



SAAB

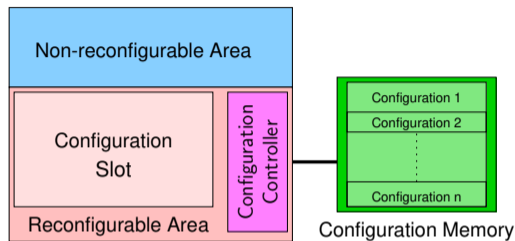
Designing Adaptive Systems with Run-Time Reconfigurable Architectures

FT2025, Stockholm, Sweden, October 14-15, 2025

Ingo Sander (KTH), Denis Loubach (ITA), Gabriel Duarte (ITA),
Anders Åhlander (Saab), and Ingemar Söderquist (Saab)

Contact: Ingo Sander (ingo@kth.se)

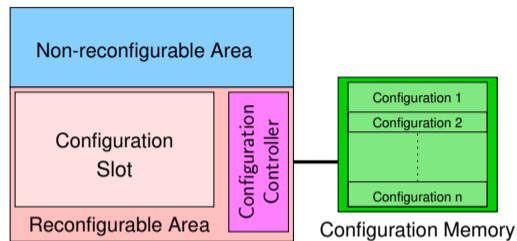
Partial and run-time reconfigurable FPGAs



Partial and run-time reconfigurable FPGAs offer huge potential for design of adaptive systems

- New functions can be loaded from a configuration memory into the reconfigurable area during run-time, while the rest of the FPGA continues with its operation

Partial and run-time reconfigurable FPGAs



Partial and run-time reconfigurable FPGAs offer huge potential for design of adaptive systems

- New functions can be loaded from a configuration memory into the reconfigurable area during run-time, while the rest of the FPGA continues with its operation

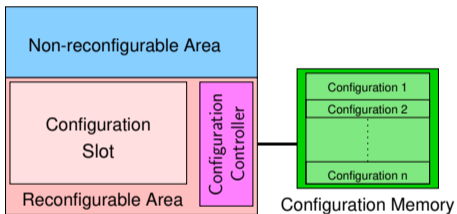
? How to exploit this technology for system design?

- Current lack of system-level design methodologies to exploit technology
- Suitable high-level models and refinement steps have to be identified

Modelling of Adaptivity

Basic Idea: Adaptive Process

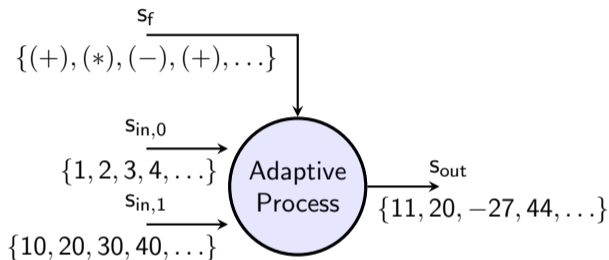
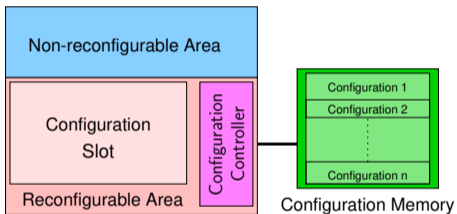
Run-Time Reconfigurable FPGA



Modelling of Adaptivity

Basic Idea: Adaptive Process

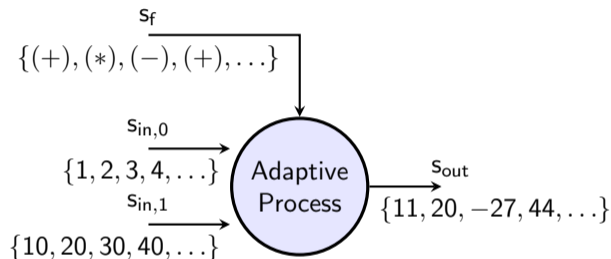
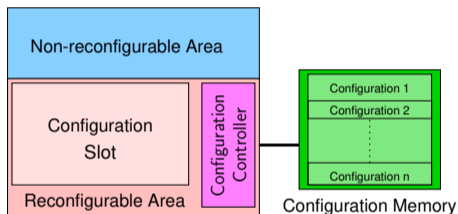
Run-Time Reconfigurable FPGA



Modelling of Adaptivity

Basic Idea: Adaptive Process

Run-Time Reconfigurable FPGA

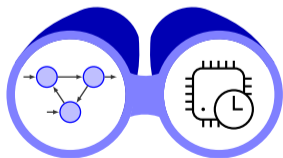


- Functions can be used as signal values
- **Adaptive Process** executes function that is provided at time instance for execution

Functionality is provided from the outside and "loaded" into the adaptive process.

