Conference on Design and Architectures for Signal and Image Processing -2010
Electronic Chips & Systems design Initiative
October 26th-28th, 2010,
Edinburgh, Scotland

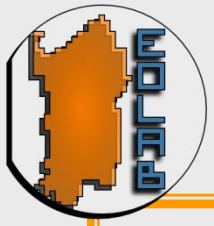
RVC: A MULTI-DECODER CAL COMPOSER TOOL

F. Palumbo, D. Pani, E. Manca and L. Raffo

EOLAB - Microelectronics Lab
DIEE , Cagliari – ITALY

M. Mattavelli and G. Roquier

EPFL, Lausanne
SWITZERLAND



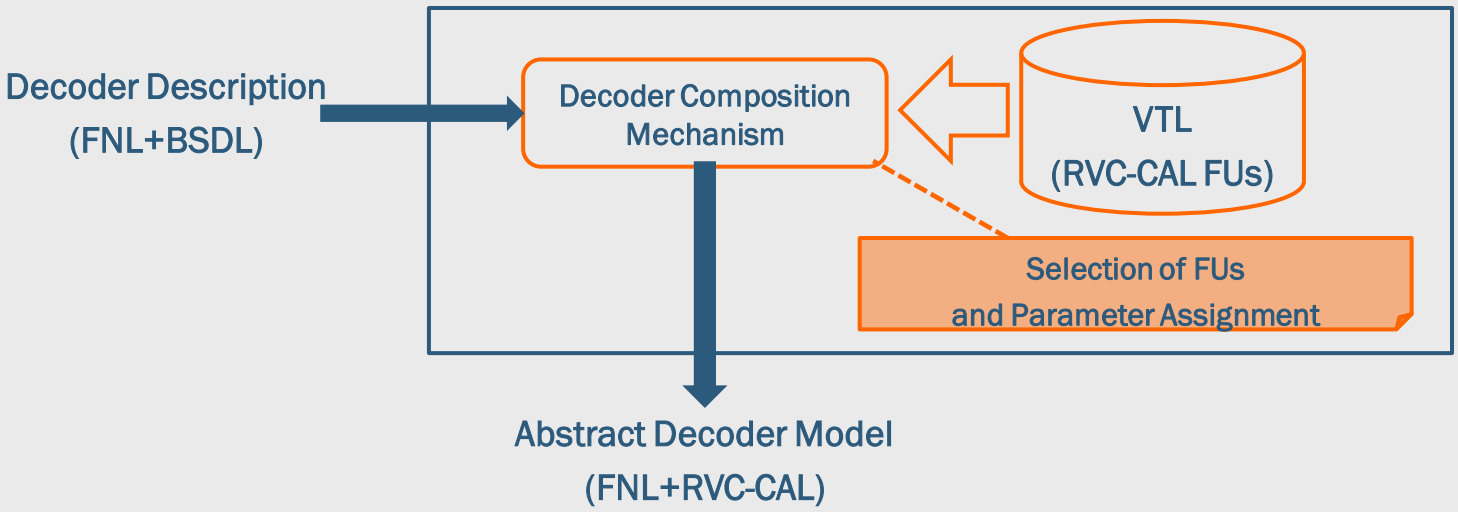
Outline

- Problem statement
- The Multi-Decoder CAL Composer tool
  - The CAL2GRAPH algorithm
  - The MERGER algorithm
- Use-Case: the parallel and the serial MPEG-4 SP
- Final remarks and exploitation

Problem Statement

- The problem of defining an efficient formalism for codecs specification has been recently addressed by the MPEG group through the development of the Reconfigurable Video Coding (RVC) framework.
- In this work we have moved one step further, conceiving the Multi-Decoder CAL Composer (MDCC) tool, based on the MPEG RVC formalism, able to characterize multi-decoder platforms.
- The MDCC is based on a platform template able to allow fast switching among codecs configurations without the need of any complete context switch.

The MDCC tool: Basis

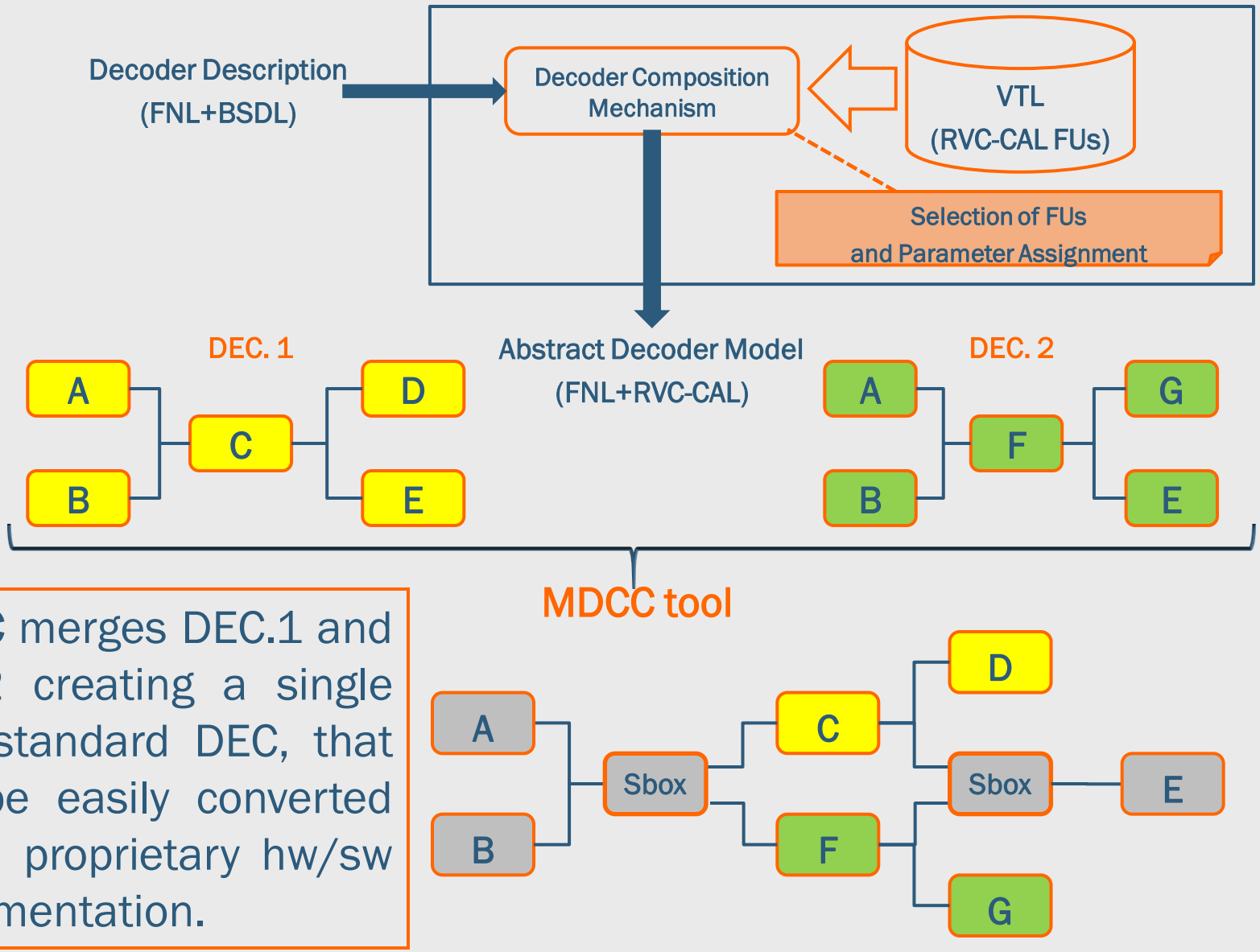


- Specification: dataflow programs composed by a network of Functional Units (FUs) belonging to a standard Video Tool Library (VTL).
- The modularity property allows to conceive implementation procedures based on static or dynamic reconfigurations.

The MDCC tool: Definition

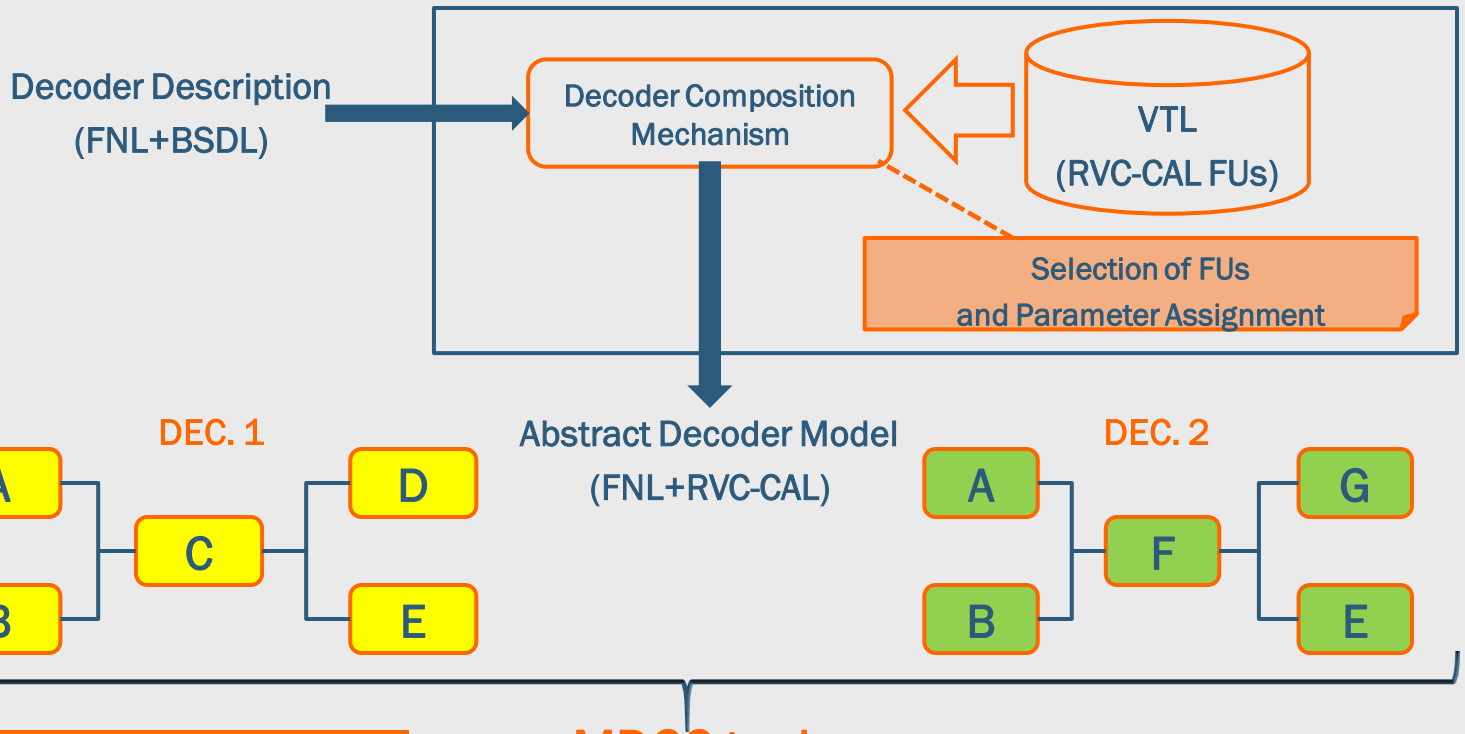
- Looking at the similarities between the decoding algorithms to be integrated in a single description, the MDCC automatically composes a single configurable multi-decoder which is formed by:
  - All the required modules necessary to accomplish the video processing task;
  - Some routing modules (Sbox) responsible of defining the correct path of the data and of guaranteeing the functionalities of each single integrated decoders;
  - A central controller, which is able to reconfigure the platform at runtime according to the information in the incoming stream.

