Formal Foundations for MARTE-SystemC Interoperability

Pablo Peñil
Fernando Herrera
Eugenio Villar

Outline

- Motivation
  - Related Work
- Focus of this work
  - Previous Work
- Interoperability UML/MARTE - SystemC
- Conclusions
- Future Work

Motivation

MDA

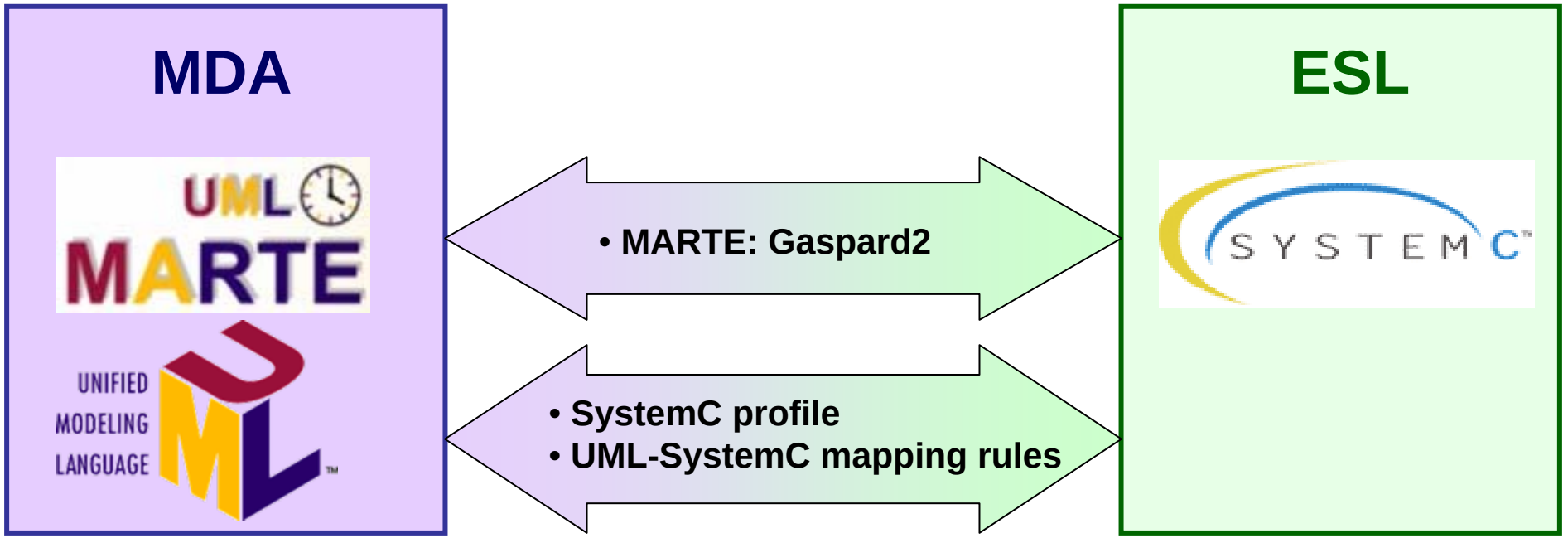
UML MARTE

UNIFIED MODELING LANGUAGE UML

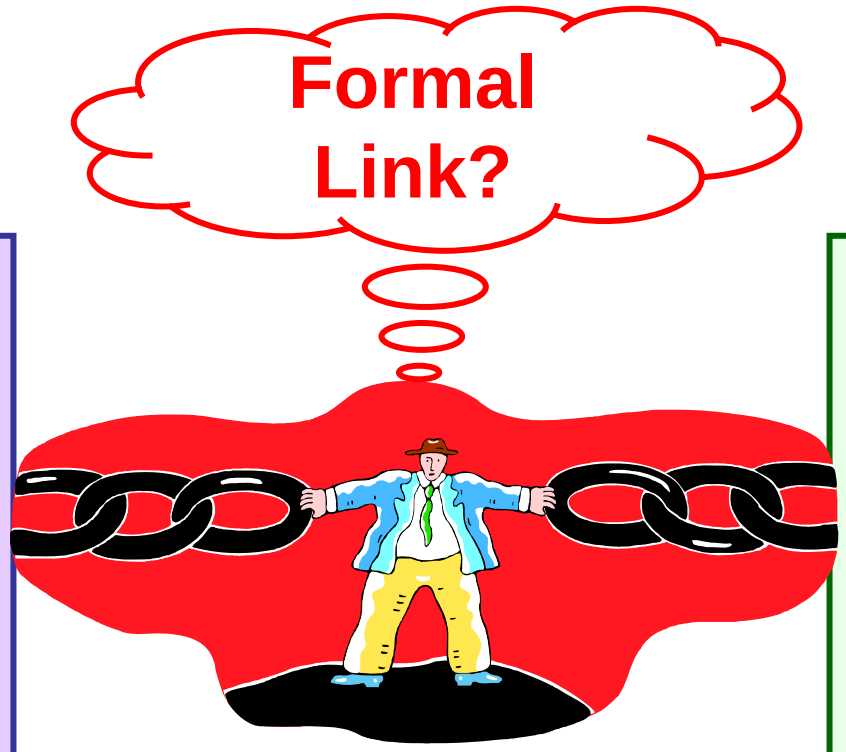
ESL

SYSTEMC™

Motivation


MDA
UML
MARTE
UNIFIED
MODELING
LANGUAGE
• MARTE: Gaspard2
• SystemC profile
• UML-SystemC mapping rules
ESL
SYSTEMC

Motivation



MDA

UML MARTE

ESL

SYSTEMC™

Motivation

- UML (Activity) diagrams
- MARTE
  - CCL
  - Communication Media

- RTL
- TLM (synchronous & asynchronous)

Formal Link?

MDA

UML MARTE

ESL

SYSTEMC™

Motivation

- UML (Activity) diagrams
- MARTE
  - CCL
  - Communication Media

- RTL
- TLM (synchronous & asynchronous)

No formal link!

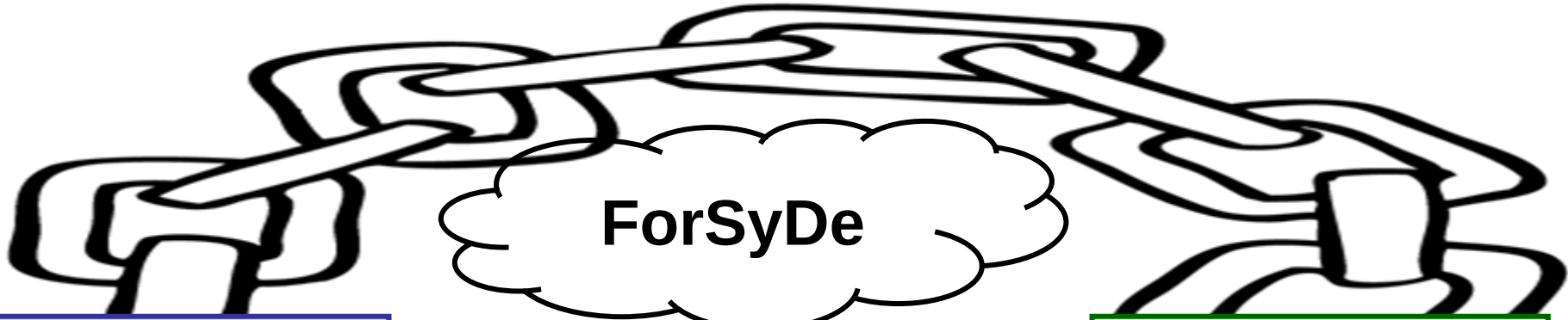
MDA

UML MARTE

ESL

SYSTEMC™

Contribution



ForSyDe

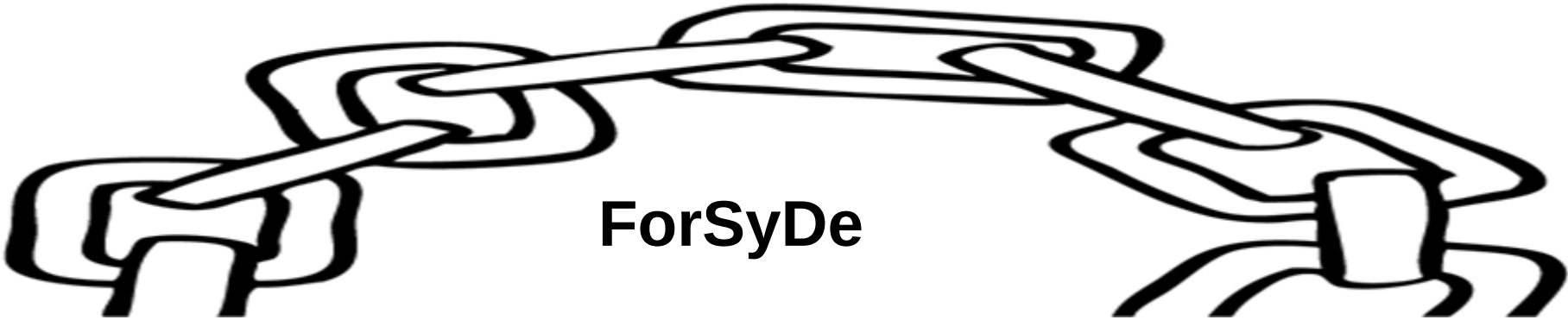
MDA

UML MARTE

ESL

SYSTEMC™

Benefits of ForSyDe as Formalism

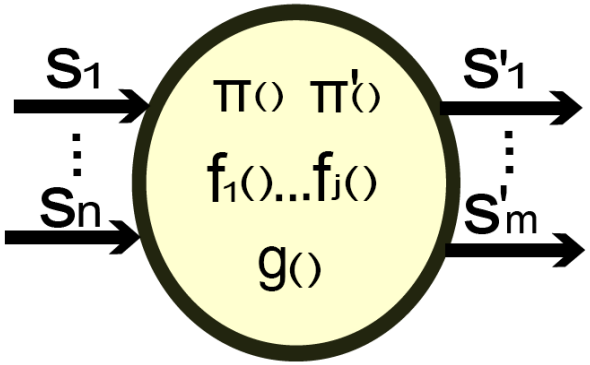


- Formal Denotation
  - Common, Unambiguous and Synthetic Representation
- Covers all different aspects of ESL design
  - Specification
    - Concurrency
    - Separation of Computation and Communication
    - Different Time abstractions (Untimed, Synchronous, …)
    - Connection of different MoCs (Heterogeneity)
  - Verification
  - Transformational Design

