

DSP Code Generation with Optimized Data Word-Length Selection

D. MENARD, O. SENTIEYS

R2D2 Team

IRISA / INRIA

ENSSAT / University of Rennes 1

name@irisa.fr



Outline of the presentation

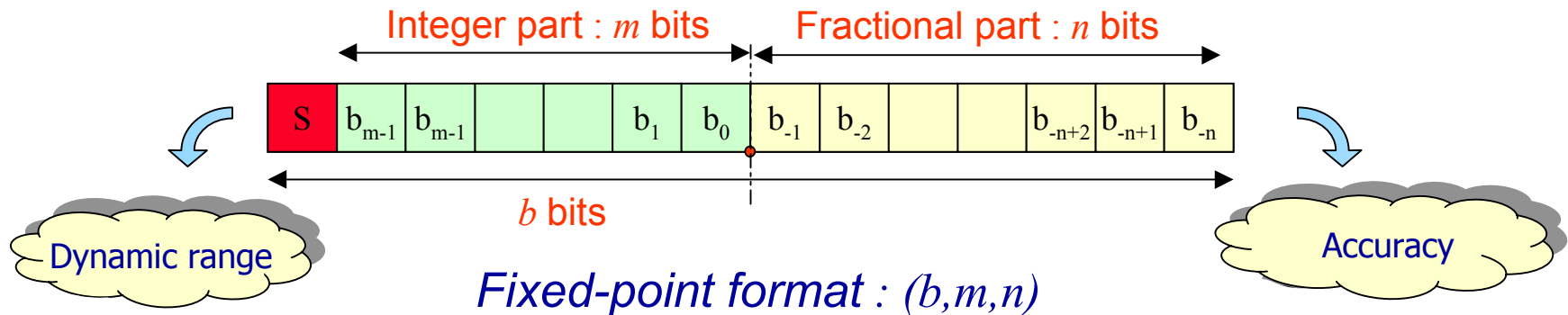
- **Motivations**
- **Floating-point to Fixed-point Conversion**
- **Data Word-Length Selection**
- **Experiments and Results**
- **Conclusion**

Motivations

- **Embedded digital signal processing systems**
 - **Specification with floating-point data types**
 - **Implementation in fixed-point architectures**
 - **Methodologies for the automatic conversion of a floating-point description into a fixed-point specification**
 - **Goals for Digital Signal Processors (DSP)**
 - ⇒ *Maximize the accuracy*
 - ⇒ *Minimize the size and the code execution time*
- ↪ **New methodology for the floating-point to fixed-point conversion**

Motivations

- **Fixed-point data specification**



- **Goals and constraints of fixed-point coding**
 - **Obtain a valid fixed-point specification**
 - ⇒ Avoid overflows
 - ⇒ Respect fixed-point arithmetic rules
 - **Maximize the accuracy**

Motivations

- **Existing methodologies :**
 - **FRIDGE [2], CoCentric Fixed Point Designer** (Synopsys)
 - **Autoscaler for C (University of Seoul) [1]**
 - ⇒ Transformation of floating-point C code into Fixed-point C code
- **Goals of our methodology**
 - **Implementation of floating-point algorithms in fixed-point DSP under accuracy constraint**
 - ⇒ Code execution time minimized under accuracy constraint
 - ⇒ The DSP architecture is taken into account for fixed-point data coding
 - **Data types supported by the DSP**

