Compiling Parametric Data Flow for Dedicated SoCs

Journées compilation

Mickaël Dardaillon Kevin Marquet Tanguy Risset
Jérôme Martin Henri-Pierre Charles

05/12/13

Context : Telecom

4G LTE-Advanced

- Up to 1 Gbps
- Control/Data multiplexing

Context : Telecom

4G LTE-Advanced

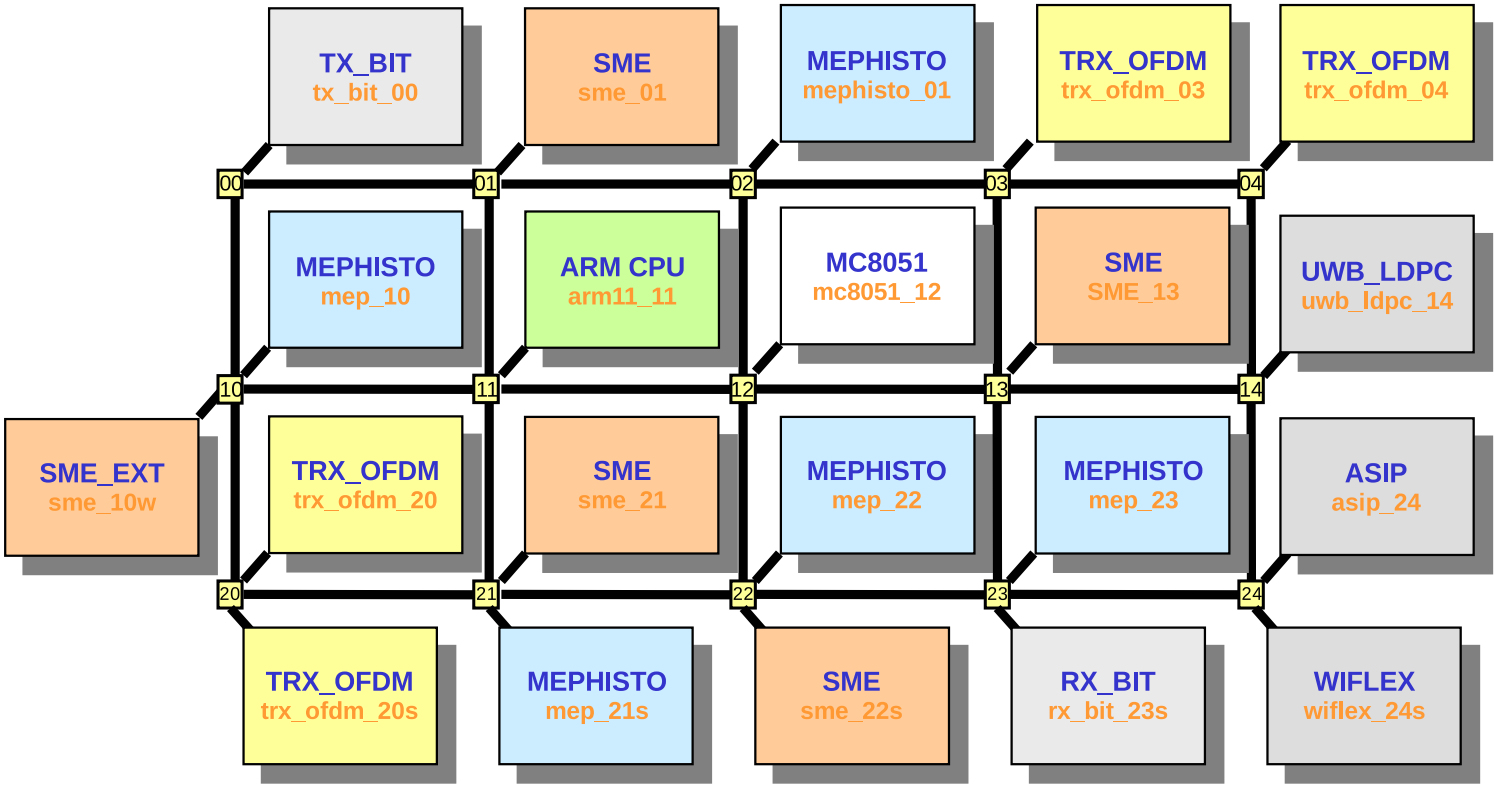
- Up to 1 Gbps
- Control/Data multiplexing

4G phone

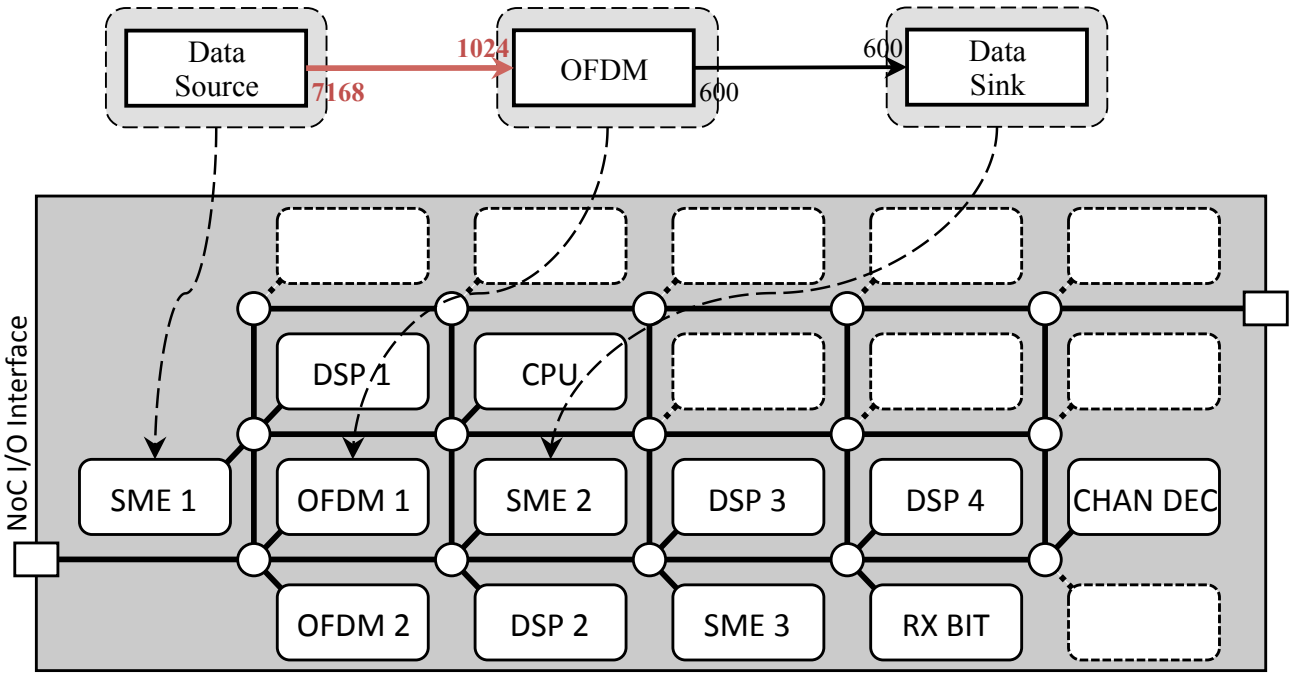
- Computing power $>$ 40 GOPS
- Latency $<$ 2 ms
- Consumption $<$ 500 mW

Platform : Magali, CEA LETI (2009)

- LTE-Advanced demonstrator
- Distributed memory
- Heterogeneous
- Network on Chip (NoC)
- Consumption $\approx$ 231mW
- Distributed assembly code



OFDM example



Output Data Source

```
config_id 0
  channel 1
  num_icc 0
  sel_credit 0
  path_to_target 2 SOUTH NORTH
  packet_size 8
  total_data_nb 7168
```

