



# Component-based Waveform Development: The Nucleus Tool Flow for Efficient and Portable SDR

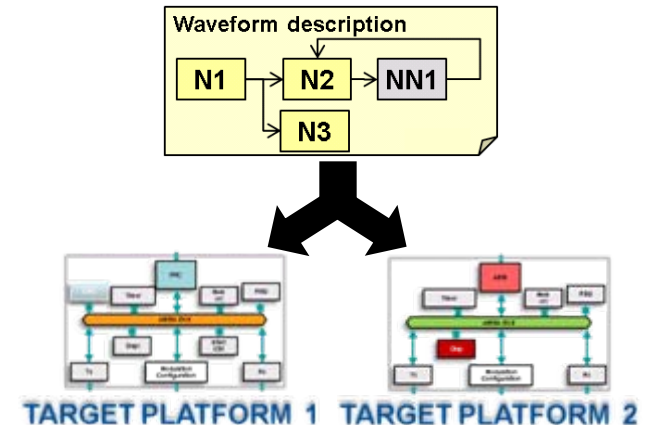
J. Castrillon, S. Schürmans, A. Stulova, W. Sheng, T. Kempf, A. Ishaque,  
R. Leupers, G. Ascheid, H. Meyr  
[Jeronimo.Castrillon@iss.rwth-aachen.de](mailto:Jeronimo.Castrillon@iss.rwth-aachen.de)

SDR Dec. 2010



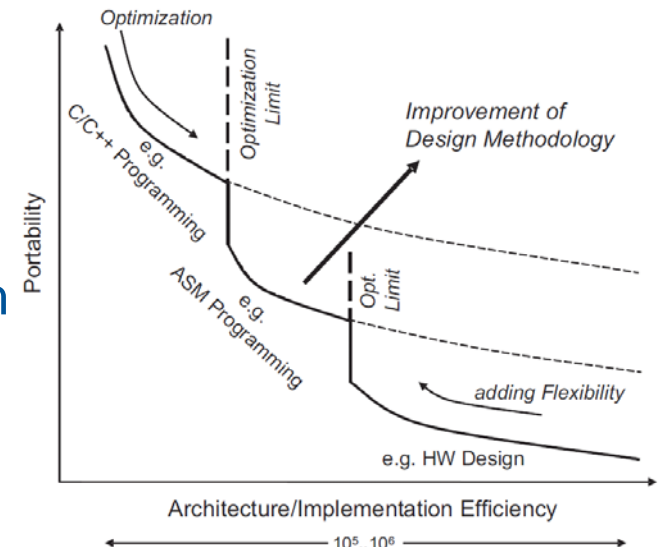
- **Benefits of Component-based SW Engineering (CBSE)**

- SW productivity increase
- Familiar block diagram paradigm
- Potentially portable specification