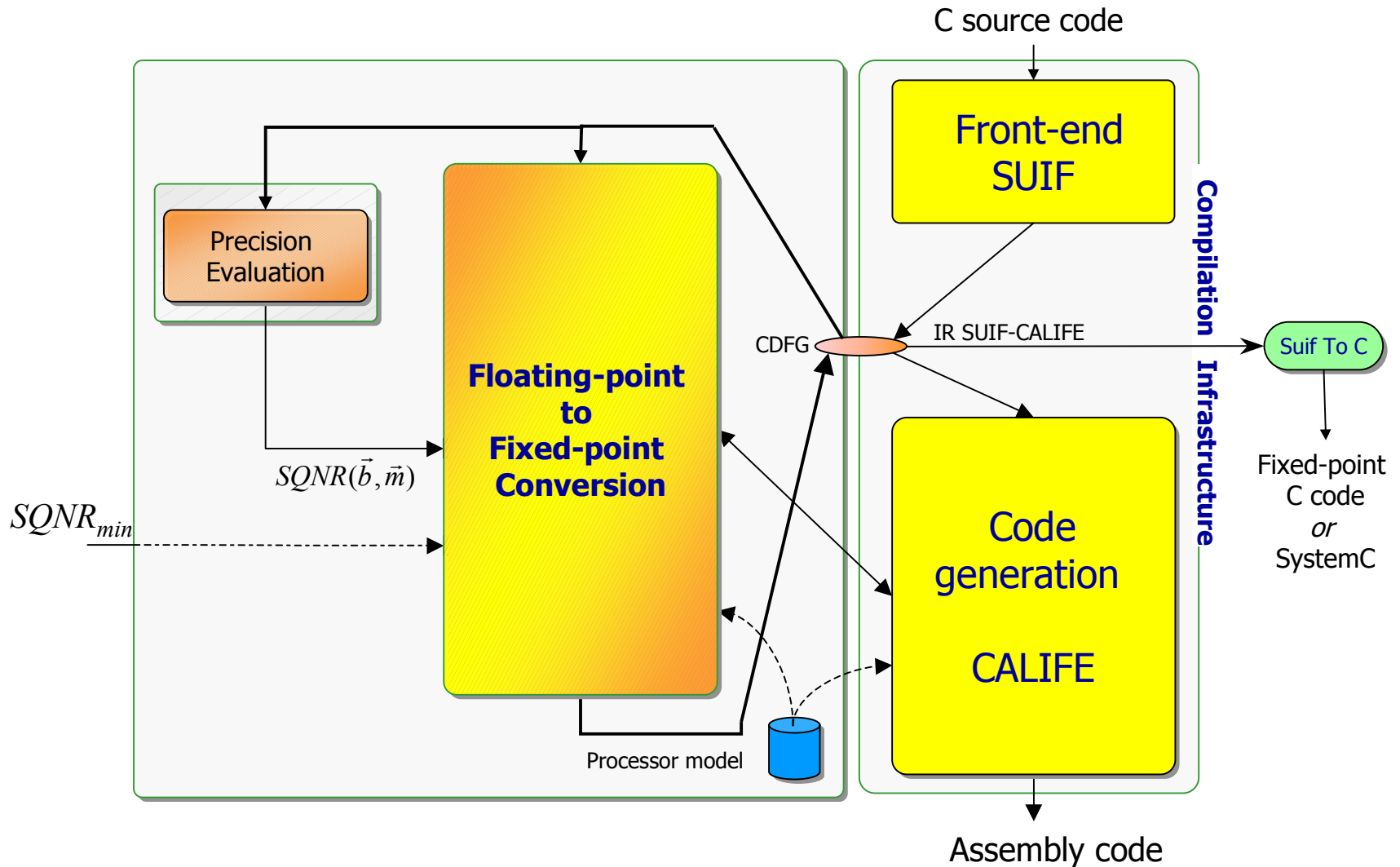
[1] K. Kum, J.Y. Kang and W.Y. Sung, *AUTOSCALER for C: An optimizing floating-point to integer C program converter for fixed-point DSP*, **IEEE Transactions on Circuits and Systems II**, pp 840-848, September 2000

[2] M. Coors, H. Keding, O. Luthje and H. Meyr, *Integer Code Generation For the TI TMS320C62x*, **ICASSP-01**, May 2001, Sate Lake City, US