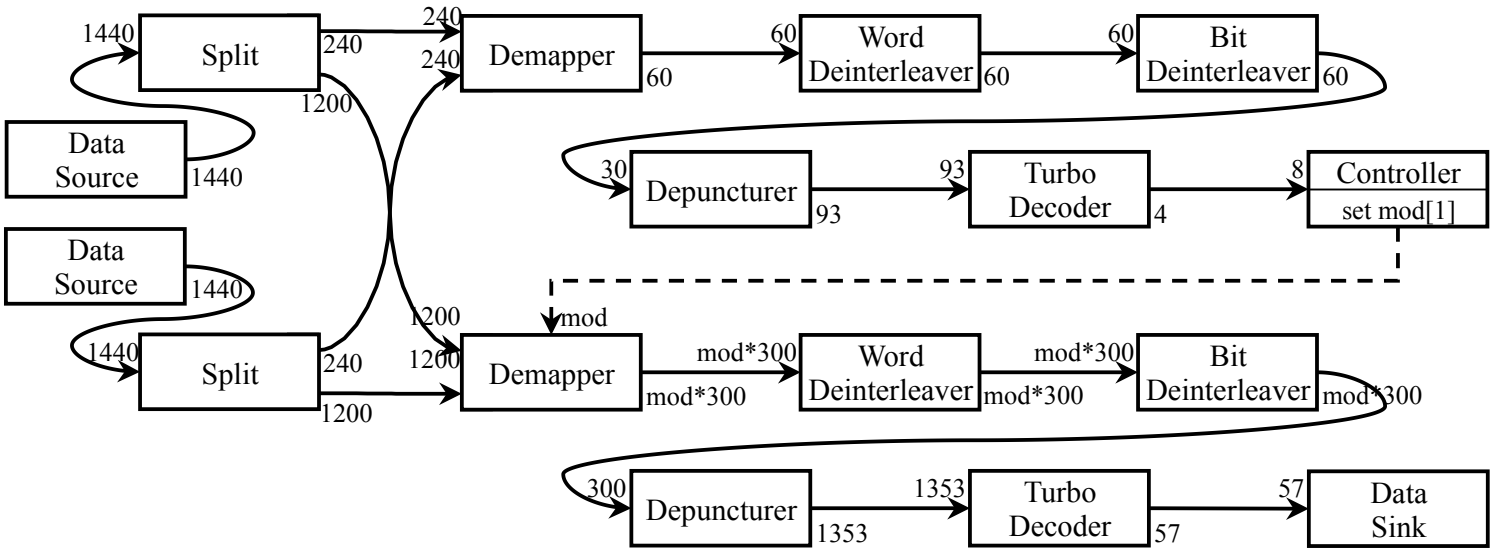
Input OFDM

```
config_id 0
  channel 1
  num_occ 2
  sel_credit 0
  path_to_target 2 NORTH SOUTH
  credit_size 8
  total_credit_nb 1024
```