ForSyDe

- ForSyDe formal metamodel:
  - *Events*
  - *Signals*
  - *Process*
    - *Process Constructors (& Operators), partition functions, etc*
- Precise definition, e.g.
  - *(Untimed) ForSyDe Event = Value*
  - *(Untimed) ForSyDe Signal = Ordered Collection of ForSyDe events*
  - *ForSyDe signal ≠ SystemC signal*

ForSyDe Process



- External View:
  - Relation on signals
- Internal View:
  - *Process constructors*

$$p(s_1 \ldots s_n) = s'_1 \ldots s'_m$$

π partition function S_n

π' partition function S'_m

$g()$ next-state function

$f_1() \ldots f_j()$ output functions

Formal Notation

The partitions functions:

$$\pi(\nu_n, s_n) = \langle a_n(z) \rangle \qquad \pi'(\nu'_m, s'_m) = \langle a'_m(z) \rangle$$

where $a_n(z)$ is a subsignal of S_n

The function ν_n gives the size of $a_n(z)$:

$$\nu_n(z) = \gamma(\omega_q) \qquad \nu'_m(z) = length(a'_m(z))$$

$$\nu_n(0) = length(a_n(0)); \quad \nu_n(1) = length(a_n(1))...$$

The outputs are calculated:

$$f_\alpha((a_1 ... a_n), \omega_q) = (a'_1 ... a'_m)$$

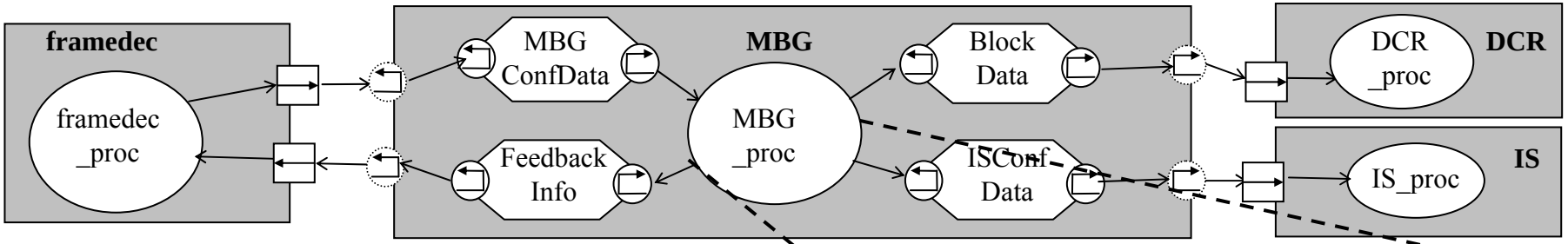
And the next internal state:

$$g((a_1 ... a_n), \omega_q) = \omega_{q+1}$$

Focus of the Formal Link

- **Specification Methodology**
  - Component-based approach
  - Concurrency and Communication Structure
  - Focus on Functionality
    - Predefined Communication Mechanisms
  - Untimed Modelling (Suitable for PIM)
    - Synthetic
    - Fast Simulation
    - Input for ESL design (implementation=refinement)

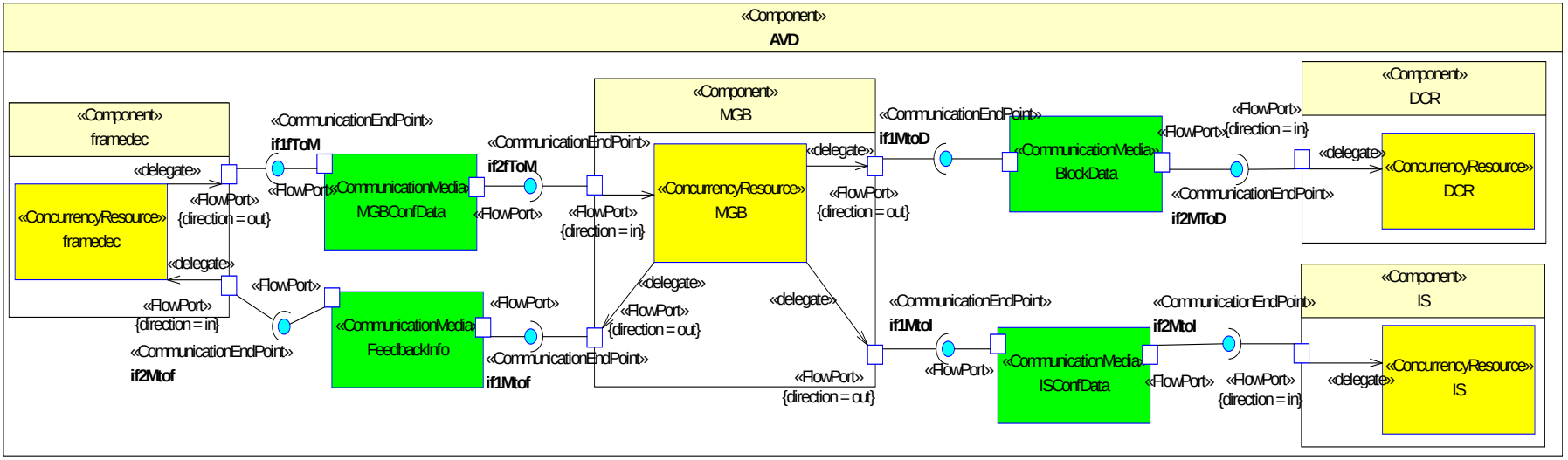
Focus of the Formal Link



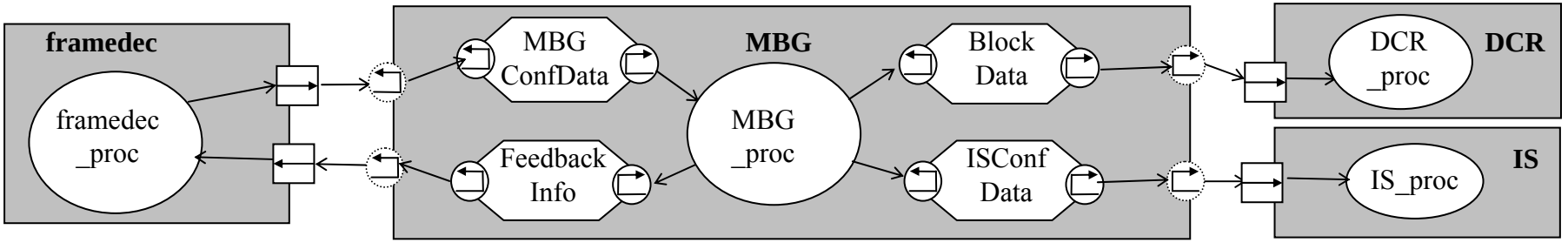

```
class MBG: public sc_module:
    uc_fifo<int>    MBG_ConfData(100),
    uc_fifo<int>    Feedback_Info(2),…
{
…SC_THREAD(MBG_proc) ;…
};

SC_MODULE(framedec)   { …SC_THREAD(framedec_proc) ;… };
SC_MODULE(DCR)        { …, SC_THREAD(DCR_proc) ;… };
SC_MODULE(IS)         { …, SC_THREAD(IS_proc) ;…  };

sc_main(int argc, char * argv ) {
  framedec    framedec_m("framedec_m");
  DCR         DCR_m("DCR_m");
  IS          IS_m("IS_m");
  MBG         mbg_m("mbg_m");

  frame_dec_m.out(MBG.in1);
  frame_dec_m.in(MBG.out1);
  …
}
```

```
(1)void MBG::MGB_proc(){
(2)  T1  invar[ ];
(3)  T2  outvar2[ ]; T3 outvar3[ ]; T4 outvar4[ ];
(4)  while (true) {
(5)    for(int i=0;i<6;i++) invar1[i]= MBGConfData.read();
(6)    Init_QFS(invar1, outvar4);
(7)    for(int i=0;i<2;i++) BlockData.write(outvar4);
(8)   do {
(9)      if (Lumablock()) {
(10)       outvar4[0] = Decode_Luma_block ();
(11)     BlockData.write (outvar4[0]);}
(12)     else {
(13)       outvar4[0] = Decode_Chroma_block();
(14)   BlockData.write (outvar4[0]) } // end if
(15)  …

(28)  } while( intra_mb_and_block_in_mb( ) );
(29) }  // end MGB process loop code
(30)} // end MGB process code
```