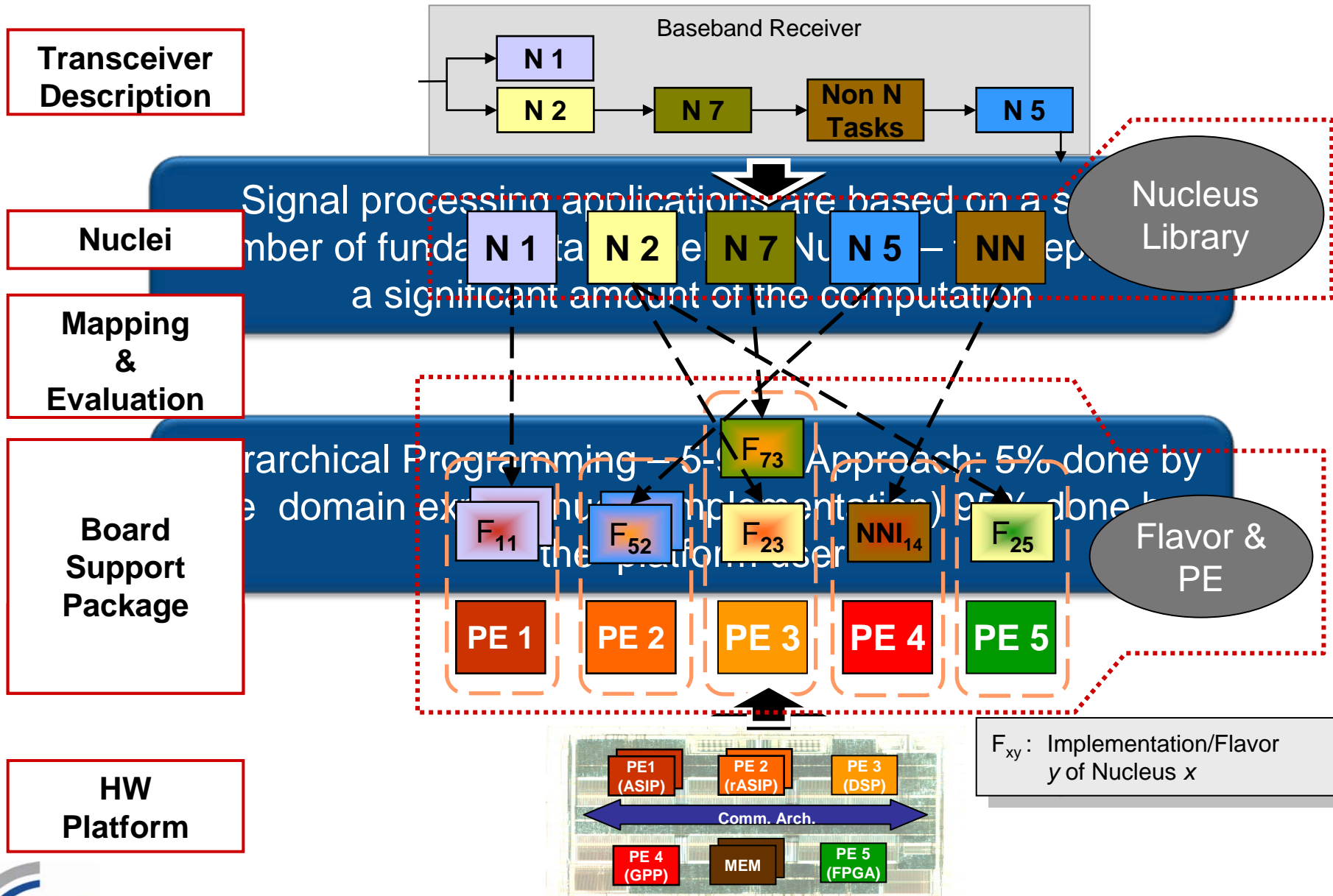
- **Problems:**

- Portability vs. efficiency
  - Big gap between high level languages and architectures
- Efficiency: normally achieved with the library approach
  - Architecture dependent