Outline

Outline

LTE application

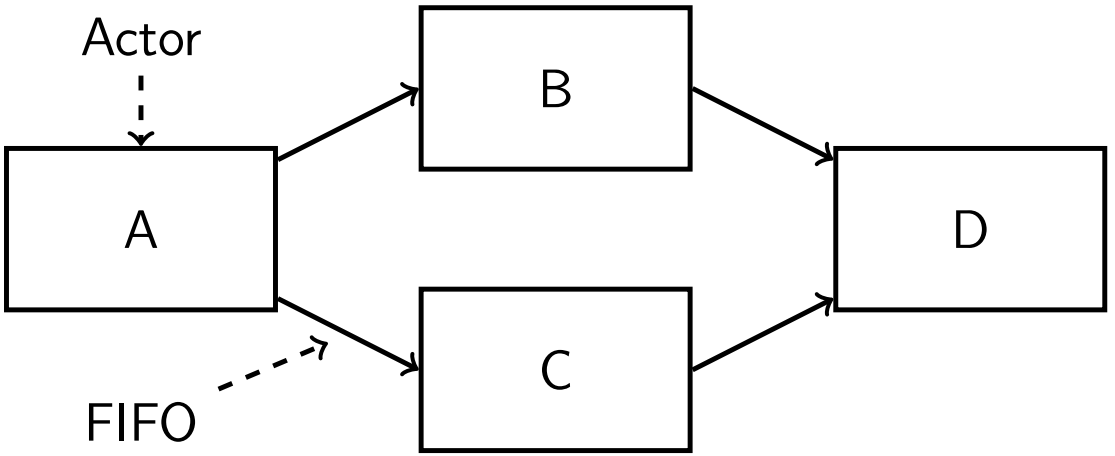


Requirements

- data driven
- reconfigurable

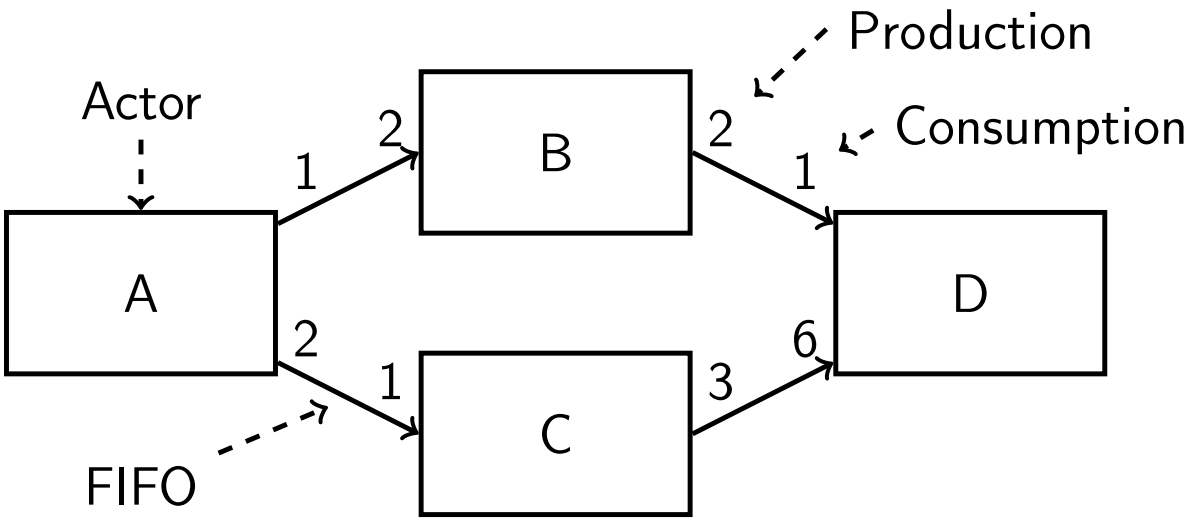
Model of computation

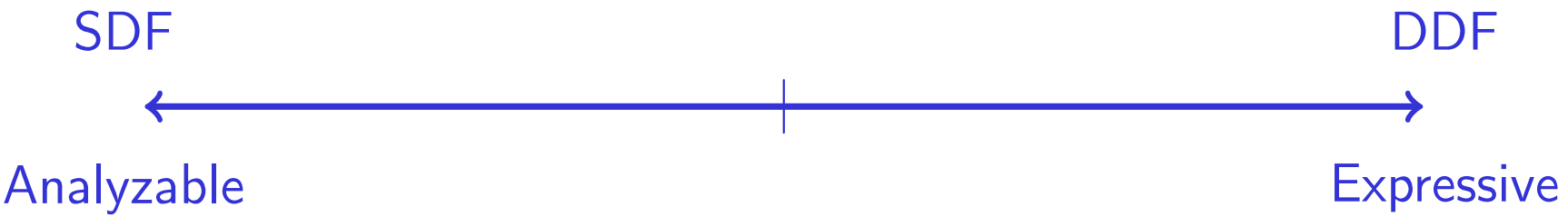
Data flow



Model of computation

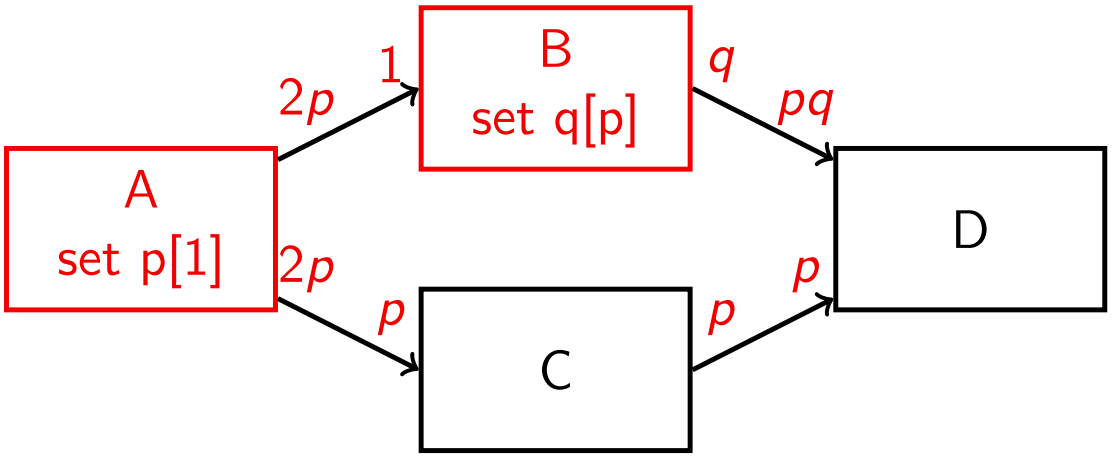
Data flow

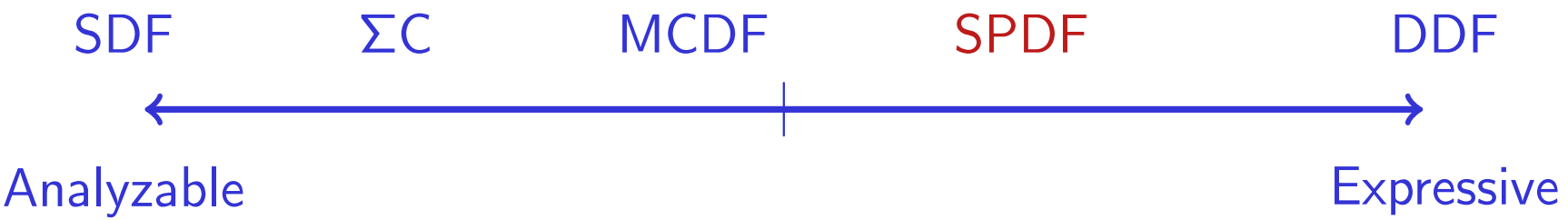




Model of computation

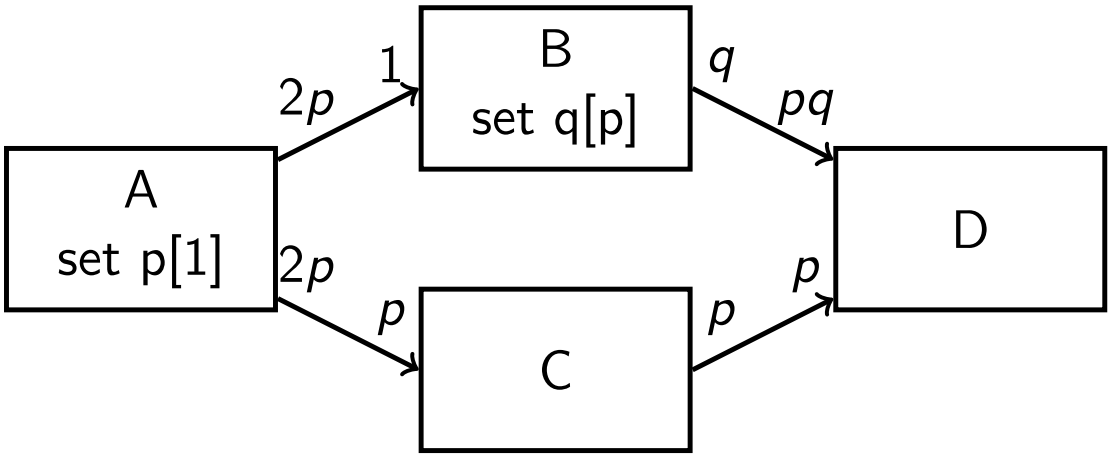
Data flow





Model of computation

Data flow



Quasi-static schedules

- A $(A; push\ p)$
- B $(pop\ p; (B^p; push\ q)^2)$
- C $(pop\ p; C^2)$
- D $(pop\ p; (pop\ q; D)^2)$

SPDF : Schedulable Parametric Data Flow, Inria (2012)

- Analyzable
- Quasi-static scheduling

A language for SPDF ?

High-level modeling

e.g. Simulink, LabView

C inspired

e.g. ΣC, StreamIt

A language for SPDF ?

High-level modeling

e.g. Simulink, LabView

C inspired

e.g. ΣC, StreamIt

C++ and API

e.g. SystemC

Same language for :

- Actor computation
- Graph construction

Based on existing tools

Actor example

```
class FFT : public Actor {
  PortIn<int> in;
  PortOut<int> out;
  ParamIn size;
  FFT(): in(size), out(size) {}
  void compute();
};
```

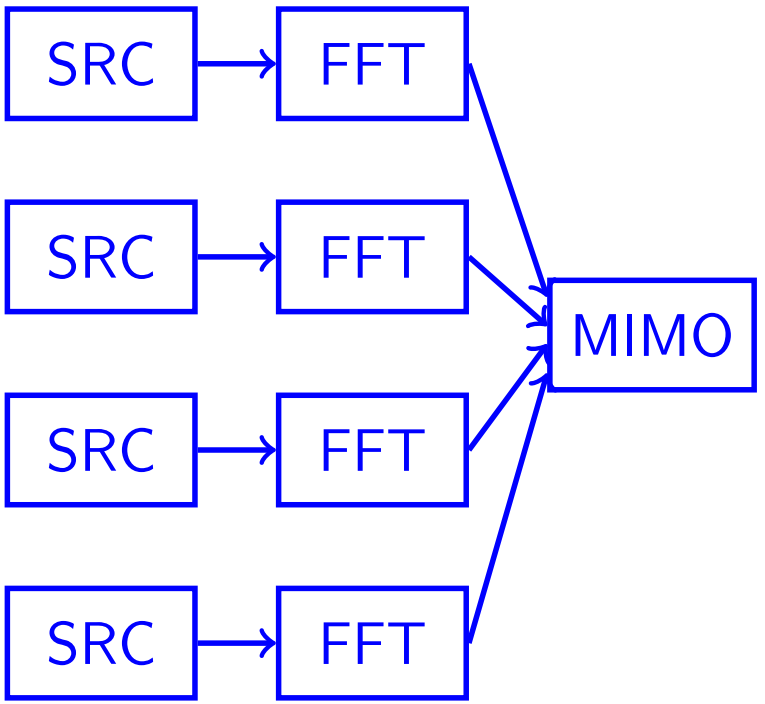
Actor example

```
class FFT : public Actor {
  PortIn<int> in;
  PortOut<int> out;
  ParamIn size;
  FFT(): in(size), out(size) {}
  void compute();
};
```

```
void FFT::compute() {
  [...]
    val1 = in.pop();
  [...]
    out.push(val2);
}
```

Graph example

```
SRC src[NB_ANT];
FFT fft[NB_ANT];
MIMO mimo(NB_ANT);
for(i = 0; i < NB_ANT; i++) {
   fft[i].in <= src[i].out;
   mimo.in[i] <= fft[i].out;
}
```

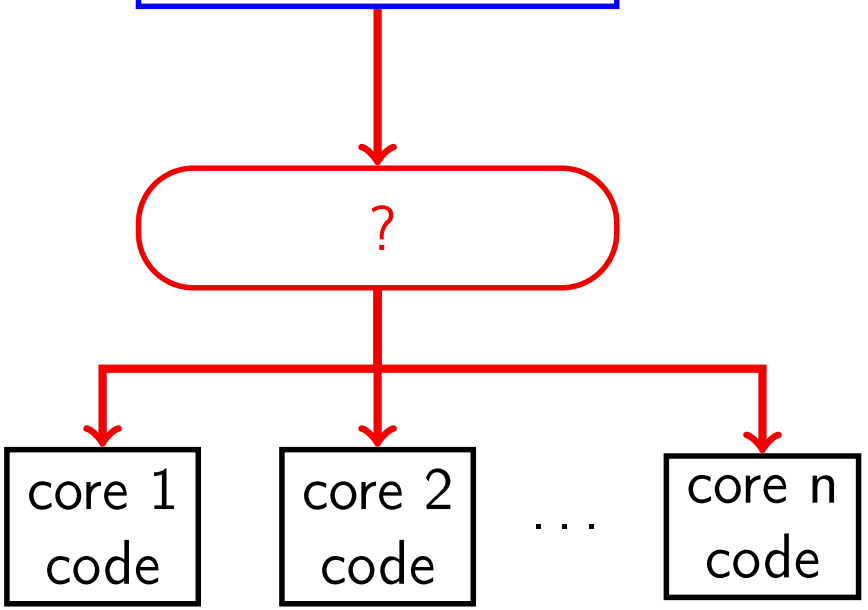


Outline

Compilation : what's needed ?