ForSyDe (Formal System Design)

- ForSyDe is a design methodology targeting heterogeneous embedded systems design
- ForSyDe envisions a correct-by-construction design process
- ForSyDe consists of modelling libraries and tools for different steps of the design flow
- ForSyDe is based on the functional programming paradigm and a sound theoretical base in form of models of computation
- The ForSyDe modelling framework is implemented in Haskell and SystemC
- ForSyDe supports several models of computation



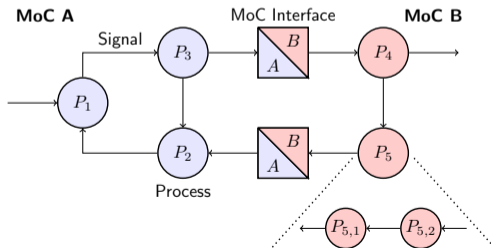
ForSyDe vision

From high-level model to real-time implementation

ForSyDe Application Model

ForSyDe follows the tagged-signal model (Lee and Sangiovanni-Vincentelli, 1998).

- An application is modelled as hierarchical concurrent process network
- Processes belonging to different models of computation communicate via MoC interfaces
- ForSyDe libraries in Haskell and SystemC to support the designer to create a formal model exist for several MoCs
 - e.g. synchronous MoC, continuous time MoC, SDF MoC, SADF MoC



ForSyDe (Synchronous Model of Computation)

Signals

$$\xrightarrow{s = \{v_0, v_1, v_2, \dots\}}$$

- A **signal** is modelled as a set of events, where each event has a **tag** and a **value**.
- In the synchronous model of computation the tag is given by the position in the signal.

ForSyDe (Synchronous Model of Computation)

Signals

$$\xrightarrow{\quad} \\ s = \{v_0, v_1, v_2, \dots\}$$

- A **signal** is modelled as a set of events, where each event has a **tag** and a **value**.
- In the synchronous model of computation the tag is given by the position in the signal.

ForSyDe model

```
1 s = signal [1,2,3,4]
```

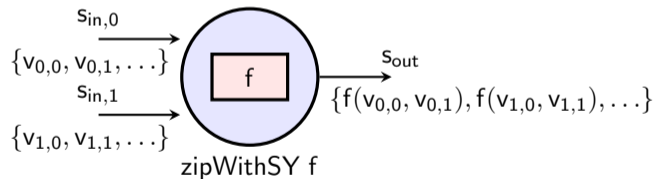
Simulation in ForSyDe (Haskell)

```
ghci> s
{1,2,3,4}
```

ForSyDe (Synchronous Model of Computation)

Processes

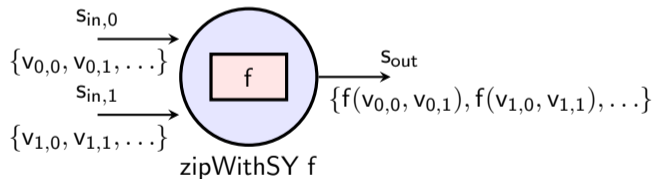
A **process** is modelled using a **process constructor** (here `zipWithSY`) that takes function(s) and/or values as arguments.



ForSyDe (Synchronous Model of Computation)

Processes

A **process** is modelled using a **process constructor** (here `zipWithSY`) that takes function(s) and/or values as arguments.



ForSyDe model

```

1  s0 = signal [1,2,3,4]
2  s1 = signal [10,20,30,40]
3  adder = zipWithSY (+)

```

Simulation in ForSyDe

```

ghci> adder s0 s1
{11,22,33,44}

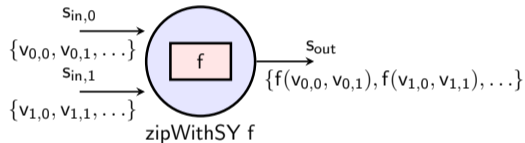
```

ForSyDe (Synchronous Model of Computation)

Basic process constructors

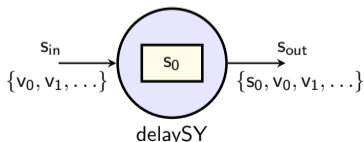
Process constructors are classified into combinational and sequential process constructors.