Source: Witte et al. WSR'08

# Nucleus Approach: Overview



- **Motivation**



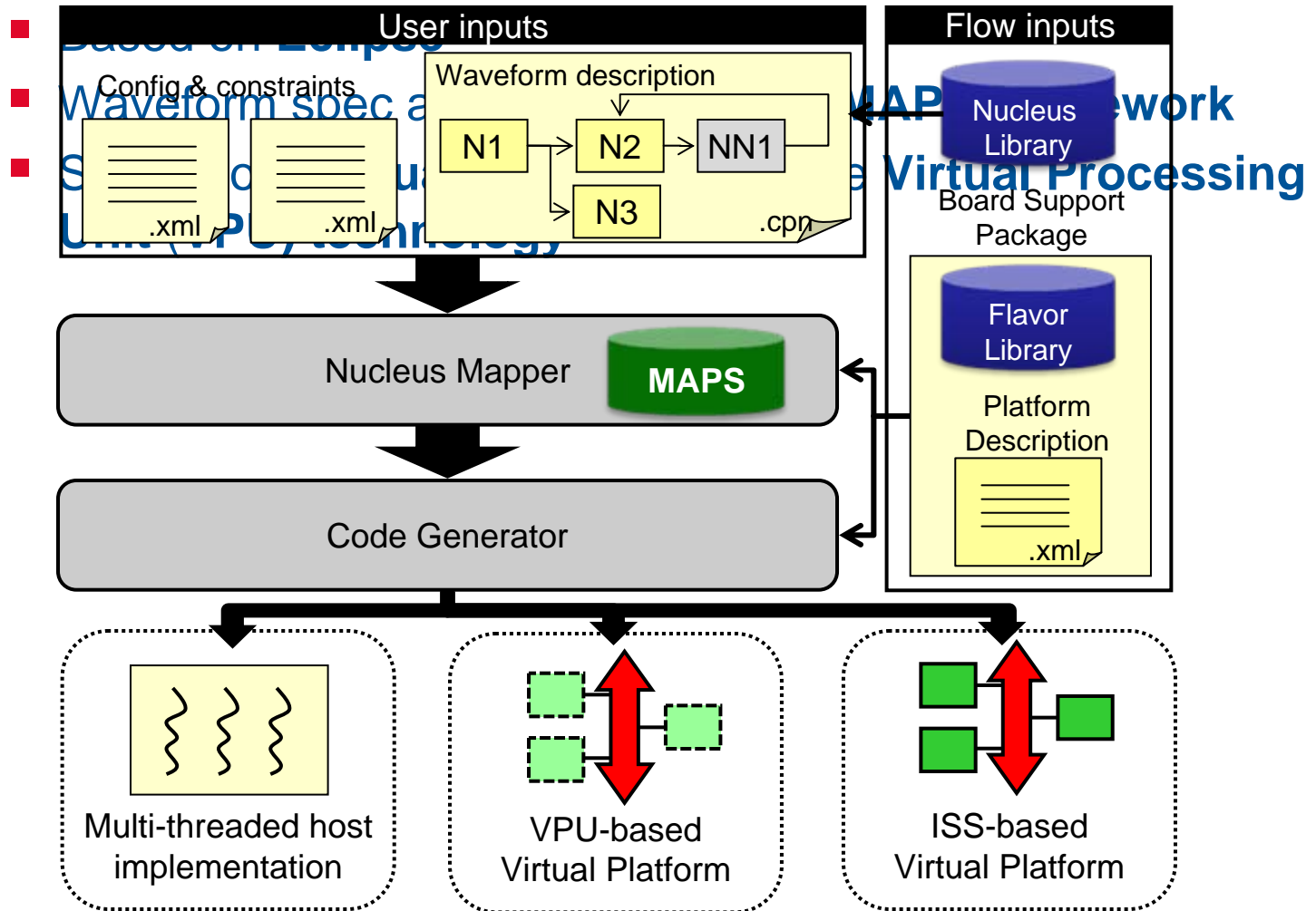
## **Tool Flow:**

- Overview
- Inputs
- Waveform mapping
- Code generation

- **Case study**

- **Summary and Outlook**

## Waveform Development Environment (WDE)



## ■ GUI:

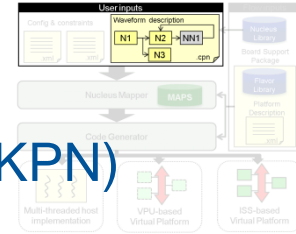
The screenshot displays the Eclipse IDE interface for the SDR'10 project. The main editor shows the 'mimo\_ofdm.cpn' file with the following C++ code snippet:

```

_PNchannel Comp r_an1, r_an2, r_an3, r_an4;
_PNchannel Comp r_demod1[N_FFT], r_demod2[N_FFT], r_demod3[N_FFT], r_demod4[N_FFT];
_PNchannel Fp r_demap[INTLV_BLOCK_LEN], r_chan[CHAN_BLOCK_LEN];
_PNchannel short src[BLOCK_LEN], r_src[BLOCK_LEN];
_PNchannel int fake;
_PNprocess source=SOURCE __PNout(src, r_an1, r_an2, r_an3, r_an4);
_PNprocess fft1=FFT __PNin(r_an1) __PNout(r_demod1) __PNparam(1);
_PNprocess fft2=FFT __PNin(r_an2) __PNout(r_demod2) __PNparam(2);
_PNprocess fft3=FFT __PNin(r_an3) __PNout(r_demod3) __PNparam(3);
_PNprocess fft4=FFT __PNin(r_an4) __PNout(r_demod4) __PNparam(4);
_PNprocess chanest=CHANEST __PNin(r_demod1, r_demod2, r_demod3, r_demod4) __PNout(fake);
_PNprocess soft_demap=SOFT_DEMAP __PNin(r_demod1, r_demod2, r_demod3, r_demod4, fake) __PNout(f);
_PNprocess deintrl=DEINTRL __PNin(r_demap) __PNout(r_chan);
_PNprocess softin_viterbi_chandec=SOFTIN_VITERBI_CHANDEC __PNin(r_chan) __PNout(r_src);
_PNprocess ber=BER __PNin(r_src, r_src);
    
```

The right sidebar shows the Palette with various blocks and edges. The bottom status bar indicates the selected object is 'Property Function latency'.