The MDCC tool: Working Principle (1)



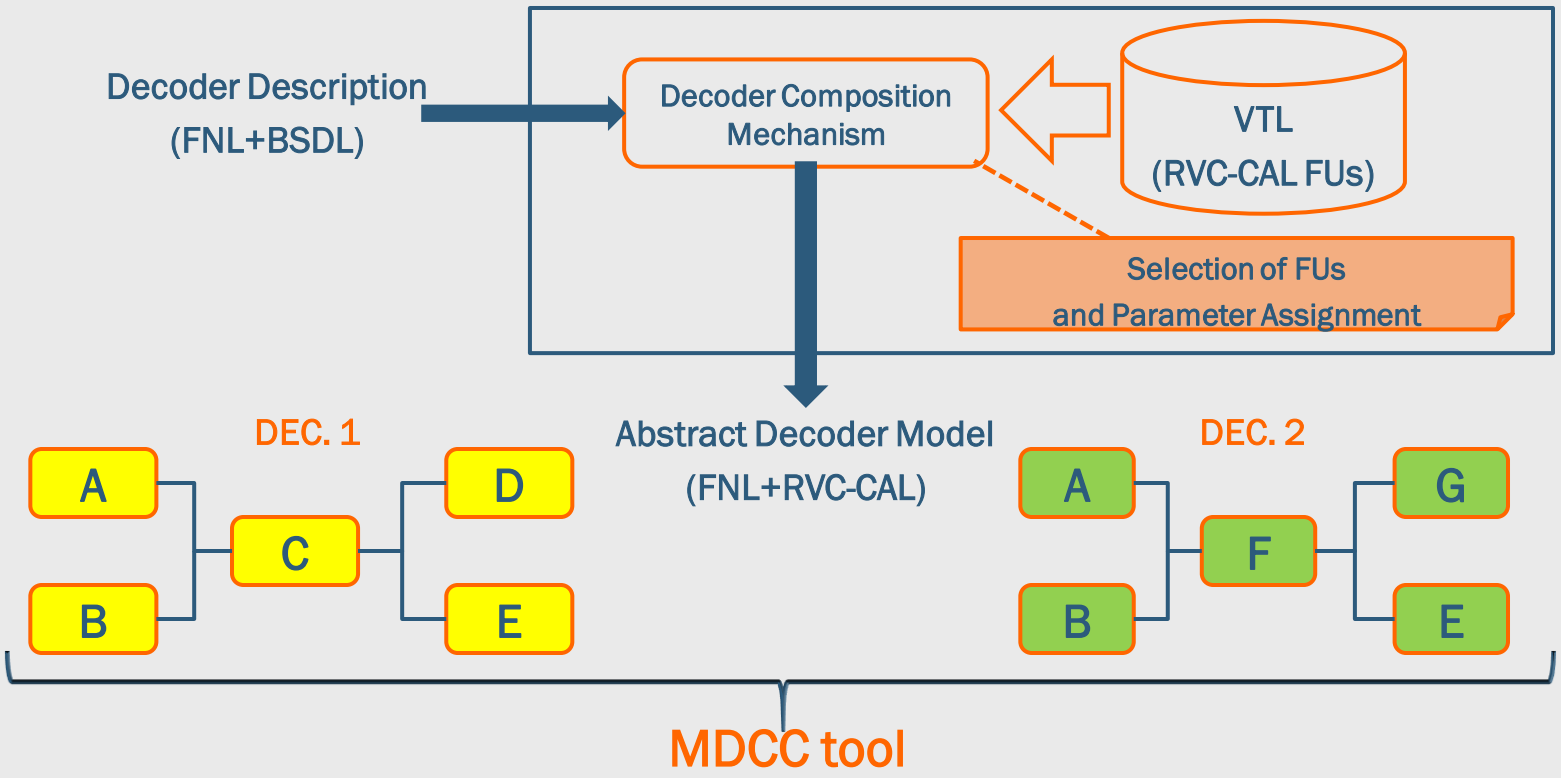
MDCC merges DEC.1 and DEC.2 creating a single multi-standard DEC, that can be easily converted into a proprietary hw/sw implementation.

The MDCC tool: Working Principle (2)

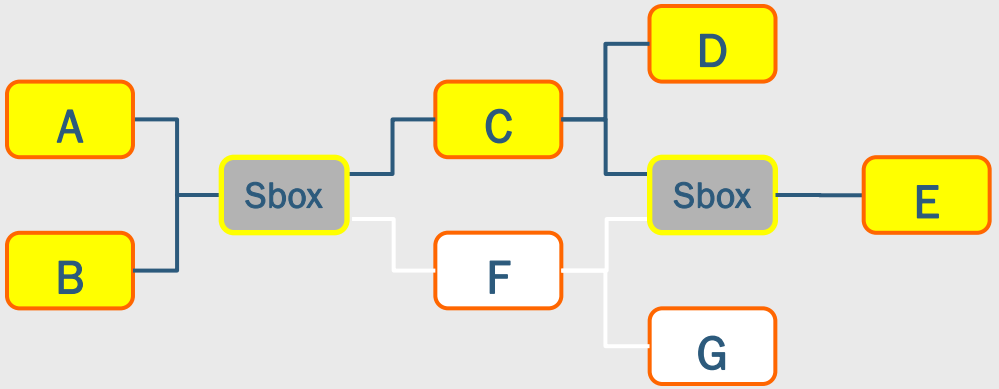


MDCC analyzes the RVC "abstract specifications" of DEC.1 and DEC.2 and inserts the Sboxes to enable the non-shared items just when needed.

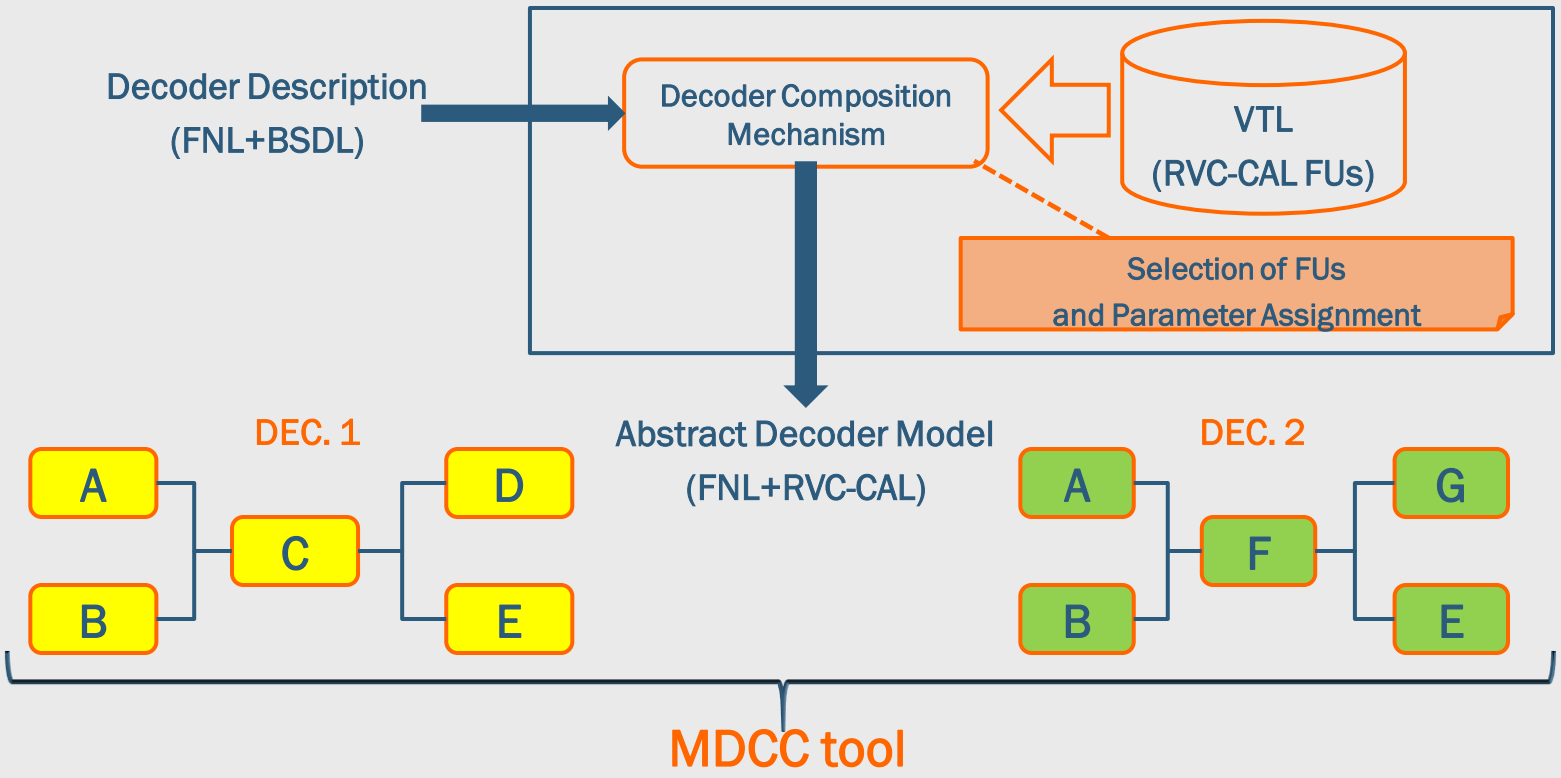
The MDCC tool: Working Principle (3)



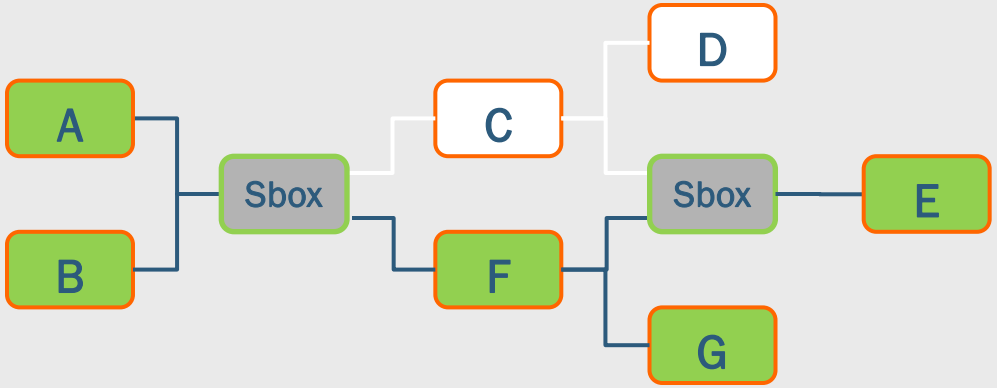
Sbox units allow to change, at runtime, the topology in order to define which DEC is in use.