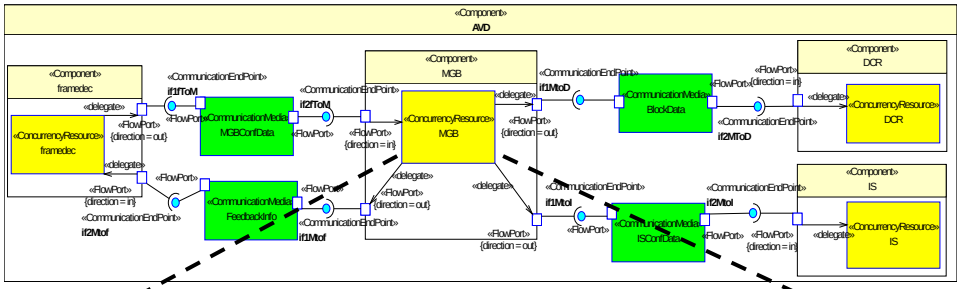
Focus of the Formal Link

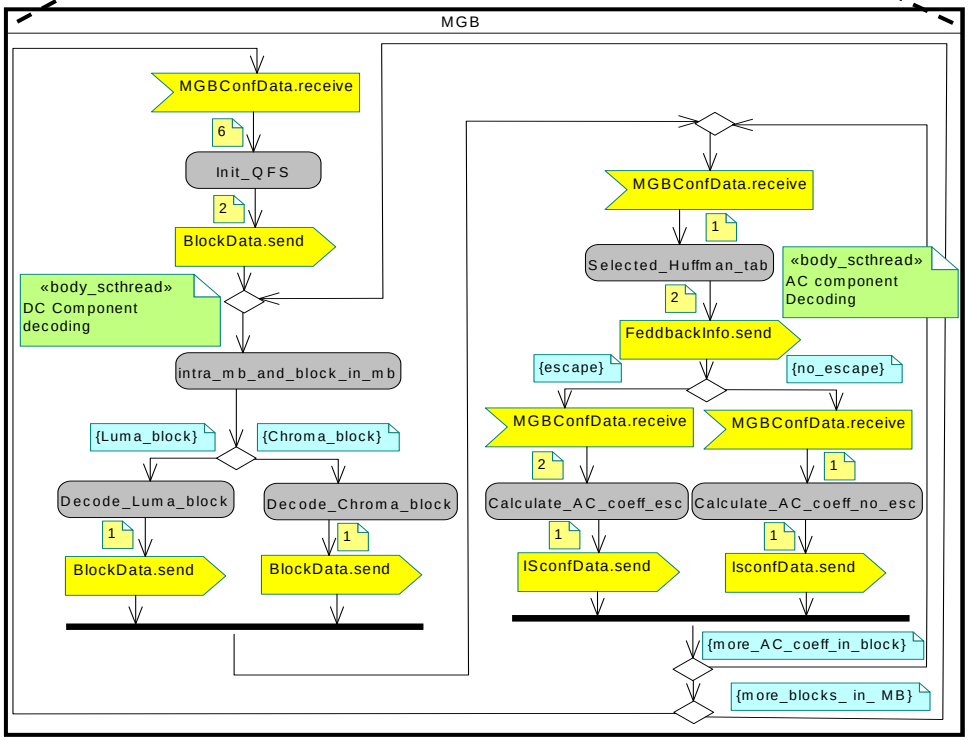


- **UML/MARTE**
  - Structured Diagram (Composite Diagram)
  - GRM profile (ConcurrencyResource, CommunicationMedia)



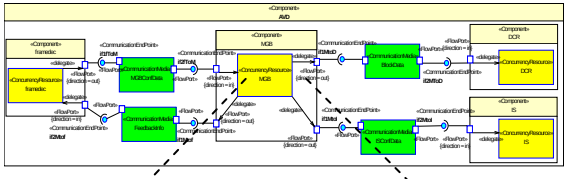
Focus of the Formal Link

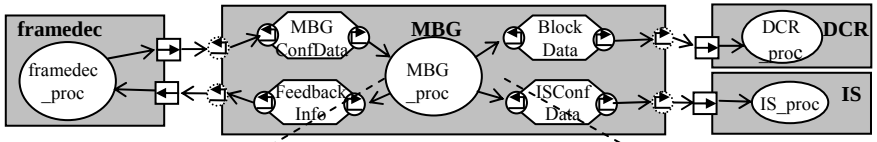




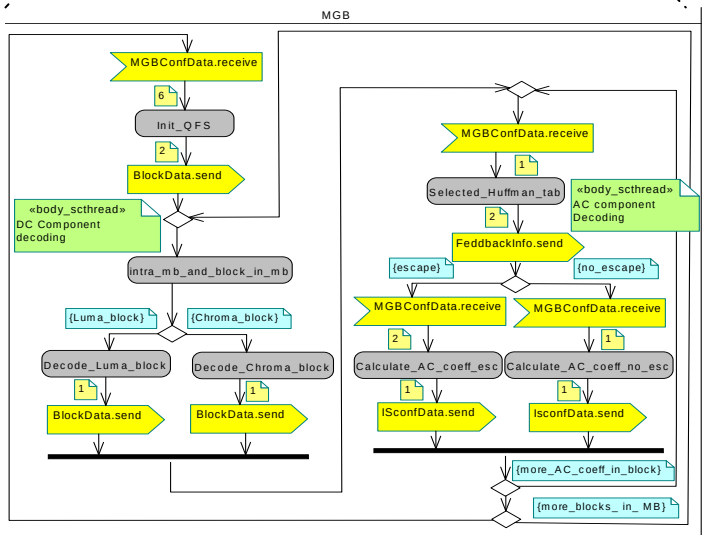
- Activity Diagram for a ConcurrencyResource
  - AcceptEventActions
  - SendObjectAction
  - AccessMultiplicity
  - CallOperationAction
  - ControlNode
  - …

Link as a mapping





Mapping?

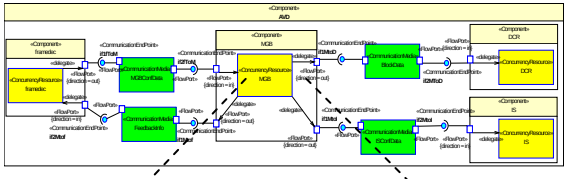


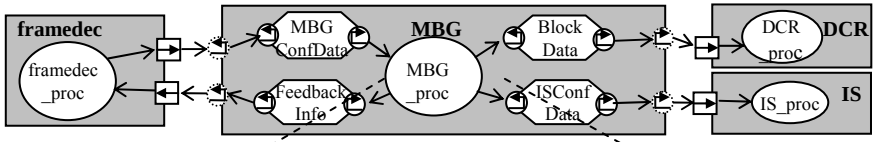

```
(1)void MBG::MGB_proc(){
(2)  T1  invar[ ];
(3)  T2  outvar2[ ]; T3 outvar3[ ]; T4 outvar4[ ];
(4)  while (true) {
(5)    for(int i=0;i<6;i++) invar1[i]= MBGConfData.read();
(6)    Init_QFS(invar1, outvar4);
(7)    for(int i=0;i<2;i++) BlockData.write(outvar4);
(8)   do {
(9)     if (Lumablock()) {
(10)      outvar4[0] = Decode_Luma_block ();
(11)      BlockData.write (outvar4[0]); }
(12)    else {
(13)      outvar4[0] = Decode_Chroma_block();
(14)      BlockData.write (outvar4[0]); } // end if
(15)    do {
(16)      invar[0]= MBGConfData.read()
(17)      Selected_Huffman_tab (invar, outvar2);
(18)      for(int i=0;i<2;i++) FeedbackInfo.write(outvar2[i]);
(19)      if ( escape() ) {
(20)        for(int i=0;i<2;i++) invar[i] = MBGConfData.read();
(21)        Calculate_AC_coeff_esc (invar, outvar4[0]);
(22)        ISConfData.write(outvar4); }
(23)      else {
(24)        invar[0] = MBGConfData.read();
(25)        Calculate_AC_coeff_no_esc(invar, outvar4[0]);
(26)        ISConfData.write(outvar4); } // end if
(27)    } while( more_AC_coeff_in_block () );
(28)  } while( intra_mb_and_block_in_mb( ) );
(29) }  // end MGB process loop code
(30)} // end MGB process code
```

Link…just a mapping?

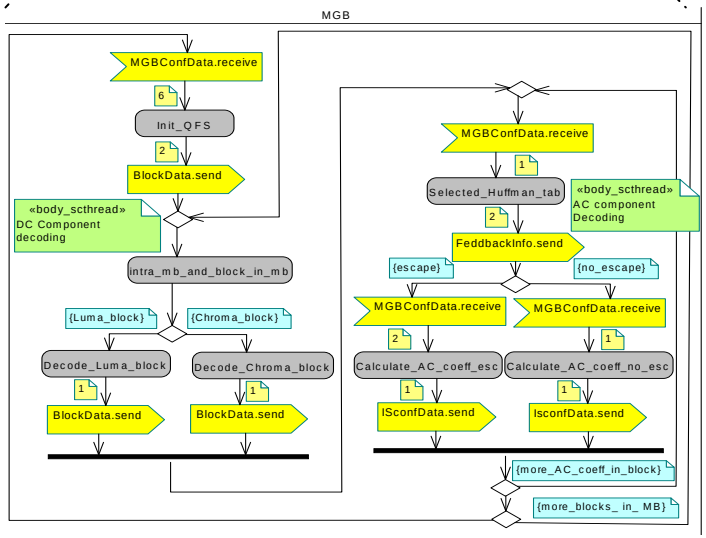
- **No perfect match between SystemC and UML/MARTE elements**
- **Interoperability can and should be more flexible:**
  - A mismatch between the UML Component and Module hierarchical structure means non equivalence?
  - What type of SystemC processes relate to Concurrency Resources?
- **How SystemC DE semantics relates to the UML/MARTE semantics?**

