Compilation de ReactiveML

Rapport de stage pour le M2 du Mpri

Louis Jachiet
sous la direction de Louis Mandel

École Normale Supérieure
Équipe Parkas

3 septembre 2013

ReactiveML

- OCaml
- Systèmes interactifs
- Du parallélisme synchrone

Exemple de parallélisme synchrone

```
loop
    print_string "tic ";
    pause;
    pause;
end
||
loop
    pause;
    print_endline "tac!";
    pause;
end
```

Sortie :

Instant :

1	2	3	4	5	6	7	8

Exemple de parallélisme synchrone

```
loop
    print_string "tic ";
    pause;
    pause;
end
||
loop
    pause;
    print_endline "tac!";
    pause;
end
```

Sortie :

```
tic
```

Instant : | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |

Exemple de parallélisme synchrone

```
loop
    print_string "tic ";
    pause;
    pause;
end
||
loop
    pause;
    print_endline "tac!";
    pause;
end
```

Sortie :

```
tic tac!
```

Instant :

1	2	3	4	5	6	7	8

Exemple de parallélisme synchrone

```
loop
    print_string "tic ";
    pause;
    pause;
end
||
loop
    pause;
    print_endline "tac!";
    pause;
end
```

Sortie :

```
tic tac!
tic
```

Instant : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Exemple de parallélisme synchrone

```
loop
    print_string "tic ";
    pause;
    pause;
end
||
loop
    pause;
    print_endline "tac!";
    pause;
end
```

Sortie :

```
tic tac!
tic tac!
```

Instant : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Exemple de parallélisme synchrone

```
loop
    print_string "tic ";
    pause;
    pause;
end
||
loop
    pause;
    print_endline "tac!";
    pause;
end
```

Sortie :

```
tic tac!
tic tac!
tic
```

Instant : | 1 | 2 | 3 | 4 | **5** | 6 | 7 | 8 |

Exemple de parallélisme synchrone

```
loop
    print_string "tic ";
    pause;
    pause;
end
||
loop
    pause;
    print_endline "tac!";
    pause;
end
```

Sortie :

```
tic tac!
tic tac!
tic tac!
```

Instant : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Exemple de parallélisme synchrone

```
loop
    print_string "tic ";
    pause;
    pause;
end
||
loop
    pause;
    print_endline "tac!";
    pause;
end
```

Sortie :

```
tic tac!
tic tac!
tic tac!
tic
```

Instant : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Exemple de parallélisme synchrone

```
loop
    print_string "tic ";
    pause;
    pause;
end
||
loop
    pause;
    print_endline "tac!";
    pause;
end
```

Sortie :

```
tic tac!
tic tac!
tic tac!
tic tac!
```

Instant : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Exemple de communication

```
signal tic in
    loop
        emit tic; pause ; pause
    end
||
    loop
        present tic
        then (print_endline "Tac!"; pause)
        else ()
    end
```

Sortie :

Instant : | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Exemple de communication

```
signal tic in
    loop
        emit tic; pause ; pause
    end
||
    loop
        present tic
        then (print_endline "Tac!"; pause)
        else ()
    end
```

Sortie :

```
Tac!
```

Instant : **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8**

Exemple de communication

```
signal tic in
    loop
        emit tic; pause ; pause
    end
||
    loop
        present tic
        then (print_endline "Tac!"; pause)
        else ()
    end
```

Sortie :

```
Tac!
```

Instant :

1	2	3	4	5	6	7	8

Exemple de communication

```
signal tic in
    loop
        emit tic; pause ; pause
    end
||
    loop
        present tic
        then (print_endline "Tac!"; pause)
        else ()
    end
```

Sortie :

```
Tac!
Tac!
```

Instant :

1	2	3	4	5	6	7	8

Exemple de communication

```
signal tic in
    loop
        emit tic; pause ; pause
    end
||
    loop
        present tic
        then (print_endline "Tac!"; pause)
        else ()
    end
```

Sortie :

```
Tac!
Tac!
```

Instant :

1	2	3	**4**	5	6	7	8

Exemple de communication

```
signal tic in
    loop
        emit tic; pause ; pause
    end
||
    loop
        present tic
        then (print_endline "Tac!"; pause)
        else ()
    end
```

Instant :