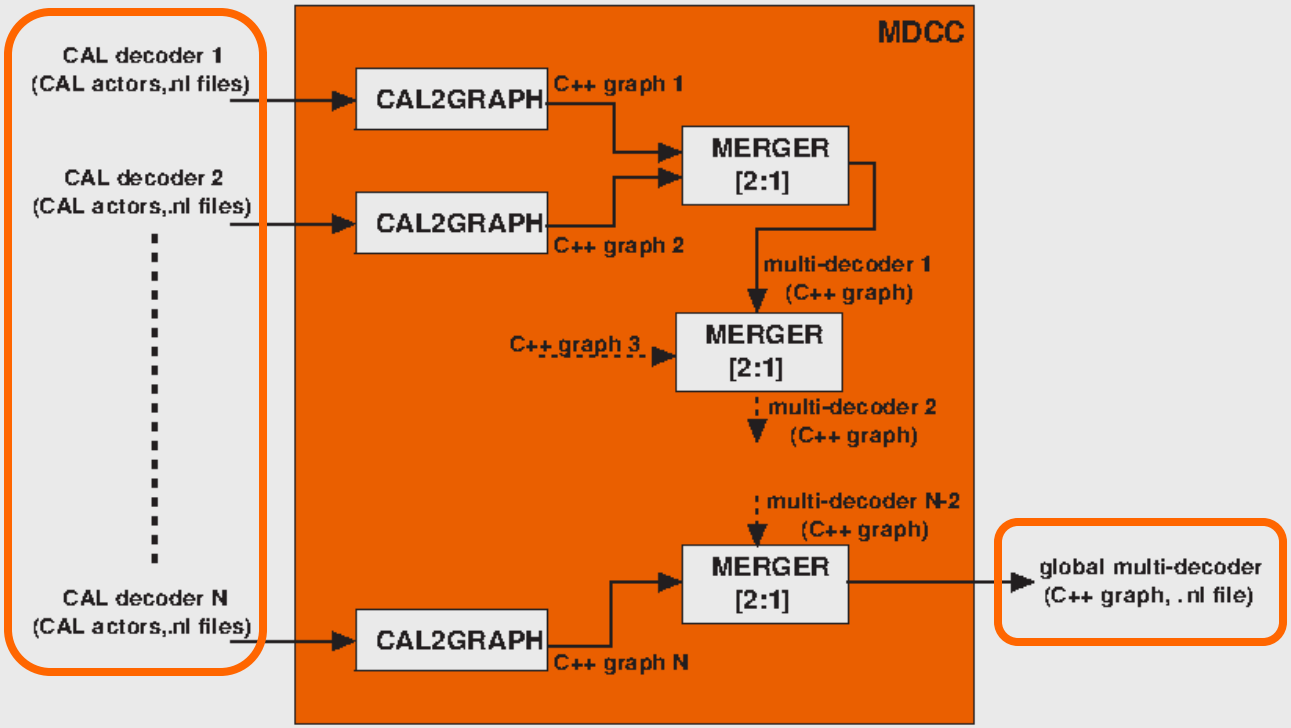
The MDCC tool: Working Principle (4)



MDCC allows to switch from DEC.1 to DEC.2 according to the incoming bitstream, without any complete context switch.

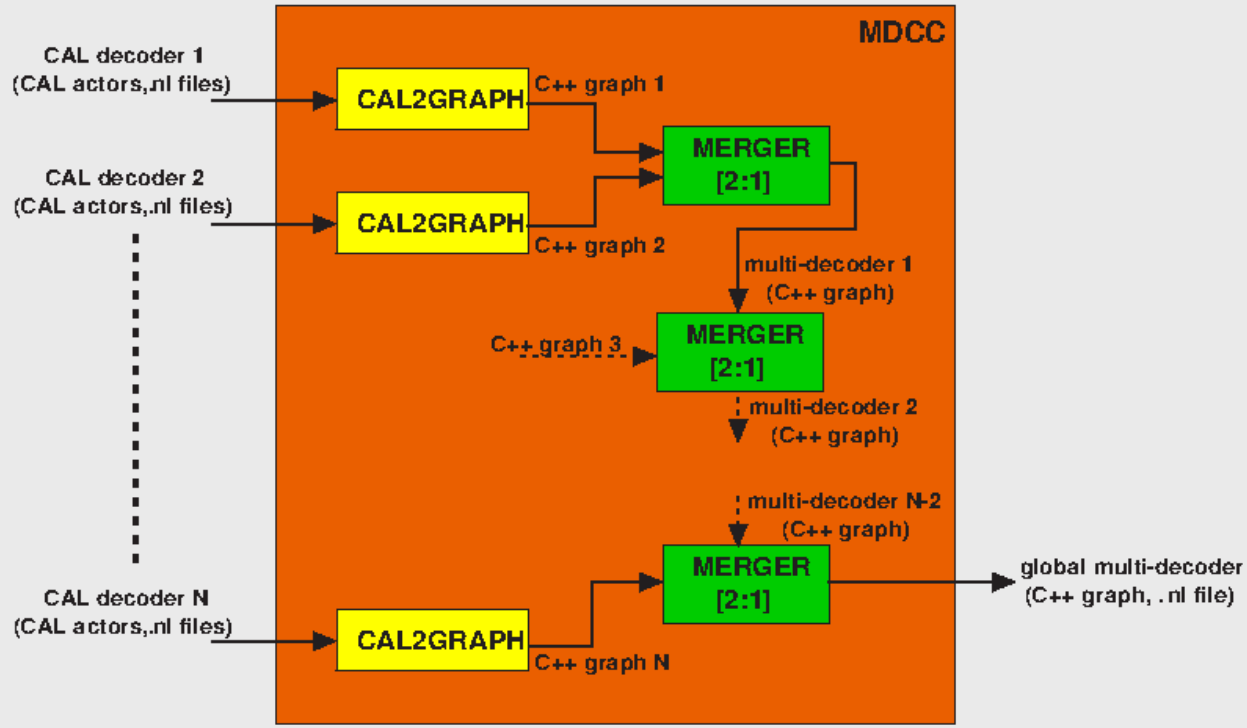


The MDCC tool: Input and Output



- The MDCC tool:
  - as inputs receives the RVC-CAL abstract DECs model
  - as outputs creates the C++ directed graph (DG) of the multi-decoder environment

The MDCC tool: Building Blocks

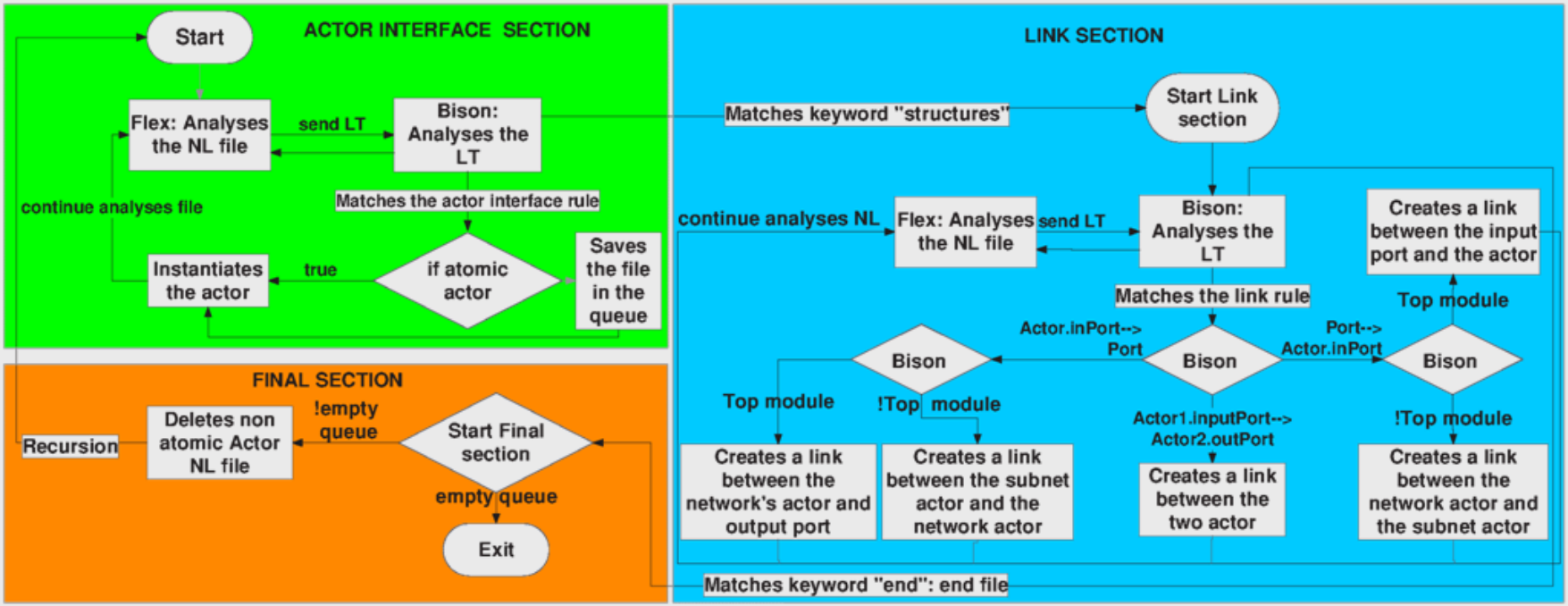


- The MDCC tool is composed of:
  - the CAL2GRAPH interpreter, which creates the C++ DG of a single DEC
  - the MERGER, which creates the multi-decoder C++ DG starting from two DGs.

The CAL2GRAPH: Basis

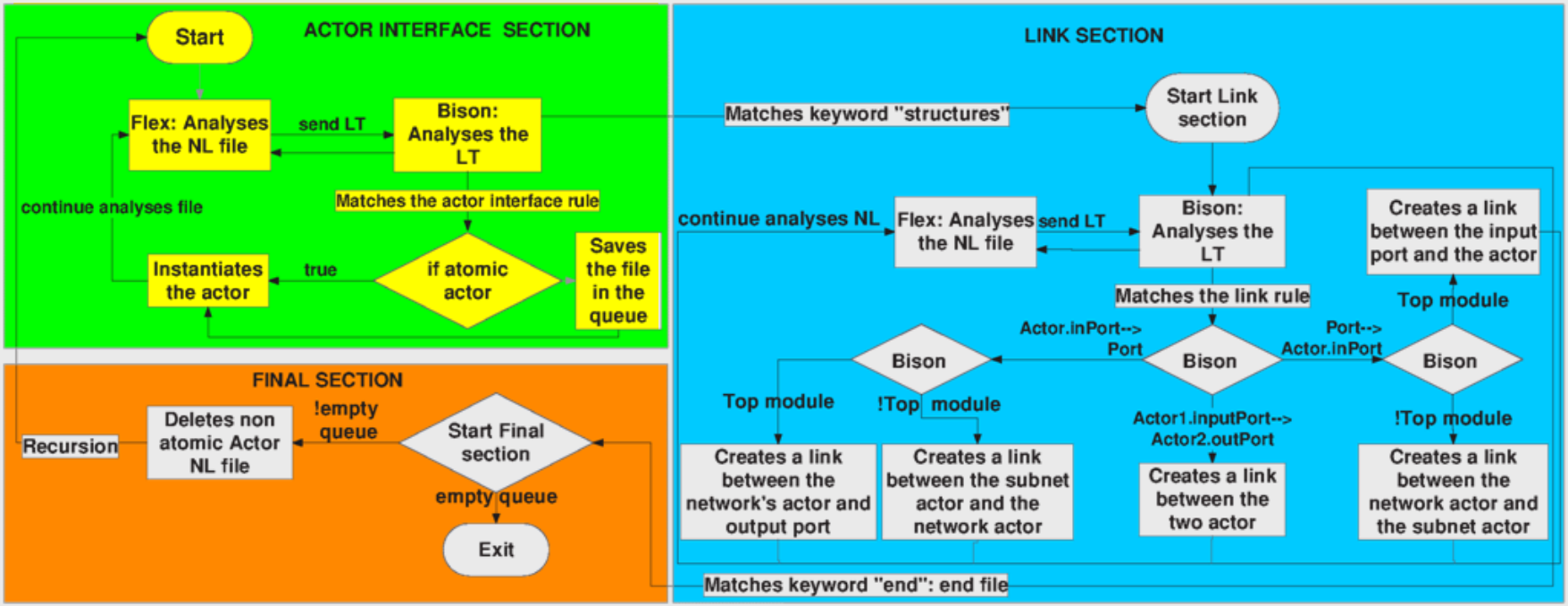
- It operates on the single DEC basis
- It generates a flattened C++ DG, composed of atomic actors only
  - Nodes : atomic actors
  - Arcs: connection among actors
- It works in three steps:
  - FNL analysis, using a FLEX lexical analyzer, to determine the Lexical Tokens (LTs)
  - LTs processing, using a Bison grammar interpreter, to create the DG
  - Recursive iteration until the DG is not completely atomic.