Link by means of a Formal Metamodel





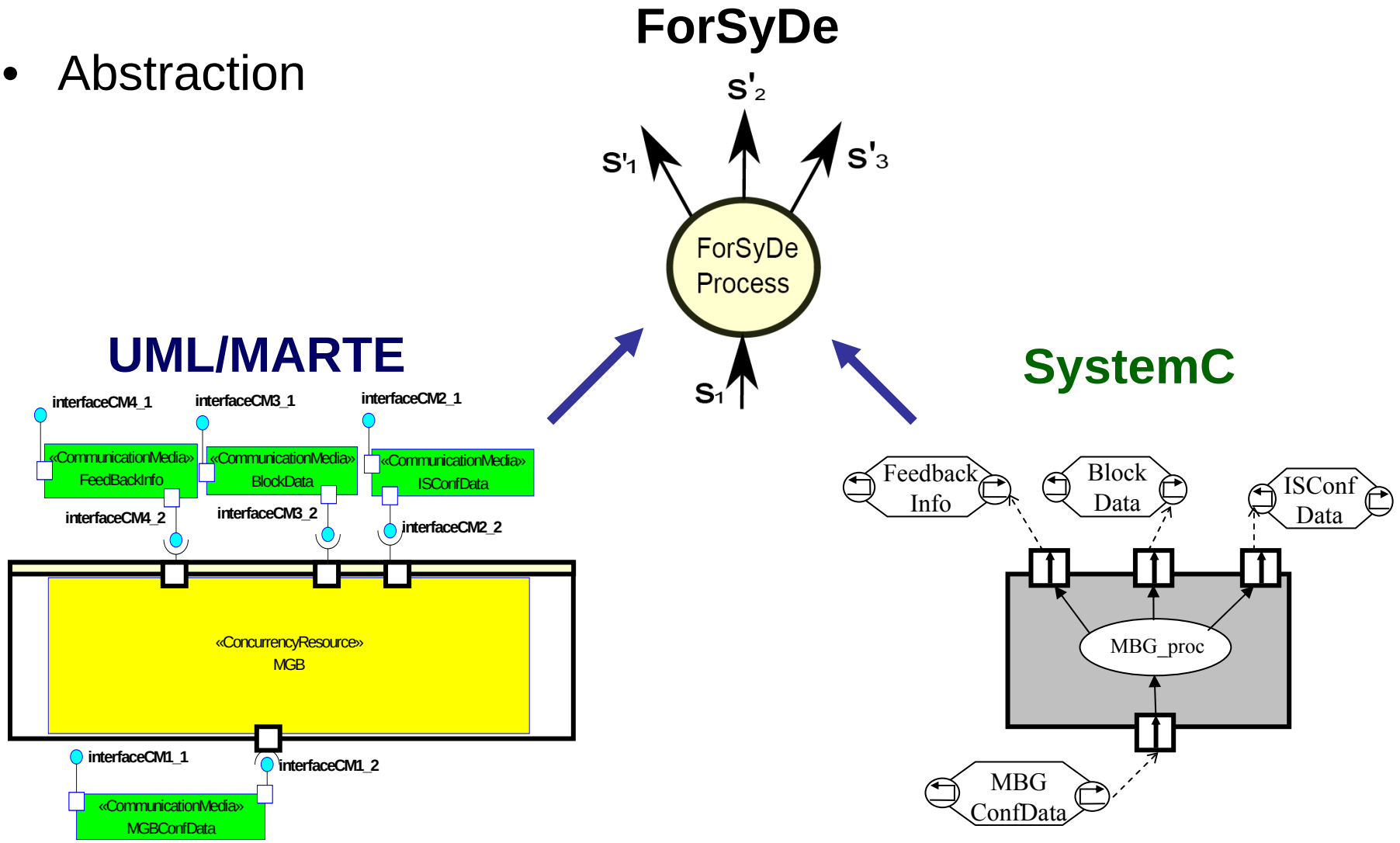
ForSyDe




```
(1)void MBG::MGB_proc(){
(2)  T1  invar[ ];
(3)  T2  outvar2[ ]; T3 outvar3[ ]; T4 outvar4[ ];
(4)  while (true) {
(5)    for(int i=0;i<6;i++) invar1[i]= MBGConfData.read();
(6)    Init_QFS(invar1, outvar4);
(7)    for(int i=0;i<2;i++) BlockData.write(outvar4);
(8)   do {
(9)     if (Lumablock()) {
(10)      outvar4[0] = Decode_Luma_block ();
(11)      BlockData.write (outvar4[0]); }
(12)    else {
(13)      outvar4[0] = Decode_Chroma_block();
(14)      BlockData.write (outvar4[0]); } // end if
(15)    do {
(16)      invar[0]= MBGConfData.read()
(17)      Selected_Huffman_tab (invar, outvar2);
(18)      for(int i=0;i<2;i++) FeedbackInfo.write(outvar2[i]);
(19)      if ( escape() ) {
(20)        for(int i=0;i<2;i++) invar[i] = MBGConfData.read();
(21)        Calculate_AC_coeff_esc (invar, outvar4[0]);
(22)        ISConfData.write(outvar4); }
(23)      else {
(24)        invar[0] = MBGConfData.read();
(25)        Calculate_AC_coeff_no_esc(invar, outvar4[0]);
(26)        ISConfData.write(outvar4); } // end if
(27)    } while( more_AC_coeff_in_block () );
(28)  } while( intra_mb_and_block_in_mb( ) );
(29) }  // end MGB process loop code
(30)} // end MGB process code
```

ForSyDe Link

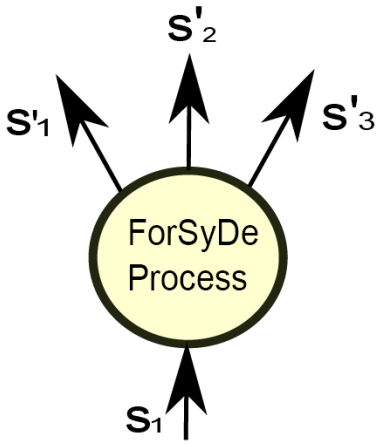
- Abstraction



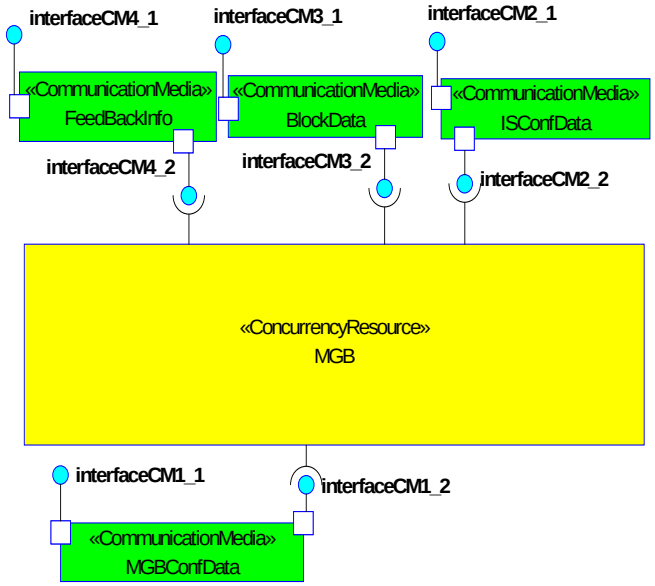
ForSyDe Link

- Abstraction
  - No need to consider Component wrapping

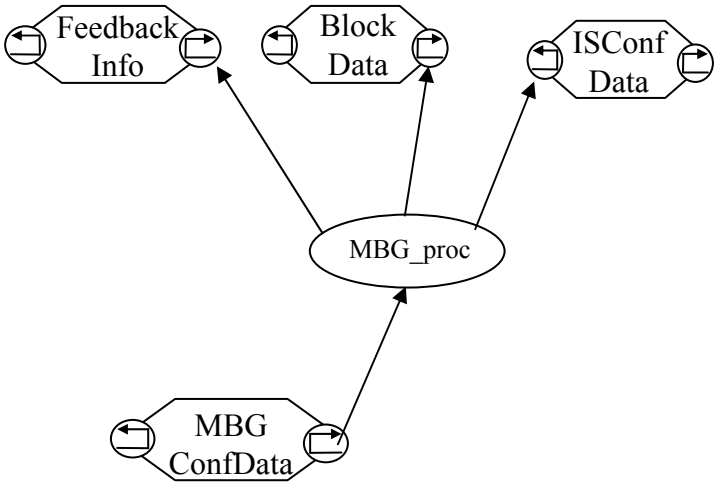
ForSyDe



UML/MARTE



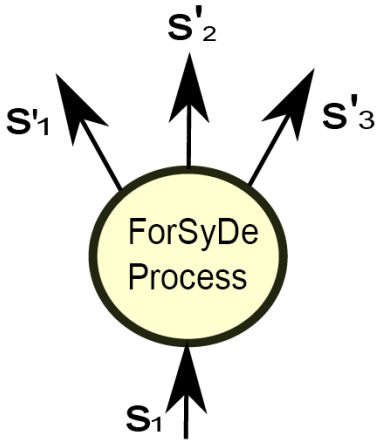
SystemC



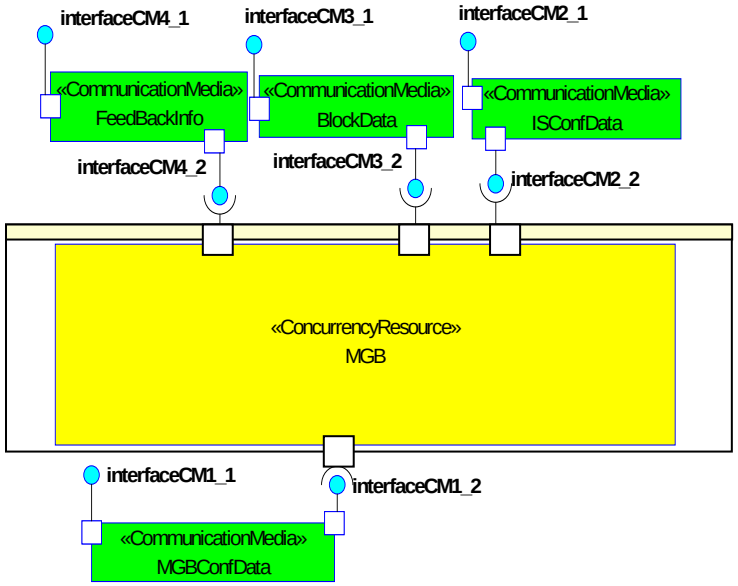
ForSyDe Link

- Abstraction
  - No need to consider Component wrapping
  - Flexibility in the interoperability