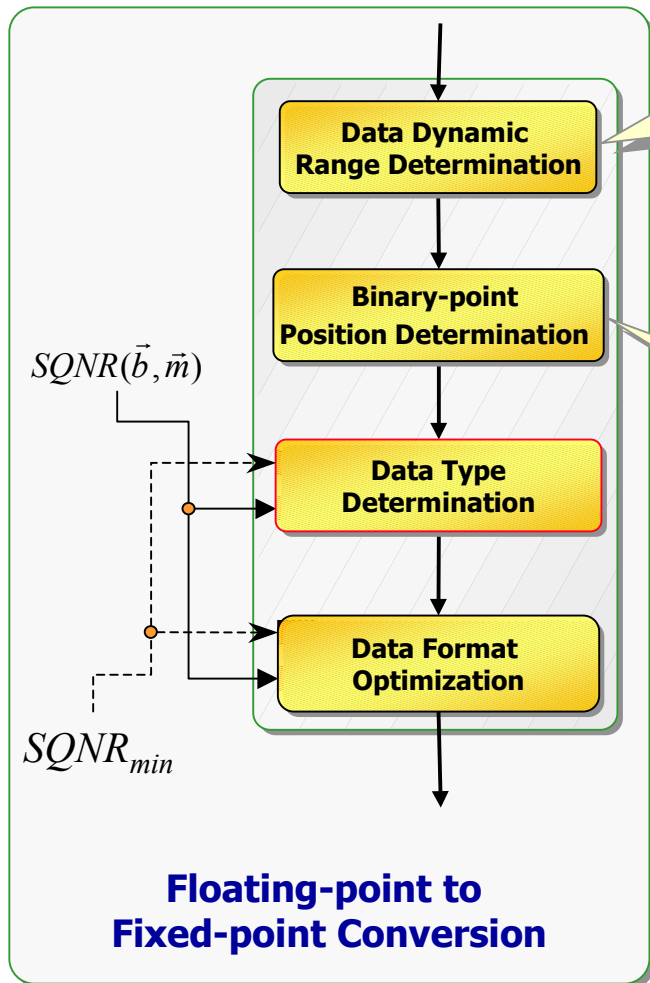
Part 2

Methodology for the Floating-point to Fixed-point Conversion

Methodology structure



Floating-to-Fixed Point Conversion



- **Goal**

- Determination of the data dynamic range

- **Method**

- Technique based on an analytical approach : interval arithmetic, L1 norm

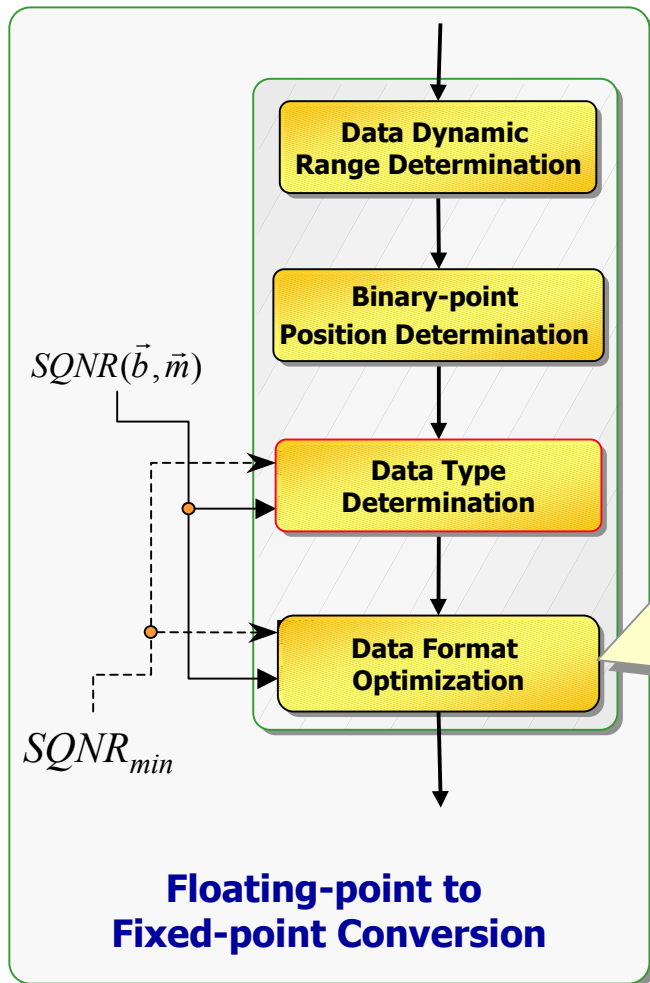
- **Goals**

- Determination of the data binary-point position
- Insertion of the scaling operations

- **Method**

- Propagation of the binary-point position through the Data Flow Graph
 - A rule is defined for each kind of operator

Floating-to-Fixed Point Conversion



• Goals

- Optimization of the scaling operation location
 - The scaling operations are moved to reduce the code execution time

⇒ Minimize the code execution time as long as the accuracy constraint is fulfilled

• Method

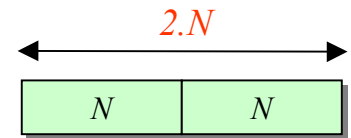
- DSP without Instruction Level Parallelism (ILP) (conventional DSP)
 - The optimization is achieved before the code generation
- DSP with ILP (VLIW)
 - The optimization is achieved during the scheduling stage

Part 3

Data Word-Length Selection

Data word-length in DSP

- **DSP native data word-length : N**
 - N is equal to 16 bits for most of the DSP
 - N can be customized for ASIP or some DSP cores
- **Multi-precision instructions**
 - The data are stored in memory with a greater precision
 - The word-length of a multi-precision data is a multiple of the DSP natural word-length (N)
 - ⇒ **Increase the computation accuracy**
 - ⇒ **Increase the code execution time**



Double precision operations	Classical operations	
	Addition	Multiplication
Addition	2	
Multiplication	3	4

