System Modelling of a large FPGA project: the SKA Tile Processing Module

Carolina Belli

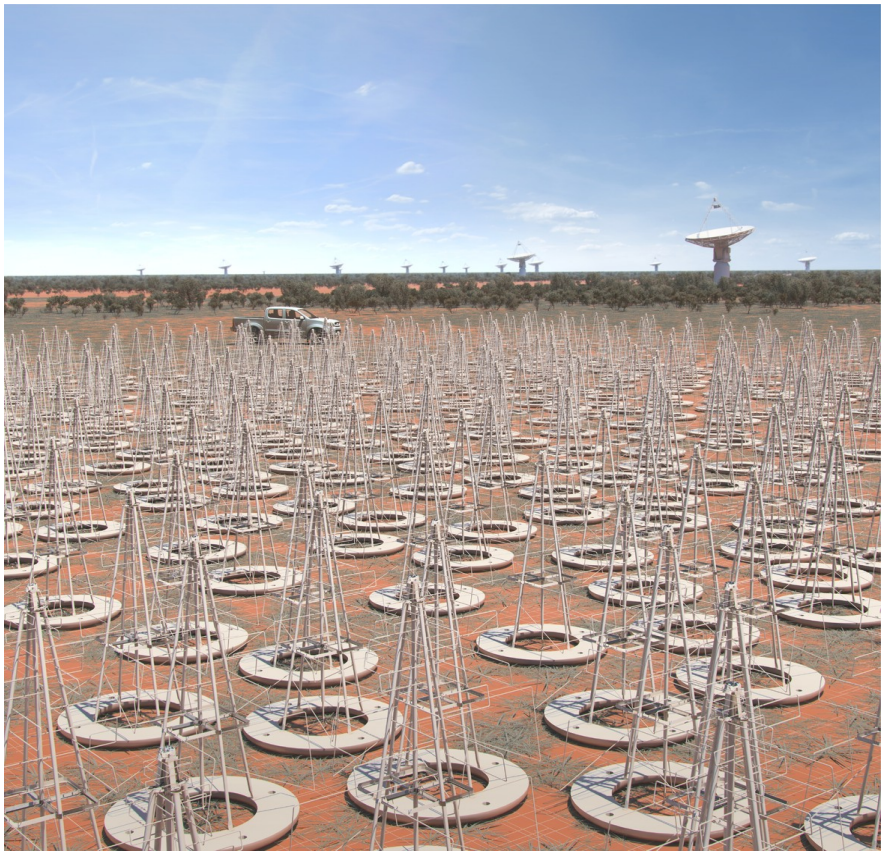
Applicazioni FPGA in ambito Astrofisico – Torino, 18-20 Maggio 2016

SKA – Low Frequency Aperture Array (LFAA)

SKA: large scale project with remarkable scientific relevance.

The "Low-Frequency Aperture Array" (LFAA) element is the set of antennas, of board amplifiers and local processing required for the Aperture Array telescope of SKA.

- *The LFAA covers the lowest frequency band for the SKA telescope: 50MHz - 350MHz.*
- *LFAA is an all-electronic telescope, based on stationary antennas and having a capability enabled by advanced signal processing and computing.*



Source: www.skatelescope.org

System Modelling Language (SysML)

- *Unified Modelling Language (UML): general-purpose, developmental, modeling language in the field of software engineering, it is intended to provide a standard way to visualize the design of a system.*
- SysML: UML's customization for engineering applications.
- SysML main characteristics and advantages:
  - Interdisciplinary approach;
  - Identification of customer needs and project's main functionalities from the very beginning of the development cycle;
  - Business and technical needs are considered in parallel during the design (e.g. costs and schedule, required and expected performances) at different scale levels;
  - The model represents the system in its totality and consistency – a change in one of the diagrams entails changes in all the connected ones.

Source: The Unified Modelling Language User Guide, (2 ed.). Addison-Wesley. 2005.

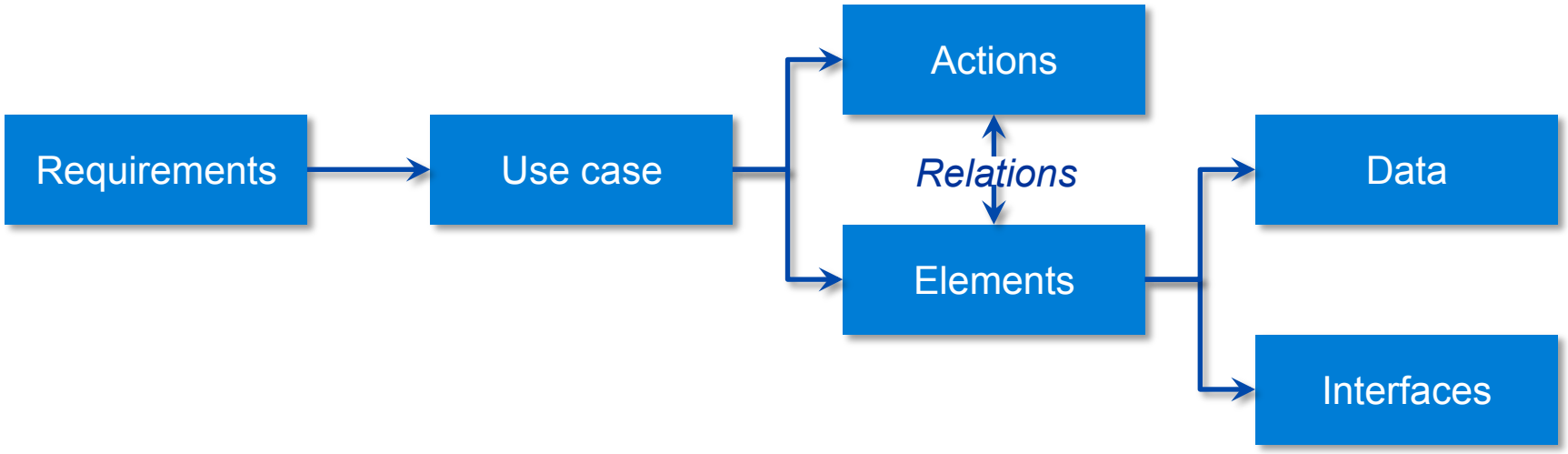
SysML application to SKA

Main step in the application of SysML to SKA:

- Science requirements of the SKA:
  - What the SKA must do;
  - How the SKA will perform.
- Applicability in the real world:
  - SKA must be built in the real world;
  - SKA will be operated by humans;
  - SKA will use existing or projected technology;
  - SKA will exist within a legal framework;
  - SKA must respect the environment.
- Iterative process: additional requirements and constraints to be taken into account during SKA designing.