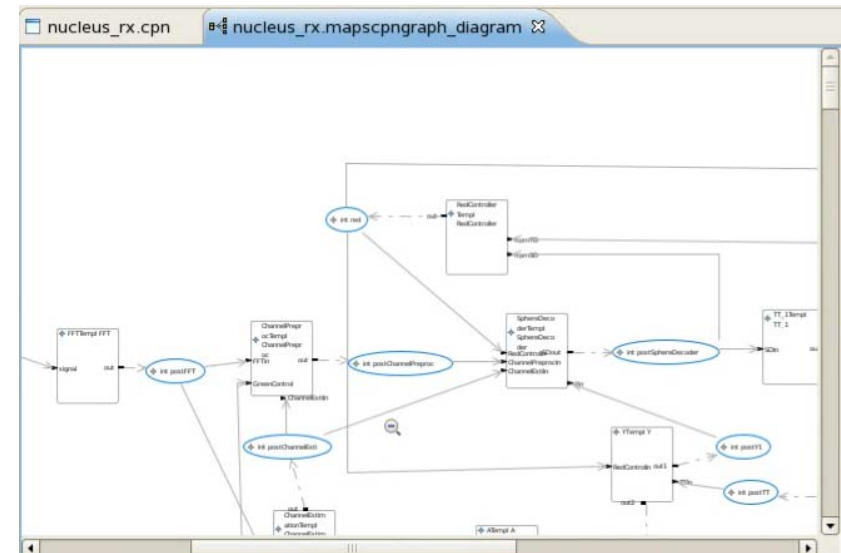
- **Waveform description: MAPS CPN Language**
  - General dataflow model: Kahn Process Networks (KPN)
  - Small extension to C (few new keywords)



```

__PNkpn myproc __PNin(int A)
__PNout(char B[3][3])
{
    int i, j;
    __PNout(B)
    for (i = 0; i < 3; i++)
        for (j = 0; j < 3; j++)
            B[i][j] = 5;
    while (1)
        __PNin(A) __PNout(B)
        for (i = 0; i < 3; i++)
            for (j = 0; j < 3; j++)
                B[i][j] = (char)A;
}
...
    
```