ForSyDe

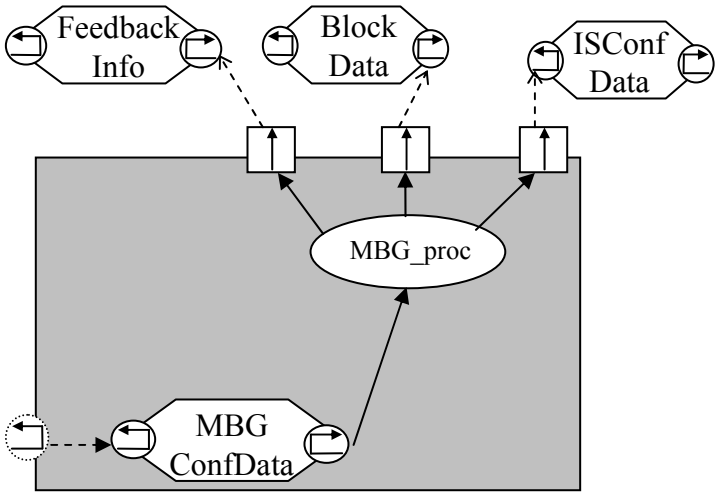


UML/MARTE

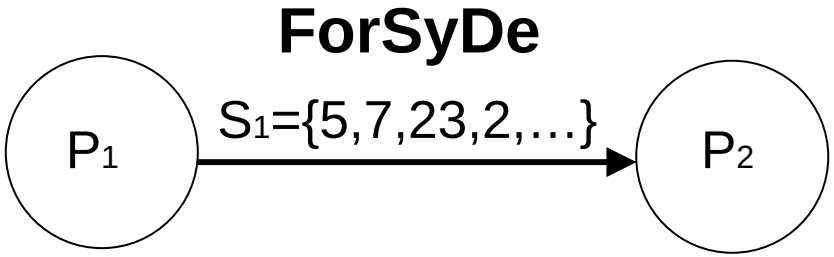


=

SystemC

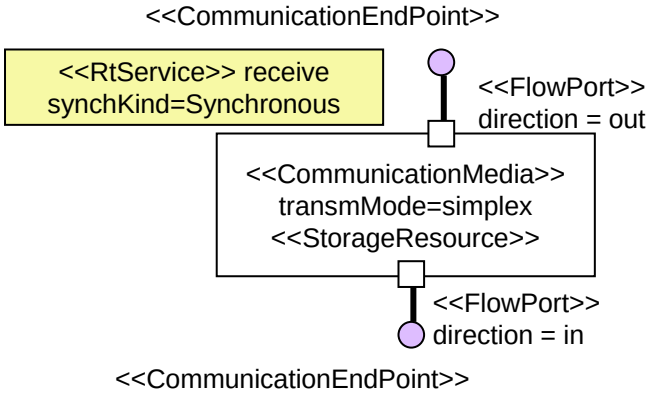


ForSyDe Link: Communication Abstraction

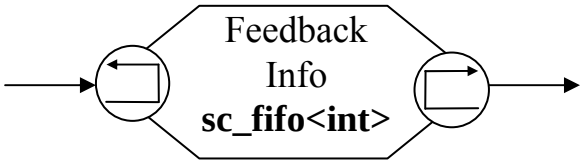


- Ordered data flow, independently on:
  - The semantic description based on UML/MARTE stereotypes
  - The executive semantics associated to the SystemC channel

UML/MARTE



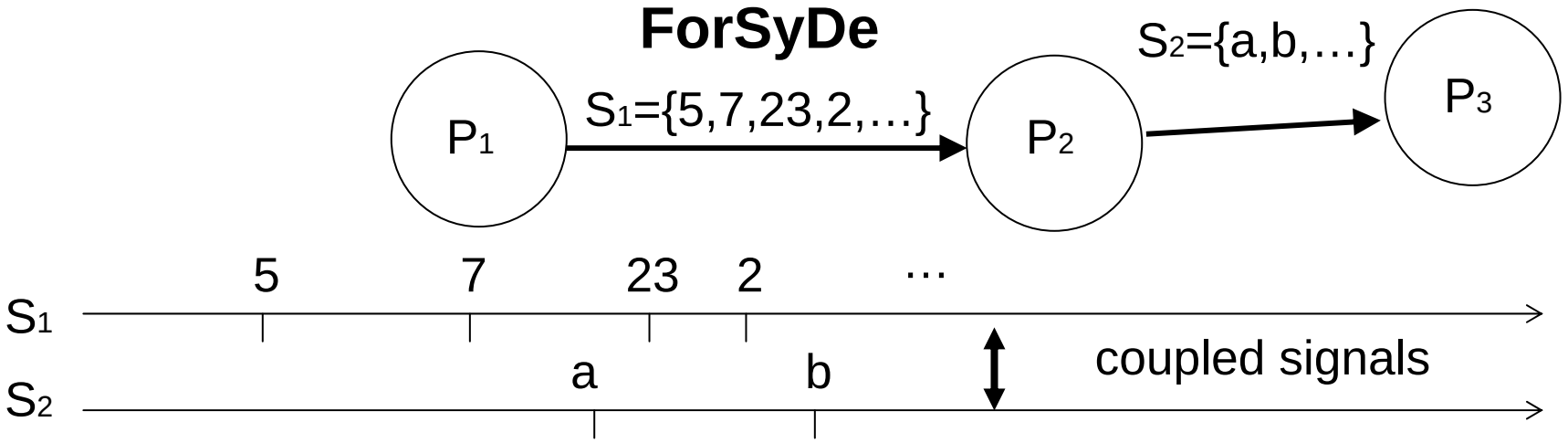
SystemC



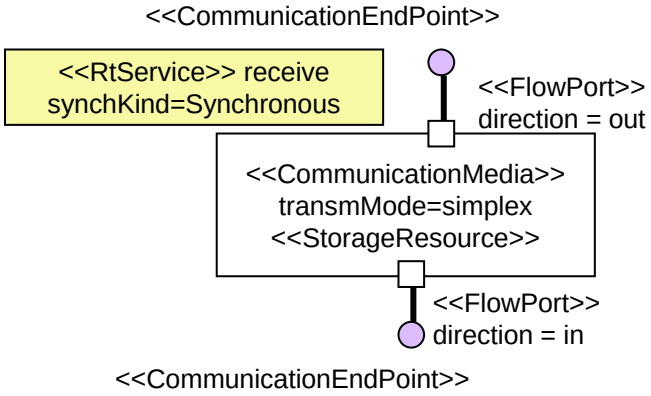
- Block if full
- write data, enqueuing it in the internal buffer
- …

- Block if empty
- Read data from inner buffer
- …

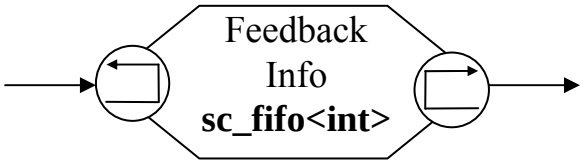
ForSyDe Link: Communication Abstraction



UML/MARTE



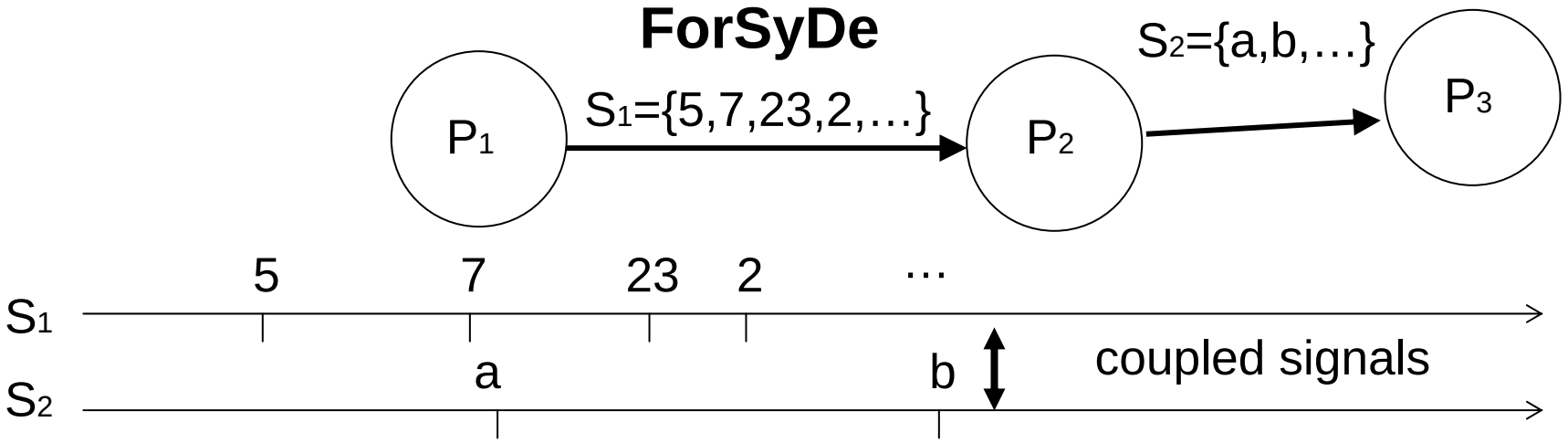
SystemC



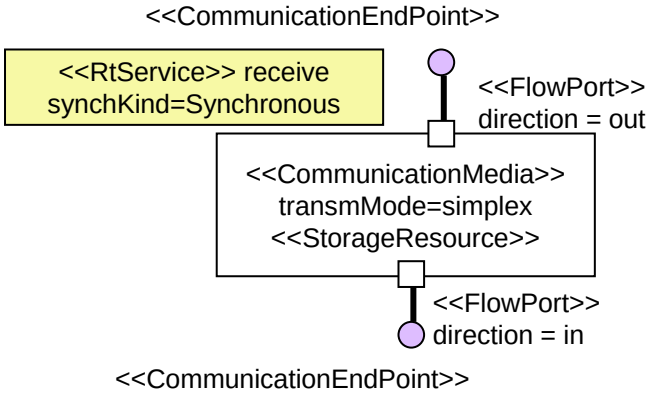
- Block if full
- write 5 enqueuing it in the internal buffer
- …