SysML approach to system design

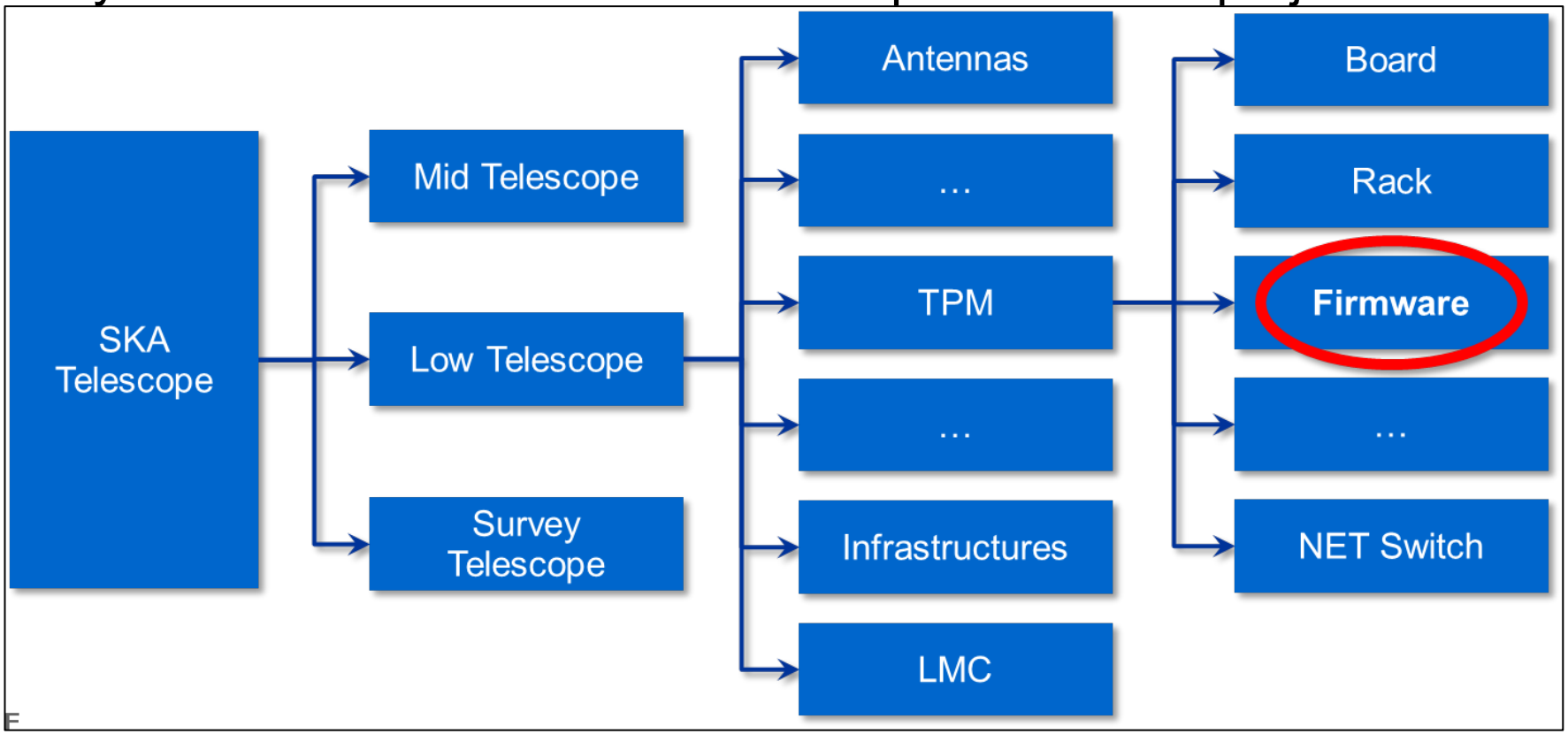
- From requirements to data:
  - Identification of requirements;
  - Development of system Use Case;
  - Definition of main functionality: elements and actions undertaken by elements;
  - Definition of elements characteristics: data and interfaces between elements.



SysML application to LFAA

SKA Low Telescope – SysML model:

- Composed by several diagrams.
- Highlight different views of the same aspects.
- Identify connections between different aspects and the project.

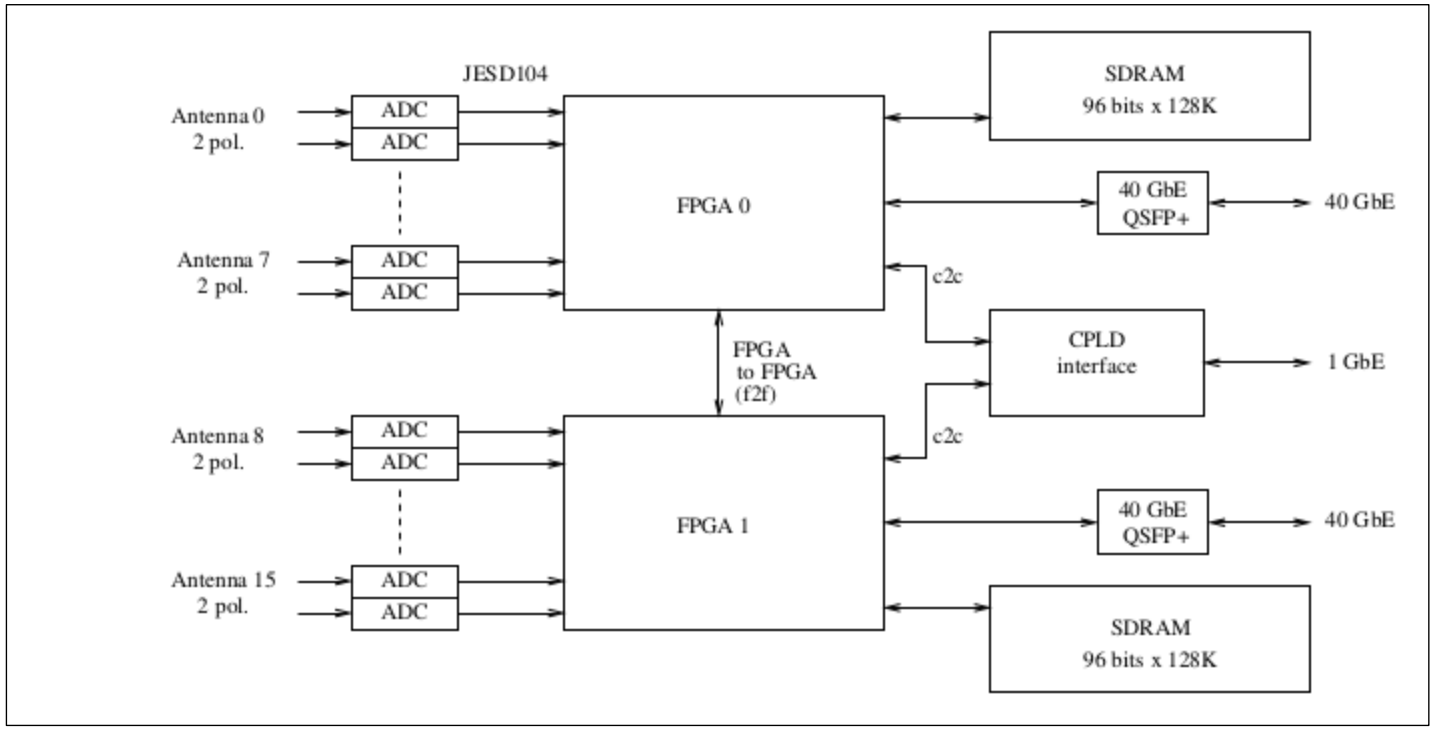


Structural decomposition

Tile Processing Module (TPM)

LFAA Station:

- 256 antennas: 16 tiles of 16 antennas.
- Each tile: served by a Tile Processing Module (TPM);
- Each TPM served by two FPGAs.