Host

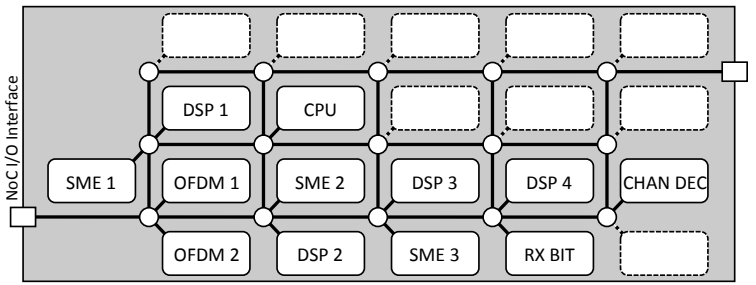
```
SRC src[NB_ANT];
FFT fft[NB_ANT];
MIMO mimo(NB_ANT);
for(i = 0; i < NB_ANT; i++) {
  fft[i].in <= src[i].out;
  mimo.in[i] <= fft[i].out;
}
```



Target

Constraints

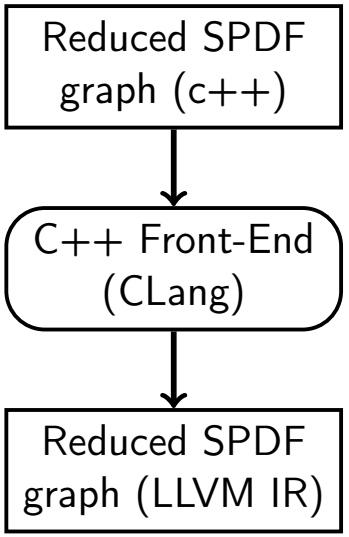
- C++ graph construction
- Use existing Front-End
- LLVM bytecode
- platform independent DSP API



Compilation flow

Ascii file representation

```
SRC src[NB_ANT];
FFT fft[NB_ANT];
MIMO mimo(NB_ANT);
for(i = 0; i < NB_ANT; i++) {
  fft[i].in <= src[i].out;
  mimo.in[i] <= fft[i].out;
}
```

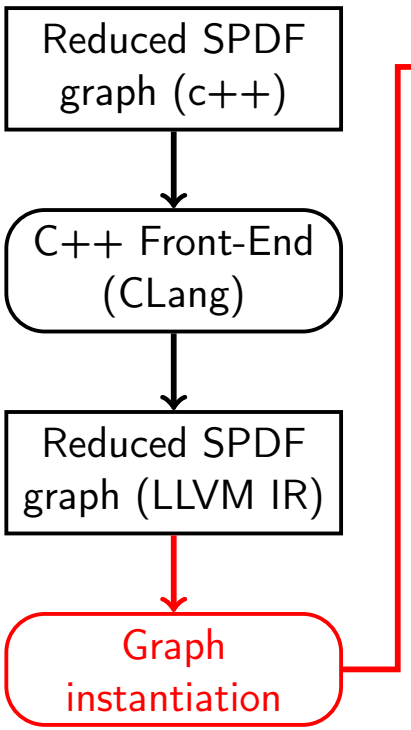


Binary memory representation

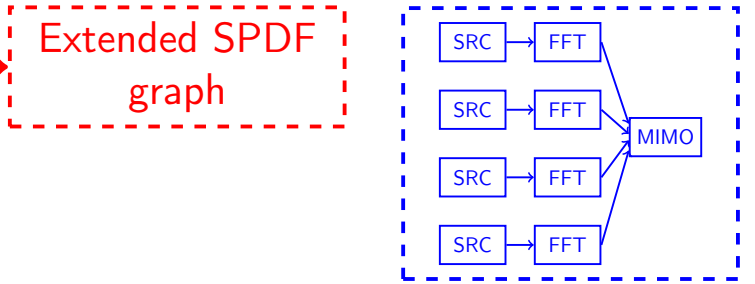
Compilation flow

Ascii file representation

```
SRC src[NB_ANT];
FFT fft[NB_ANT];
MIMO mimo(NB_ANT);
for(i = 0; i < NB_ANT; i++) {
  fft[i].in <= src[i].out;
  mimo.in[i] <= fft[i].out;
}
```



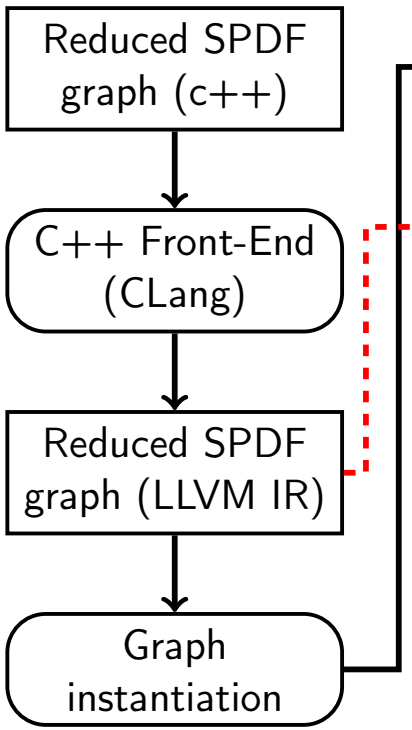
Binary memory representation



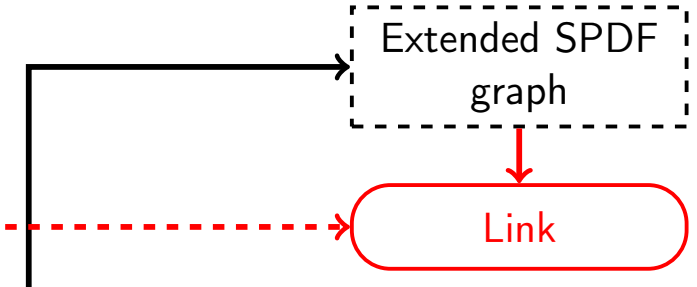
Compilation flow

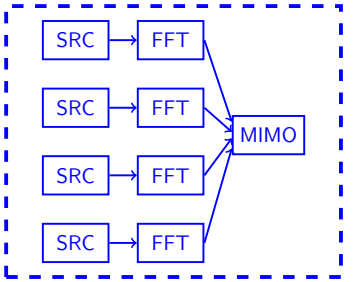
Ascii file representation

```
SRC src[NB_ANT];
FFT fft[NB_ANT];
MIMO mimo(NB_ANT);
for(i = 0; i < NB_ANT; i++) {
  fft[i].in <= src[i].out;
  mimo.in[i] <= fft[i].out;
}
```