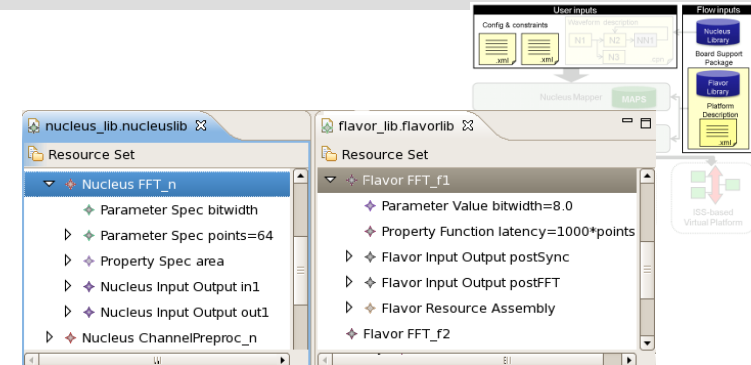
.cpn



.vcpn

## ■ Libraries for Nuclei and Flavors

- XML representation
- Efforts in finding common parameters and properties



## ■ Constraints:

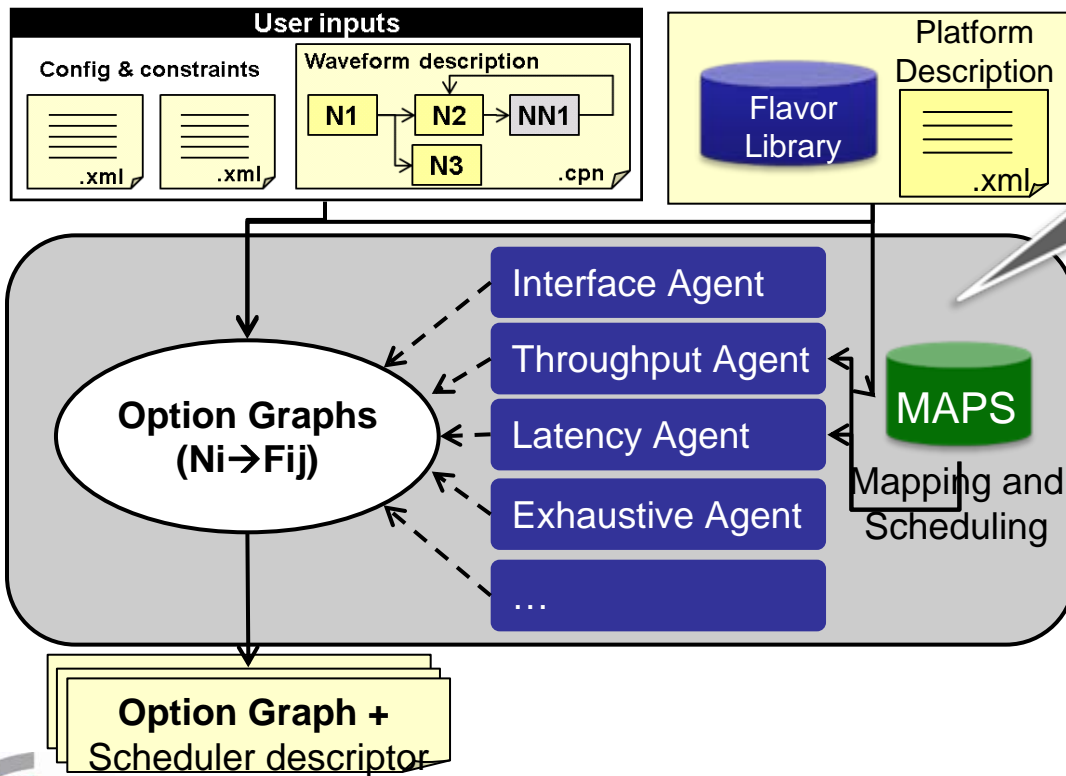
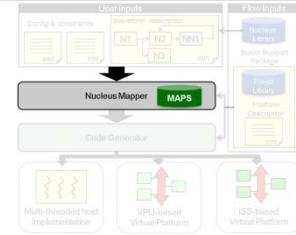
- XML representation
- Supported: latency along paths, throughput at different locations, multi-tasking

## ■ Configuration:

- XML representation
- Allow to configure several parameters of the tool



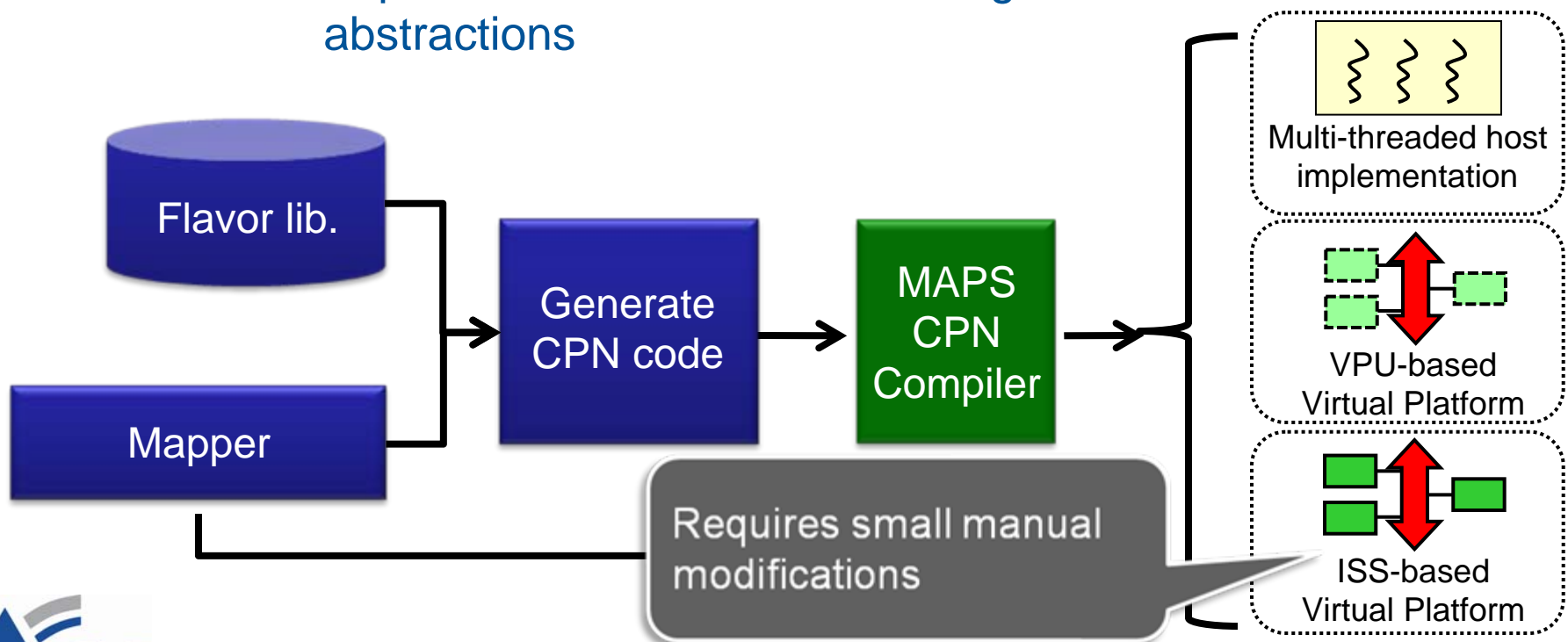
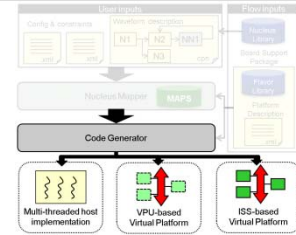
- **Mapping: Finding a flavor for every nucleus**
  - Architecture: list of Option-Graphs that is refined by running Agents
  - Export feasible options for further analysis