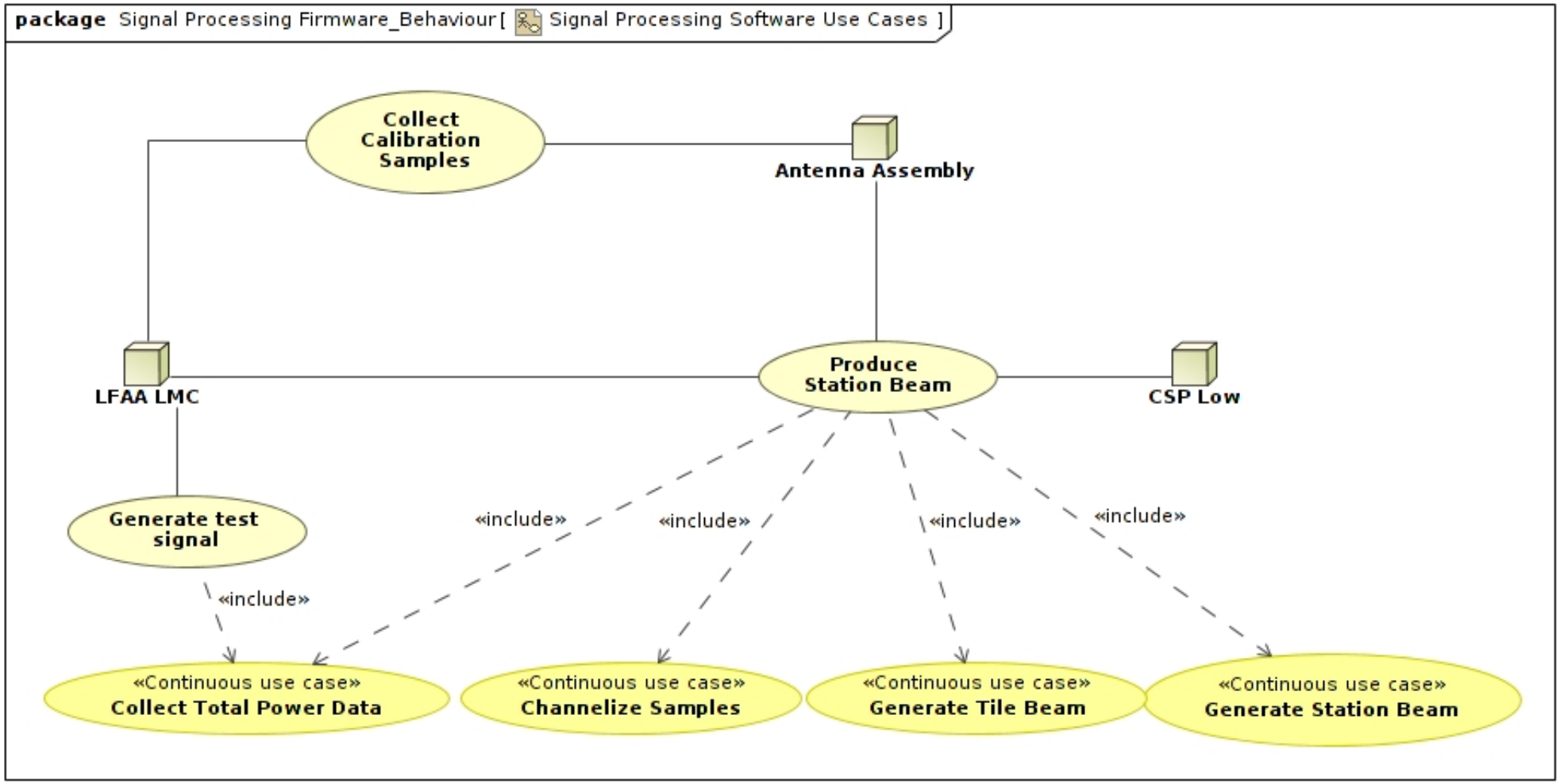
Tile Processing Module structure

TPM diagrams

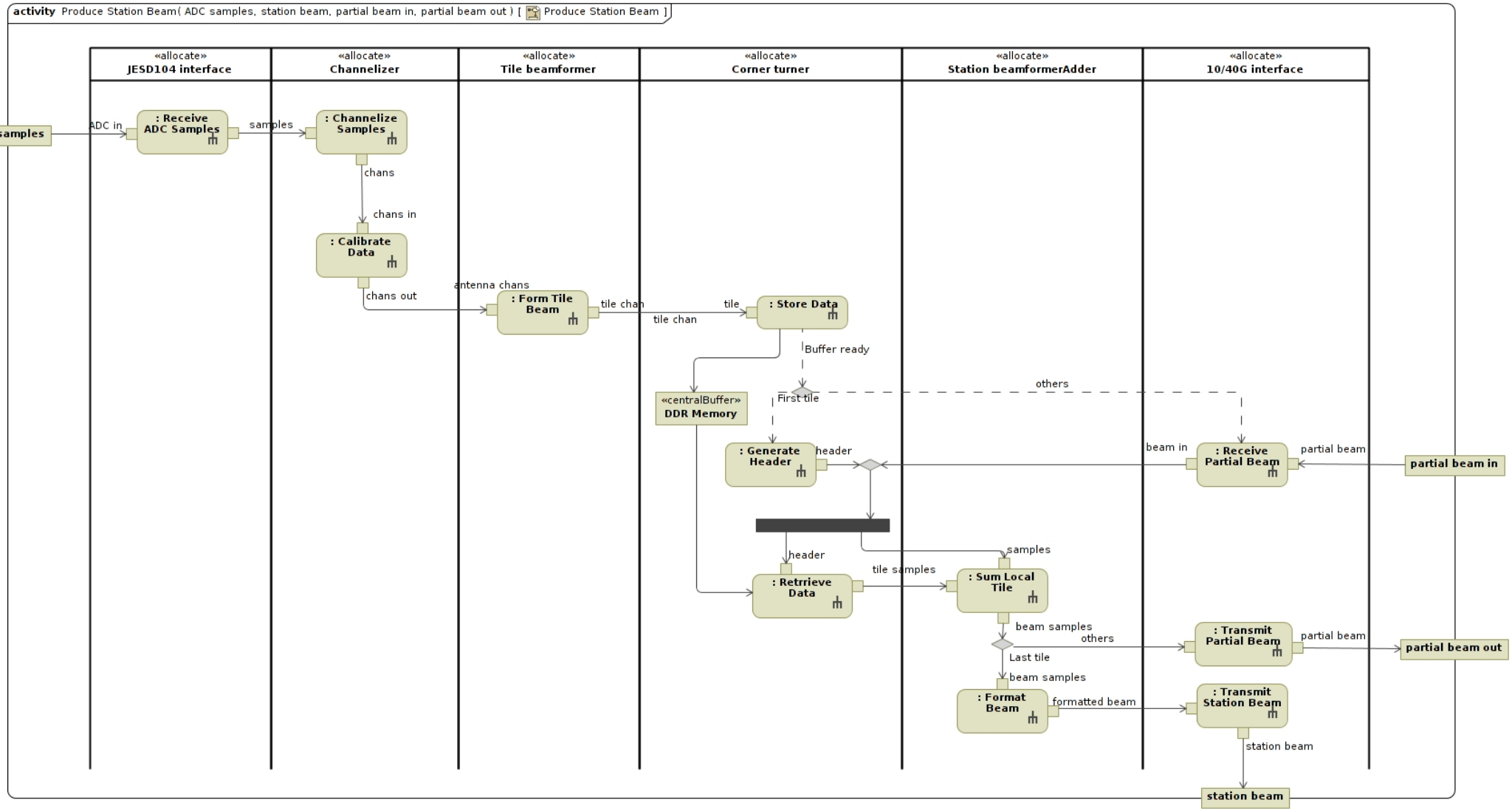
Up to now, the TPM model include the following types of diagram for parts and sub-parts of the project:

- Use Case Diagram (uc): UC diagrams document how a system will be used, identifying which are the actors and how they interact with the system.
- Activity Diagram (act): ACT diagrams document what happens when a use case is executed. They can show alternatives and parallel activities within a workflow.
- Block Definition Diagram (bdd): BDDs describe the architecture of a system and represent the system hierarchy in terms of system and sub-systems. They are constituted by blocks, defining their features and any relationships between them.
- Internal Block Diagram (ibd): IBDs capture the internal structure of a block in terms of properties and connections among properties.