The CAL2GRAPH: Building Blocks



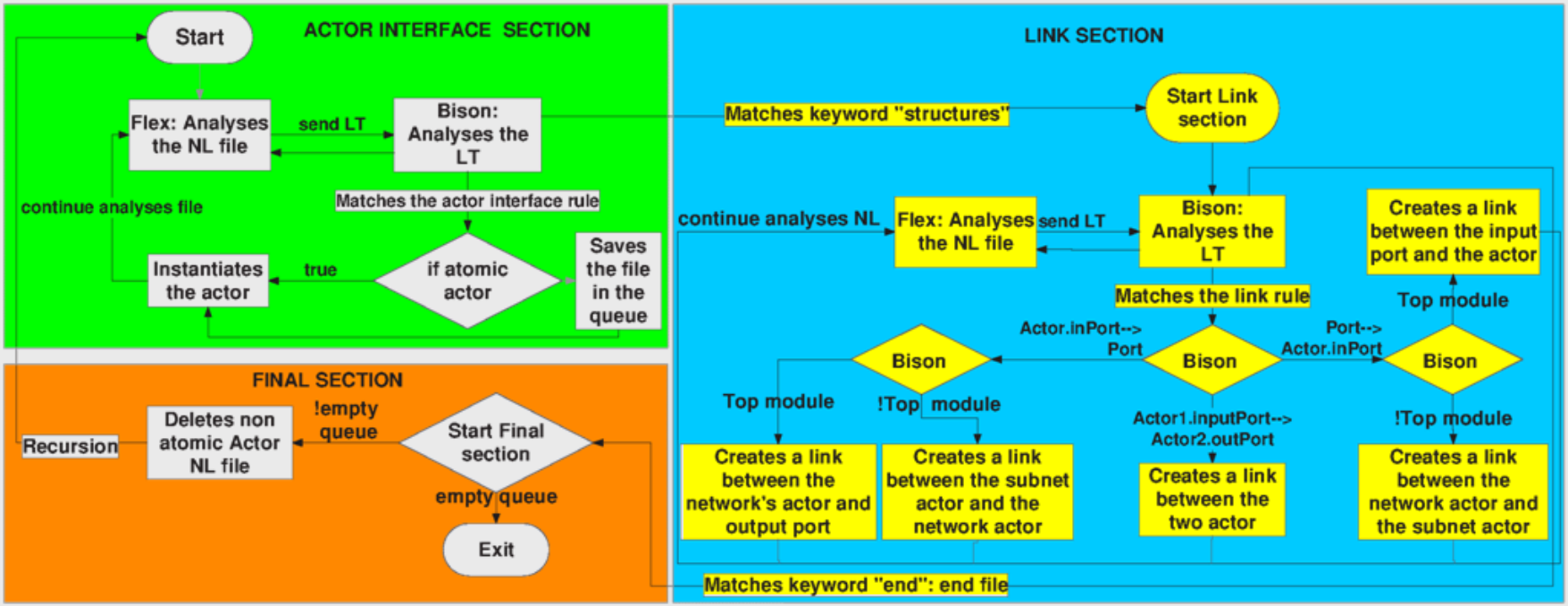
- Actor Interface Section -> nodes management
- Link Section -> connection management
- Final Section -> recursion management

The CAL2GRAPH: Actor Interface Section



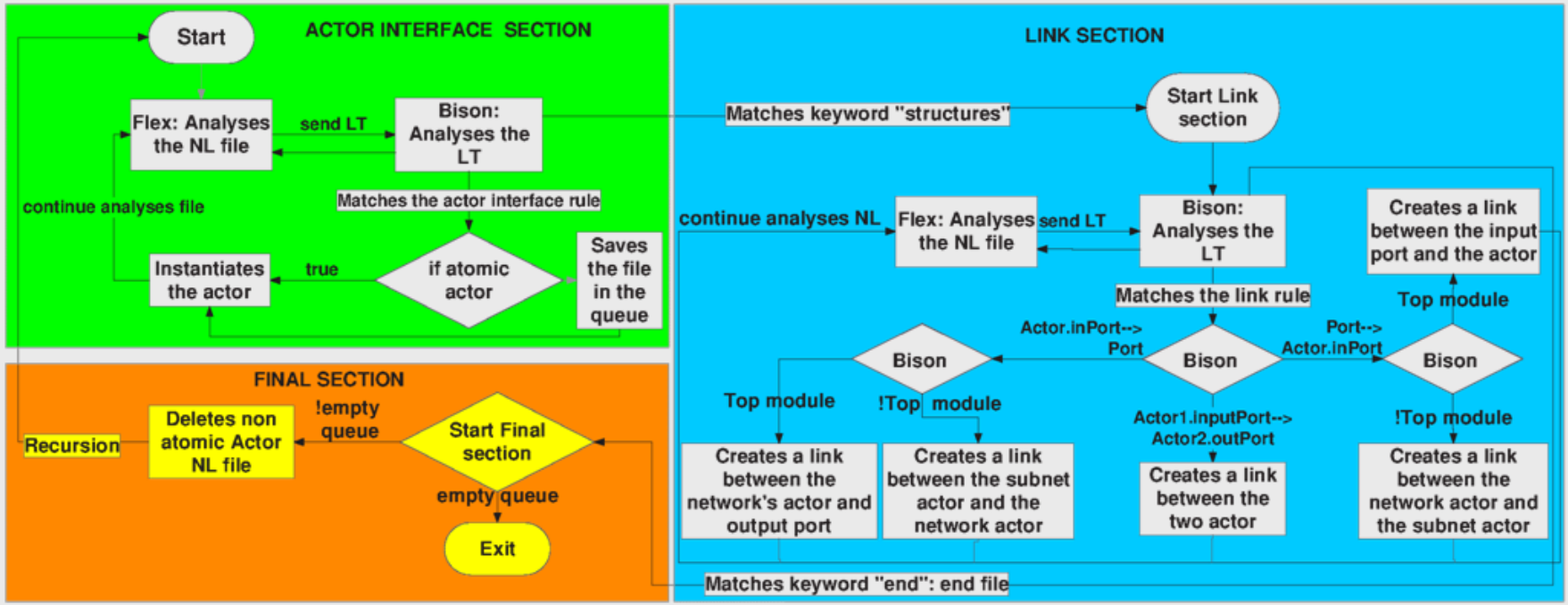
- Actor Interface Section -> nodes management through the FNL parsing:
  - as soon as an actor is found a node is instantiated in the DG
  - if the instantiated node is not atomic a reference is pushed in the recursion queue

The CAL2GRAPH: Link Section



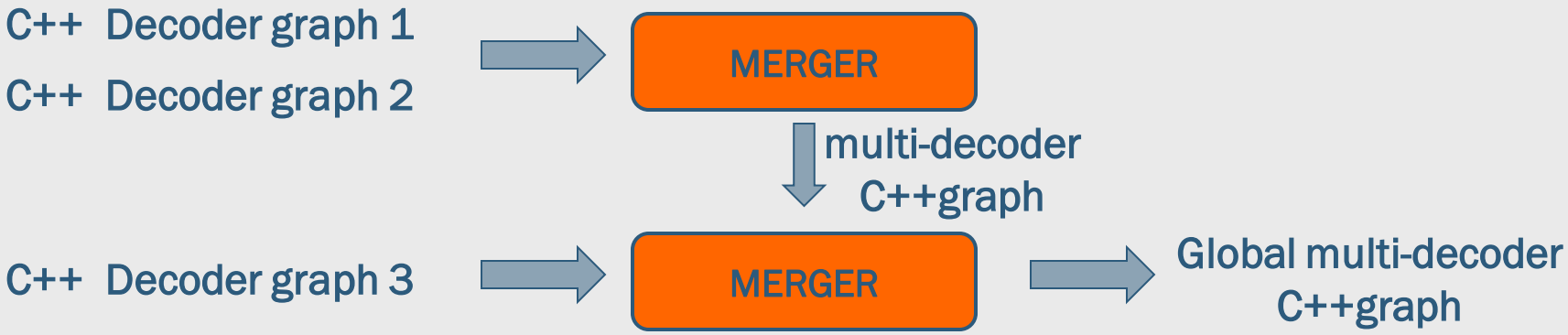
- Link Section -> connection management through the FNL parsing:
  - global input to network port or sub-network input port with network output port
  - network port to global output or sub-network output port with network input port
  - Generic intermediate nodes connection

The CAL2GRAPH: Final Section



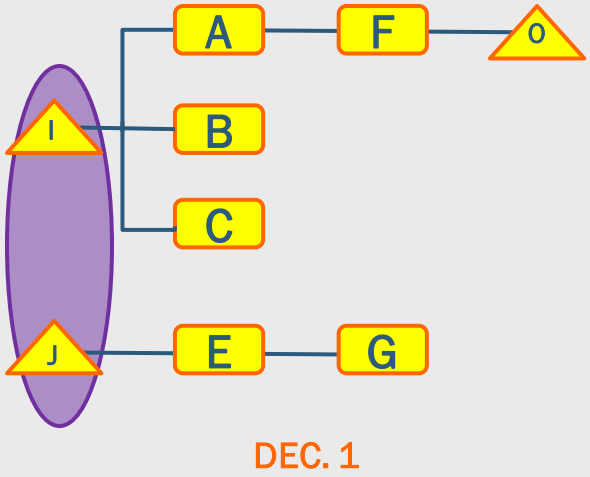
- Final Section -> recursion management through the recursion queue analysis:
  - If it is empty the DG is complete
  - If it is not empty the DG graph is still not completely atomic and recursion can take place, by popping a reference from the recursion queue and executing the CAL2GRAPH again

The MERGER: Basis



- It operates on a couple of C++ DGs
- It generates the global multi-decoder C++ DG of the entire multi-standard system
  - Integrating the whole set of actors of the original DGs.
  - Preserving the connections among nodes.
  - Placing the Sbox units are at the crossroads between different paths, to allow more than one DG to share a common dataflow where some actors are in common and some others are not.

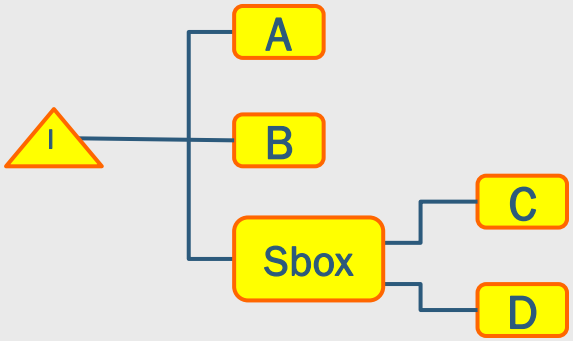
The MERGER: Application Example (1)

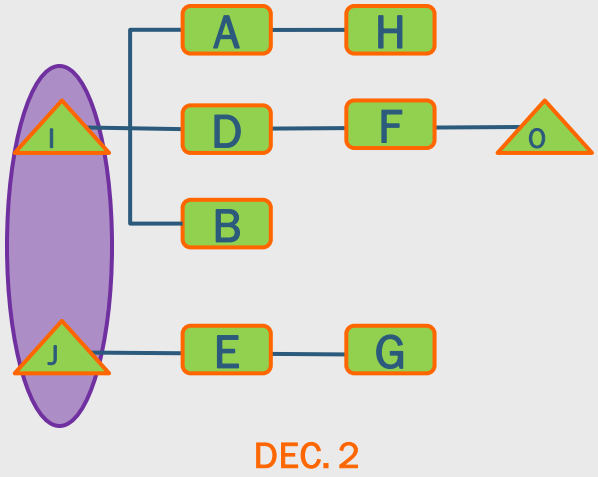


DEC. 1

Father Nodes
[I,J]

Children Nodes
[A,B,C,D]

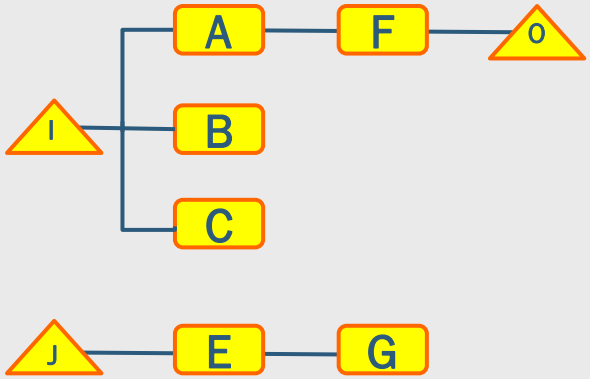




DEC. 2

- The MERGER starts from the root of the two dataflows
- Three iterations on I:
  - I to A, I to B -> the same on both reported as they are
  - I to C/D -> Sbox to be inserted
- Children Set: [A,B,C,D]