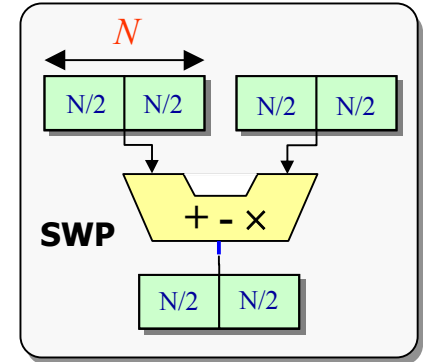
Data word-length in DSP

- **SWP instructions (Sub-Word Parallelism)**

- Several operations executed in parallel on a same operator

- ⇒ **Decrease the code execution time**

- ⇒ **Decrease the computation accuracy**

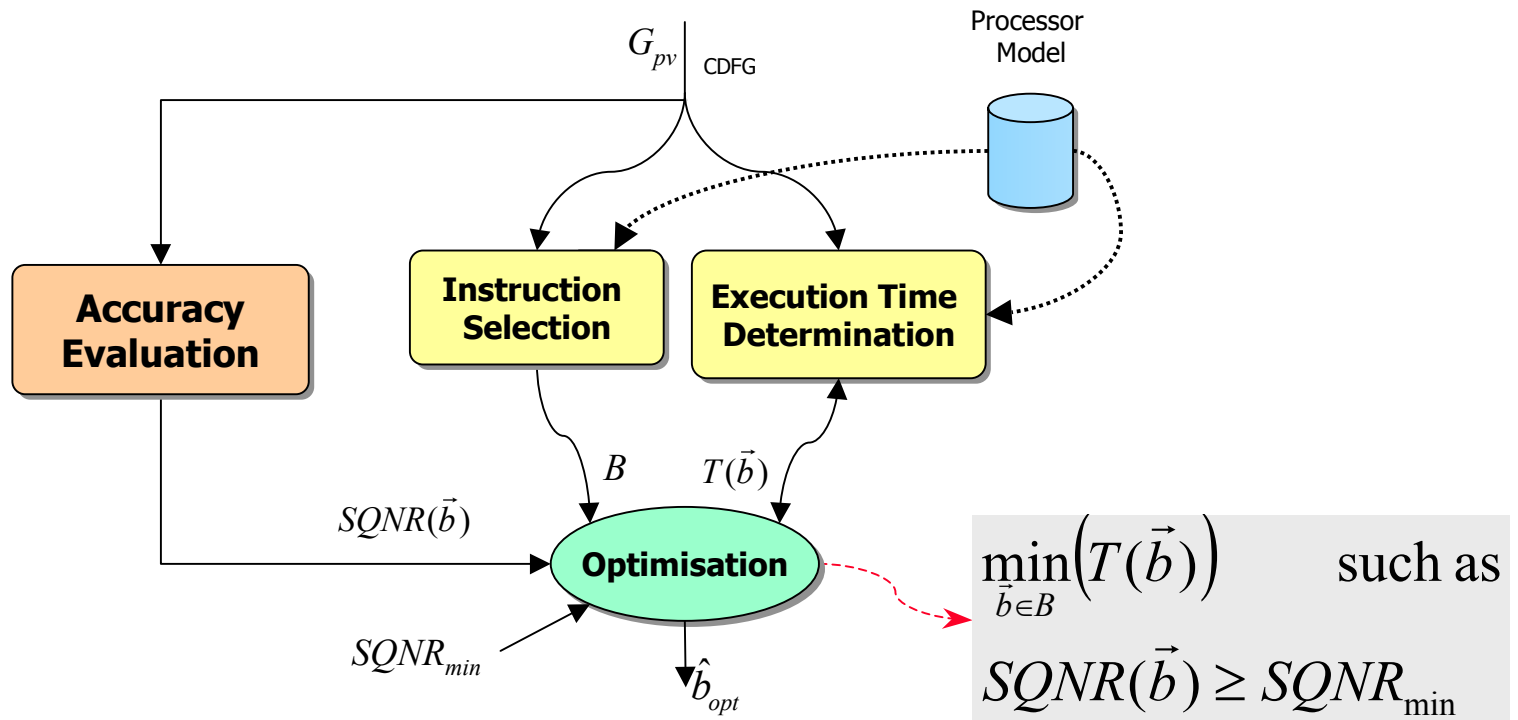


- DSP with SWP capabilities can manipulate a wide range of data types

Processor	Data type (bits)
TMS320C64x (T.I.)	8, 16, 32, 40, 64
TigerSHARC (A.D.)	8, 16, 32, 64
SP5, UniPhy (3DSP)	8, 16, 24, 32, 48
CEVA-X1620 (CEVA)	8, 16, 32, 40
ZSP500 (LSI Logic)	16, 32, 40, 64
OneDSP (Siroyan)	8, 16, 32, 44, 88

Data word-length optimization

- Minimize the code execution time under an accuracy constraint
 - Select the set of instructions which minimizes the code execution time $T(b)$ and fulfil the precision constraint $SQNR_{min}$



Accuracy evaluation

- **Technique based on an analytical approach [3]**
 - **Reduce the evaluation time compared to the simulation based approaches**
 - ⇒ Simulation based approach drawbacks
 - Long simulation time
 - Fixed-point format optimization process requires multiple simulations
- **Accuracy evaluation metric**
 - **Signal to Quantization Noise Ratio (SQNR)**
 - ⇒ Computation of the SQNR expression $SQNR(\vec{b})$ according to the data word-length \vec{b}