Binary memory representation

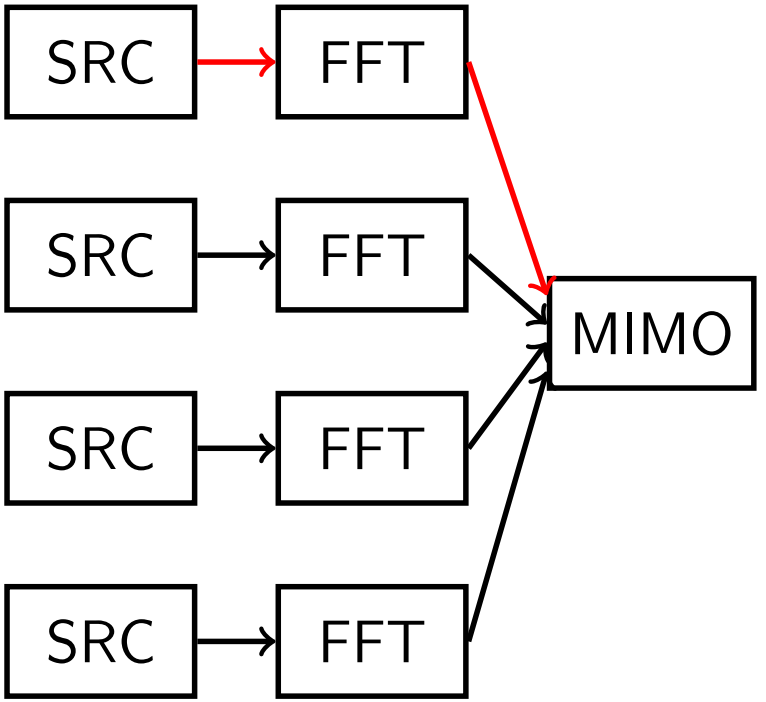




Link

Compute method

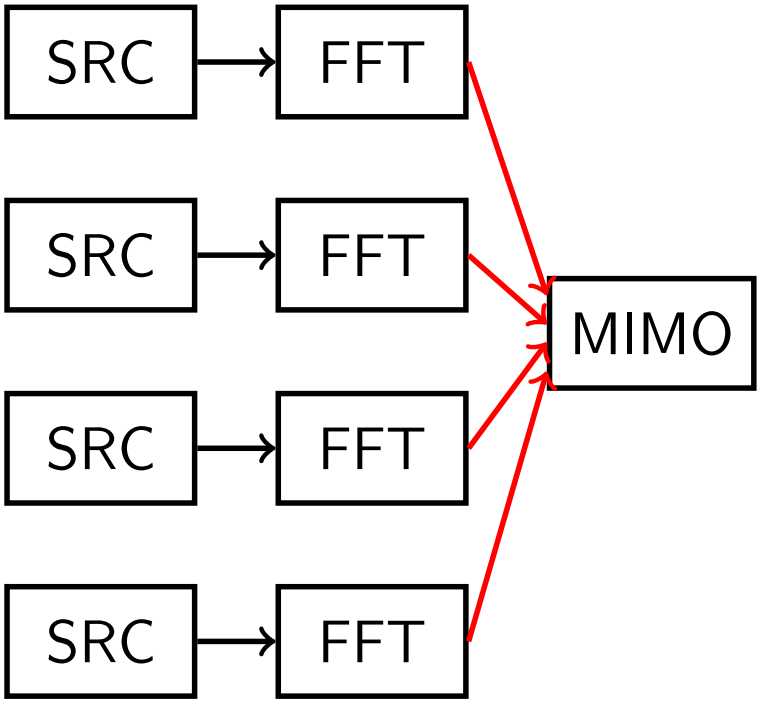
```
void FFT::compute() {
   [...]
     val1 = in.pop();
   [...]
     out.push(val2);
}
```



Link

Compute method

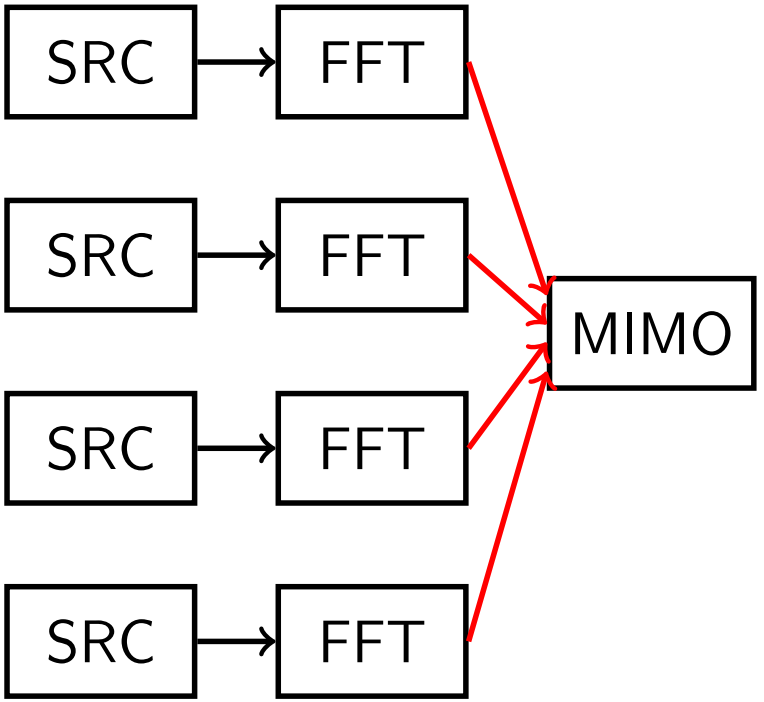
```
void MIMO::compute() {
   [...]
   for(i=0; i<nb_ant; i++) {
      val[i] = in[tab[i]].pop();
   [...]
}
```



Link

Compute method

```
void MIMO::compute() {
   [...]
   for(i=0; i<nb_ant; i++) {
      val[i] = in[tab[i]].pop();
   [...]
}
```



IR LLVM

```
define void @_ZN4MIMO7computeEv(%class.MIMO* %this)
  %1 = gep %class.MIMO* %this, i32 0, i32 1
  %2 = gep %class.MIMO* %this, i32 0, i32 2
  %3 = call i32 @_ZN6PortIn13popE(%class.PortIn* %1)
```

Link

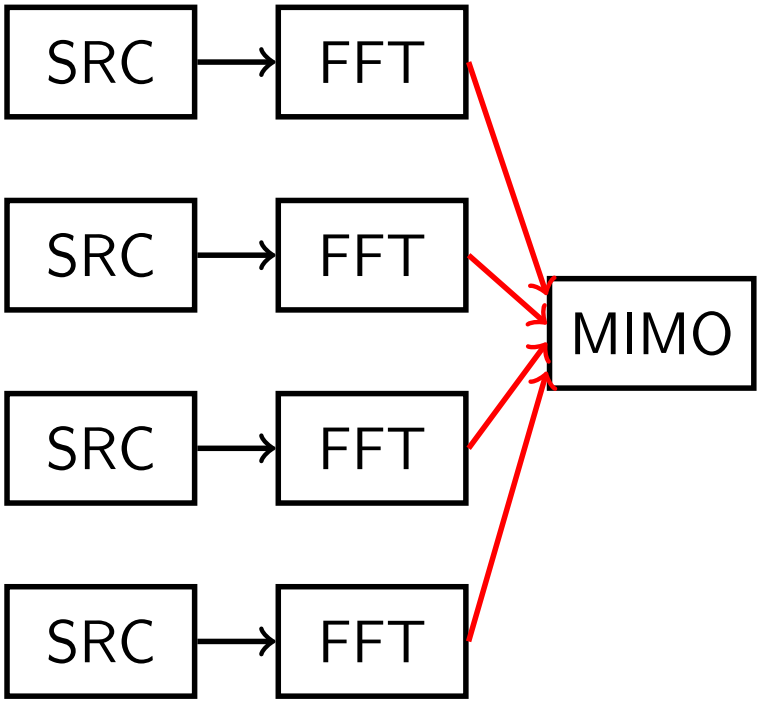
Compute method

```
void MIMO::compute() {
    [...]
    for(i=0; i<nb_ant; i++) {
        val[i] = in[tab[i]].pop();
    [...]
}
```