1	2	3	4	5	6	7	8

Sortie :

```
Tac!
Tac!
Tac!
```

Exemple de communication

```
signal tic in
    loop
        emit tic; pause ; pause
    end
||
    loop
        present tic
        then (print_endline "Tac!"; pause)
        else ()
    end
```

Instant :

1	2	3	4	5	6	7	8

Sortie :

```
Tac!
Tac!
Tac!
```

Exemple de communication

```
signal tic in
    loop
        emit tic; pause ; pause
    end
||
    loop
        present tic
        then (print_endline "Tac!"; pause)
        else ()
    end
```

Instant :

1	2	3	4	5	6	7	8

Sortie :

```
Tac!
Tac!
Tac!
Tac!
```

Exemple de communication

```
signal tic in
    loop
        emit tic; pause ; pause
    end
||
    loop
        present tic
        then (print_endline "Tac!"; pause)
        else ()
    end
```

Instant :

1	2	3	4	5	6	7	8

Sortie :

```
Tac!
Tac!
Tac!
Tac!
```

Problématique

```
signal a in
    signal b in
      present a then print_string "42"
    ||
      present b then emit a
    ||
      emit b
```

Instant : 1

Sortie :

Problématique

```
signal a in
    signal b in
      present a then print_string "42"
    ||
      present b then emit a
    ||
      emit b
```

Instant : 1

Sortie :

Problématique

```
signal a in
    signal b in
        present a then print_string "42"
      ||
        present b then emit a
      ||
        emit b
```

Instant : 1

Sortie :

Problématique

```
signal a in
    signal b in
        present a then print_string "42"
      ||
        present b then emit a
      ||
        emit b
```

Instant : 1

Sortie :

Problématique

```
signal a in
    signal b in
        present a then print_string "42"
      ||
        present b then emit a
      ||
        emit b
```

Instant : 1

Sortie :

Problématique

```
signal a in
    signal b in
        present a then print_string "42"
      ||
        present b then emit a
      ||
        emit b
```

Instant : 1

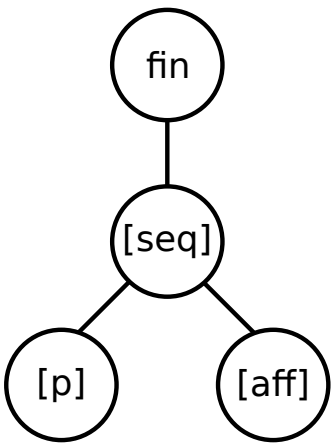
Sortie : 42

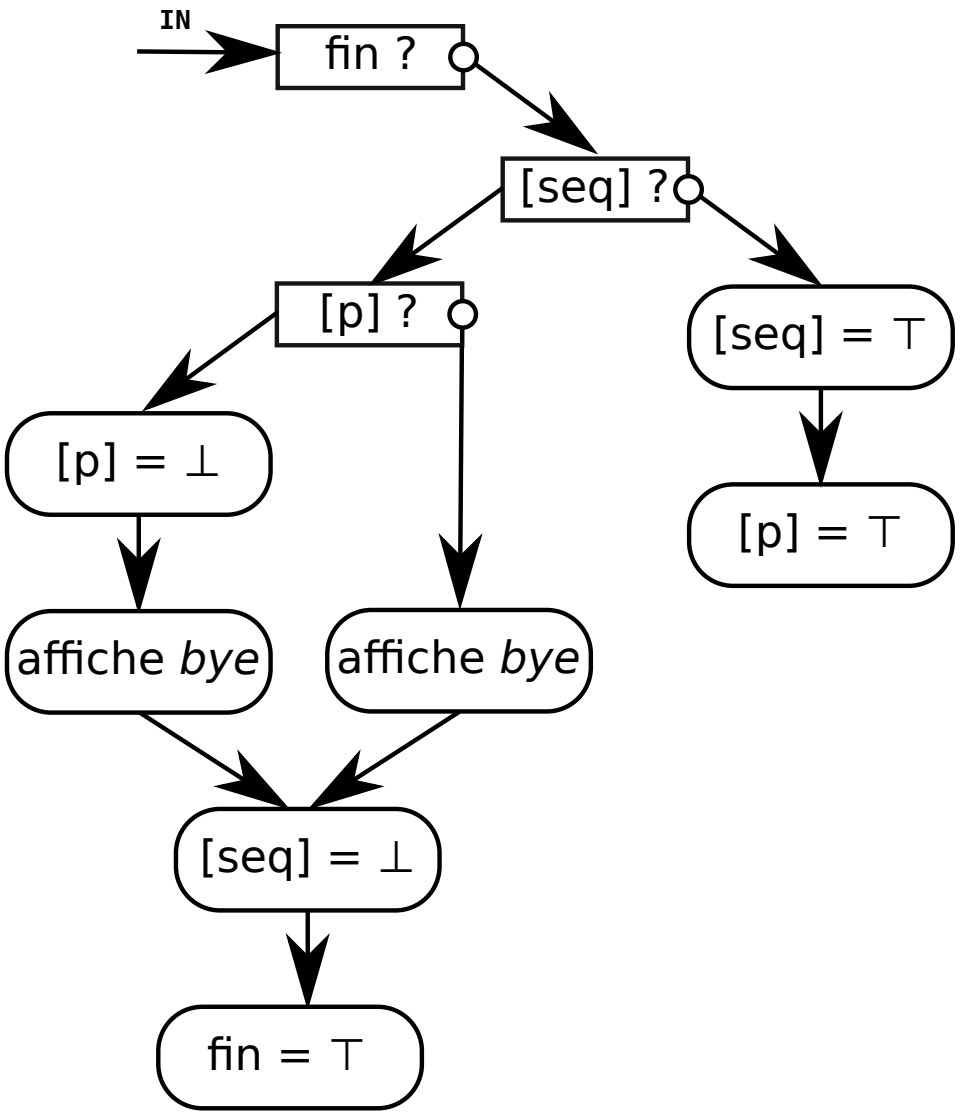
Esterel	ReactiveML
Langage synchrone	
Langage impératif simple	OCAML
Embarqué critique	Système interactif
Ordonnancement statique	Ordonnancement dynamique
Années 80	Années 2000