- Block if empty
- Read 5 in buffer
- …

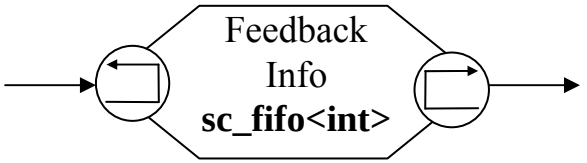
ForSyDe Link: Communication Abstraction



UML/MARTE



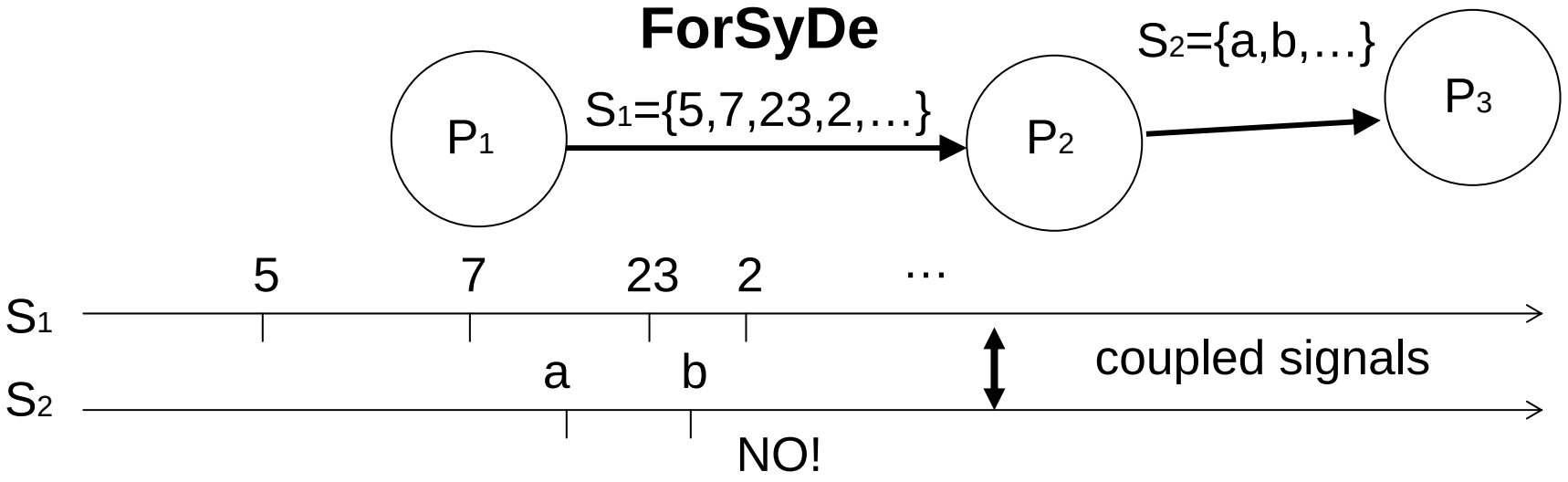
SystemC



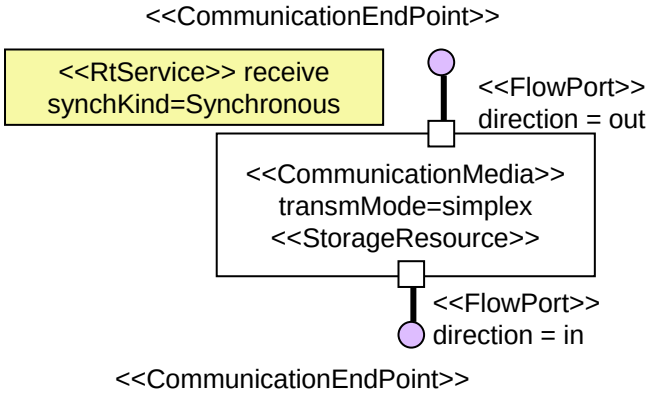
- Block if full
- write 5 enqueuing it in the internal buffer
- …

- Block if empty
- Read 5 in buffer
- …

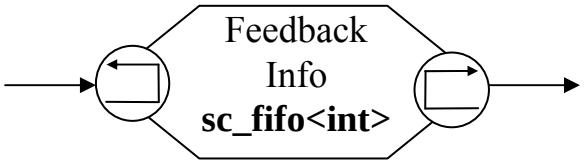
ForSyDe Link: Communication Abstraction



UML/MARTE



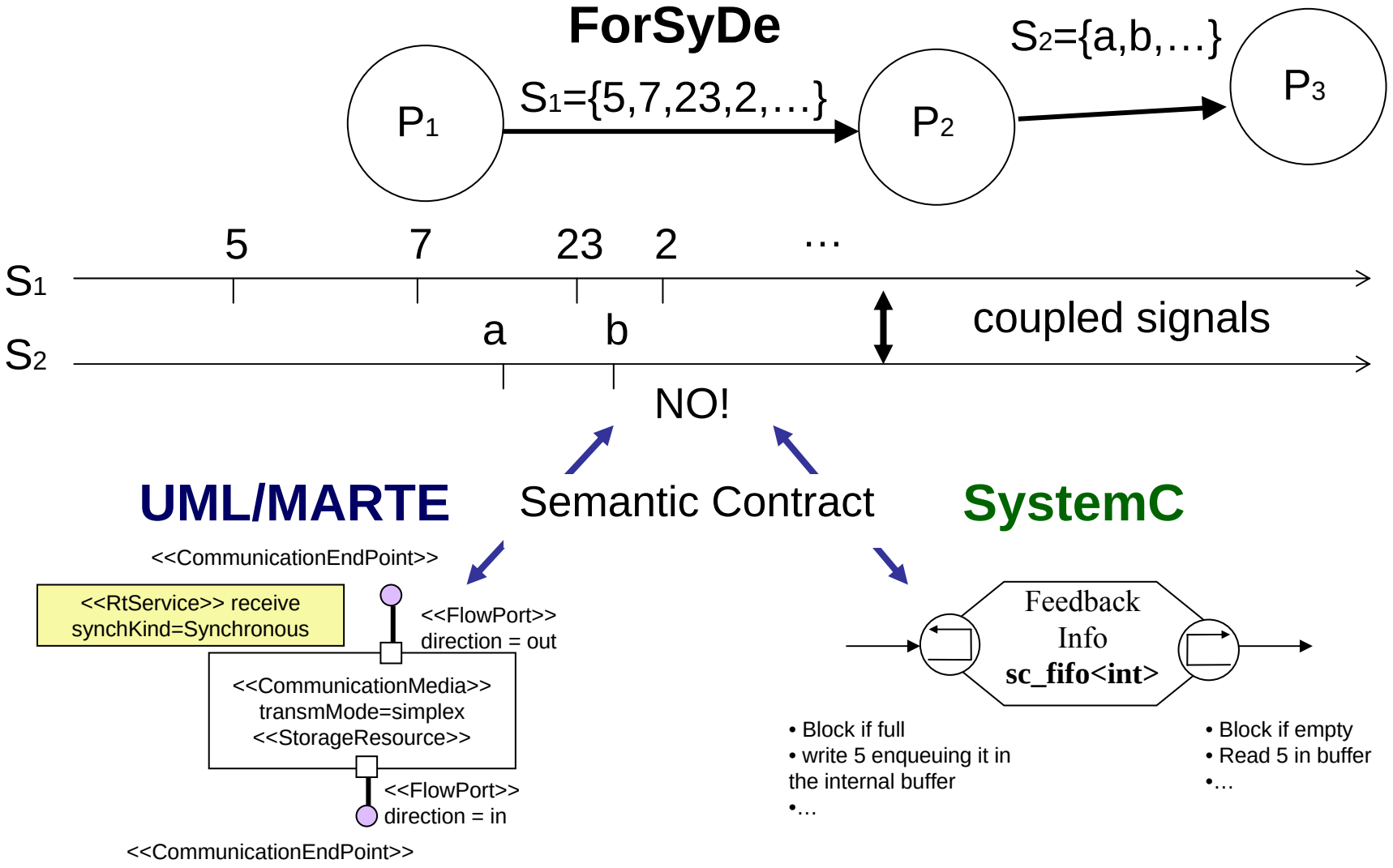
SystemC



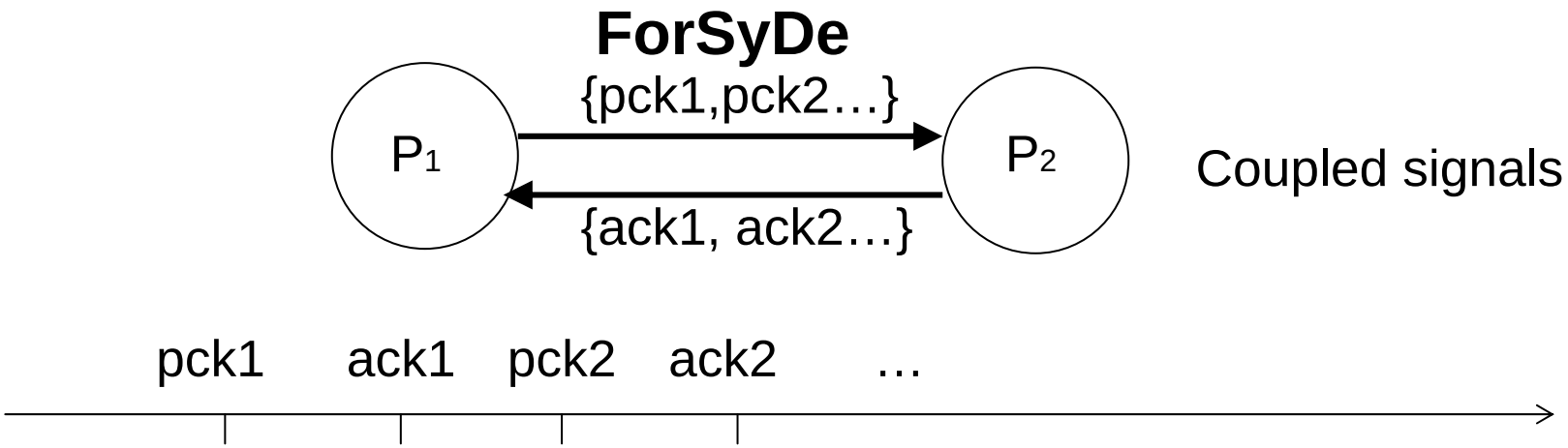
- Block if full
- write 5 enqueuing it in the internal buffer
- …