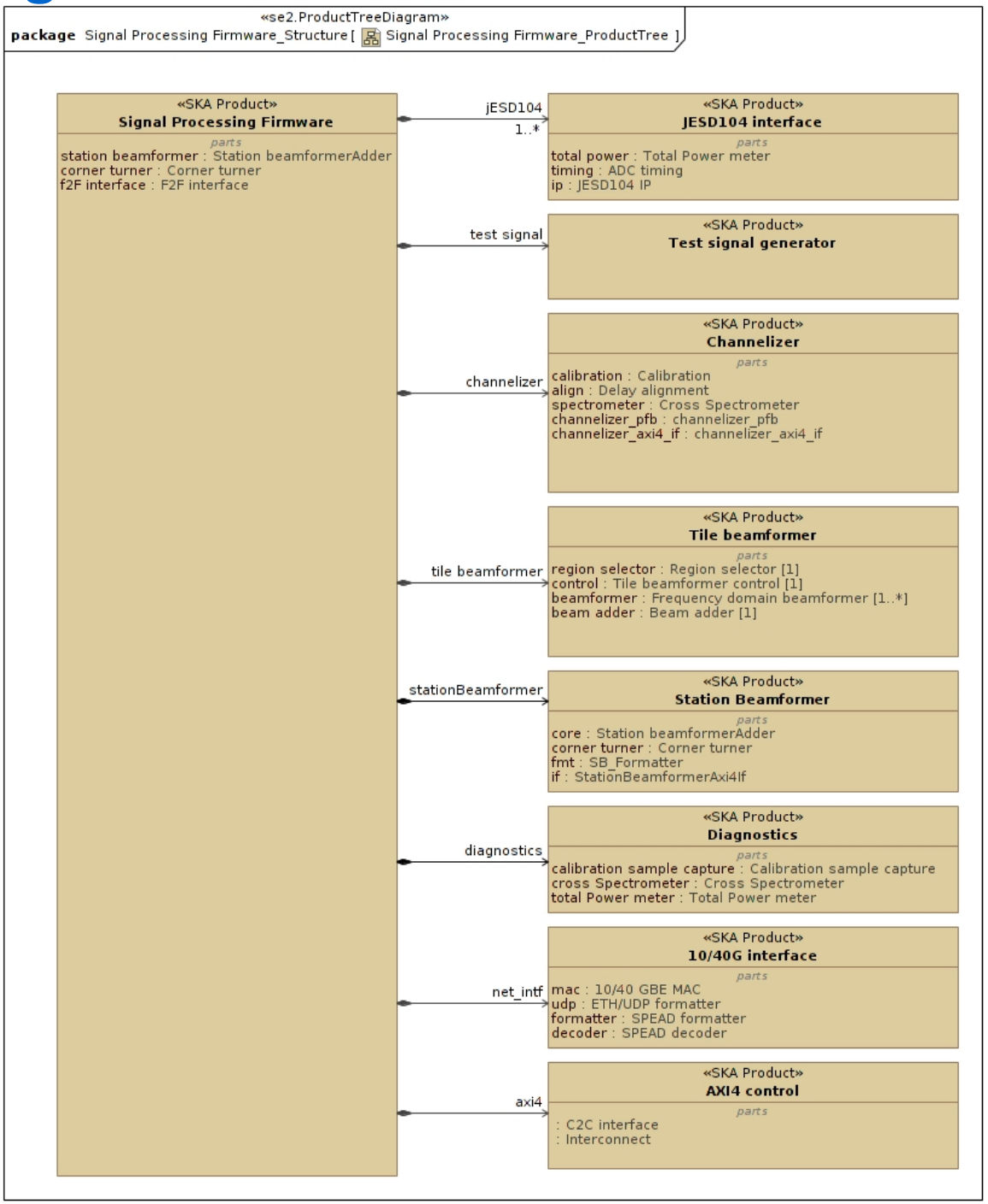
Tile Processing Module – Use Case Diagram



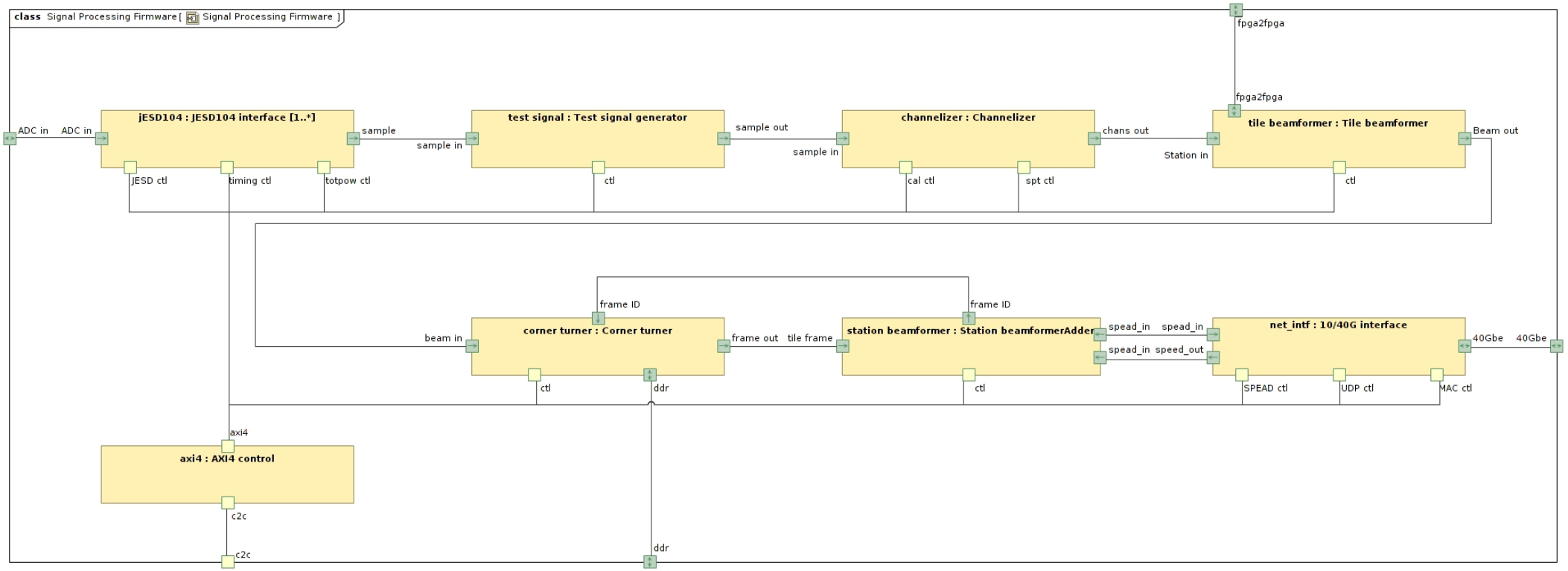
Tile Processing Module – Activity Diagram