[3] D. Menard and O. Sentieys. *Automatic Evaluation of the Accuracy of Fixed-point Algorithms*. DATE'02, Paris, March 2002

Code execution time estimation

- **Processor model**

- **Data flow instruction set which includes the SWP, the classical and the multi-precision instructions**

Instruction j_k	Function γ_k	Execution time t_k	I/O operand word-length		
			b^{in1}	b^{in2}	b^{out}
j_1	MULT	0.25	8	8	16
j_2	MULT	0.5	16	16	32
j_3	MULT	1	32	32	64
j_4	ADD	0.25	16	16	16
j_5	ADD	0.5	32	32	32
j_6	ADD	1	64	64	64

- **Code execution time estimation**

- **Goals**

- ⇒ Compare and select two instruction series
- ⇒ Simple model to obtain a small evaluation time

Code execution time estimation

- Execution time T estimation technique

$$T = \sum_i t_i \cdot n_i$$

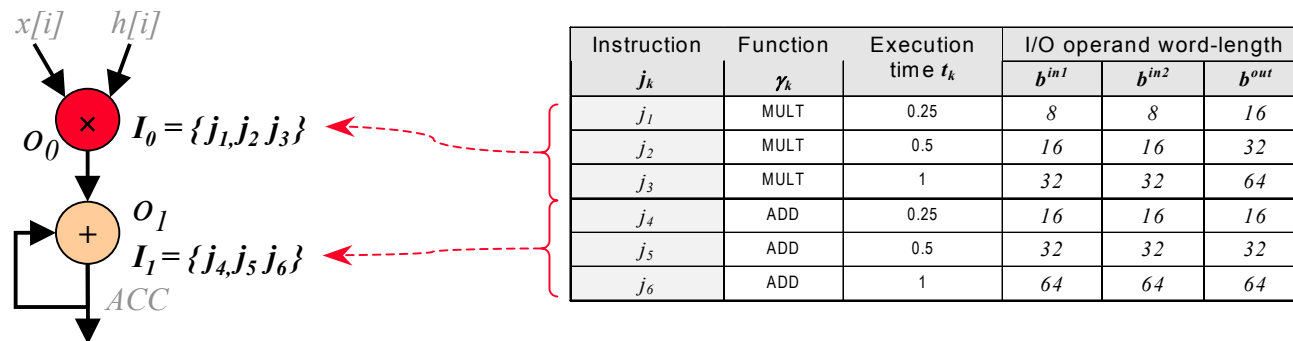
Operation o_i
execution time

Number of time that the
operation o_i is executed

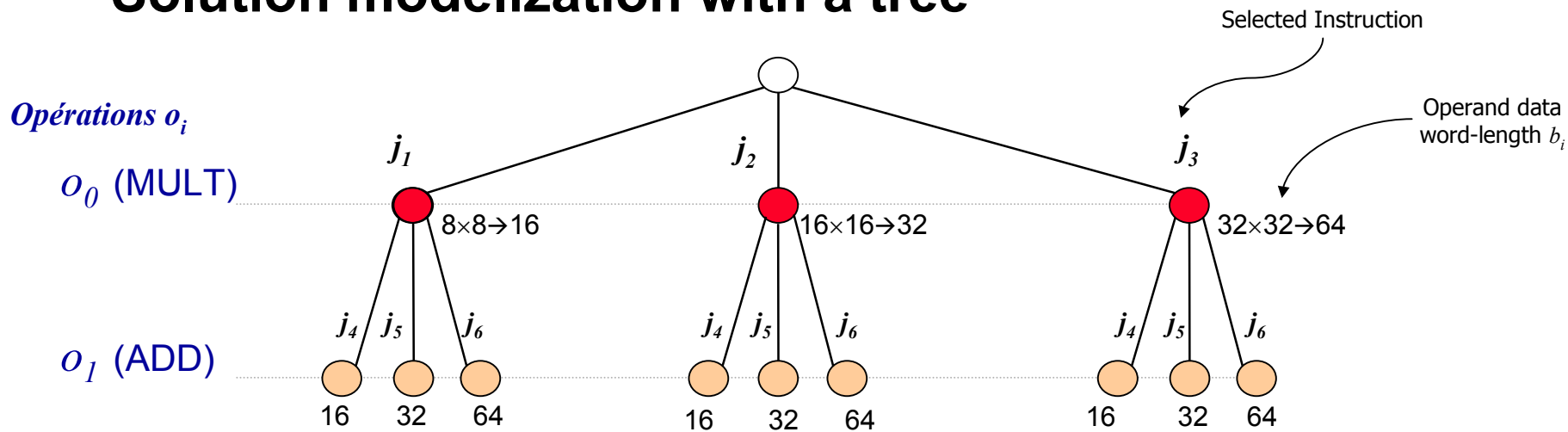
- **DSP without Instruction Level Parallelism (ILP)**
 - ⇒ Accurate results
- **DSP with ILP**
 - ⇒ Estimation of the vertical code execution time
 - ⇒ The gains due to code parallelization are closed for two instruction series
 - **Classical and SWP instructions use the same functional unit at the same clock cycle**