Provides timing analysis, mapping and scheduling

```
// an ordered list of smart-agents
Mapper.regPass(interface);
Mapper.regPass(critical_path);
Mapper.regPass(throughput);
...
```

- **Option graphs include:**
  - A mapping of nuclei to flavors
  - A scheduler descriptor
- **Code-gen and simulation**
  - Pick the right implementation from the Flavor library
  - Export simulation with scheduling information: different abstractions



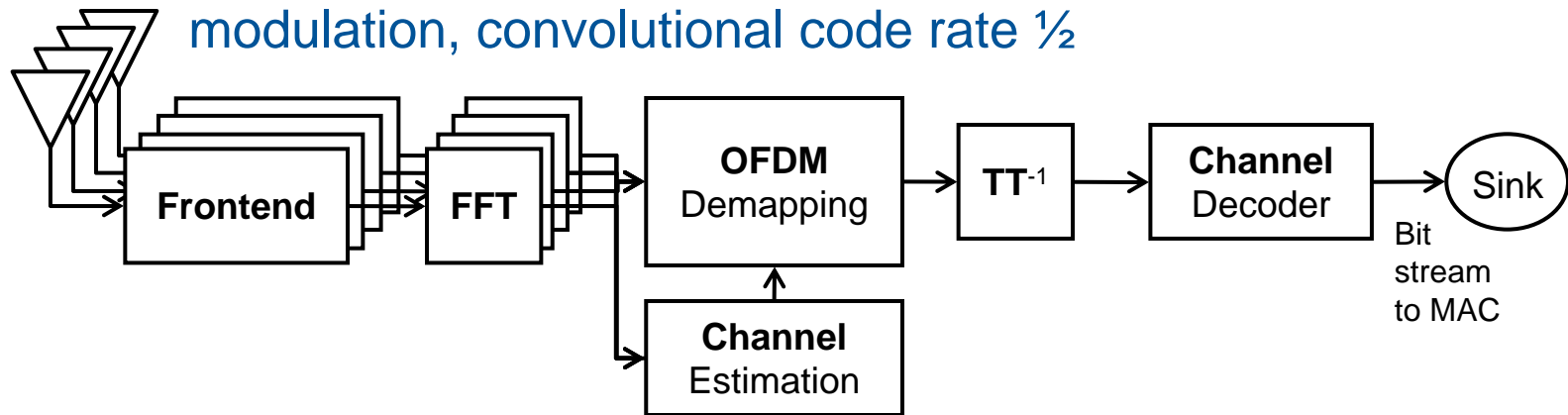
- **Motivation**
- **Tool Flow:**
  - Overview
  - Inputs
  - Waveform mapping
  - Code generation

