A Runtime Reconfiguration Design Targeting Avionics Systems

Denis S. Loubach
Assistant Professor
dloubach@fem.unicamp.br

Advanced Computing, Control & Embedded Systems Laboratory
University of Campinas - UNICAMP

ACCES Lab
UNICAMP

September 2016

Agenda

1. About Denis Loubach
2. Introduction
3. Background
4. Proposed Design
5. Case Study
6. Results
7. Conclusion
8. Ack

About Denis Loubach

Assistant Professor at UNICAMP, since 2014

Background

- PhD in Electronics and Computer Engineering, ITA, 2012
  Brazilian Aeronautics Institute of Technology - ITA
- MSc (ITA, 2007) and BSc in Computer Engineering (UMC, 2004)

Industry experience as development engineer (*e.g.*, Embraer 2011-2014)

Research Interests:

- Reconfigurable computing
- Embedded systems
- Real time systems
- Real time operating systems
- Model-based design

About Denis Loubach

Assistant Professor at UNICAMP, since 2014

Background

- PhD in Electronics and Computer Engineering, ITA, 2012
  Brazilian Aeronautics Institute of Technology - ITA
- MSc (ITA, 2007) and BSc in Computer Engineering (UMC, 2004)

Industry experience as development engineer (*e.g.*, Embraer 2011-2014)

Research Interests:

- Reconfigurable computing
- Embedded systems
- Real time systems
- Real time operating systems
- Model-based design

About Denis Loubach

Assistant Professor at UNICAMP, since 2014

Background

- PhD in Electronics and Computer Engineering, ITA, 2012
  Brazilian Aeronautics Institute of Technology - ITA
- MSc (ITA, 2007) and BSc in Computer Engineering (UMC, 2004)

Industry experience as development engineer (*e.g.*, Embraer 2011-2014)

Research Interests:

- Reconfigurable computing
- Embedded systems
- Real time systems
- Real time operating systems
- Model-based design

About Denis Loubach

Assistant Professor at UNICAMP, since 2014

Background

- PhD in Electronics and Computer Engineering, ITA, 2012
  Brazilian Aeronautics Institute of Technology - ITA
- MSc (ITA, 2007) and BSc in Computer Engineering (UMC, 2004)

Industry experience as development engineer (*e.g.*, Embraer 2011-2014)

Research Interests:

- Reconfigurable computing
- Embedded systems
- Real time systems
- Real time operating systems
- Model-based design

Contact

Email: `dloubach@fem.unicamp.br`

Academic page: `http://www.fem.unicamp.br/~dloubach/`

Lab page: `http://www.fem.unicamp.br/~acceslab/`

Introduction

We introduce in this paper a **runtime reconfiguration design** that can be applied to avionics systems, mainly concerning **performance** and **power consumption** aspects

We implemented the design by using a **heterogeneous SoC**[1] with two major areas in the hardware architecture: a **hard processor area** and a **reconfigurable area**

The former has the authority to manage which configuration will be programmed/used in a given time. The developed design enables both **full** reconfiguration (FR) and **partial reconfiguration** (PR)

[1] System on chip

Introduction

We introduce in this paper a **runtime reconfiguration design** that can be applied to avionics systems, mainly concerning **performance** and **power consumption** aspects

We implemented the design by using a **heterogeneous SoC**[1] with two major areas in the hardware architecture: a **hard processor area** and a **reconfigurable area**

The former has the authority to manage which configuration will be programmed/used in a given time. The developed design enables both **full** reconfiguration (FR) and **partial reconfiguration** (PR)

[1] System on chip

Introduction

We introduce in this paper a **runtime reconfiguration design** that can be applied to avionics systems, mainly concerning **performance** and **power consumption** aspects

We implemented the design by using a **heterogeneous SoC**[1] with two major areas in the hardware architecture: a **hard processor area** and a **reconfigurable area**

The former has the authority to manage which configuration will be programmed/used in a given time. The developed design enables both **full** reconfiguration (FR) and **partial reconfiguration** (PR)

[1] System on chip

Background

Reconfigurable computing

- Programmable *vs.* reconfigurable device
- Full *vs.* partial reconfiguration
- Partition *vs.* static area

FPGA[2] compared to GPP[3] and ASIC[4]

Main reconfiguration techniques

- Runtime *vs.* dynamic reconfiguration

Configuration bitstream modes

- And/or mode
- Scrub mode
- Scrub clear/set mode

[2] Field-programmable gate array

[3] General purpose processors

[4] Application-specific integrated circuit

Background

Reconfigurable computing

- Programmable *vs.* reconfigurable device
- Full *vs.* partial reconfiguration
- Partition *vs.* static area

FPGA[2] compared to GPP[3] and ASIC[4]

Main reconfiguration techniques

- Runtime *vs.* dynamic reconfiguration

Configuration bitstream modes

- And/or mode
- Scrub mode
- Scrub clear/set mode

[2] Field-programmable gate array

[3] General purpose processors

[4] Application-specific integrated circuit

Background

Reconfigurable computing

- Programmable *vs.* reconfigurable device
- Full *vs.* partial reconfiguration
- Partition *vs.* static area

FPGA[2] compared to GPP[3] and ASIC[4]

Main reconfiguration techniques

- Runtime *vs.* dynamic reconfiguration

Configuration bitstream modes

- And/or mode
- Scrub mode
- Scrub clear/set mode

[2] Field-programmable gate array
[3] General purpose processors
[4] Application-specific integrated circuit

