names not consistent in different units
- Reduce the number of signals, they are really messing up the code:-s
- Grouping ports by interface or direction?
- manchester encoder should be a function
- Avoid using complex data types in entity ports (var, counter)
- State machine outputs style :-s

wf_incr_counter

nfip_rst_i and reinit_counter_i do exactly the same thing. Remove nfip_rst_i.

wf_wb_controller

Check that wb_we = '0' to ack a read transaction!!

“reading/writing from an invalid address will hang the bus by not ack-ing the transaction. Is it the expected behavior? “

wf_cons_frame_validator

Replace

```
if rx_fss_crc_fes_manch_ok_p_i = '1' then                    -- checking the RP_DAT.Data.Length
                                                             -- byte, when the FES arrives.
  if unsigned(rx_byte_index_i ) = (unsigned(cons_lgth_byte_i) + 5) then
    s_cons_lgth_byte_ok <= '1';                              -- rx_byte_index starts counting
                                                             -- from 0 and apart from the
                                                             -- user-data bytes, also counts the
  else                                                       -- Control, PDU_TYPE, Length,
                                                             -- the 2 CRC and the FES bytes
    s_cons_lgth_byte_ok <= '0';
  end if;
```

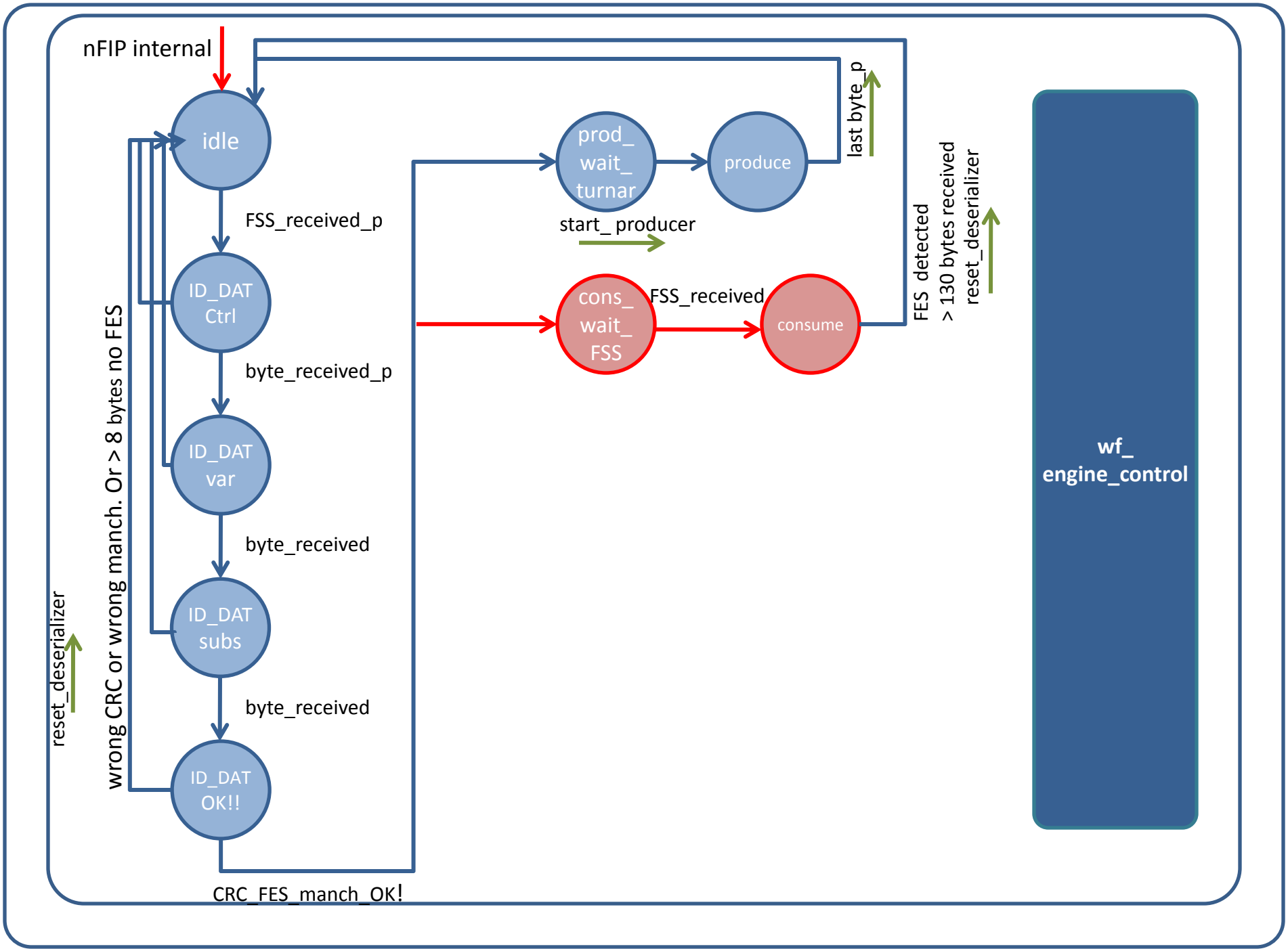
With

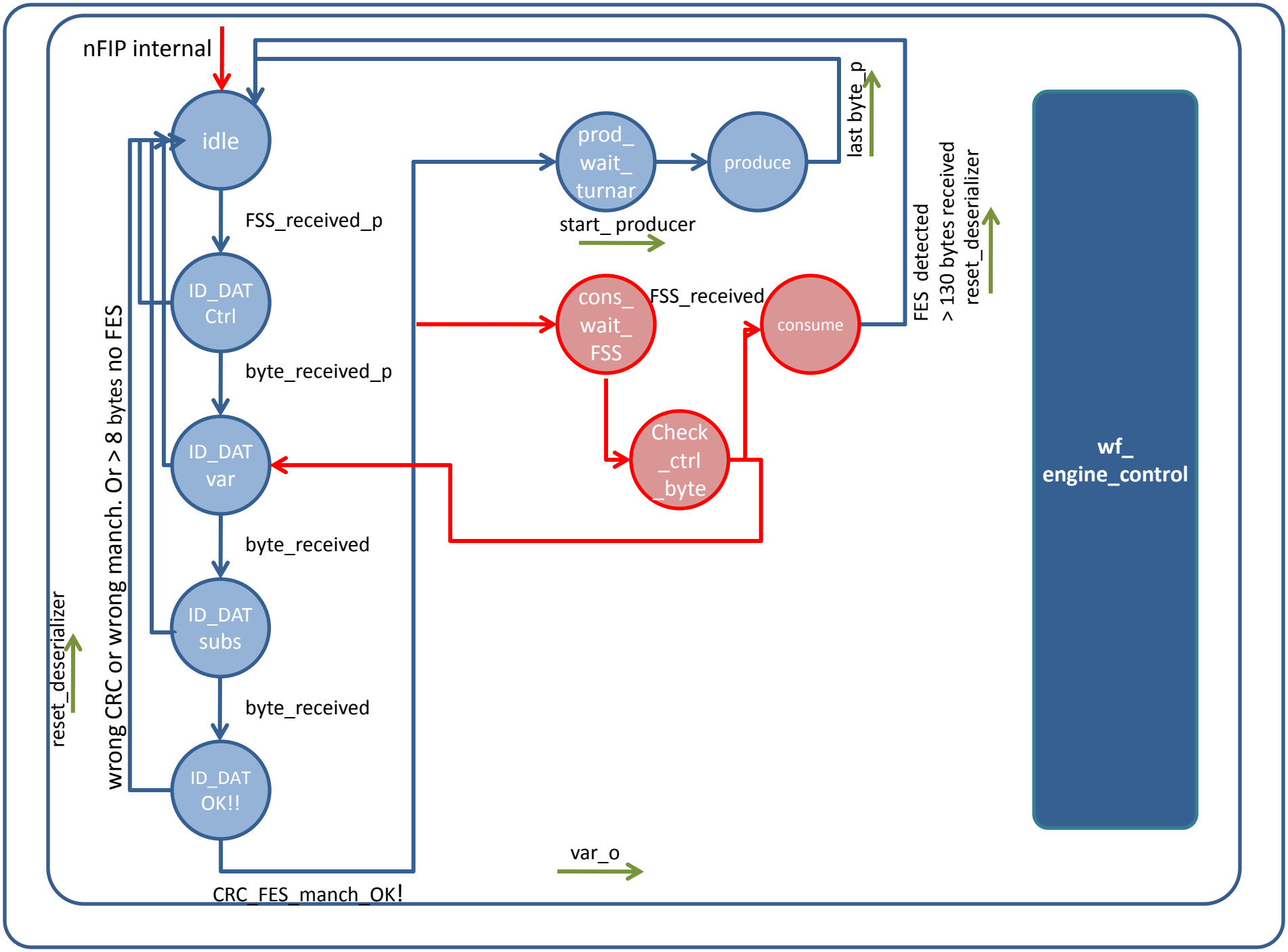
```
if rx_fss_crc_fes_manch_ok_p_i = '1' or crc_manch_wrong_p_i = '1' then -- checking the
                                                                       -- RP_DAT.Data.Length byte,
                                                                       -- when the FES arrives.
  if unsigned(rx_byte_index_i ) = (unsigned(cons_lgth_byte_i) + 5) then
    s_cons_lgth_byte_ok <= '1';                              -- rx_byte_index starts counting
                                                             -- from 0 and apart from the
                                                             -- user-data bytes, also counts the
  else                                                       -- Control, PDU_TYPE, Length,
                                                             -- the 2 CRC and the FES bytes
    s_cons_lgth_byte_ok <= '0';
  end if;
```