## **Case study**

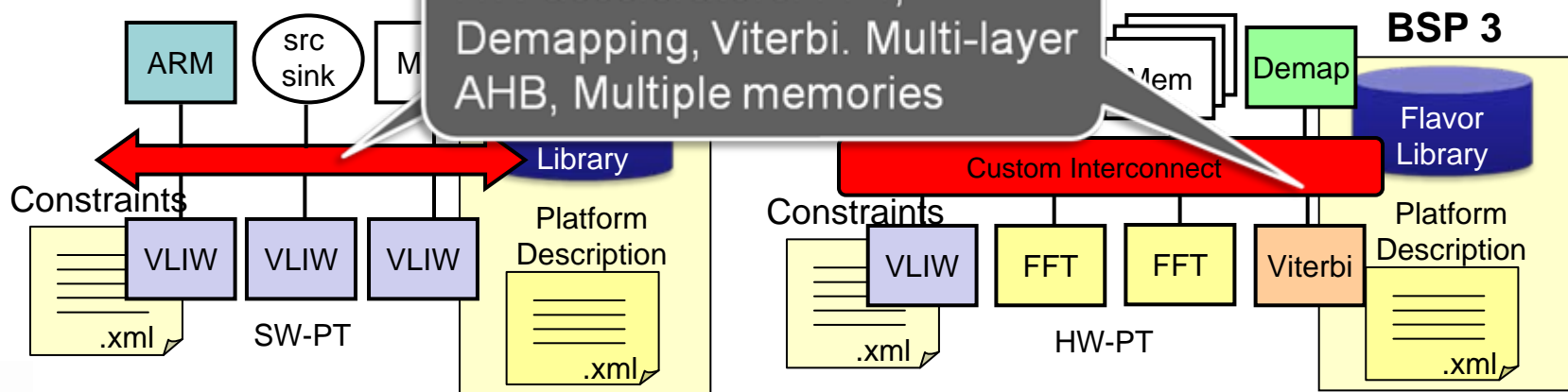
- **Summary and Outlook**

## ■ Application: MIMO OFDM Receiver

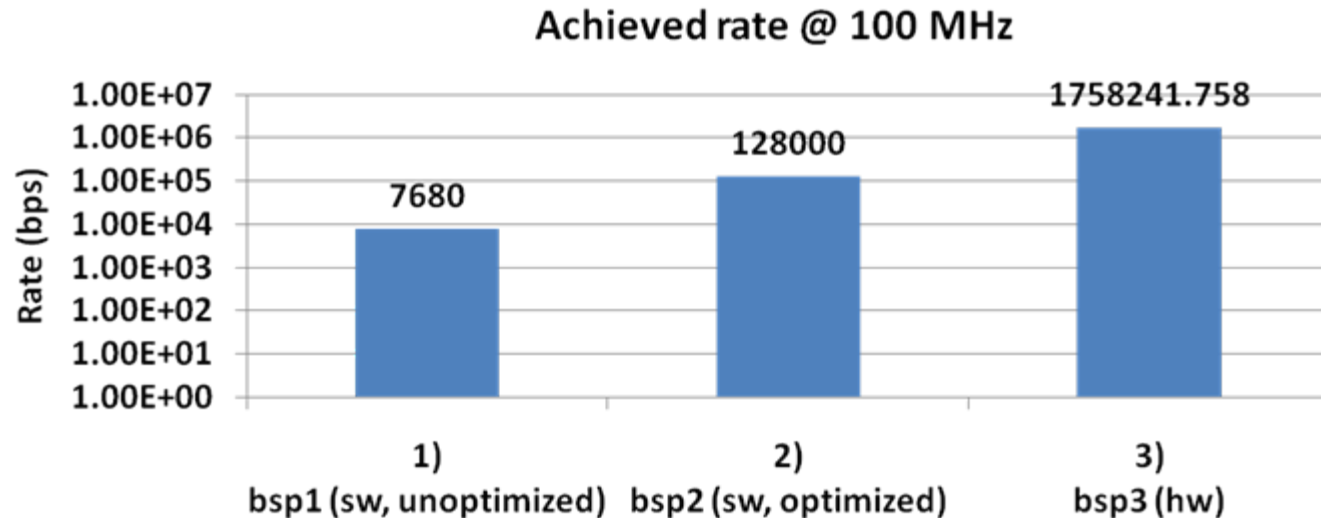
- 4 Antennas, 64 sub-carriers (48 with payload), QPSK modulation, convolutional code rate  $\frac{1}{2}$