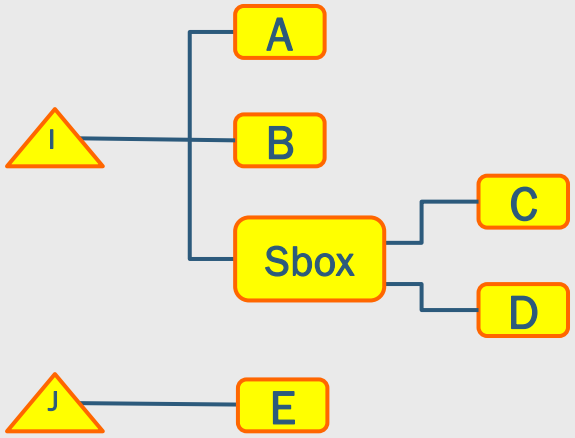
The MERGER: Application Example (2)

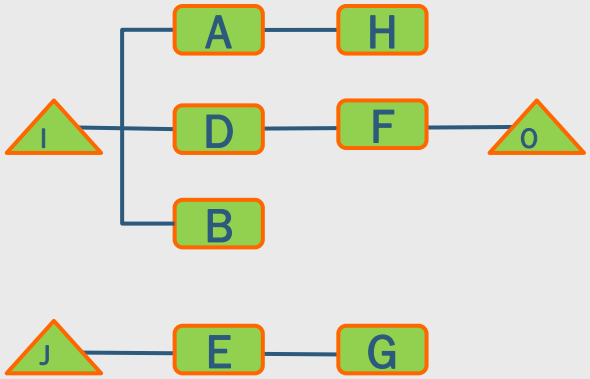


DEC. 1

Father Nodes
[I,J]

Children Nodes
[A,B,C,D,E]

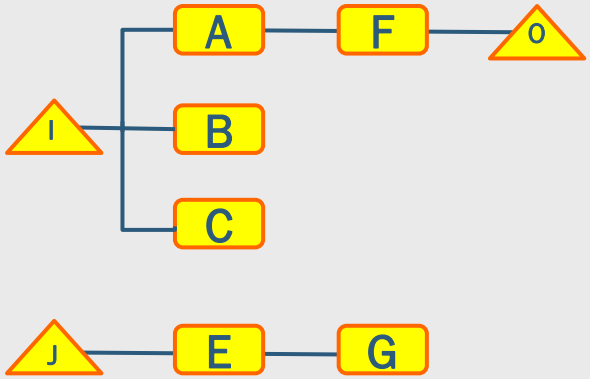




DEC. 2

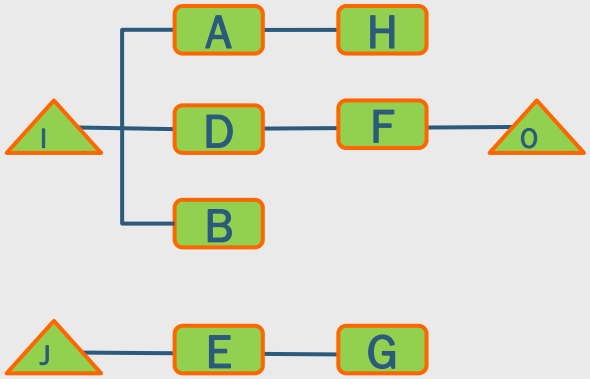
- One iteration on J:
  - J to E -> the same on both reported as it is
- Iteration on Father Nodes is complete
- Children Set: [A,B,C,D,E]. Father Nodes = Children Nodes

The MERGER: Application Example (3)



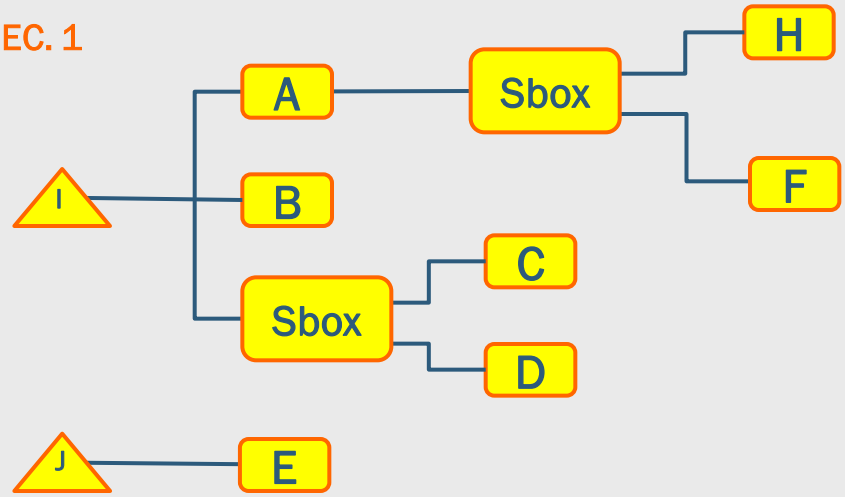
Father Nodes
[A,B,C,D,E]

Children Nodes
[F,H]



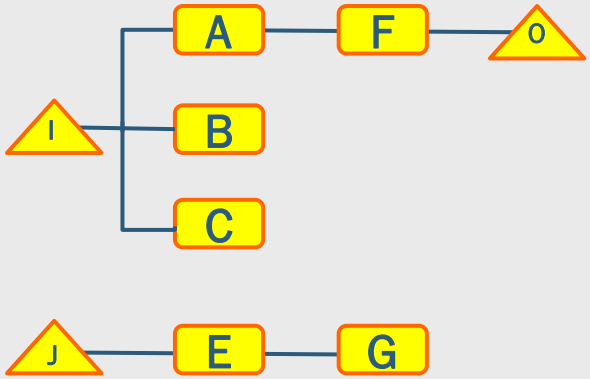
DEC. 1

DEC. 2



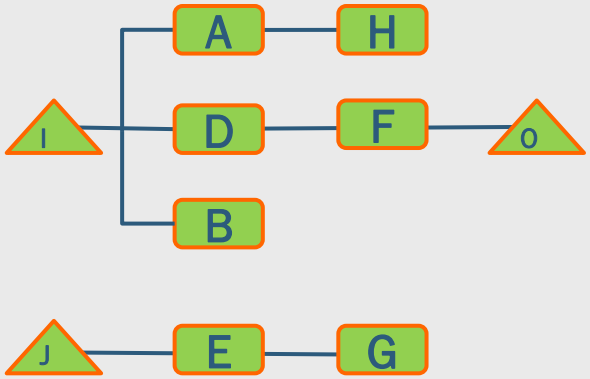
- One iteration on A:
  - A to F/H -> Sbox to be inserted
- Children Set: [F,H]

The MERGER: Application Example (4-5)

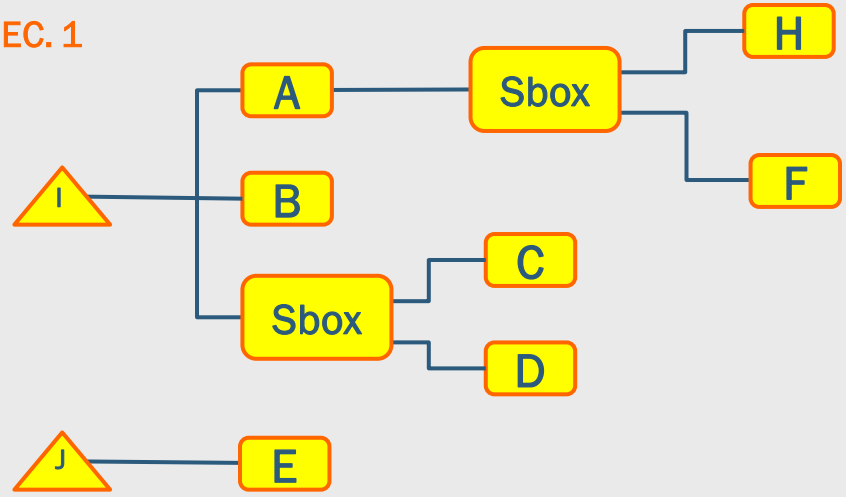


Father Nodes
[A,B,C,D,E]

Children Nodes
[F,H]