■ Combinational process constructors



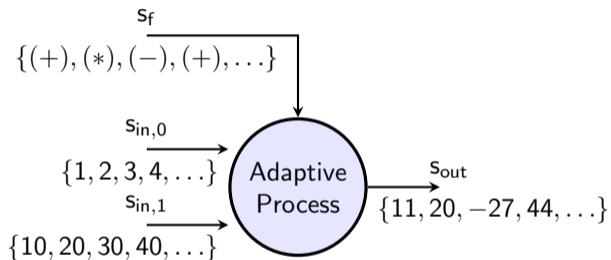
```
ghci> s1 = signal [1,2,3,4]
ghci> s2 = signal [10,20,30,40]
ghci> zipWithSY (*) s1 s2
{10,40,90,160}
```

■ Sequential process constructors

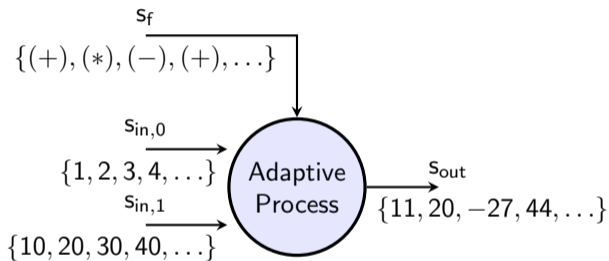


```
ghci> delaySY 0 s1
{0,1,2,3,4}
```

ForSyDe: Adaptive Process



ForSyDe: Adaptive Process



ForSyDe model

```

1  adaptive = zipWith3SY ($)
2  s_f = signal [(+), (*), (-), (+)]
3  s_0 = signal [1,2,3,4]
4  s_1 = signal [10,20,30,40]

```

Simulation in Haskell

```

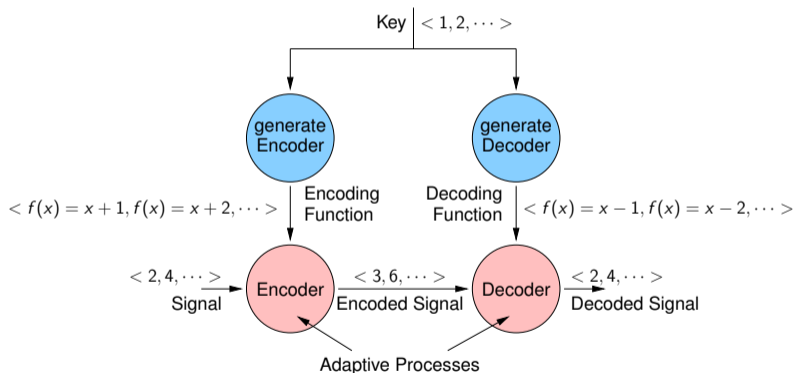
ghci> adaptive s_f s_0 s_1
{11,40,-27,44}

```

The functional programming paradigm, where functions are treated as first-class citizens, is a perfect match for modelling adaptivity.

Modelling Adaptive Systems in ForSyDe

Tutorial Encoder/Decoder Example



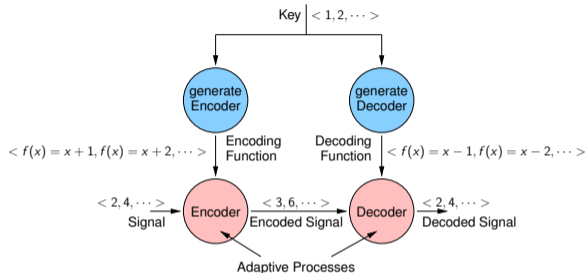
Modelling Adaptive Systems in ForSyDe

Tutorial Encoder/Decoder Example

```

1  module Adaptivity where
2
3  import ForSyDe.Shallow
4
5  system s_key s_in = (s_enc, s_dec) where
6    s_dec = decoder s_decF s_enc
7    s_encF = genEnc s_key
8    s_decF = genDec s_key
9    s_enc = encoder s_encF s_in
10
11  genEnc = mapSY (+)
12  genDec = mapSY f where
13    f x y = y - x
14  decoder = zipWithSY ($)
15  encoder = zipWithSY ($)