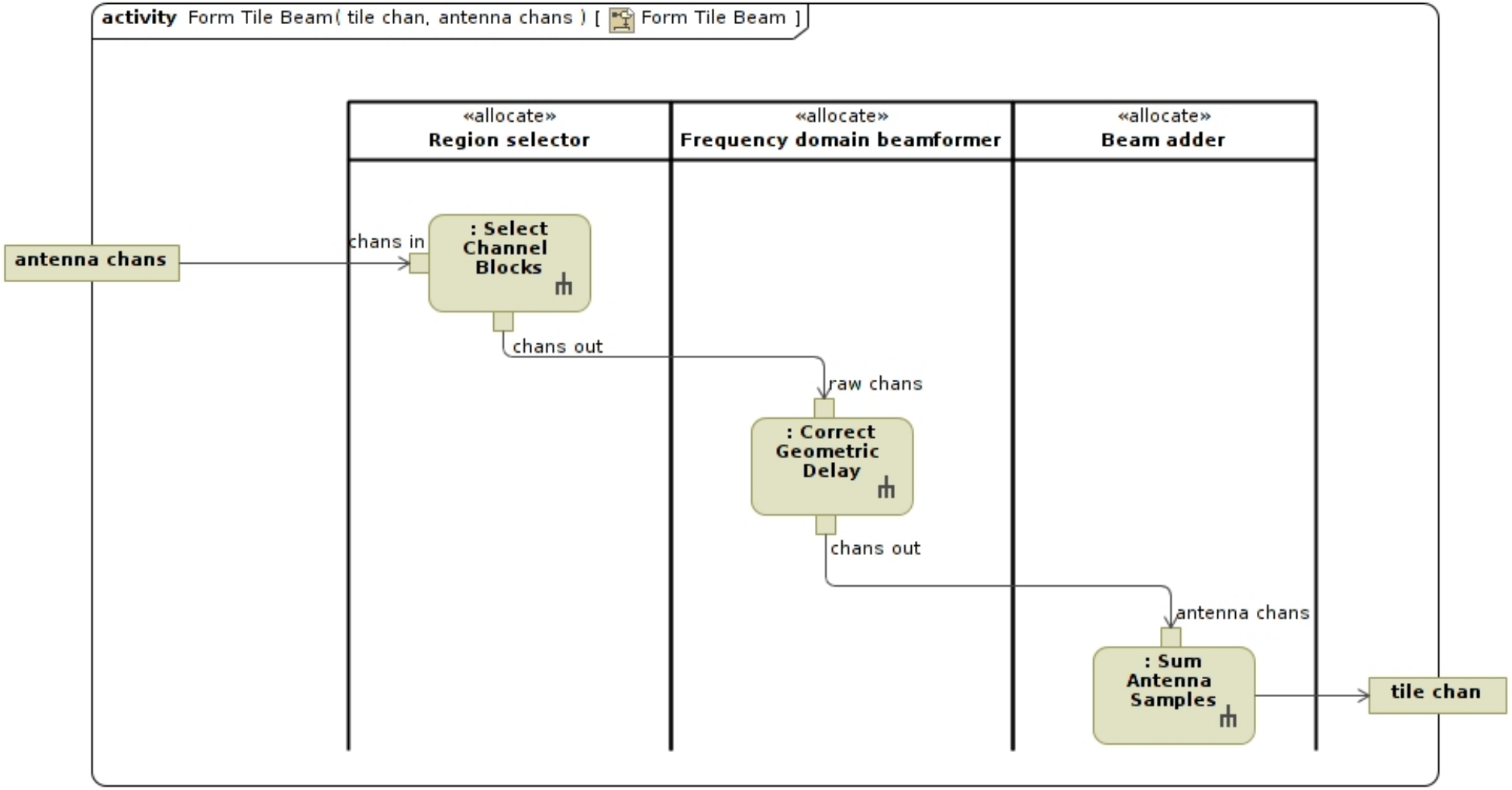
Tile Processing Module – Block Definition Diagram