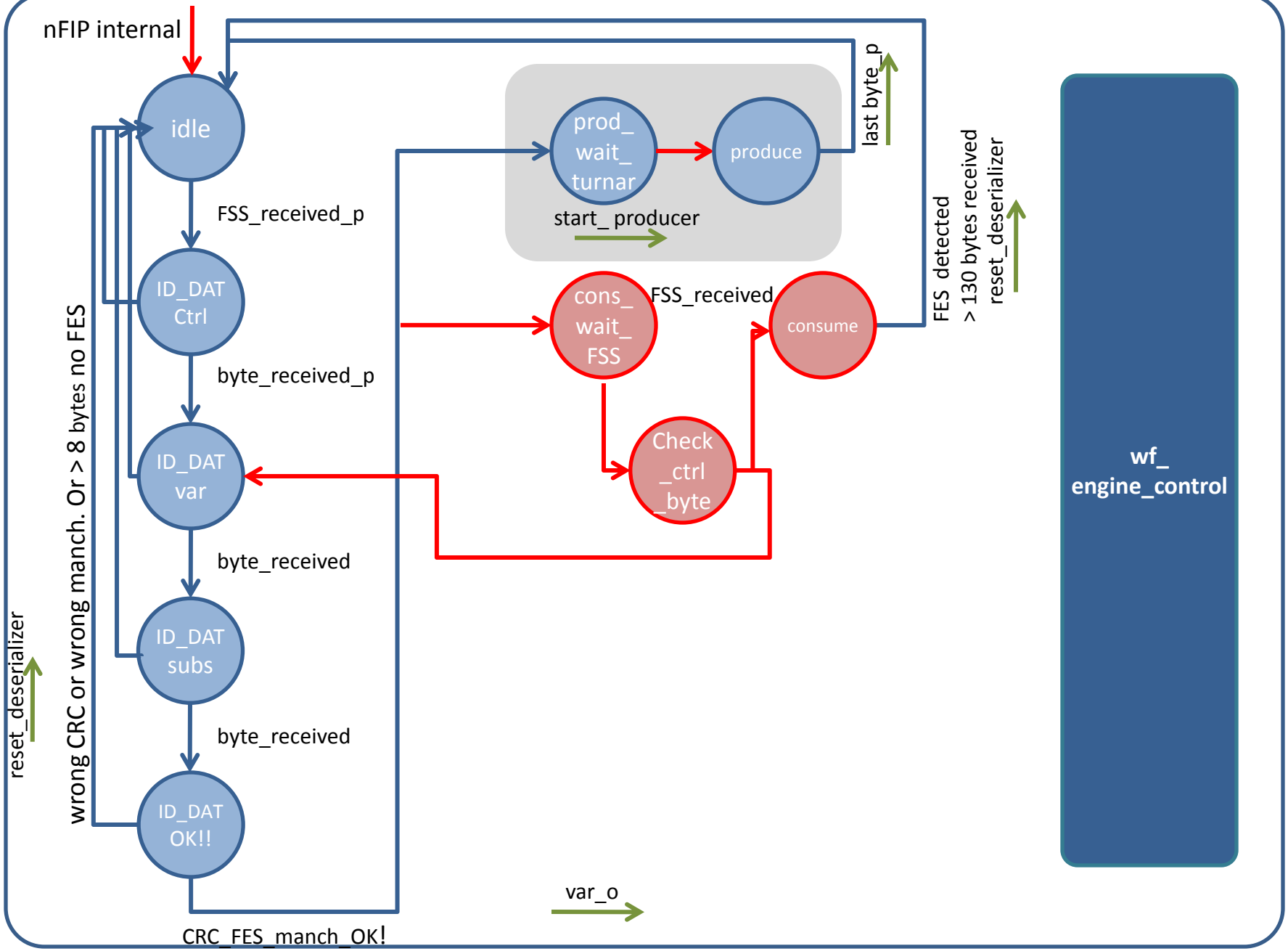
Remove checking of Ctrl and PDU_TYPE bytes; check them directly at reception?