```



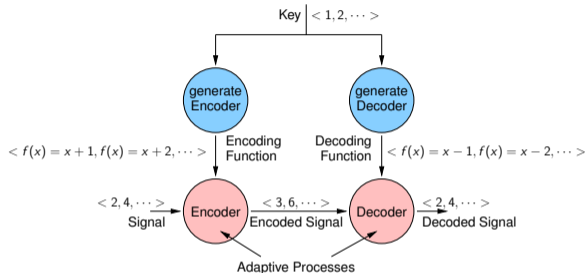
Modelling Adaptive Systems in ForSyDe

Tutorial Encoder/Decoder Example

```

1  module Adaptivity where
2
3  import ForSyDe.Shallow
4
5  system s_key s_in = (s_enc, s_dec) where
6      s_dec = decoder s_decF s_enc
7      s_encF = genEnc s_key
8      s_decF = genDec s_key
9      s_enc = encoder s_encF s_in
10
11  genEnc = mapSY (+)
12  genDec = mapSY f where
13      f x y = y - x
14  decoder = zipWithSY ($)
15  encoder = zipWithSY ($)

```

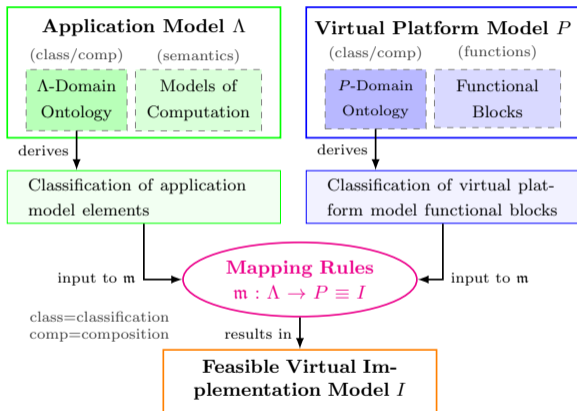


```

ghci> s_key = signal [1..5]
ghci> s_in = signal [2,4..10]
ghci> system s_key s_in
{2,4,6,8,10}

```

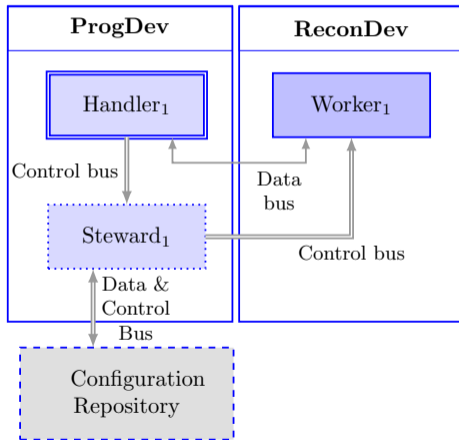
Conceptual Design Methodology for Adaptive Systems



- **Application Model** is modelled in ForSyDe
- **Virtual Platform Model** captures reconfiguration capabilities of possible target platforms
- **Mapping Rules** are used to generate a feasible **Virtual Implementation Model**
- **Virtual Implementation Model** is synthesised into final implementation

Virtual Platform Model

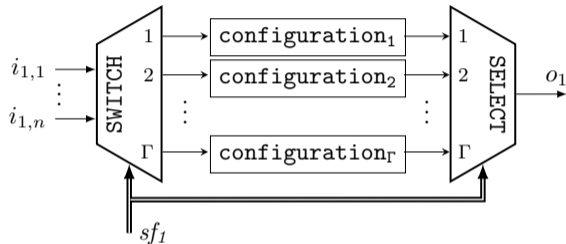
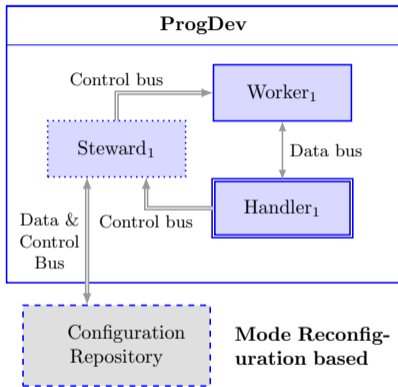
Handler, Steward, and Worker



- **Handler** initiates reconfiguration
- **Steward** prepares and executes reconfiguration
- **Worker** executes current functionality
- **Configuration Repository** contains the available functions that can be executed by the worker

Handler, Steward, and Worker can be implemented on programmable and reconfigurable devices.