Solution modelization

- Application data flow graph example (FIR filter)



- Solution modelization with a tree

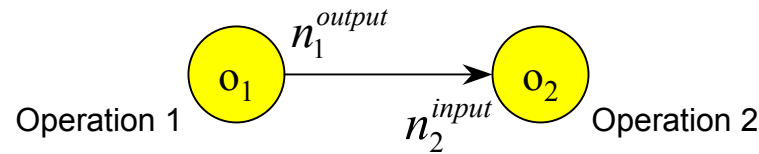


⇒ Branch & Bound algorithm

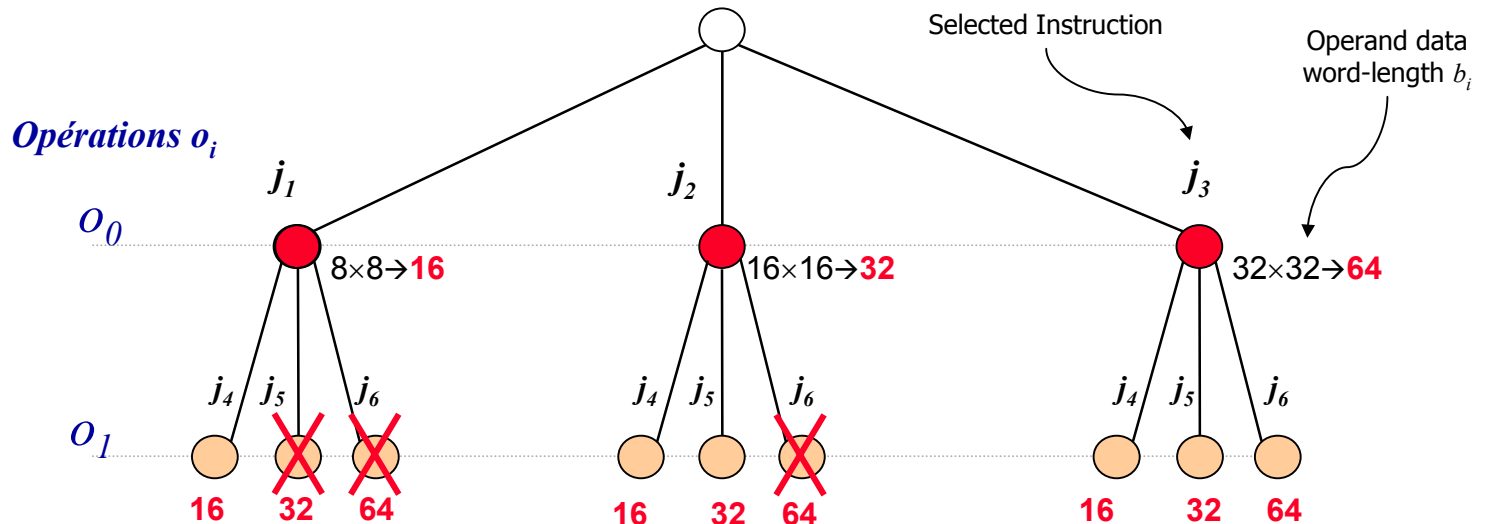
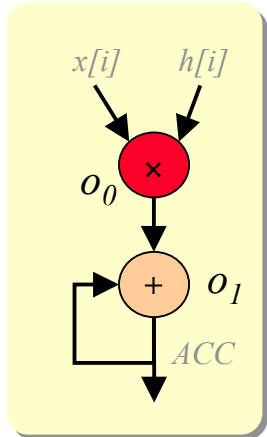
Search space limitation

- **Instruction combination restriction**

- **Fractional part word-length restriction** $n_1^{output} \geq n_2^{input}$



- **FIR filter example** (same binary-point position for the data)



Search space limitation

- **Partial solution evaluation : the exploration is stop if**
 - ⇒ The minimal execution time which can be obtained is greater than the minimal execution time already obtained
 - ⇒ The maximal SQNR which can be obtained is lower than the SQNR constraint
- **Node evaluation order**
 - The variables are proceeded according to their influence on the execution time and the SQNR
- **Reduction of the number of values per variable**
 - **Two step optimization**
 - ⇒ **1. Integer optimization leading to the solution \tilde{b}_i**
 - ⇒ **2. Branch & Bound with 2 values per variable**