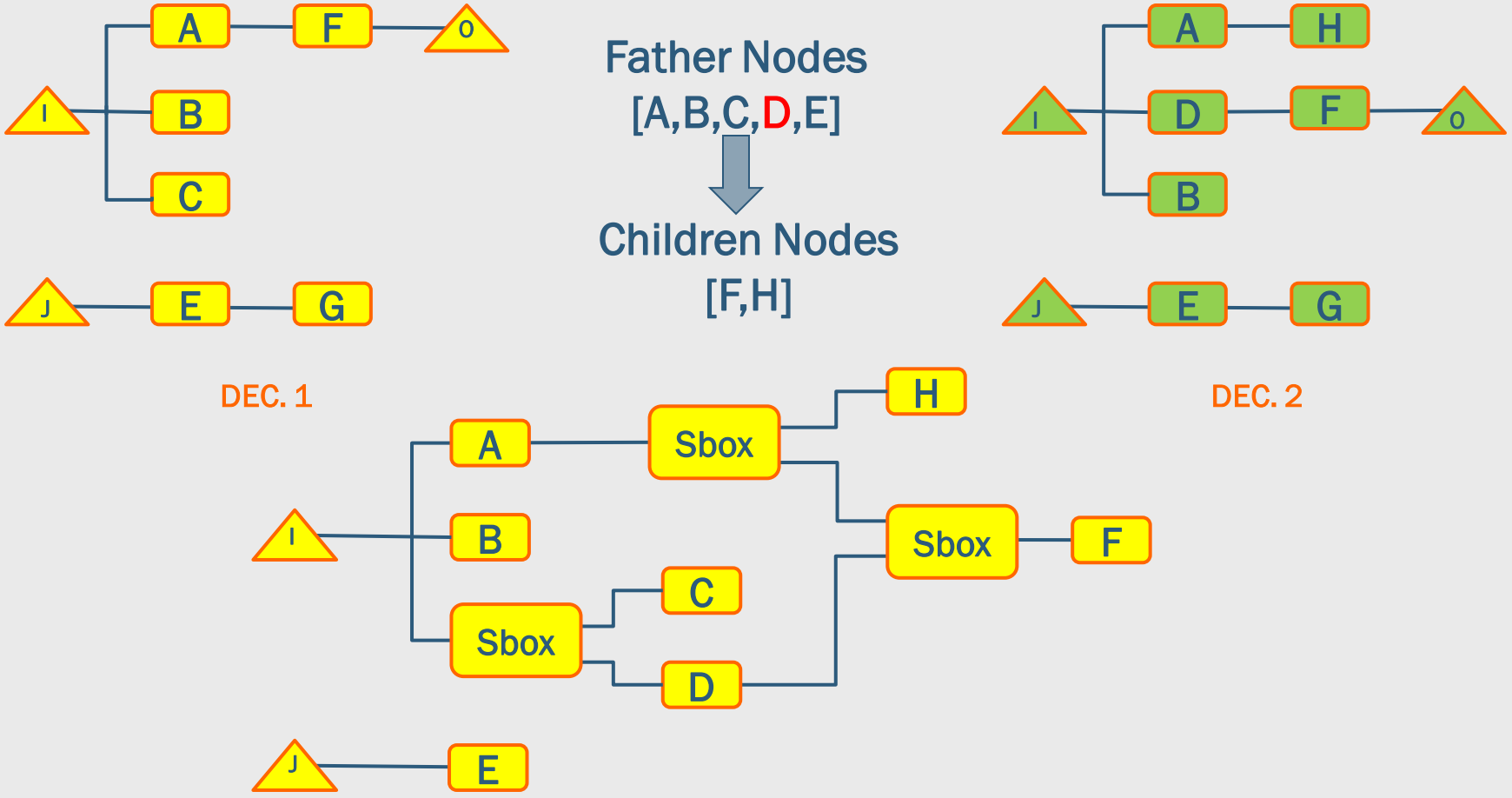
DEC. 1



DEC. 2

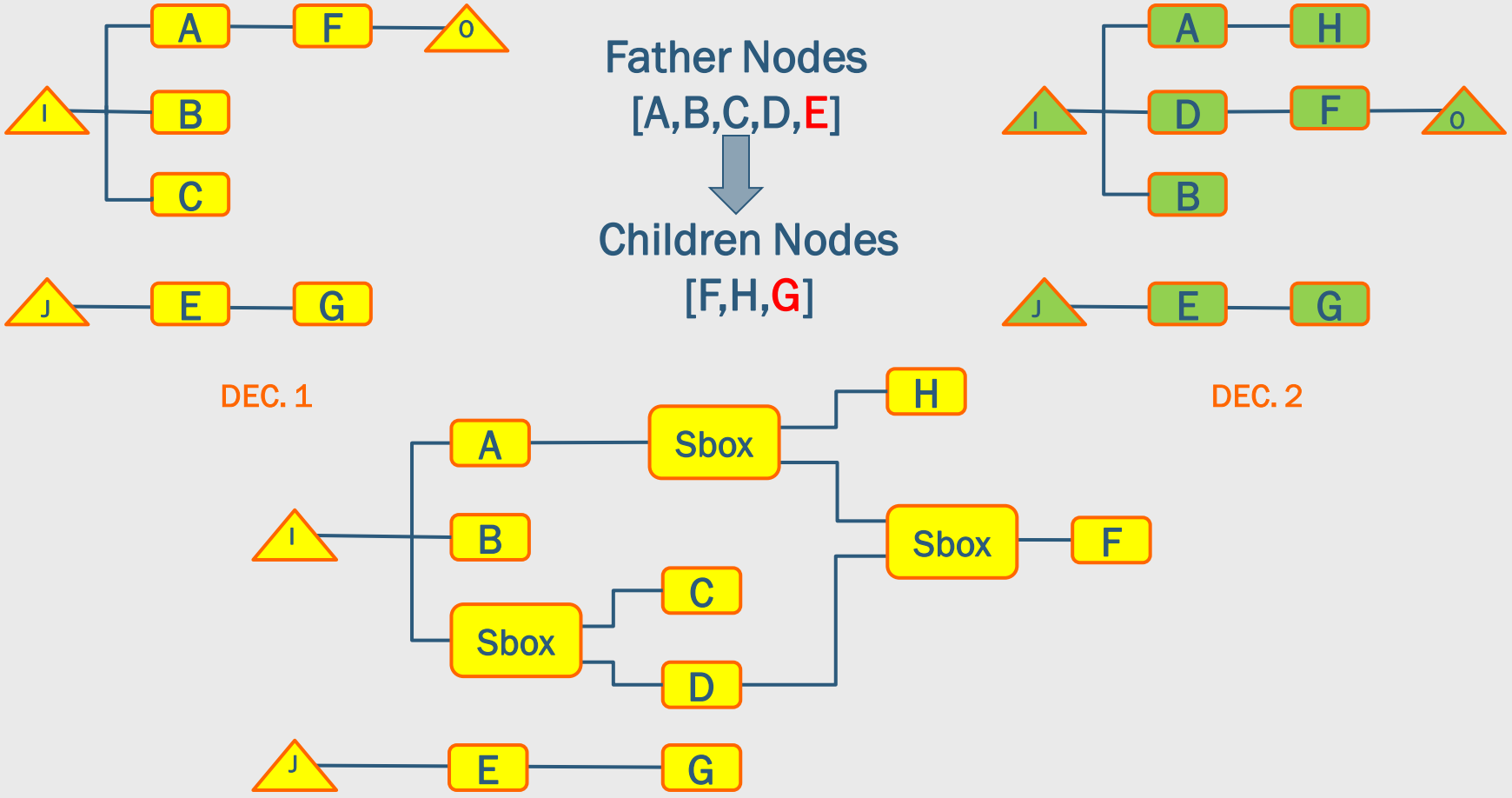
- Iteration on B and C:
  - Nothing has to be done
- Children Set: [F,H]

The MERGER: Application Example (6)



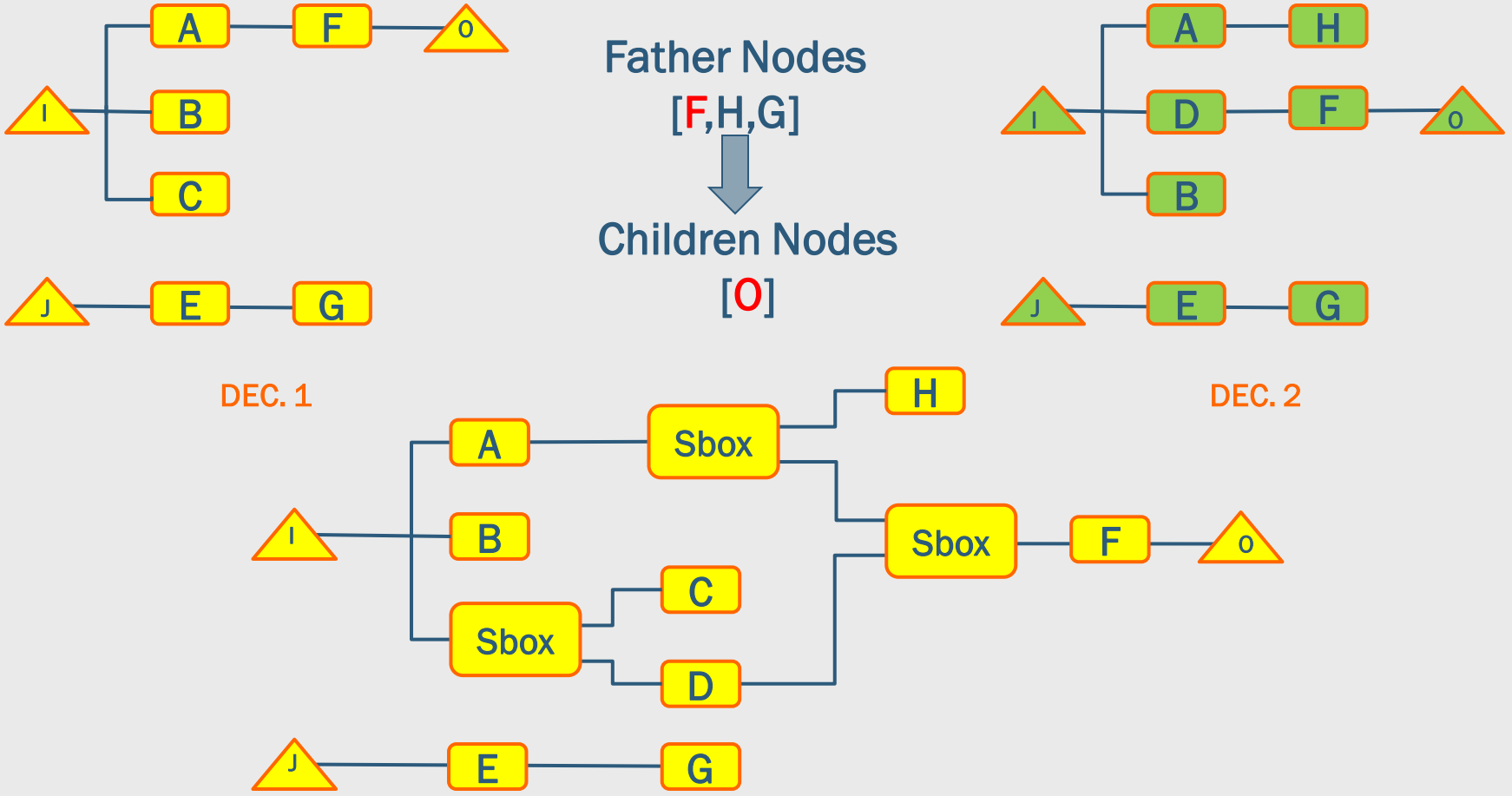
- One iteration on D:
  - D to F -> F is already in the global DG -> Sbox to be inserted
- Children Set: [F,H]

The MERGER: Application Example (7)



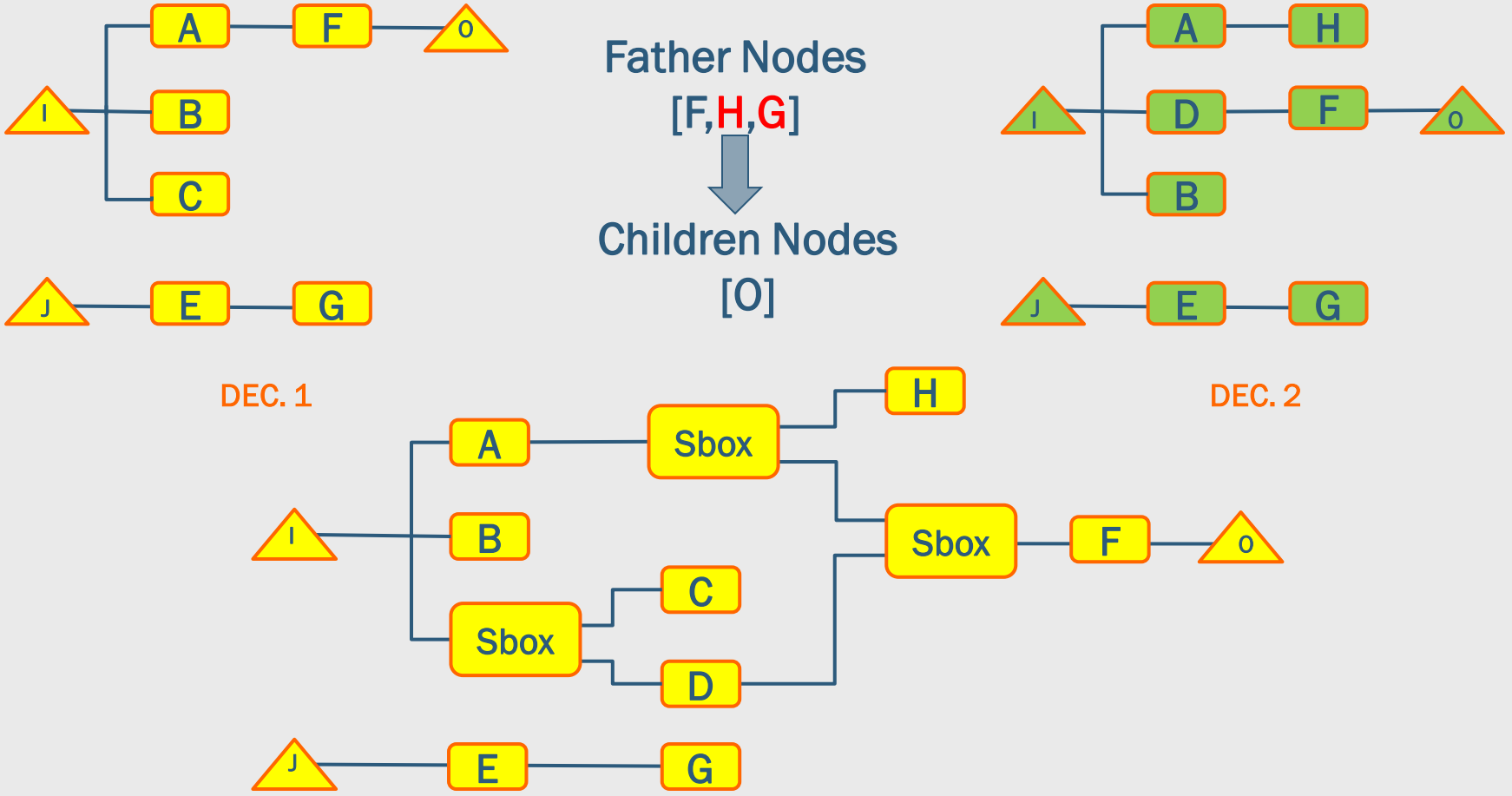
- One iteration on E:
  - E to G -> the same on both reported as it is
- Children Set: [F,H,G]. Iteration on Father Nodes is complete