Tile Processing Module – Internal Block Diagram

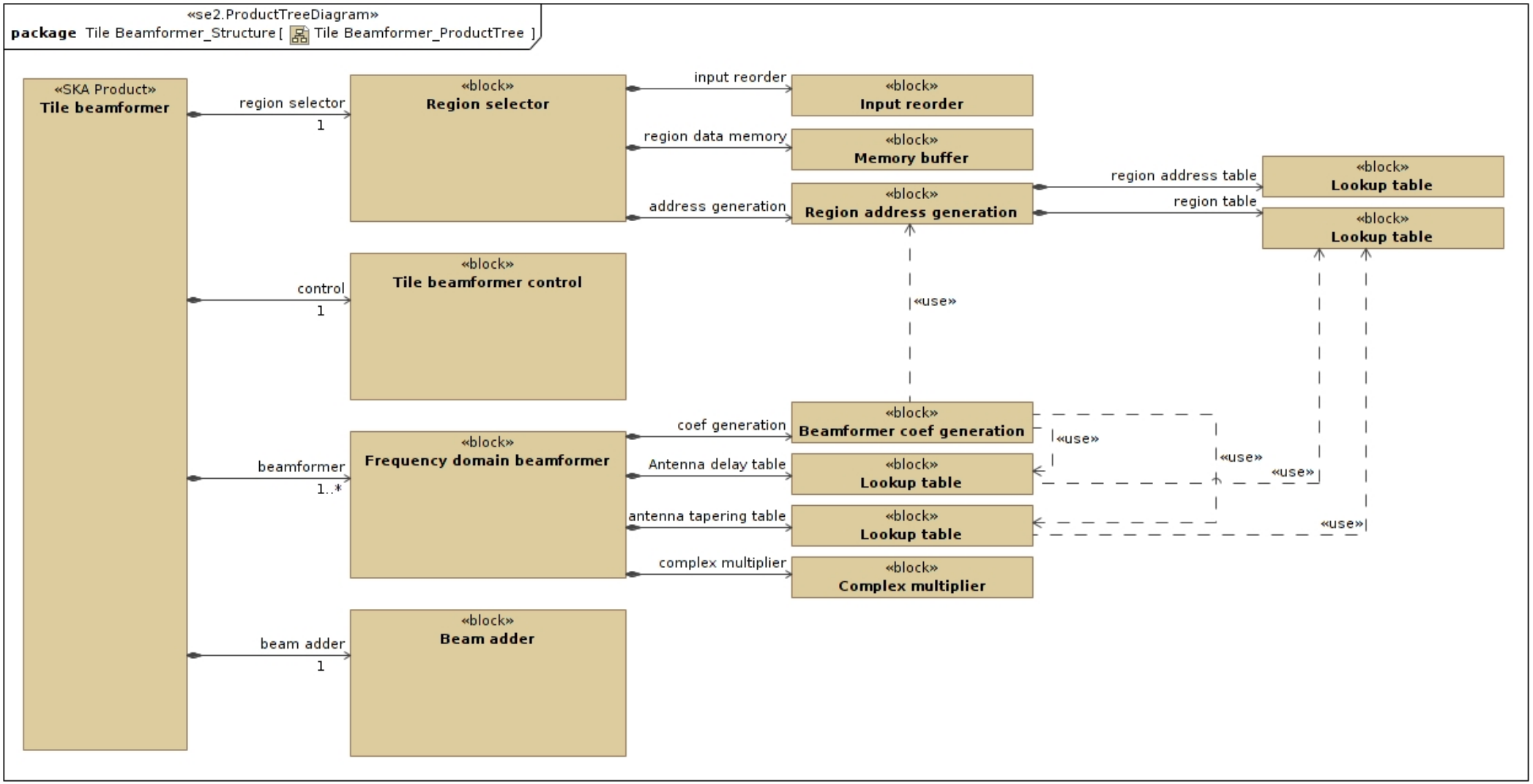


Tile Beamformer – Activity Diagram

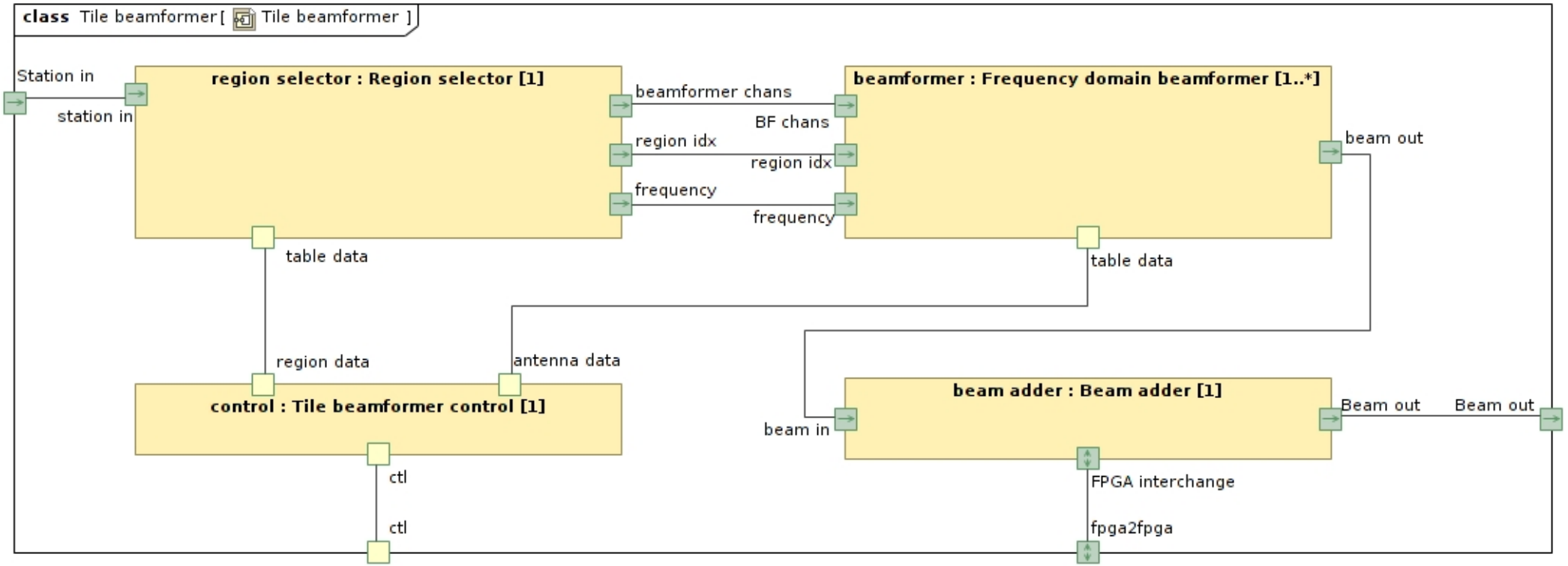


$$out(freq) = \sum_k exp(i \cdot \tau(k) \cdot freq) \cdot x(k, freq)$$

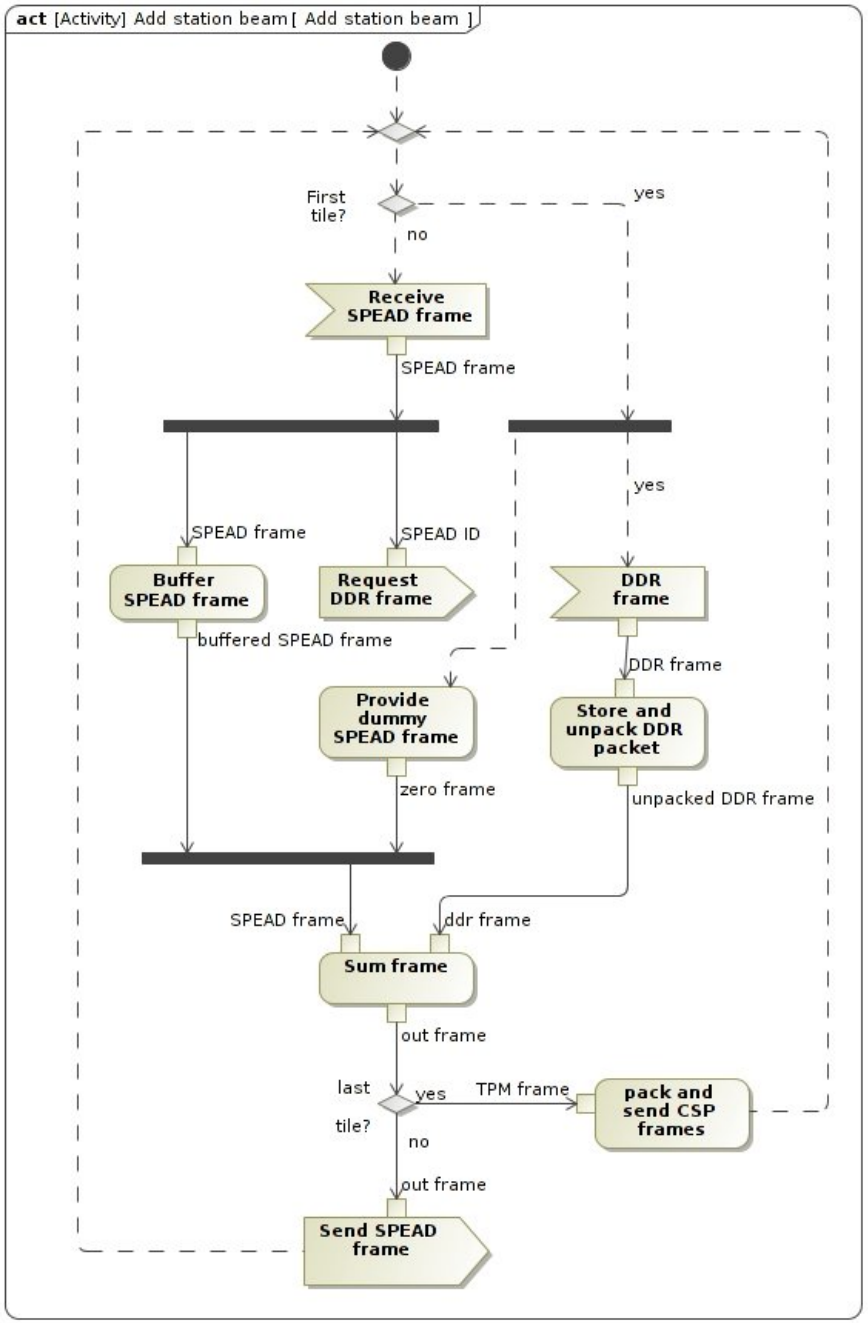
Tile Beamformer – Block Definition Diagram



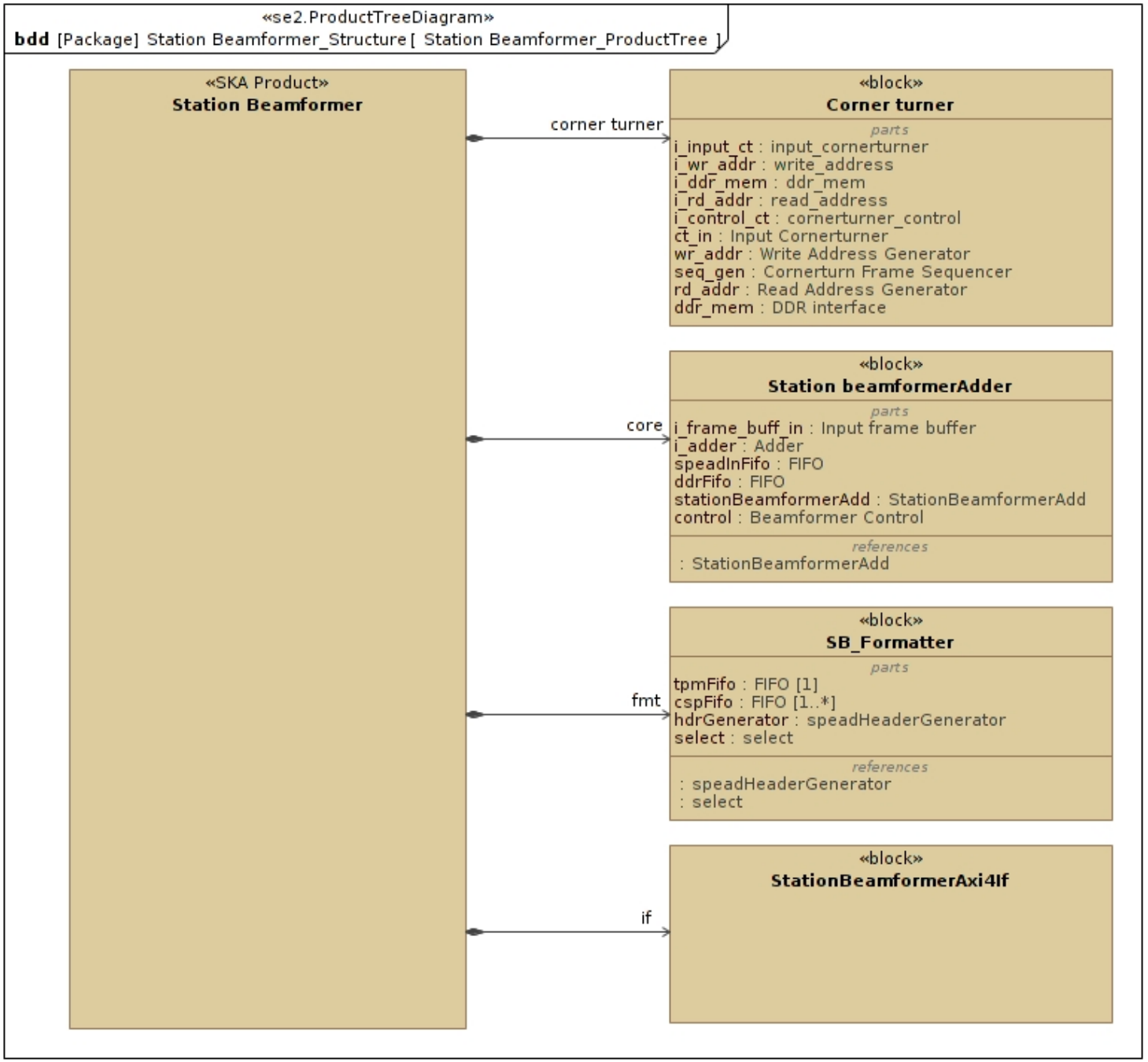
Tile Beamformer – Internal Block Diagram