wf_engine_control

Receiving 2 or more consecutive ID_DATs:


nFIP internal
idle
FSS_received_p
ID_DAT Ctrl
byte_received_p
ID_DAT var
byte_received
ID_DAT subs
byte_received
ID_DAT OK!!
CRC_FES_manch_OK!
wrong CRC or wrong manch. Or > 8 bytes no FES
reset_deserializer
prod_ wait_ turnar
produce
last byte_p
start_ producer
cons_ wait_ FSS
FSS_received
consume
FES detected
> 130 bytes received
reset_deserializer
wf_ engine_control


nFIP internal
idle
FSS_received_p
ID_DAT Ctrl
byte_received_p
ID_DAT var
byte_received
ID_DAT subs
byte_received
ID_DAT OK!!
CRC_FES_manch_OK!
wrong CRC or wrong manch. Or > 8 bytes no FES
reset_deserializer
prod_ wait_ turnar
produce
last byte_p
start_ producer
cons_ wait_ FSS
FSS_received
consume
Check _ctrl _byte
FES detected
> 130 bytes received
reset_deserializer
var_o
wf_ engine_control


nFIP internal
idle
FSS_received_p
ID_DAT Ctrl
byte_received_p
ID_DAT var
byte_received
ID_DAT subs
byte_received
ID_DAT OK!!
CRC_FES_manch_OK!
wrong CRC or wrong manch. Or > 8 bytes no FES
reset_deserializer
prod_ wait_ turnar
produce
start_ producer
last byte_p
cons_ wait_ FSS
FSS_received
consume
Check _ctrl _byte
FES detected
> 130 bytes received
reset_deserializer
wf_ engine_control
var_o

wf_engine_control

Receiving 2 or more consecutive ID_DATs:

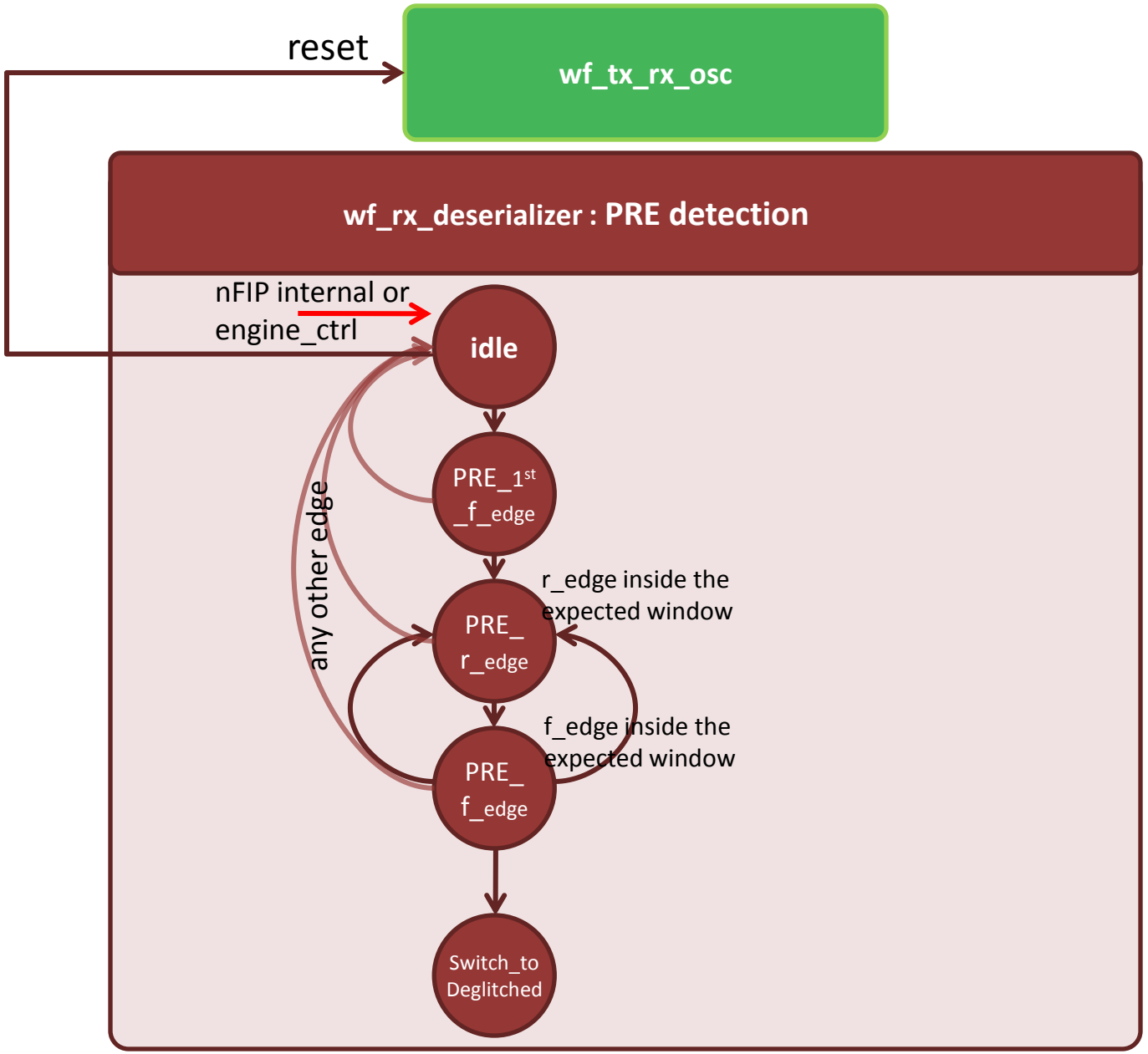
Remove “for loops” for var identification!

Check for FES (not only byte_ready_p) in all the states!

Timeouts relying only to system clock!

wf_rx_deserializer

*“POSSIBLE SERIOUS BUG: The bit-window locking seems to be using the first transition in the RXD signal to synchronize the counter in rx_tx_osc. A small glitch in the signal just before the preamble could cause a valid frame to be dropped because of an invalid data window. RX clock should be locked *after* receiving the preamble, not before - this is the purpose of preambles in all data links.”*


reset
wf_tx_rx_osc
wf_rx_deserializer : PRE detection
nFIP internal or
engine_ctrl
idle
PRE_1st
_f_edge
any other edge
r_edge inside the
expected window
PRE_
r_edge
f_edge inside the
expected window
PRE_
f_edge
Switch_to
Deglitched

wf_rx_deserializer

*“POSSIBLE SERIOUS BUG: The bit-window locking seems to be using the first transition in the RXD signal to synchronize the counter in rx_tx_osc. A small glitch in the signal just before the preamble could cause a valid frame to be dropped because of an invalid data window. RX clock should be locked *after* receiving the preamble, not before - this is the purpose of preambles in all data links.”*

Condition to return to idle in switch_to_deglitched state!!

“I wonder if we should have an additional condition which takes this state machine to Idle, namely the wfip link being inactive, i.e. maintaining a constant level for a long (to be defined)time. This would ensure the state machine is always sitting in the Idle state before the beginning of a frame. With the current implementation, it might well be the case, but there are manyscenarios to analyze and we risk forgetting something.”