Bien étudié

- *grc2c* chez Inria par Dumitru Potop-Butucaru [PB02]
- *CEC* à l'université de Columbia par Stephen A. Edwards [Edw94]
- *Saxo-RT* chez France Télécom par Etienne Closse et al [CPP+02]

```
pause ;
print "bye"
```





La méthode GRC (*grc2c*,*CEC*)

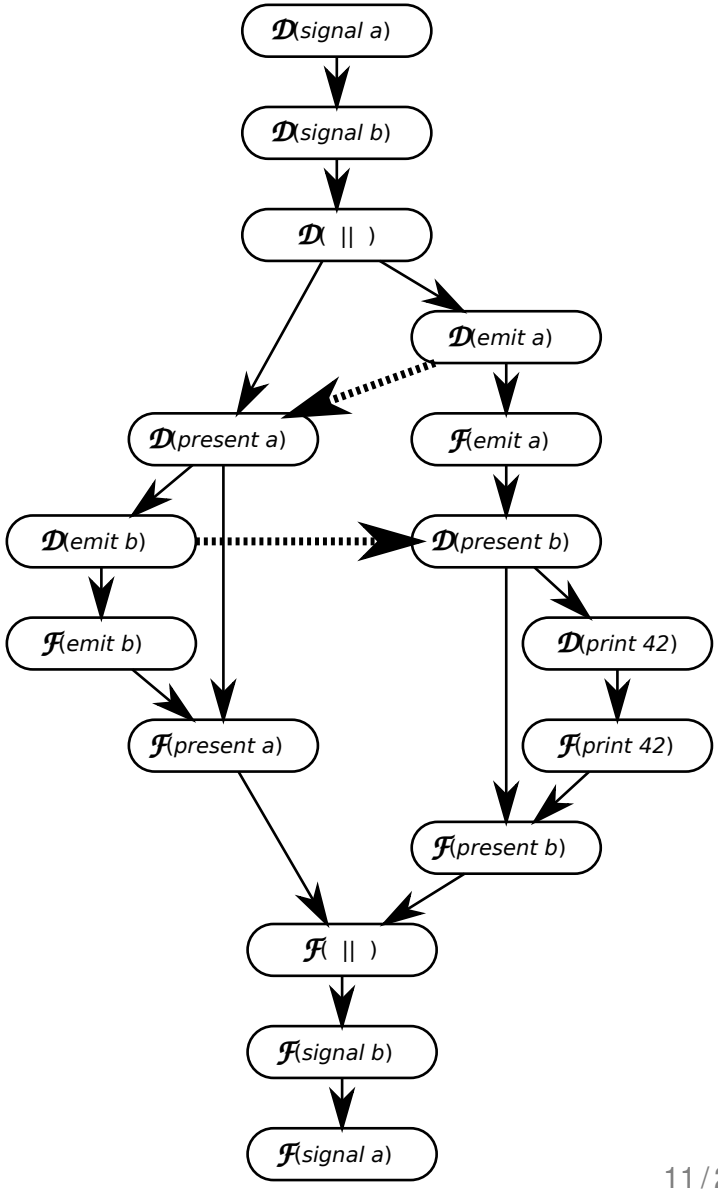
- État du programme : une liste des instructions en cours d'exécution.
- Transitions : un graphe de contrôle.
- On fait un tri topologique pour interpréter les nœuds dans l'ordre des dépendances.