Port address computation

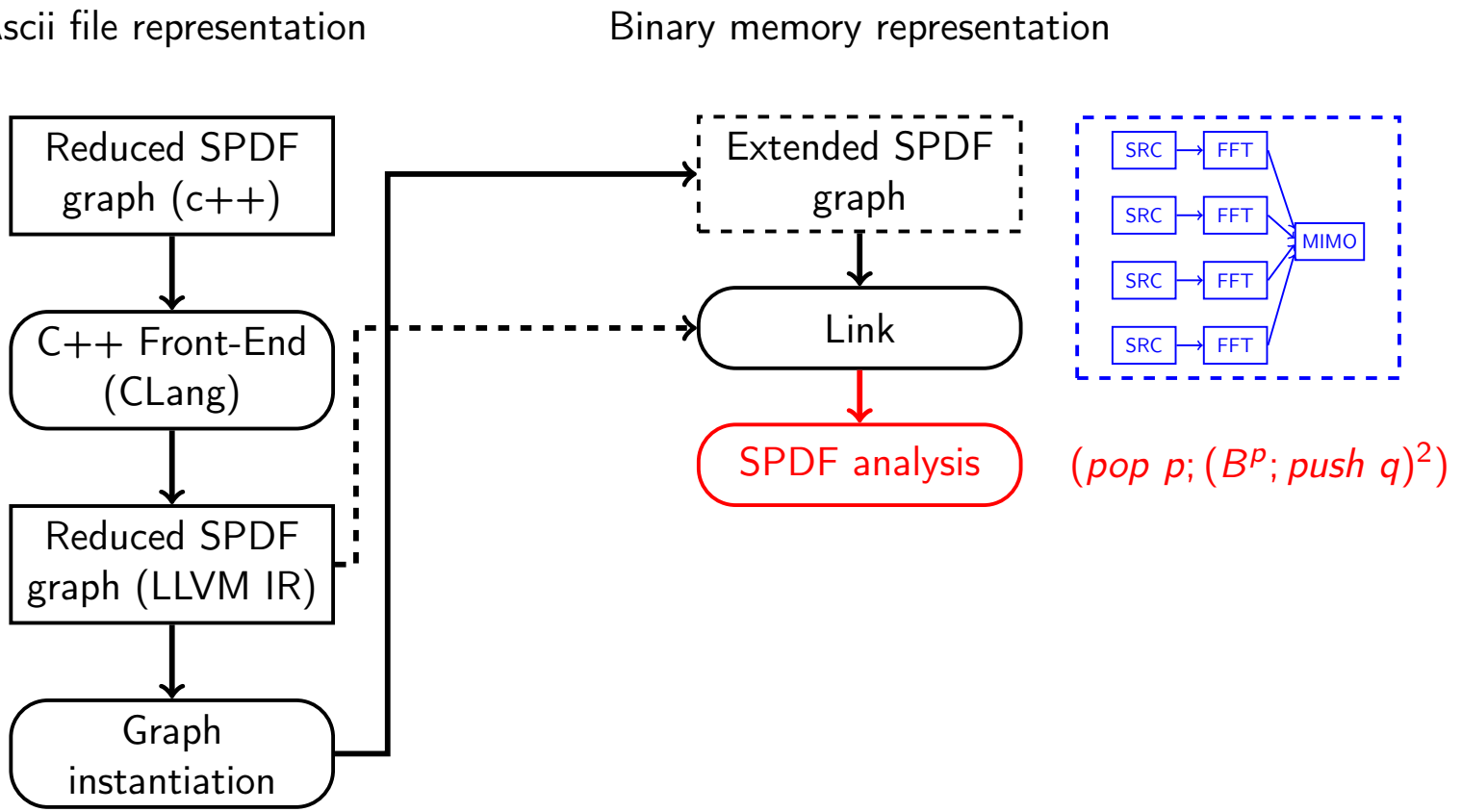
From PinaVM, Verimag (2010)

- *inlining*
- *slicing*



Compilation flow

```
SRC src[NB_ANT];
FFT fft[NB_ANT];
MIMO mimo(NB_ANT);
for(i = 0; i < NB_ANT; i++) {
  fft[i].in <= src[i].out;
  mimo.in[i] <= fft[i].out;
}
```

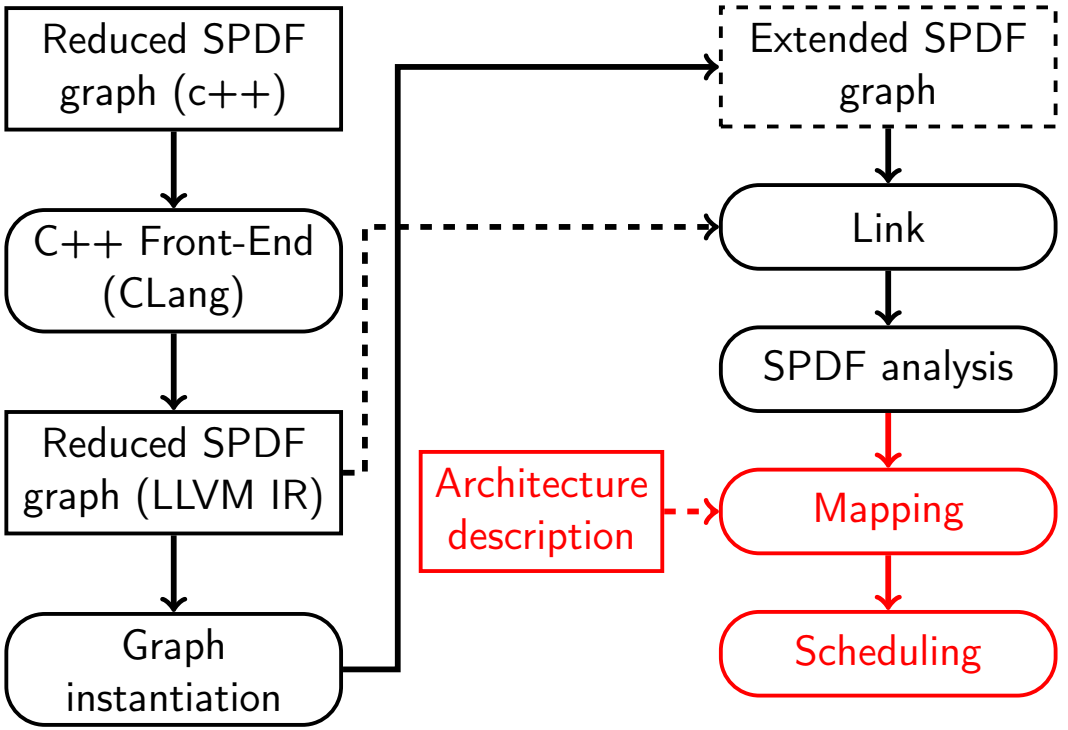


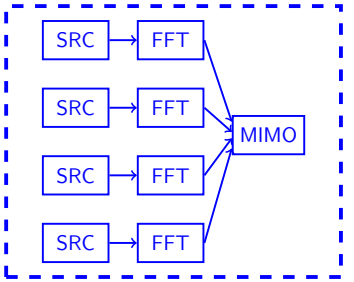
Compilation flow

Ascii file representation

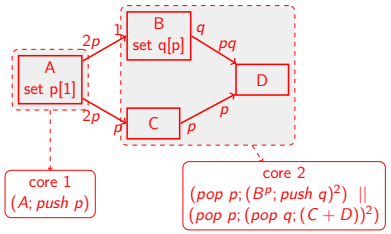
Binary memory representation

```
SRC src[NB_ANT];
FFT fft[NB_ANT];
MIMO mimo(NB_ANT);
for(i = 0; i < NB_ANT; i++) {
  fft[i].in <= src[i].out;
  mimo.in[i] <= fft[i].out;
}
```