wf_tx_rx_osc

"When the first falling edge arrives (or the following significant edges) s_rx_counter is reset and start counting. And the significant edge window is valid as long as s_rx_counter is smaller than s_jitter. It means that just after an edge if another comes before s_rx_counter = s_jitter, the edge is valid."

```
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
-- regarding significant edges:

-- looking for a significant edge inside the corresponding window
if (s_rxd_signif_edge_window = '1') and (rxd_edge_p_i = '1') then

    s_rx_manch_clk          <= not s_rx_manch_clk; -- inversion of rx_manch_clk
    s_signif_edge_found     <= '1';                -- indication that the edge was found
    s_adjac_bits_edge_found <= '0';

-- if a significant edge is not found where expected (code violation), the rx_manch_clk
-- is inverted right after the end of the signif_edge_window.
elsif (s_signif_edge_found = '0') and (s_rx_counter = s_jitter) then

  s_rx_manch_clk            <= not s_rx_manch_clk;
  s_adjac_bits_edge_found   <= '0';                -- re-initialization before the
                                                   -- next cycle
```

wf_tx_rx_osc

"When the first falling edge arrives (or the following significant edges) s_rx_counter is reset and start counting. And the significant edge window is valid as long as s_rx_counter is smaller than s_jitter. It means that just after an edge if another comes before s_rx_counter = s_jitter, the edge is valid."

```
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
-- regarding significant edges:

-- looking for a significant edge inside the corresponding window
if (s_rxd_signif_edge_window = '1') and (rxd_edge_p_i = '1') and (s_signif_edge_found = '0') then

    s_rx_manch_clk          <= not s_rx_manch_clk; -- inversion of rx_manch_clk
    s_signif_edge_found     <= '1';                -- indication that the edge was found
    s_adjac_bits_edge_found <= '0';

-- if a significant edge is not found where expected (code violation), the rx_manch_clk
-- is inverted right after the end of the signif_edge_window.
elsif (s_signif_edge_found = '0') and (s_rx_counter = s_jitter) then

  s_rx_manch_clk          <= not s_rx_manch_clk;
  s_adjac_bits_edge_found <= '0';                -- re-initialization before the
                                                 -- next cycle
```

wf_tx_rx_osc

“When the first falling edge arrives (or the following significant edges) s_rx_counter is reset and start counting. And the significant edge window is valid as long as s_rx_counter is smaller than s_jitter. It means that just after an edge if another comes before s_rx_counter = s_jitter, the edge is valid.”

```
-- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -
-- regarding significant edges:

-- looking for a significant edge inside the corresponding window
if (s_rxd_signif_edge_window = '1') and (rxd_edge_p_i = '1') and (s_signif_edge_found = '0') then

    s_rx_manch_clk          <= not s_rx_manch_clk; -- inversion of rx_manch_clk
    s_signif_edge_found     <= '1';                -- indication that the edge was found
    s_adjac_bits_edge_found <= '0';

-- if a significant edge is not found where expected (code violation), the rx_manch_clk
-- is inverted right after the end of the signif_edge_window.
elsif (s_signif_edge_found = '0') and (s_rx_counter = s_jitter) then

  s_rx_manch_clk          <= not s_rx_manch_clk;
  s_adjac_bits_edge_found <= '0';                -- re-initialization before the
                                                 -- next cycle
```

Optimizations!

wf_crc

s_q_check_mask not in sensitivity list!!

Replace “xor” with “if”!

“avoid driving outputs with comb logic unless it's really justified. Here you can move the CRC comparison to the sequential process above”
→it’s clocked in the wf_rx_deserializer

“crc_ok_p is not registered to reduce the number of flip flops, remember this is a rad-hard design”
→TMR?

wf_inputs_synchronizer

Same style for synchronization!

Clarify FD_RXCDN & maybe remove FD_RXD filtering from here
-> finally, remove completely use of FD_RXCDN to make nanoFIP independent of Alstom's FIELDRIVE

Robust edge detection with 7 flip-flops!

Inhibit detection for a while after a detected edge!

Remove synchronization of DAT_I in slone mode

Remove synchronization of WorldFIP settings

TMR & Synchronization: TMR everything

We know all inputs are well registered. Are all outputs registered as well?

Almost!

Wf_model_constr_decoder:

```
------------------------------------------------------------------------------------------------
--
--! @todo
--!  -> select_id_o not the output of a dff:-s
--
------------------------------------------------------------------------------------------------
```

Outcome – Summary of Basic changes

- *Direct deglitching of FD_RXD (also the PRE) independent of the wf_rx_osc*
- *Timeouts for all state machines*


THANK YOU