## ■ Two target virtual platforms: Portability and efficiency analysis

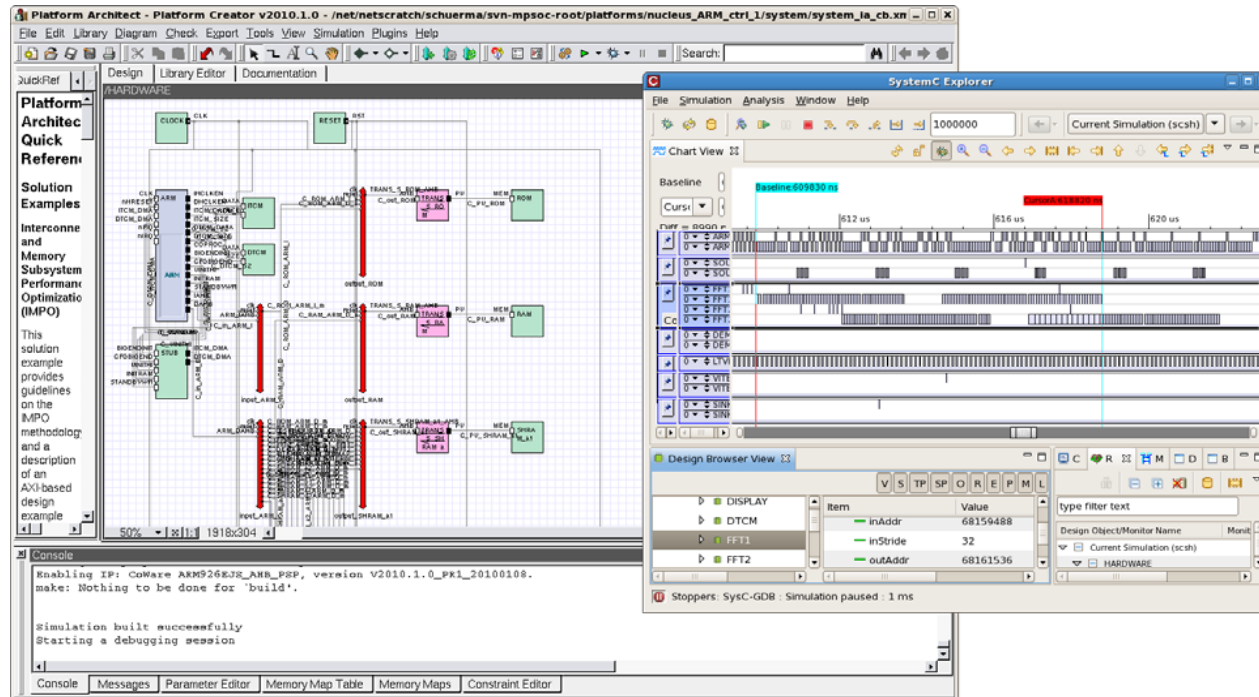


- **Best solutions found by the tool for each BSP with the same application specification (w/o simulation)**



- **Tool performance:**
  - Option-graphs analysis: from ~600 to 3 options in ~15s (on a dual core AMD Opteron, 2.6 GHz.)

## ■ Exporting simulations:



- Actual rate on cycle accurate platform 16 % slower than predicted

- **Motivation**
- **Tool Flow:**
  - Overview
  - Inputs
  - Waveform mapping
  - Code generation
- **Case study**

## **Summary and Outlook**

## ■ Summary:

- A flow from KPN waveform description to implementation
- Portability: high level description based on components (*nuclei*)
- Efficiency: tool selects efficient implementations (*flavors*) from BSP
- Integrated with state-of-the-art ESL tools

## ■ Outlook:

- Characterization of new nuclei
- Implementation and characterization of new flavors
- Analysis of iterative decoding





# Thanks!

## Questions?

**ACK:** This work has been supported by the UMIC (Ultra High-Speed Mobile Information and Communication) research centre. [www.umic.rwth-aachen.de](http://www.umic.rwth-aachen.de)

Contact: [Jeronimo.Castrillon@iss.rwth-aachen.de](mailto:Jeronimo.Castrillon@iss.rwth-aachen.de)