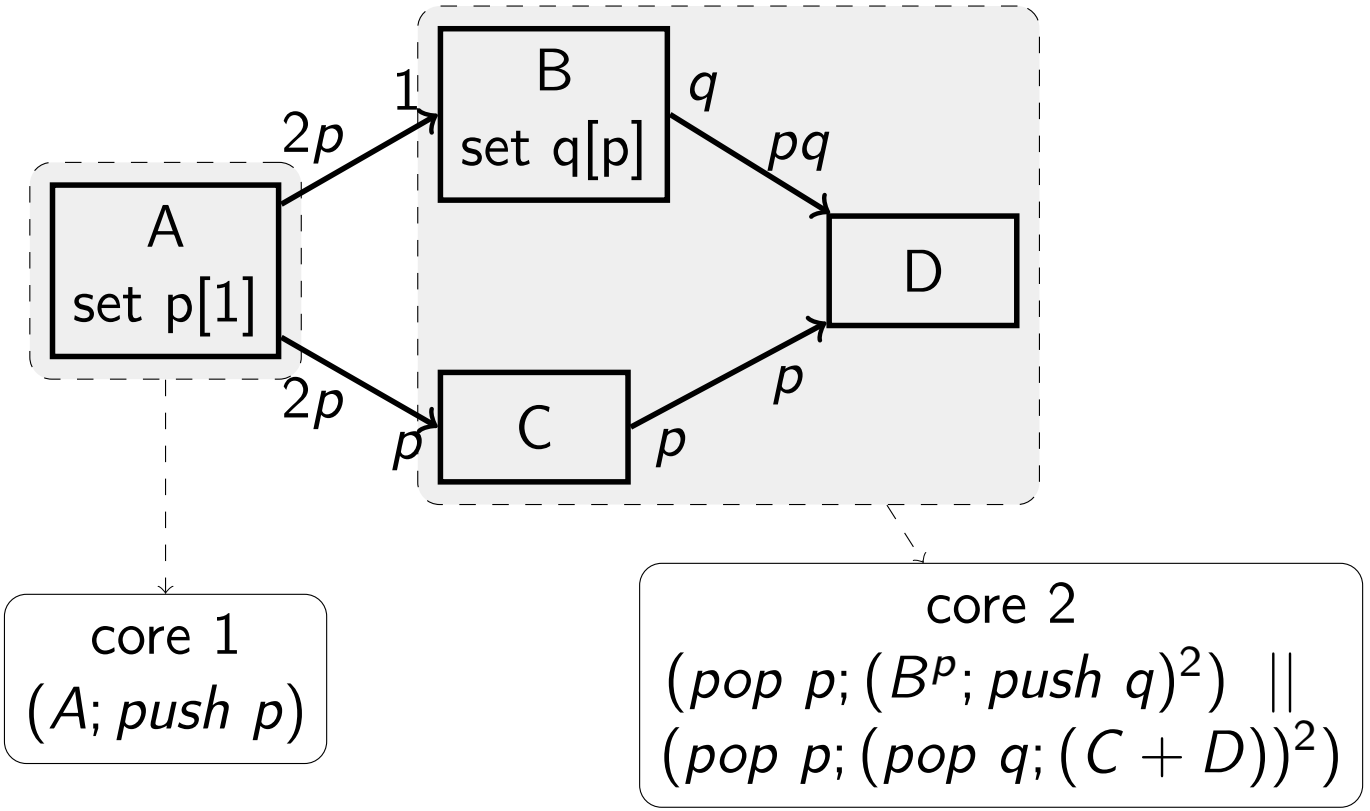




$(pop\ p; (B^p; push\ q)^2)$



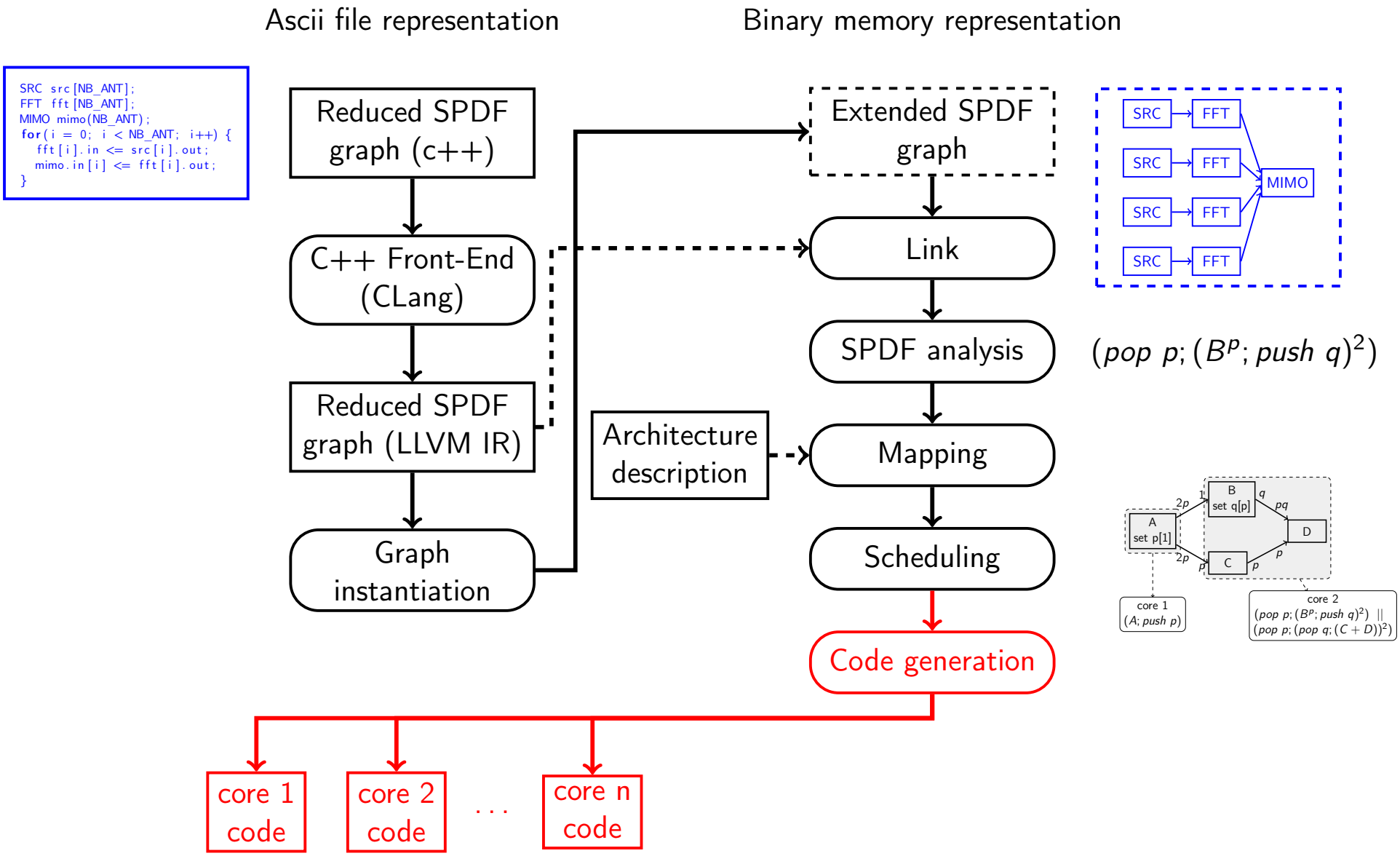
Mapping



Fusion

- inter-actor optimization
- n actors $\equiv$ 1 core
- solved on SDF (StreamIt)
- extended to SPDF