The MERGER: Application Example (8)



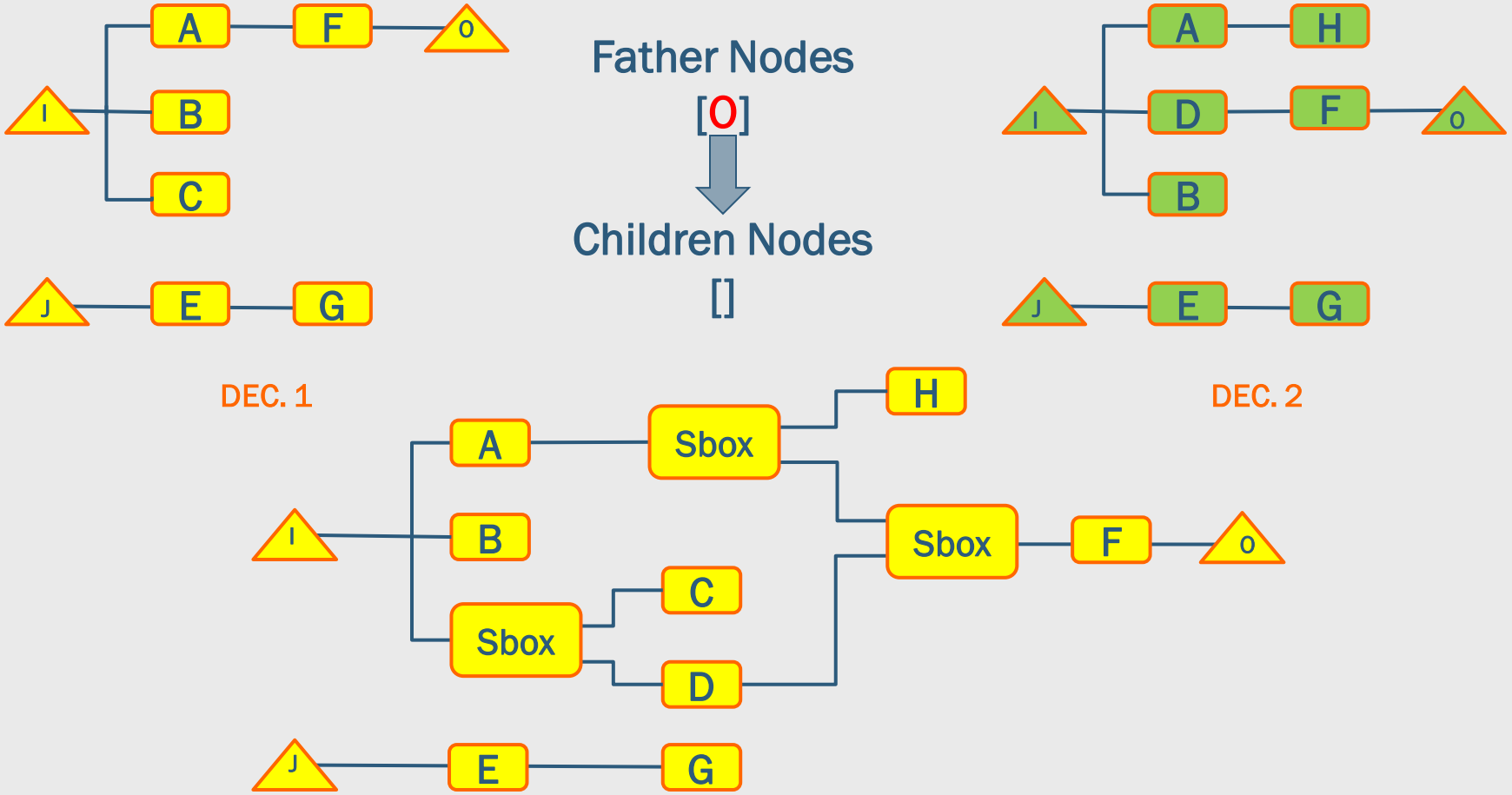
- One iteration on F:
  - F to O -> the same on both reported as it is
- Children Set: [O]

The MERGER: Application Example (9-10)



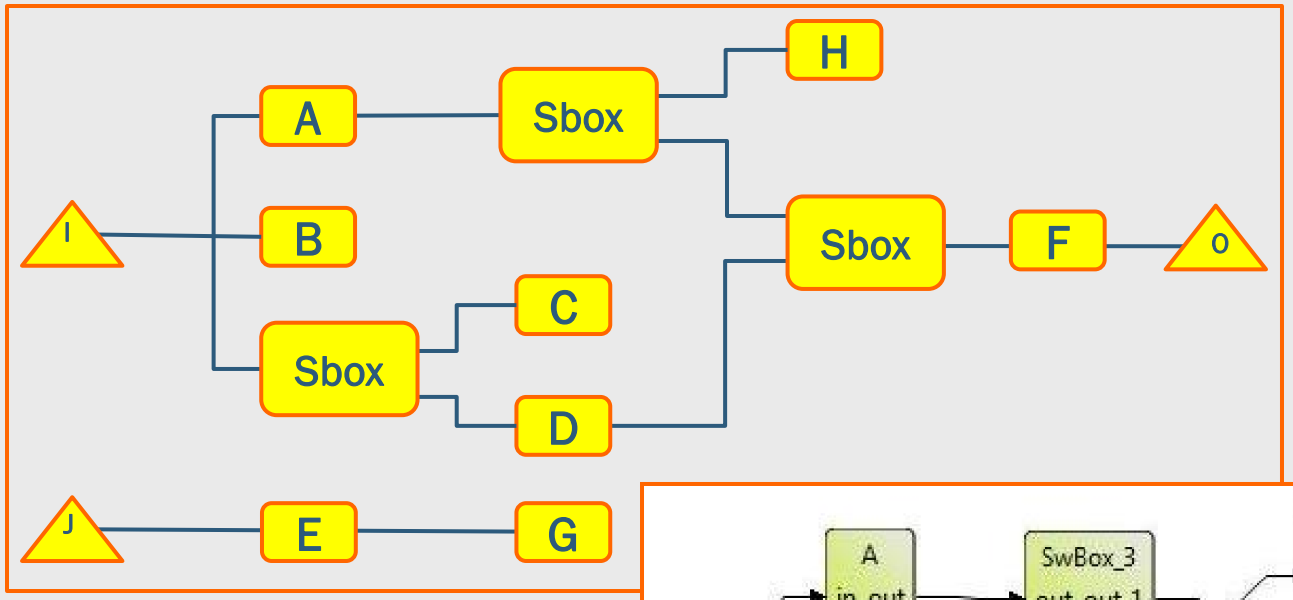
- Iteration on H and G:
  - Nothing has to be done
- Children Set: [O]

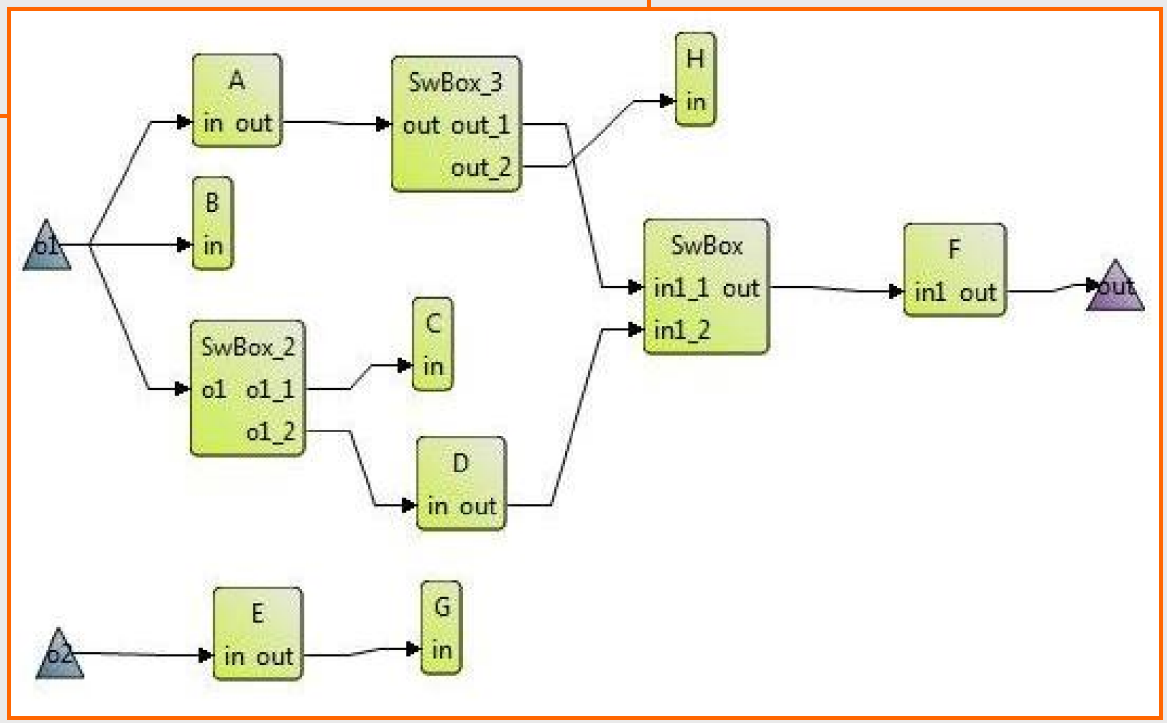
The MERGER: Application Example (11)



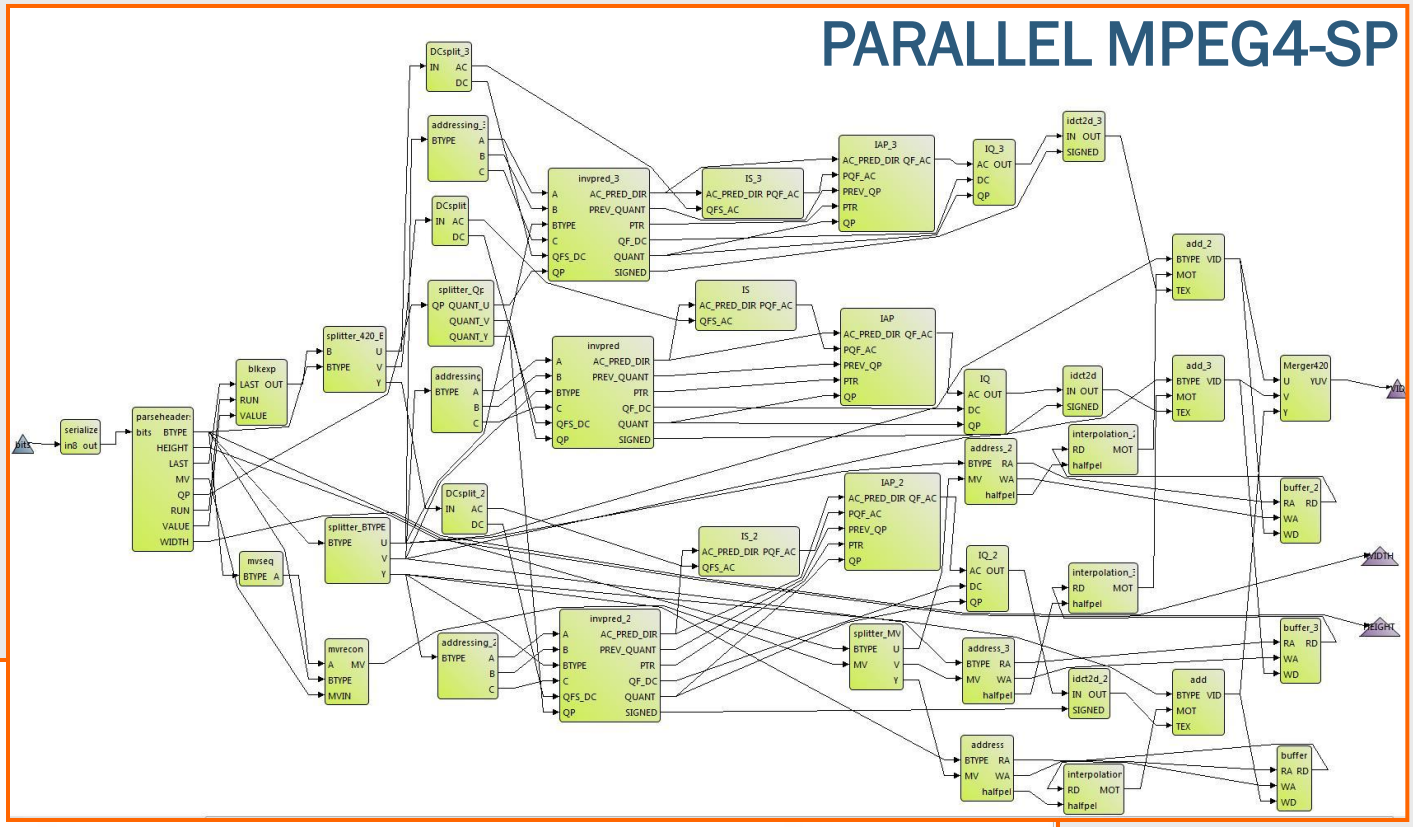
- One iteration on O:
  - O is a global output, nothing has to be done
- Children Set: []. The MERGER terminates.