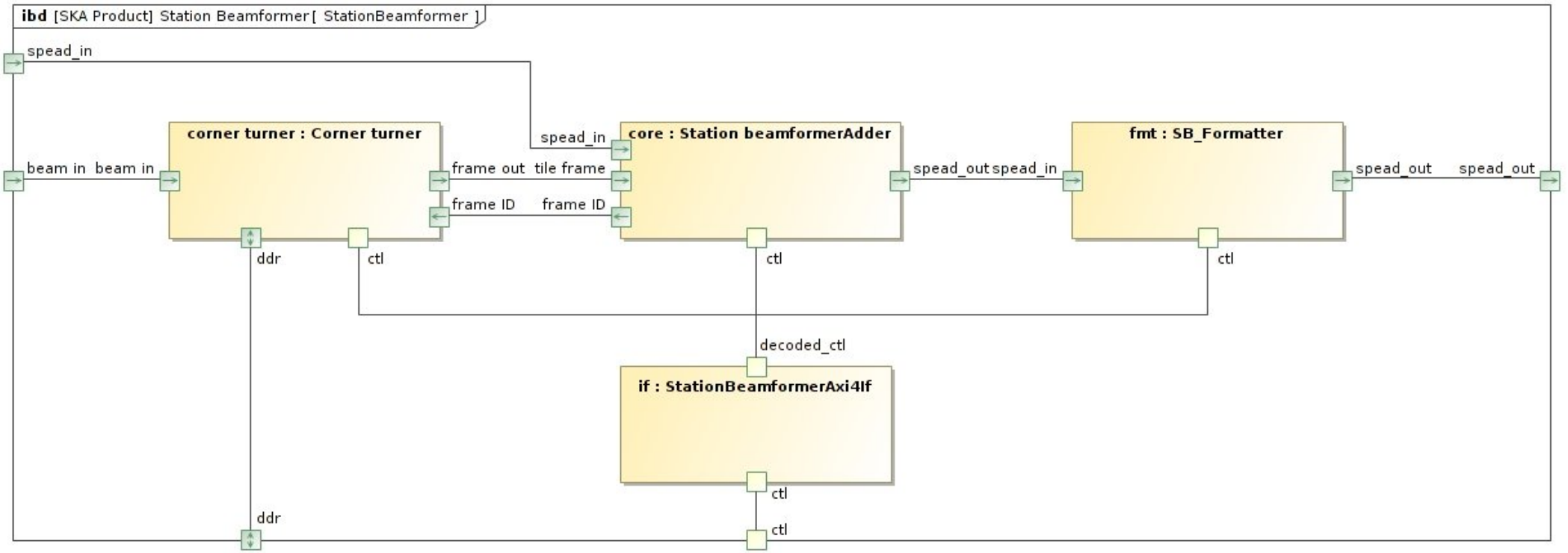
Station Beamformer – Activity Diagram



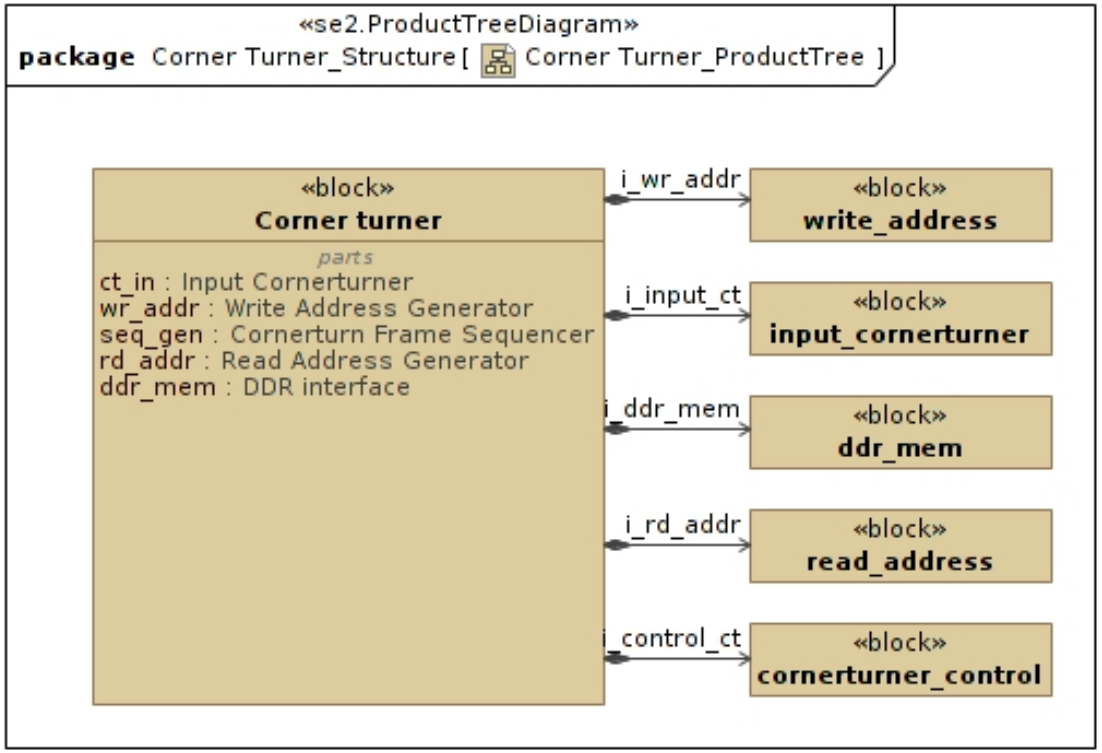
Station Beamformer – Block Definition Diagram