La méthode Saxo-RT

- État du programme : liste des nœuds actifs de ce graphe.
- Transition : à chaque instant un nœud peut activer ou désactiver d'autres nœuds pour l'instant courant ou d'après.

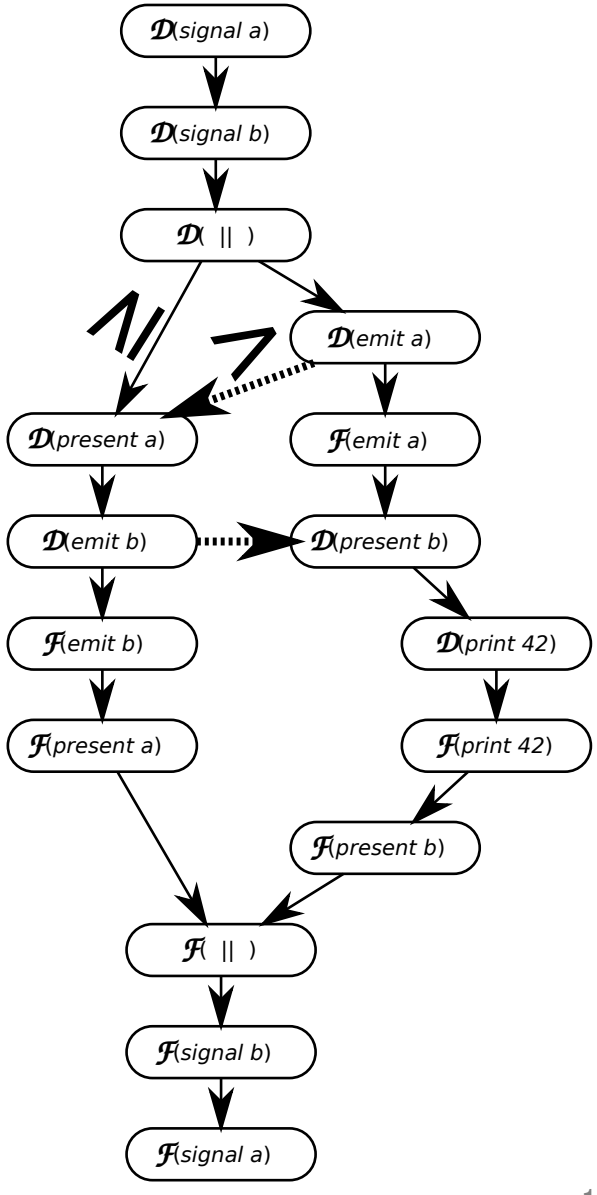
Saxo-RT : exemple

```
signal a in
   signal b in
      present a
         then emit b
         else ()
   ||
      emit a;
      present b
         then print "42"
         else ()
   end
end
```