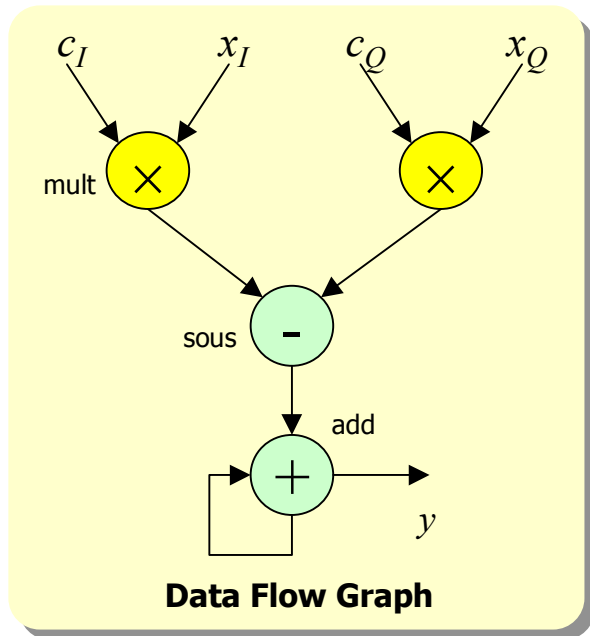
$$b_i^{\min} \leq \tilde{b}_i \leq b_i^{\max} \quad \text{with} \quad b_i^{\min}, b_i^{\max} \in B$$