- Block if empty
- Read 5 in buffer
- …

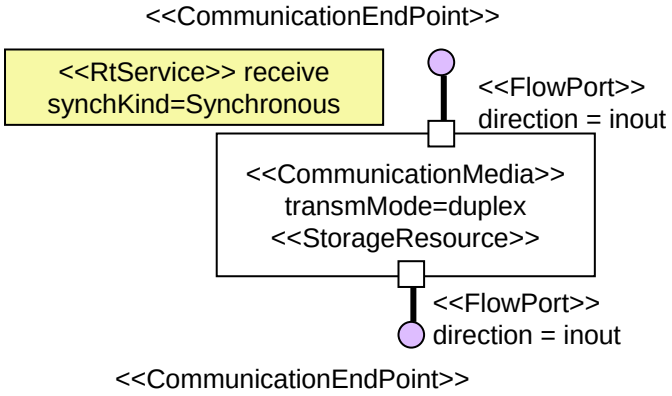
ForSyDe Link: Communication Abstraction



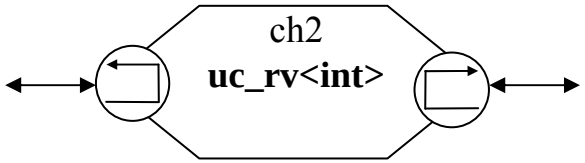
ForSyDe Link: Communication Abstraction



UML/MARTE

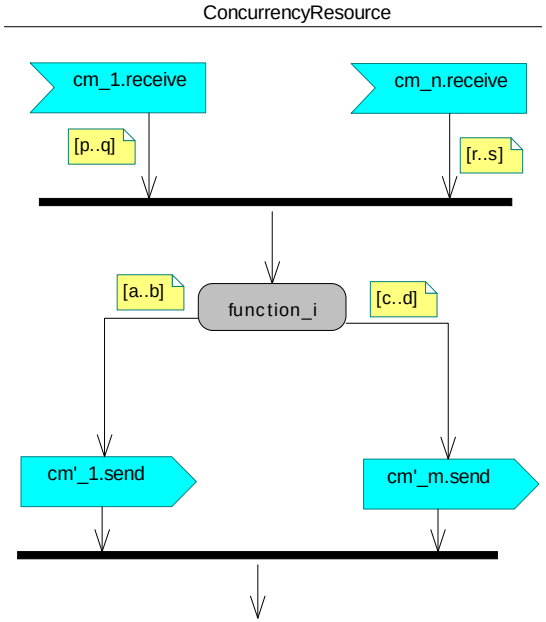


SystemC



Functionality

UML/MARTE



ForSyDe

$$mealyU(\gamma, g, f, w_0) = p$$

where

$$p(\dot{s}) = \dot{s}'$$

$$f(w_i, \dot{a}_i) = \dot{a}'_i$$

$$g(w_i, \dot{a}_i) = w_{i+1}$$

$$\pi(\nu, \dot{s}) = \langle \dot{a}_i \rangle,$$

$$\nu(i) = \gamma(w_i)$$

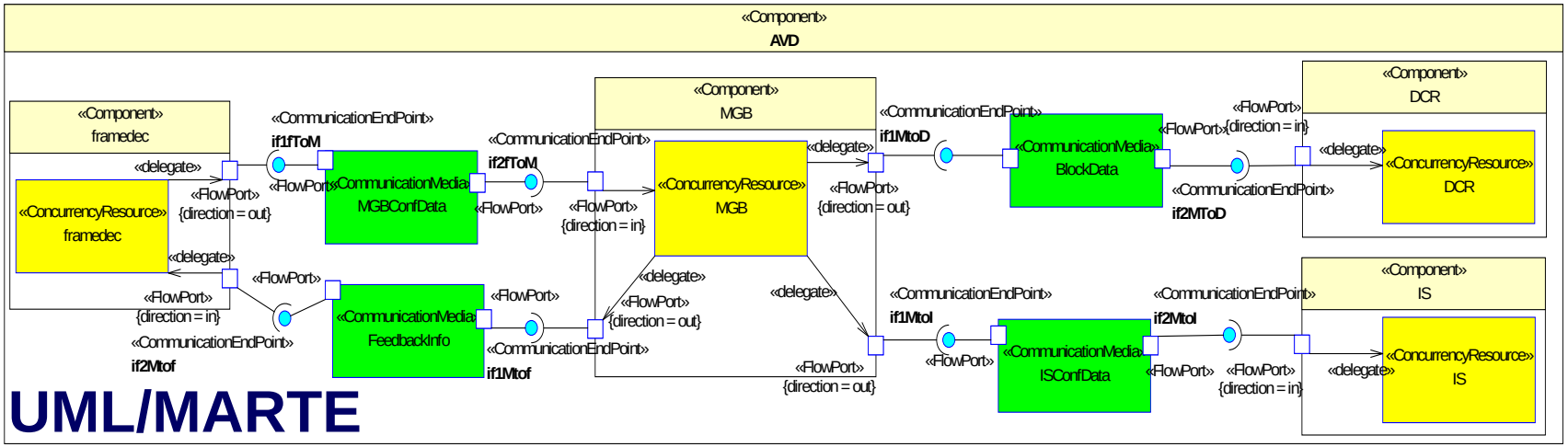
$$\pi(\nu', \dot{s}') = \langle \dot{a}'_i \rangle,$$

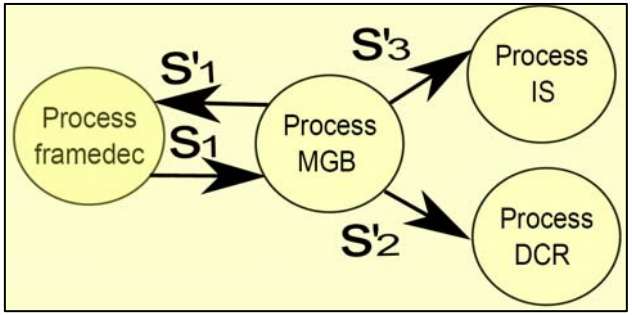
$$\nu'(i) = \#f(w_i, \dot{a}_i)$$

SystemC

```
vp=ch_1.read();...;vq=ch_1.read();
...
vr=ch_N.read();...;vs=ch_N.read();


function_i(vp,...,vq,...,vr,...vs,
                va,...,vb,...,vc,...vd);


chp_1.write(va);...;chp_1.write(vb);
...
chp_N.write(vc);...;chp_N.write(vd);
```

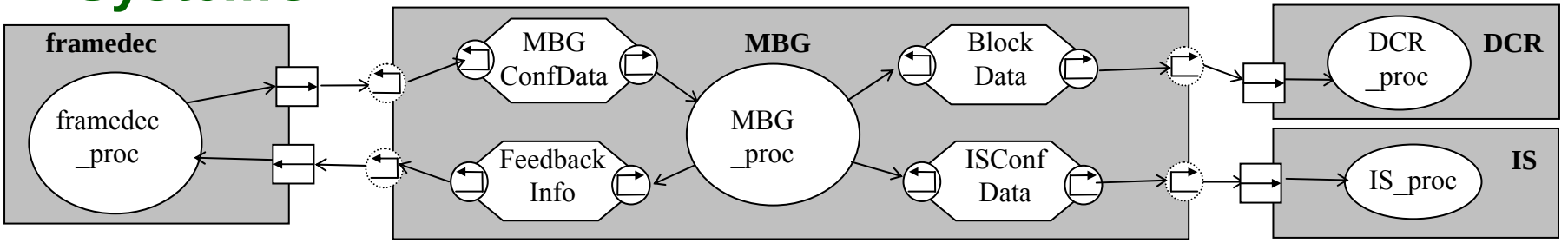
Example (AVD)