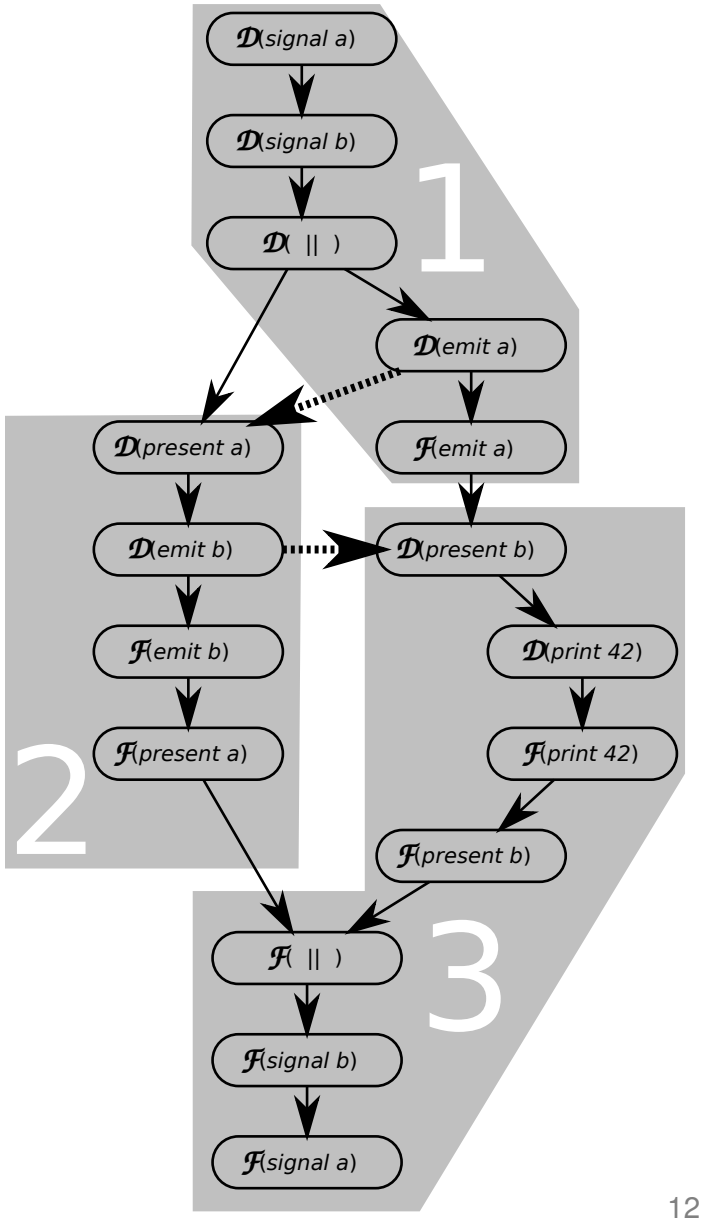
Analyse : exemple

```
signal a in
   signal b in
      present a
         then emit b
         else ()
   ||
      emit a;
      present b
         then print "42"
         else ()
   end
end
```



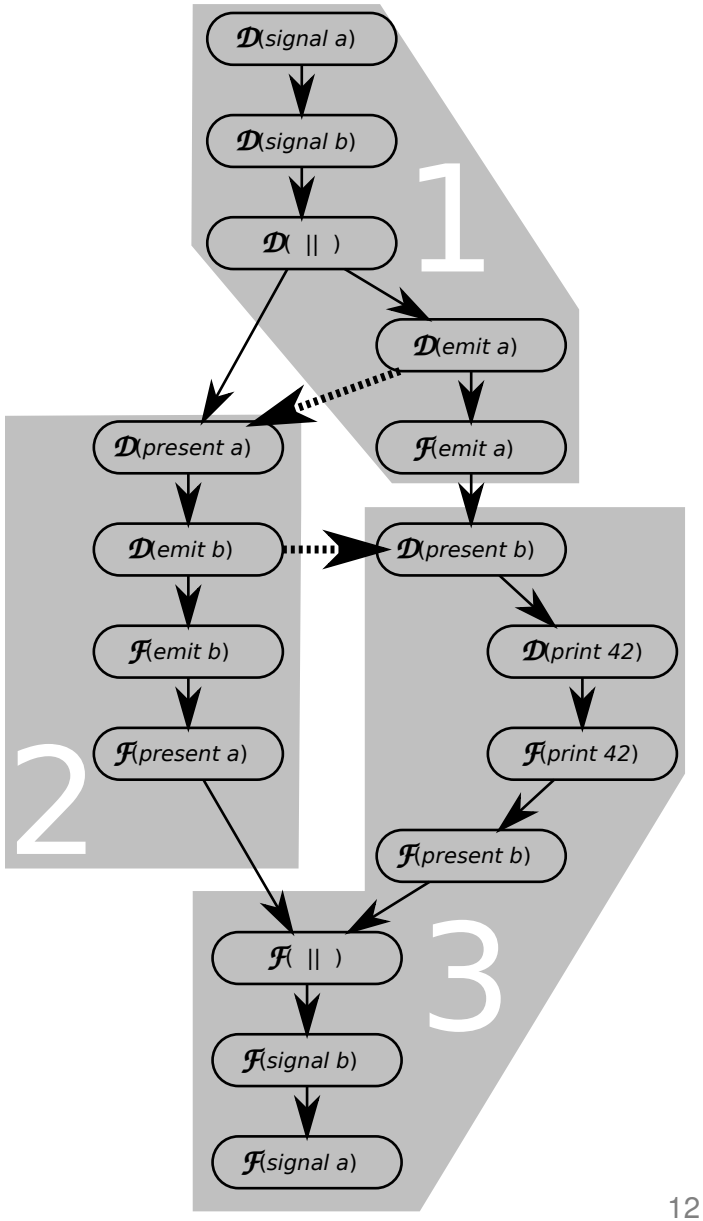
Analyse : exemple

```
signal a in
   signal b in
      present a
         then emit b
         else ()
   ||
      emit a;
      present b
         then print "42"
         else ()
   end
end
```