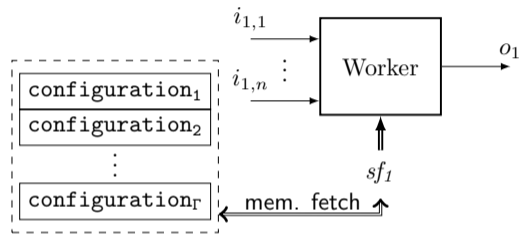
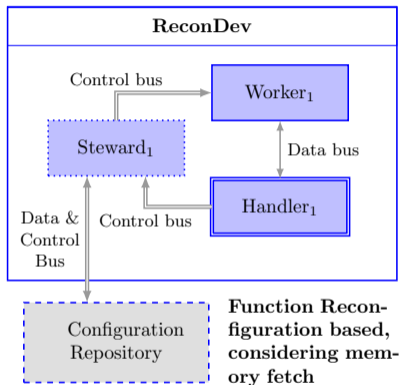
Mode Reconfiguration



Implementation in hardware

Reconfiguration is implemented on a programmable device. All configurable functions are pre-implemented on the device and selected at run-time.

Function Reconfiguration

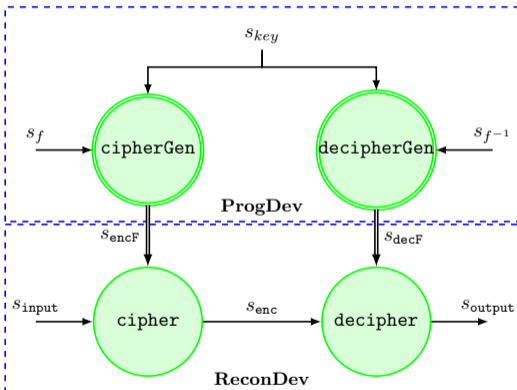


Implementation on run-time reconfigurable
FPGA

Reconfiguration is implemented on a run-time reconfigurable device. The available functions are stored in a configuration repository and loaded at run-time into the reconfigurable device.

Tutorial Encoder/Decoder Example

Partitioning

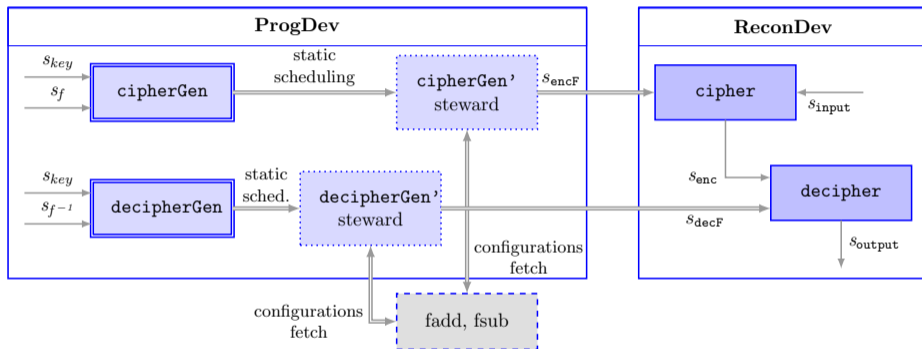


Partitioning decision:

- Handler and Steward shall be implemented on programmable device
- Worker shall be implemented on reconfigurable device

Tutorial Encoder/Decoder Example

Virtual Implementation Model



- Following the mapping rules the virtual implementation model is generated
- To yield the final implementation, the virtual implementation model has to be synthesised into the final implementation (ongoing work)

Conclusion

Runtime reconfigurable architectures

- Balanced trade-off between performance and flexibility
- Can dynamically change adaptive part of the hardware implementation **without changing other parts of the system**

Methodology

- Functional high-level adaptivity ForSyDe model based on formal MoCs
- Virtual platform model
- Formal base provides an important step towards **correct-by-construction**

Conclusion

Runtime reconfigurable architectures

- Balanced trade-off between performance and flexibility
- Can dynamically change adaptive part of the hardware implementation **without changing other parts of the system**