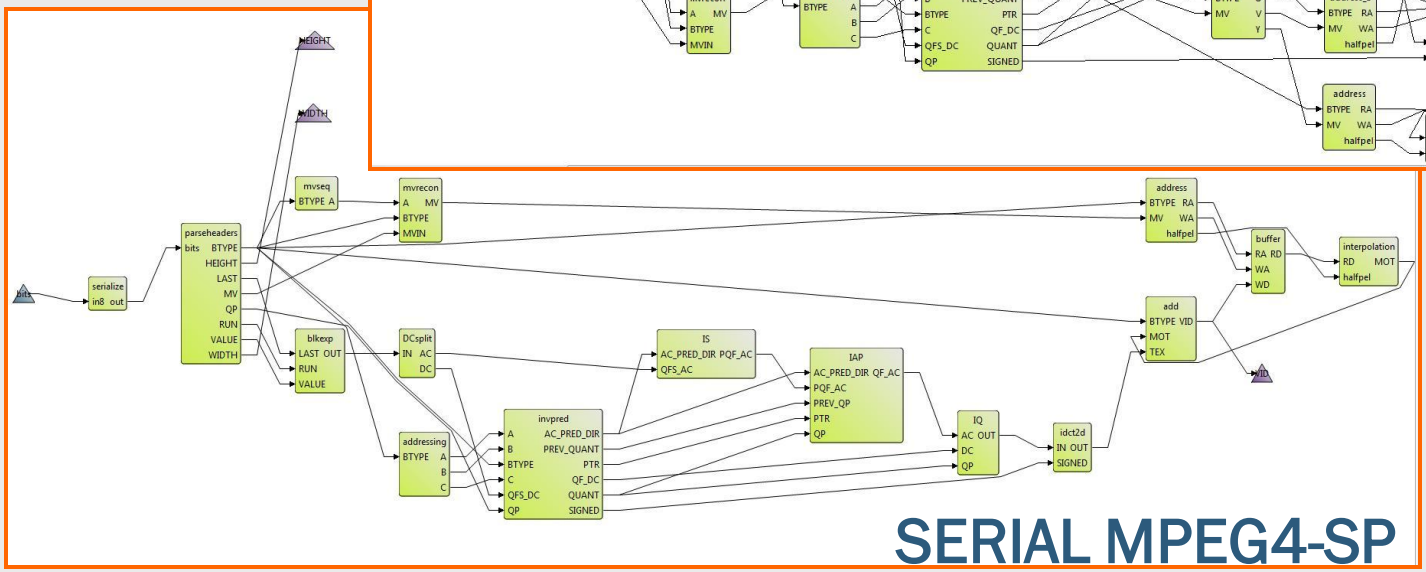
The MERGER: Application Example





Use Case: Parallel and Serial MPEG-4 SP(1)





DASIP 2010 – October 26th-28th, Edinburgh, Scotland

Use Case: Parallel and Serial MPEG-4 SP(2)

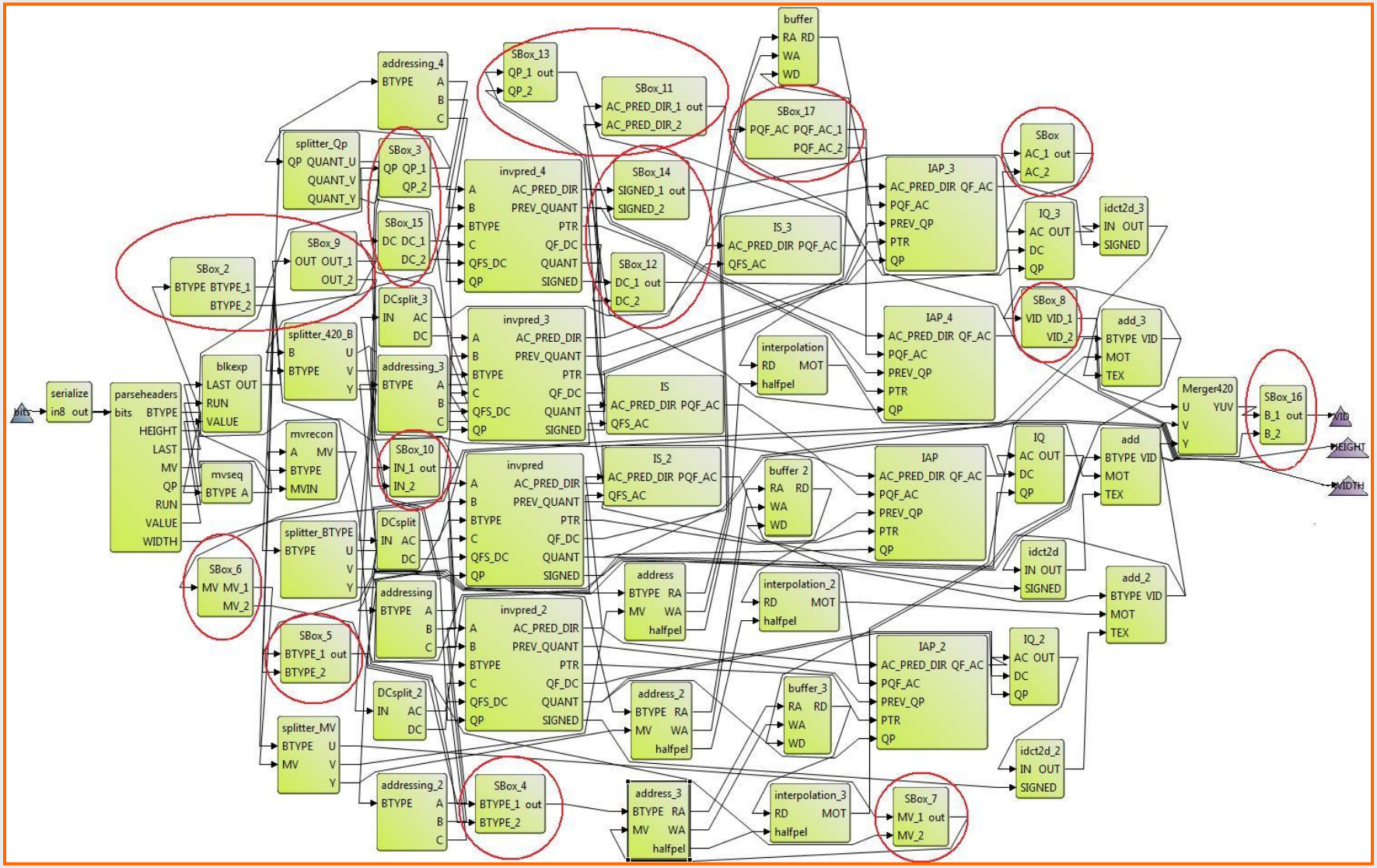
Actor	Parallel SP	Serial SP	Actor Sum	MDCC Applied
serialize	1	1	2	1
parseheaders	1	1	2	1
blkexp	1	1	2	1
mvseq	1	1	2	1
mvrecon	1	1	2	1
DCsplit	3	1	4	3
splitter_420_B	1	0	1	1
splitter_MV	1	0	1	1
spliter_Qp	1	0	1	1
splitter_BTYPE	1	0	1	1
DCRaddressing_16x16	1	0	1	1
Algo_DCRaddressing_8x8	2	0	2	2
Algo_DCRaddressing	0	1	1	1
Algo_DCRinvpred_luma_16x16	1	0	1	1
Algo_DCRinvpred_chroma_8x8	2	0	2	2
Algo_DCRinvpred	0	1	1	1
IS	3	1	4	3
Algo_IAP_16x16	1	0	1	1
Algo_IAP	0	1	1	1
Algo_IAP_8x8	2	0	2	2
IQ	3	1	4	3
idct2d	3	1	4	3
add	3	1	4	3
address	3	1	4	3
buffer	3	1	4	3
interpolation	3	1	4	3
Merger420	1	0	1	1
Sbox	0	0	0	17

- Multi-decoder units:

  46 instead of 59.
- Multi-decoder units + Sbox:

  64 instead of 59.
- It is less costly to integrate Sbox units than more complex ones (e.g. DCT and FFT).
- The more the integrated dataflows the more will be the instantiated Sbox units.

Use Case: Parallel and Serial MPEG-4 SP(3)



Final Remarks

- Exploiting the modularity property of the RVC formalism it is possible to implement an automatic method for video codecs formal definition.
- The proposed Multi-Decoder CAL Composer tool has been already successfully tested on factitious dataflows and also on a MPEG-4 SP use case, merging its serial and parallel descriptions.
- This tool is fully compatible with other RVC-CAL state of the art tools: all the outputs are provided along with the FNL descriptions.

The MDCC tool Possible Exploitations

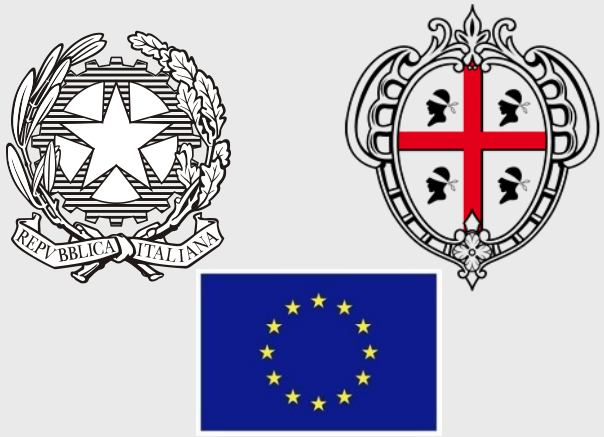
- From the hardware viewpoint:
  - Completing the MDCC with a tool called graph2HW in order to accomplish the automatic creation of the HDL description of the overall multi-decoder platform.

- From the software viewpoint:
  - Completing the MDCC with a high level profiler tool in order to allow to bridge the gap between hardware physical implementation and software development. This tool will operate at the direct graph level combining lower level back-annotated information with higher level functional information.

Acknowledgements

The research leading to these results has received funding from:

the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 248424, MADNESS Project



the Region of Sardinia, Young Researchers Grant, PO Sardegna FSE 2007-2013, L.R.7/2007 “Promotion of the scientific research and technological innovation in Sardinia” under grant agreement CRP-18324 RPCT Project

Conference on Design and Architectures for Signal and Image Processing -2010
Electronic Chips & Systems design Initiative
October 26th-28th, 2010,
Edinburgh, Scotland

RVC: A MULTI-DECODER CAL COMPOSER TOOL

Francesca Palumbo, Ph.D.
francesca.palumbo@diee.unica.it
EOLAB - Microelectronics Lab
Dept. of Electrical and Electronics Eng.
University of Cagliari (ITALY)