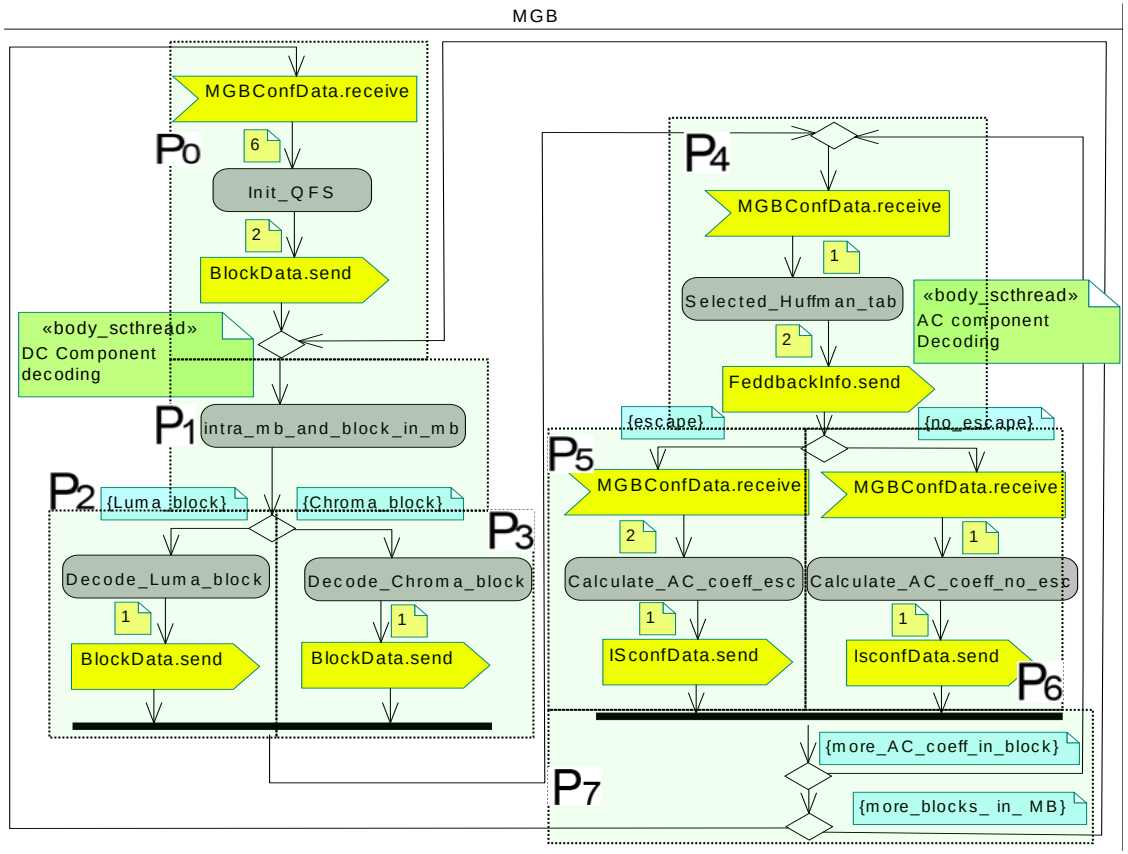
ForSyDe

SystemC



Example (MBG process)

UML/MARTE



SystemC

```
(1)void MBG::MGB_proc(){
(2)  T1  invar[ ];
(3)  T2  outvar2[ ]; T3 outvar3[ ]; T4 outvar4[ ];
(4)  while (true) {
(5)    for(int i=0;i<6;i++) invar1[i]= MBGConfData.read();
(6)    Init_QFS(invar1, outvar4);
(7)    for(int i=0;i<2;i++) BlockData.write(outvar4);
(8)   do {
(9)     if (Lumablock()) {
(10)      outvar4[0] = Decode_Luma_block ();
(11)      BlockData.write (outvar4[0]); }
(12)    else {
(13)      outvar4[0] = Decode_Chroma_block();
(14)      BlockData.write (outvar4[0]); } // end if
(15)    do {
(16)      invar[0]= MBGConfData.read()
(17)      Selected_Huffman_tab (invar, outvar2);
(18)      for(int i=0;i<2;i++) FeedbackInfo.write(outvar2[i]);
(19)      if ( escape() ) {
(20)        for(int i=0;i<2;i++) invar[i] = MBGConfData.read();
(21)        Calculate_AC_coeff_esc (invar, outvar4[0]);
(22)        ISConfData.write(outvar4); }
(23)      else {
(24)        invar[0] = MBGConfData.read();
(25)        Calculate_AC_coeff_no_esc(invar, outvar4[0]);
(26)        ISConfData.write(outvar4); } // end if
(27)    } while( more_AC_coeff_in_block () );
(28)  } while( intra_mb_and_block_in_mb( ) );
(29) }  // end MGB process loop code
(30)} // end MGB process code
```

Example (MBG process)

ForSyDe

MGB = **mealyU**(γ, g, f, ω_0)
MGB(s_1) = <s'_1, s'_2, s'_3>

If ($state_i = \omega_0$) then
$\nu_{s1}(i) = 6$, $\pi(\nu_{s1}, s_1) = <a1_i>$
$a3'_i = f_0(a1_i)$ = Init_QFS($a1_i$)
$\nu_{s'3}(i) = 2$. $\pi(\nu_{s'3}, s'_3) = < a3'_i>$
$state_{i+1} = g(\omega_0) = \omega_1$

elseif ($state_i = \omega_1$) then
$g(w_i)$ = Decode_Type_of_Block(w_i)

elseif ($state_i = \omega_2$) then
$a3'_i = f_2()$ = Decode_Luma_block()
$\nu_{s'3}(i) = 1$, $\pi(\nu_{s'3}, s'_3) = <a3'_i>$
$state_{i+1} = g(\omega_2) = \omega_4$

elseif ($state_i = \omega_3$) then
$a3'_i = f_3()$ = Decode_Chroma_block()
$\nu_{S'3}(i) = 1$, $\pi(\nu_{s'3}, s'_3) = < a3'_i >$
$state_{i+1} = g(\omega_2) = \omega_4$

elseif ($state_i = \omega_4$) then
$\nu_{s1}(i) = 1$, $\pi(\nu_{s1}, s_1) = <a1_i>$
$a1'_i = f_4(a1_i)$ = Selected_Huffman_tab(a1)
$\nu_{s'1}(i) = 2$, $\pi(\nu_{s'1}, s'_1) = <a1'_i>$

elseif ($state_i = \omega_5$) then
$\nu_{s1}(i) = 2$, $\pi(\nu_{s1}, s_1) = <a1_i>$
$a2'_i = f_5(a1_i)$=Calculate_AC_coeff_esc($a1_i$)
$\nu_{s'2}(i) = 1$, $\pi(\nu_{S'2}, s'_2) = <a2'_i>$
$state_{i+1} = g(\omega_5) = \omega_7$

elseif ($state_i = \omega_6$) then
$\nu_{s1}(i) = 1$, $\pi(\nu_{s1}, s_1) = <a1_i>$
$a2'_i = f_6(a1_i)$=Calculate_AC_coeff_no_esc($a1_i$)
$\nu_{s'2}(i) = 1$, $\pi(\nu_{S'2}, s'_2) = <a2'_i>$
$state_{i+1} = g(\omega_6) = \omega_7$

elseif (state = ω_7)
$g1(\omega'_t)$ = more_AC_coeff_in_block(ω_t)
$state_{i+1} = g2(\omega'_t)$ = more_blocks_in_MB(ω'_t)

Example (Zoom)

If ($state_i = \omega_0$) then

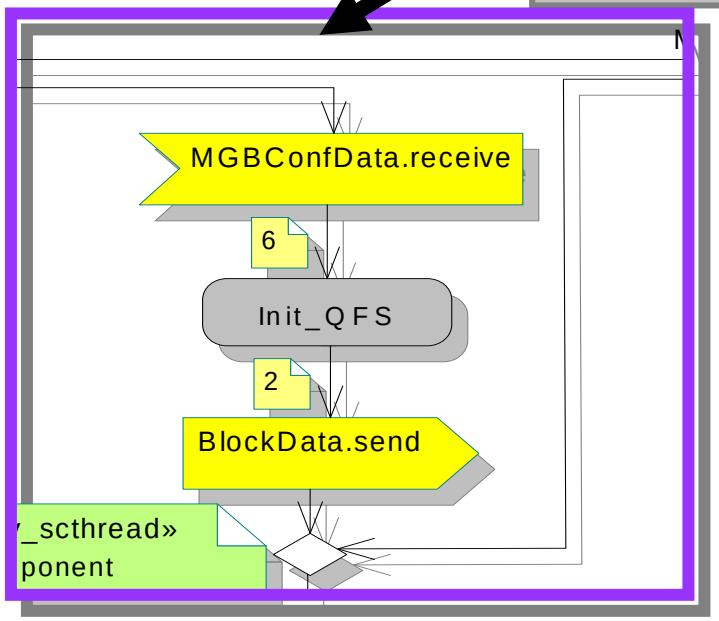
$\nu_{s1}(i) = 6$, $\pi(\nu_{s1}, s_1) = <a1_i>$

$a3'_i = f_0(a1_i) = Init\_QFS(a1_i)$

$\nu_{s'3}(i) = 2$. $\pi(\nu_{s'3}, s'_3) = < a3'_i>$

$state_{i+1} = g(\omega_0) = \omega_1$

UML/MARTE



SystemC

```
...
(5) for(int i=0;i<6;i++)
        invar1[i]= MBGConfData.read();
(6) Init_QFS(invar1, outvar4);
(7) for(int i=0;i<2;i++)
        BlockData.write(outvar4);
(8) …
```

Conclusions

- **ForSyDe link between MARTE and SystemC**
  - Solid Interoperability: Crucial for the adoption of MARTE/SystemC flows
  - Formal support for ESL related design activities
- **This work focusing on untimed models (crucial for PIMs as suitable input to ESL design)**
- **ForSyDe as semantic contract reflecting basic information:**
  - Concurrency and Communication Structure
  - Causal Order relationships (Untimed model)
- **… and enabling a flexible interoperability (beyond mapping)**

Future Work

- Automation of the generation of executable SystemC specifications from the UML/MARTE models

- Extension to Synchronous models

Thanks

- Professors A. Jatsch and I. Sander

- And funding

- Further Questions:
  - Of course, now
  - Poster (COMPLEX)
  - authors: {pablop, fherrera, evillar}@teisa.unican.es