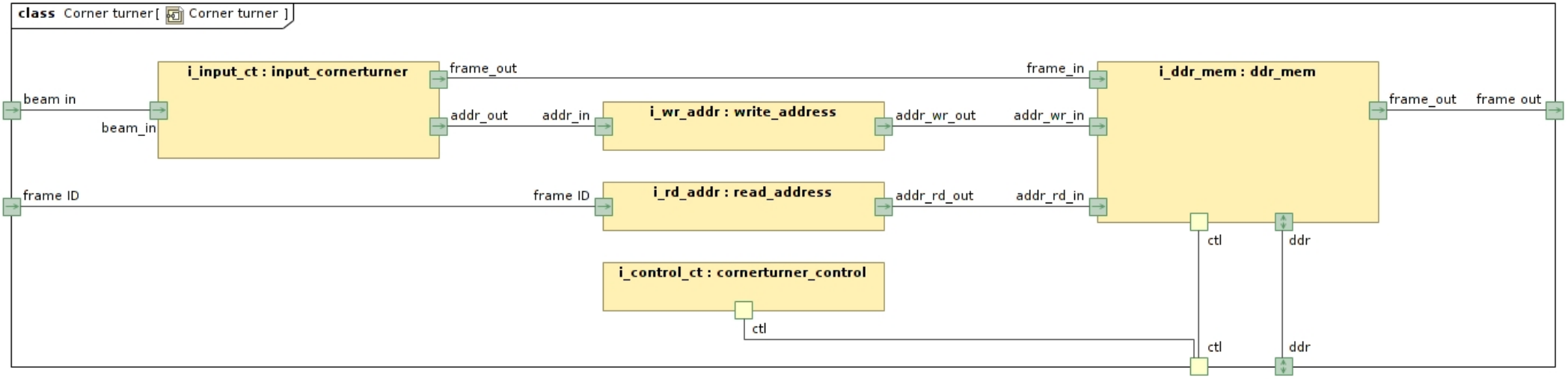


Station Beamformer – Internal Block Diagram



Cornerturner – Block Definition Diagram and Internal Block Diagram





Cornerturner – VHDL code

```
-------------------------------------------------------------------------------------------------------
-- Title      : cornerturner
-- library    : cornerturner
-- Project    : SKA
-------------------------------------------------------------------------------------------------------
-- File       : cornerturner.vhd
-- Author     :   <comore@arcetri.astro.it>
-- Last modified by : $Author: comore $
-- Company    : INAF - OAA
-- Created    : 2015-12-02
-- Platform   :
-- Standard   : VHDL'93
-------------------------------------------------------------------------------------------------------
-- Description:
--! DDR based corner turner for the TPM beamformer. Stores a whole elementary integration time
--! in DDR, and returns it as a set of contiguous time samples for a few channels at a time
--!
--! Input frames contain a single time sample for all frequency channels. Each sample is composed in
--! total by g_data_w bits. In TPM standard case there are 4 values per sample, representing real
--! and imaginary parts per two polarizations. Each value will be separately summed in the
--! beamformer with the corresponding value from the other tiles, but the cornerturner considers
--! each sampel as an atomic entity.
--! Frame length is g_in_frame_length samples, received in the same number of clock cycles.
--!
--! Frames stored in memory contain g_in_nof_frames time samples, for g_tmp_nof_out_channels
--! channels. Each time sample is composed of g_values_per_sample signed values with g_in_data_w
--! bits each. Each frame is (g_in_nof_frames x g_tpm_nof_out_channels) sample long.
--! Consecutive groups of g_tpm_nof_out_channels are stored in the 2**g_bank_w memory banks.
--!
--! Memory word size is g_ddr_word_size. It must accomodate one time sample, with the unused bits
--! set to 0. Memory address is expressed as a vector of size (g_addr_w), with bits in order (msb
--! to lsb) row-column-bank. The actual number of bits used may be less, and is controlled by the
--! generics g_addr_w, g_row_w; g_col_w, g_bank_w. These must match the actual memory chips used.
--! Memory uses a time multiplexing factor g_ddr_tmf, i.e. this number of memory words are
--! transferred to the memory controller at each (internal) clock cycle.
--!
--! A write cycle contains A total of g_burst_len memory words are accessed at each read cycle.
--! The necessary number of consecutive burst read, possibly intermixed with write accesses,
--! are used to fill a whole tpm frame.
--!
--! Output frames contain g_tpm_frame_length time samples, for g_nof_out_channels channels.
--!
--! TPM frames are sequenced in the hyerarchic order:
--!
--! 2**(inner_chan_loop) frames with the same time interval, referring to the same number of
--! consecutive groups of (g_tpm_nof_out_channels) frequency channels
--!
--! The sequence is repeated for enough consecutive time intervals in order to fill the integration
--! period
--!
--! Sequence is repeated in order to send all frequency channels
--!
--! Integration time is measured in CSP frames, given in input signal inner_chan_loop.
--!
--! Each CSP frame contains  g_csp_nof_out_chans frequency channels, and g_csp_frame_length time
--! samples.
--!
-------------------------------------------------------------------------------------------------------
-- Libraries:

library ieee;
use ieee.std_logic_1164.all;
use IEEE.NUMERIC_STD.all;
```

```
library common_lib, ddr_lib, cornerturner_lib;
use common_lib.all;
use ddr_lib.ddr_pkg.ALL;
--
-- entity declaration
--
ENTITY cornerturner IS
  GENERIC (
    -- DDR
    g_ddr_word_size : INTEGER := 64;  --! DDR word size
    g_ddr_tmf       : INTEGER := 8;    --! int.clk 200 MHz, DDR clk 800 MT/s );
    g_addr_w        : INTEGER := 29;   --! Total DDR address space
    g_row_w         : INTEGER := 14;   --! Row address space -- 128 M word DDR memory
    g_col_w         : INTEGER := 10;   --! Column address space -- 1k-word row size
    g_bank_w        : INTEGER := 3;    --! 8 DDR banks
    g_burst_len     : INTEGER := 64;   --! Number of memory words per read/write burst
    -- Input frame
    g_data_w        : INTEGER := 48;   --! bits per sample (real) in DDR words
    g_in_frame_length: INTEGER := 192;--! input frame length (number of channels)
    g_nof_frames    : INTEGER := 8;    --! Frames in input cornerturner
    -- Output frame
    g_tpm_nof_chans : INTEGER := 4;   --! Frequency channels in an output frame
    g_tpm_frame_len : INTEGER := 256  --! Number of read memory bursts in an output frame
    );
  PORT (
    -- Clock and control
    --
    dsp_clk        : IN  STD_LOGIC; --! Signal processign clock
    dsp_rst        : IN  STD_LOGIC; --! Signal processing reset
  -- control parameters
    int_block_len  : IN  UNSIGNED(11 DOWNTO 0) := TO_UNSIGNED(102*8-1,12); -- Integration block length (in TPM frames)
    first_tile     : IN  STD_LOGIC;  --! = true for first tile in chain
    csp_frame_size : IN  STD_LOGIC_VECTOR(3 DOWNTO 0) := "0111"; --! Number (minus 1) of TPM frames in a CSP frame
    inner_chan_loop: IN  STD_LOGIC_VECTOR(1 DOWNTO 0) := "00"; --! Log2 of number of channels in inner loop for CSP frames
    max_out_chan   : IN  STD_LOGIC_VECTOR(7 DOWNTO 0) := x"2F";--! Maximum transmitted channel
    --
    -- Cascaded TPM frame - used if first_tile=0
    casc_frame_stb : IN  STD_LOGIC := '0'; --! Frame strobe from previous tile in chain
    casc_frame_id  : IN  STD_LOGIC_VECTOR(47 DOWNTO 0) := (OTHERS => '0'); --! Frame ID from previous tile in chain
    casc_frame_out_id : OUT STD_LOGIC_VECTOR(47 DOWNTO 0); --! Frame ID of the generated frame
    casc_frame_rdy : OUT STD_LOGIC; --! Ready for frame strobe from previous tile in chain
    --
    -- Input frames, in dsp_clk domain
    data_in      : IN  STD_LOGIC_VECTOR(g_data_w-1 DOWNTO 0); --! Input data
    sop_in       : IN  STD_LOGIC; --! Input data start of packet
    eop_in       : IN  STD_LOGIC; --! Input data end of packet
    dav_in       : IN  STD_LOGIC; --! Input data valid
    rdy_in       : OUT STD_LOGIC; --! Input data ready
    --
    -- Out data, in ddr_clk domain
    data_out     : OUT STD_LOGIC_VECTOR(g_ddr_word_size*g_ddr_tmf-1 DOWNTO 0); --! Output data
    sop_out      : OUT STD_LOGIC;  --! Output data start of packet
    eop_out      : OUT STD_LOGIC;  --! Output data end of packet
    dav_out      : OUT STD_LOGIC;  --! Output data valid
    rdy_out      : IN  STD_LOGIC;  --! Output data ready
    --
    -- DDR PHY, in ddr_clk domain
    ddr_clk      : IN  STD_LOGIC;  --! DDR clock
    ddr_phy_out  : OUT t_ddr_phy_out; --! DDR address and control signals
    ddr_phy_io   : INOUT t_ddr_phy_io); --! DDR input/output signals
END ENTITY cornerturner;
```

Thank you for your attention