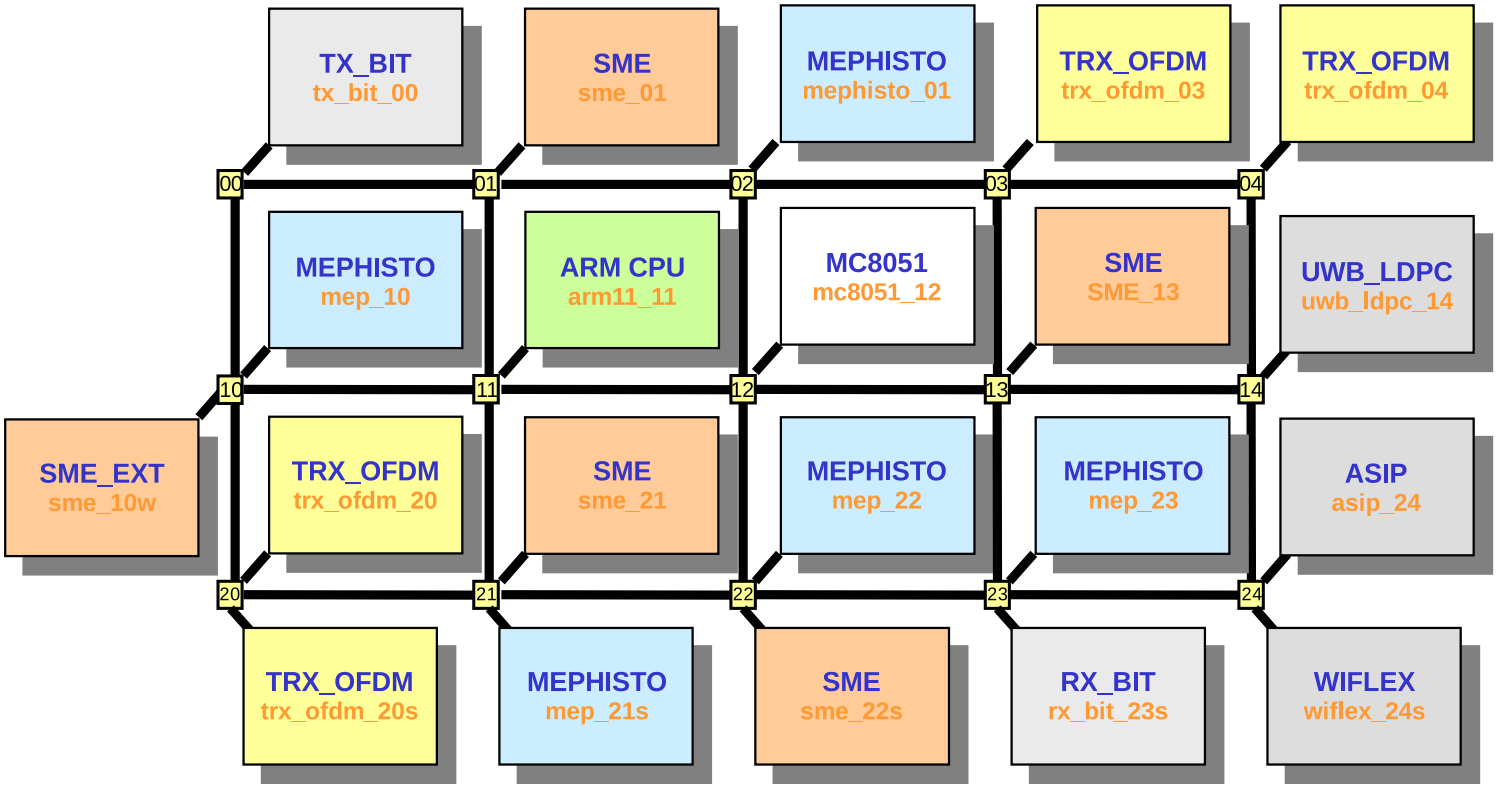
Compilation flow



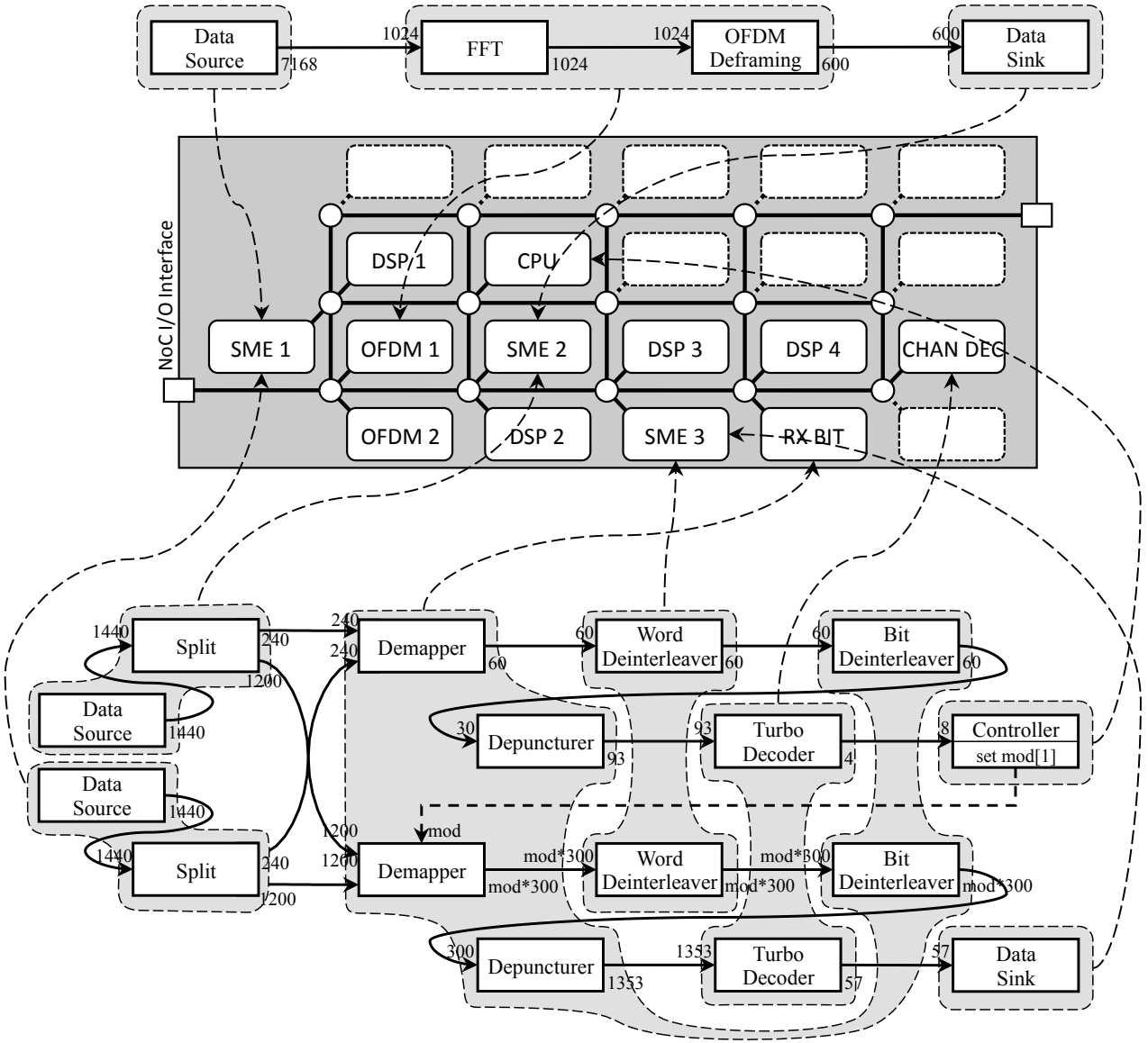
Outline

Platform : Magali, CEA LETI (2009)

- ✓ Core configuration
- ✓ Communications
- ✓ Distributed controllers
- ✓ Central controller



Applications



Results

Application	Hand-written code #lines (C - ASM) / time	PWF code #lines / time
FFT	150 l - 200 l / 1 week	60 l / 1 h
demodulation	300 l - 600 l / 1 month	160 l / 4 h
parametric demod.	500 l - 800 l / 3 months	260 l / 8 h

Results

Application	Hand-written code #lines (C - ASM) / time	PWF code #lines / time
FFT	150 l - 200 l / 1 week	60 l / 1 h
demodulation	300 l - 600 l / 1 month	160 l / 4 h
parametric demod.	500 l - 800 l / 3 months	260 l / 8 h

Application	Hand-written	Generated	RVM
FFT	149 μs	168 μs (+13%)	500 μs (+236%)
demodulation	180 μs	283 μs (+57%)	-
parametric demod.	288 μs	558 μs (+94%)	-

Conclusion

Compilation flow

- C++ and API
- Parametric Data Flow
- Graph instantiation/Link
- Actor Fusion

Magali platform

- Core configuration
- Communications
- Distributed controllers
- Central controller

Future works

- Central controller optimization
- New platforms support
- Scheduling

Language : Actor

```
class actSource : public Actor {
  public:
    PortOut<int> lout;
    ParamIn size;
    void compute() {
      int* data = readFile("ofdm_demod_ant1", size.
          get());
      lout.push(data, size.get());
    }
    actSource()
      :Actor("actSource", gen_00w), size("size"),
          lout("out",size) {}
};
```

Language : Graph

```
int main (void) {
  actSize size;
  actCtrl ctrl;
  actSource src;
  actFFT fft;
  actSink snk;

  ctrl.lin <= size.lout;
  fft.lin <= src.lout;
  snk.lin <= fft.lout;

  src.size <= ctrl.param*SIZE_FFT;
  fft.size <= SIZE_FFT;
  snk.size <= SIZE_FFT;
```