Methodology

- Functional high-level adaptivity ForSyDe model based on formal MoCs
- Virtual platform model
- Formal base provides an important step towards **correct-by-construction**

Challenge: Generating an efficient implementation

- Mapping of a high-level model that does not require and contain low-level architectural knowledge into an efficient partial reconfigurable HDL implementation
- Important step has been taken by providing an abstract virtual platform model; more work in particular on synthesis to the final implementation is required

Ongoing and Future Work

Synthesis implementation flow

- Supporting the synchronous MoC
- Implementing mode and function reconfiguration
- Goal: automated synthesis tool
- Vendor-independent

Extensions

- Add support for SADF MoC
- Heterogeneous Modelling
- Different controller implementations:
 - Hardware controller
 - Software controlling run-time reconfigurable hardware

Ongoing and Future Work

Synthesis implementation flow

- Supporting the synchronous MoC
- Implementing mode and function reconfiguration
- Goal: automated synthesis tool
- Vendor-independent

Extensions

- Add support for SADF MoC
- Heterogeneous Modelling
- Different controller implementations:
 - Hardware controller
 - Software controlling run-time reconfigurable hardware

Challenge: Correctness during reconfiguration

- **During reconfiguration:** Input data cannot be processed and needs to be discarded or buffered
- **After reconfiguration:** Process possibly buffered data
- **Challenge:** How to achieve functional and timely correctness?

Supporting Project and Links to ForSyDe

The work in this presentation has been supported by the following projects:

- Designing Adaptive Systems with Run-Time Reconfigurable Architectures (Vinnova, Sweden, 2023–2025)
- FLYMOV Engineering Research Center for the Aerial Mobility of the Future (FAPESP, Brazil, 2023–2027)

More information on ForSyDe

- ForSyDe web page: <https://forsyde.github.io/>
- ForSyDe tools are developed under a permissive open source license and publicly available on <https://github.com/forsyde>



Further Reading

- The initial concepts for modelling adaptivity in ForSyDe have been formulated in ([Sander and Jantsch, 2008](#)).
- An overview of ForSyDe with emphasis on the modelling framework is given in ([Sander et al., 2017](#)).
- The ForSyDe implementation of the scenario-aware data-flow model is described in ([Bonna et al., 2019](#)).
- The concept of a systematic design flow using virtual implementation models was introduced in ([Loubach et al., 2021](#)).
- Heterogeneous modelling using several models of computation has been addressed in ([Duarte et al., 2025](#)).

References I

- Ricardo Bonna, Denis S. Loubach, George Ungureanu, and Ingo Sander. Modeling and simulation of dynamic applications using scenario-aware dataflow. *ACM Trans. Des. Autom. Electron. Syst.*, 24(5):58:1–58:29, August 2019. ISSN 1084-4309. doi: 10.1145/3342997. URL <http://doi.acm.org/10.1145/3342997>.
- Gabriel C. Duarte, Denis S. Loubach, and Ingo Sander. High-level reconfigurable embedded system design based on heterogeneous models of computation. *IEEE Access*, 13:63918–63934, 2025. doi: 10.1109/ACCESS.2025.3559695.
- Edward A. Lee and Alberto Sangiovanni-Vincentelli. A framework for comparing models of computation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(12):1217–1229, December 1998.
- Denis S. Loubach, Ricardo Bonna, George Ungureanu, Ingo Sander, and Ingemar Söderquist. Classification and mapping of model elements for designing runtime reconfigurable systems. *IEEE Access*, 9:156337–156360, 2021. doi: 10.1109/ACCESS.2021.3129899.
- Ingo Sander and Axel Jantsch. Modelling adaptive systems in ForSyDe. *Electronic Notes in Theoretical Computer Science (ENTCS)*, 200(2):39–54, 2008. ISSN 1571-0661. doi: <http://dx.doi.org/10.1016/j.entcs.2008.02.011>. URL <http://web.it.kth.se/~ingo/Papers/ENTCS2008.pdf>. First Workshop on Verification of Adaptive Systems (VerAS 2007).
- Ingo Sander, Axel Jantsch, and Seyed-Hosein Attarzadeh-Niaki. ForSyDe: System design using a functional language and models of computation. In Soonhoi Ha and Jürgen Teich, editors, *Handbook of Hardware/Software Codesign*, pages 99–140. Springer Netherlands, 2017.