Part 4

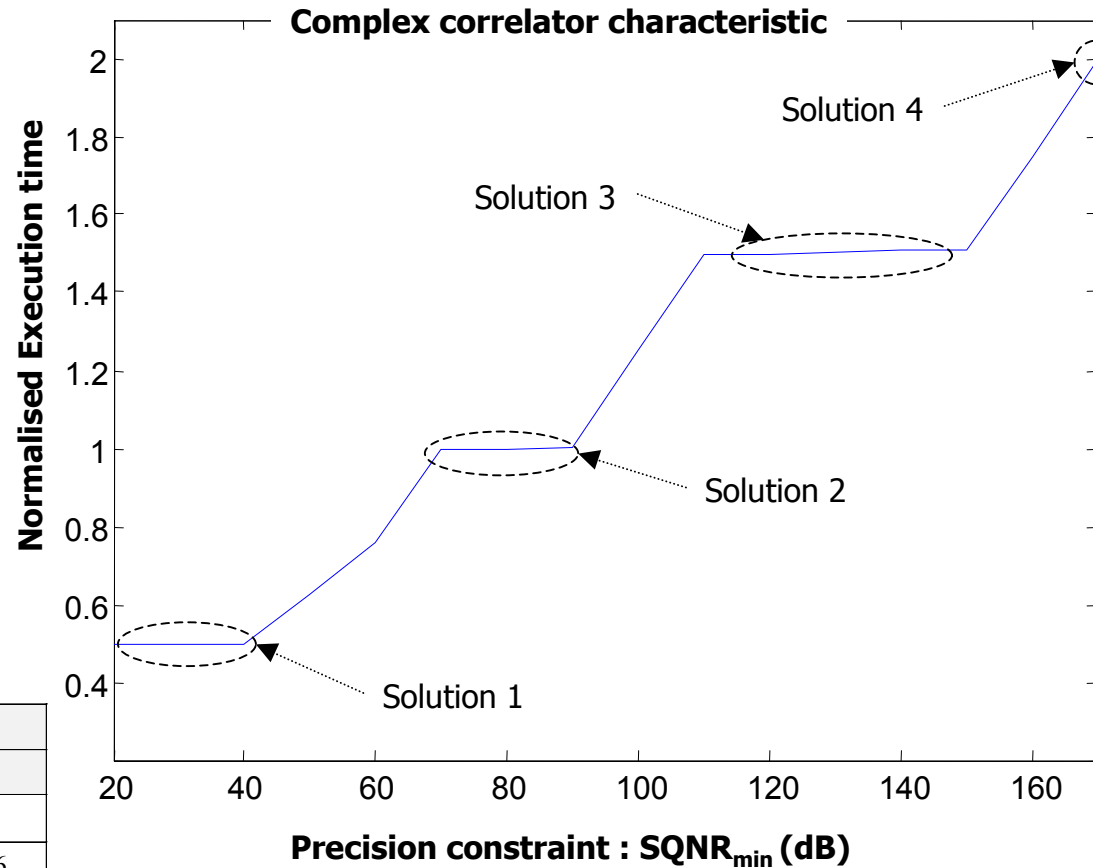
Experiments and Results

Complex correlator

- Complex correlator characteristic

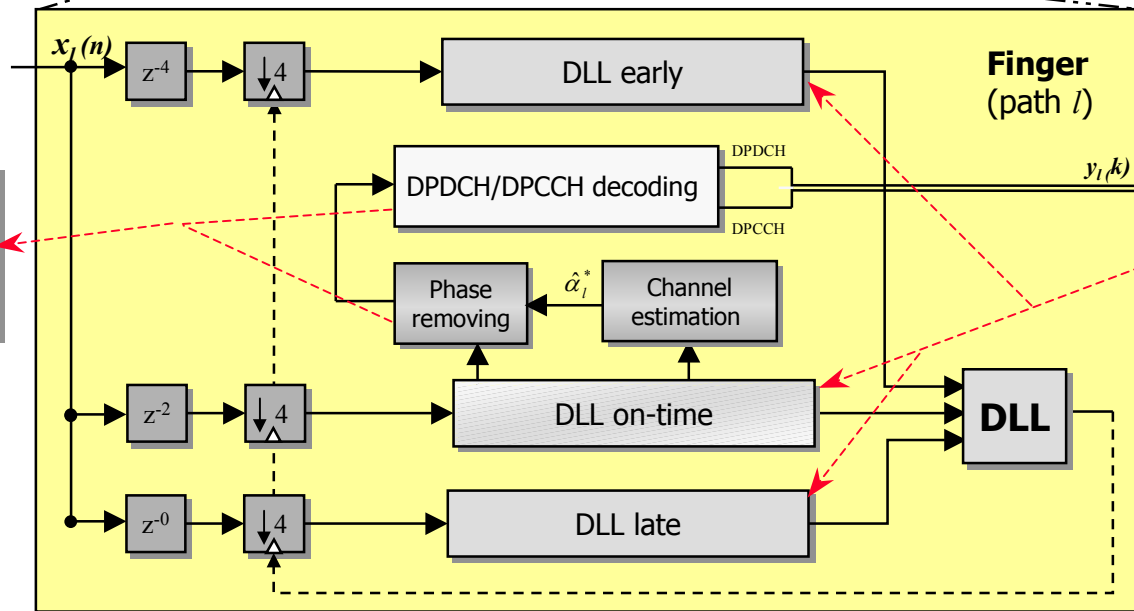
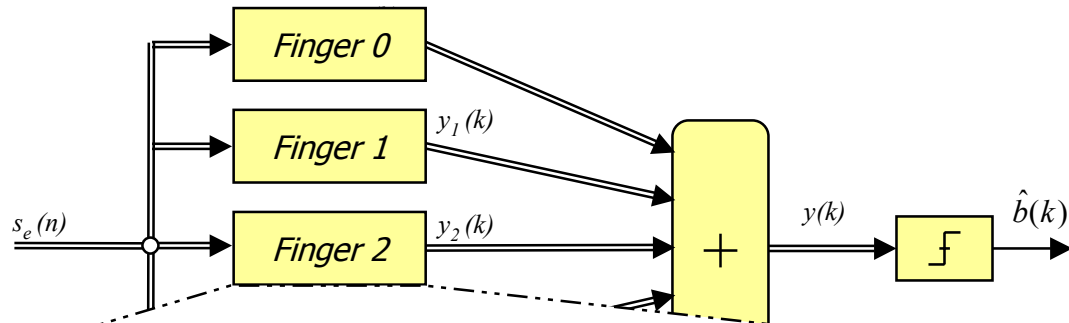


Operation and data types (bits)				
Solutions	mult	sous	add	y
1	8×8→16	16+16→16	16+16→16	8
2	16×16→32	32+32→32	32+32→32	16
3	32×16→32	32+32→32	32+32→32	32
4	32×16→64	64+64→64	64+64→64	32



Rake receiver

- Used in the WCDMA receiver for UMTS
- Application description



• Symbol decoding subsystem

- 10 MULT
- 8 ADD

• Synchronization subsystem

- 3*10 MULT
- 3*6 ADD

Rake Receiver

- The SQNR constraint is defined according to the receiver performance (Bit Error Rate)
- Target processor C64x (Texas Instrument)
- SWP improvement factor :
$$F = \frac{T_{exec-NoSWP}}{T_{exec-SWP}}$$

Number of fingers grouped in the same loop	SWP improvement factor F	
	Symbol decoding subsystem	Synchronization subsystem
1	2.83	1.91
2	2.79	2.79
4	3.51	3.18

Optimization time : 180s

- The code execution time can be reduced dramatically

Conclusion

- **New methodology for the Floating-point to Fixed-point conversion**
 - **The architecture is taken into account**
- **Data word-length selection**
 - **Minimize the code execution time under accuracy constraint**
 - ⇒ This approach allows to
 - **Analyze the trade-off between the code execution time and the accuracy**
 - **Reduce the code execution time**
- **Perspective: methodology for the case of ASIC / FPGA**
 - **Several phases of this methodology can be reused**