Analyse : exemple

```
signal a in
   signal b in
      pause for 1 level
      present a
         then emit b
         else ()
   ||
      emit a;
      pause for 2 levels
      present b
         then print "42"
         else ()
   end
end
```



Principe de notre méthode

- On divise les instants en n niveaux
- Le niveau d'une instruction qui teste un signal doit être supérieur strictement au niveau d'une instruction qui l'émet
- On insère dans le programme des pauses de un ou plusieurs niveaux

Compilation

La compilation est similaire à celle de ReactiveML :

- Transformation Continuation Passing Style
- Un moteur d'exécution qui réveille les processus
- Une bibliothèque de fonctions

Limites à l'analyse : la modularité

```
let rec process wait s =
  present s then () else wait s
```

```
let rec process present_s () =
  present s then true else false
```

Limites à l'analyse : les signaux ?

```
let rec process prevSignal =
  let stock = ref sInit in
  fun sCur ->
    let prev = !stock in
    signal nouv in
      stock := nouv ;
      prev
```

Productions

Un compilateur d'un fragment d'Esterel :

- compile vers OCaml
- méthode Grc
- ~ 1000 lignes d'OCaml

Un compilateur d'un fragment de ReactiveML :

- source à source sur le langage intermédiaire de ReactiveML
- implémente la méthode présentée
- ~ 500 lignes d'OCaml

Performances

```
loop
  signal s in
    present s
    then pause
    else ()
  ||
    emit s ; pause
end
```

```
loop
  signal s in
    emit s ; pause
  ||
    present s
    then pause
    else ()
end
```

Temps d'exécution sur la distribution ReactiveML		
Mauvais ordre	Bon ordre	ralentissement
9.56s	7.12s	34% (à 1%)

Temps d'exécution sur notre compilateur		
Mauvais ordre	Bon ordre	ralentissement
4.27s	4.32s	< 1%

Travaux futurs

- Améliorer l'analyse
- Compilation hybride
- Intégration complète à la distribution ReactiveML

Références

G. Berry and L. Cosserat.
The synchronous programming language Esterel and its mathematical semantics.
Lecture Notes in Computer Science, 1984.

G. Berry.
The constructive semantics of pure esterel.
1999.

G. Berry.
Chaire algorithme, machines et langages du collège de france : « l'informatique du temps et des événements ».
2013.

Frédéric Boussinot.
Reactive C : An extension of C to program reactive systems.
Software Practice and Experience, 21(4) :401–428, April 1991.

Etienne Closse, Michel Poize, Jacques Pulou, Patrick Venier, and Daniel Weil.
Saxo–rt : Interpreting esterel semantic on a sequential execution structure.
Electronic Notes in Theoretical Computer Science, 65(5) :80 – 94, 2002.
SLAP'2002, Synchronous Languages, Applications, and Programming (Satellite Event of {ETAPS} 2002).

S. Edwards.
An esterel compiler for a synchronous/ reactive development system.
Technical Report UCB/ERL M94/43, EECS Department, University of California, Berkeley, 1994.

Louis Mandel and Marc Pouzet.
ReactiveML : un langage fonctionnel pour la programmation réactive.
Technique et Science Informatiques (TSI), 27(9–10/2008) :1097–1128, 2008.

Dumitru Potop-Butucaru.
Generation of Fast C Code from Esterel Programs.
Thèse de doctorat, ENSMP/CMA, 2002.

Dumitru Potop-Butucaru, Stephen A. Edwards, and Gérard Berry.
Compiling Esterel.
2007.