Background

Reconfigurable computing

- Programmable *vs.* reconfigurable device
- Full *vs.* partial reconfiguration
- Partition *vs.* static area

FPGA[2] compared to GPP[3] and ASIC[4]

Main reconfiguration techniques

- Runtime *vs.* dynamic reconfiguration

Configuration bitstream modes

- And/or mode
- Scrub mode
- Scrub clear/set mode

[2] Field-programmable gate array

[3] General purpose processors

[4] Application-specific integrated circuit

Background

Reconfigurable computing

- Programmable *vs.* reconfigurable device
- Full *vs.* partial reconfiguration
- Partition *vs.* static area

FPGA[2] compared to GPP[3] and ASIC[4]

Main reconfiguration techniques

- Runtime *vs.* dynamic reconfiguration

Configuration bitstream modes

- And/or mode
- Scrub mode
- Scrub clear/set mode

[2] Field-programmable gate array

[3] General purpose processors

[4] Application-specific integrated circuit

Background

Reconfigurable computing

- Programmable *vs.* reconfigurable device
- Full *vs.* partial reconfiguration
- Partition *vs.* static area

FPGA[2] compared to GPP[3] and ASIC[4]

Main reconfiguration techniques

- Runtime *vs.* dynamic reconfiguration

Configuration bitstream modes

- And/or mode
- Scrub mode
- Scrub clear/set mode

[2] Field-programmable gate array

[3] General purpose processors

[4] Application-specific integrated circuit

Proposed Runtime Reconfiguration Design

The fundamental principle is to only place a partition implementation when its functionality is about to be required, and remove it as soon as it is no longer needed

This way, a **same hardware area may be reused over time** considering temporal mutually-exclusive partition implementations (**minimize the wasting** of on-chip resources)

Each partition is able to have one or more different implementations. A partition implementation is named **`prosopon`** in our design

Therefore, it is possible to have a `prosopon` that is based on performance results, another one concerned with power consumption constraints

The developed design enables both FR and PR

Proposed Runtime Reconfiguration Design

The fundamental principle is to only place a partition implementation when its functionality is about to be required, and remove it as soon as it is no longer needed

This way, a **same hardware area may be reused over time** considering temporal mutually-exclusive partition implementations (**minimize the wasting** of on-chip resources)

Each partition is able to have one or more different implementations. A partition implementation is named `prosopon` in our design

Therefore, it is possible to have a `prosopon` that is based on performance results, another one concerned with power consumption constraints

The developed design enables both FR and PR

Proposed Runtime Reconfiguration Design

The fundamental principle is to only place a partition implementation when its functionality is about to be required, and remove it as soon as it is no longer needed

This way, a **same hardware area may be reused over time** considering temporal mutually-exclusive partition implementations (**minimize the wasting** of on-chip resources)

Each partition is able to have one or more different implementations. A partition implementation is named **`prosopon`** in our design

Therefore, it is possible to have a `prosopon` that is based on performance results, another one concerned with power consumption constraints

The developed design enables both FR and PR

Proposed Runtime Reconfiguration Design

The fundamental principle is to only place a partition implementation when its functionality is about to be required, and remove it as soon as it is no longer needed

This way, a **same hardware area may be reused over time** considering temporal mutually-exclusive partition implementations (**minimize the wasting** of on-chip resources)

Each partition is able to have one or more different implementations. A partition implementation is named **prosopon** in our design

Therefore, it is possible to have a prosopon that is based on performance results, another one concerned with power consumption constraints

The developed design enables both FR and PR

Proposed Runtime Reconfiguration Design

The fundamental principle is to only place a partition implementation when its functionality is about to be required, and remove it as soon as it is no longer needed

This way, a **same hardware area may be reused over time** considering temporal mutually-exclusive partition implementations (**minimize the wasting** of on-chip resources)

Each partition is able to have one or more different implementations. A partition implementation is named **`prosopon`** in our design

Therefore, it is possible to have a `prosopon` that is based on performance results, another one concerned with power consumption constraints

The developed design enables both FR and PR

cont...

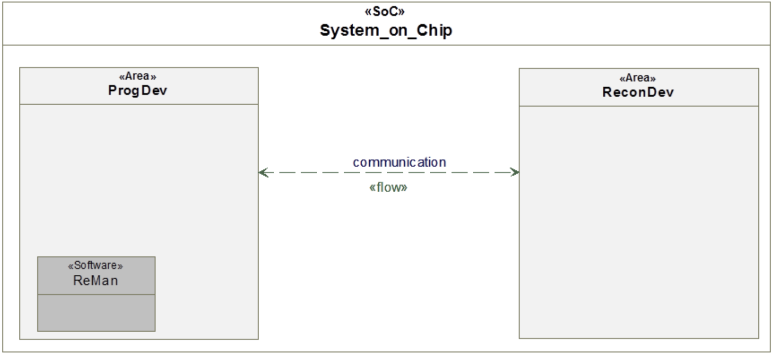

Figure: SoC with the ProgDev and ReconDev areas overview and also the ReMan software application inside the ProgDev

ProgDev[5], ReconDev[6], ReMan[7]

[5] Programmable device

[6] Reconfigurable device

[7] Reconfiguration manager

cont...

The main steps of our design are:

1. Implement the `ReconDev`
   1. Describe the hardware without partitions
   2. Identify the parts to be partitioned
   3. Define the logical area and physical area for the identified partitions
   4. Implement and test each `prosopon`
2. Generate the bitstream for FR and for each `prosopon`
3. Implement the `ReMan` in the `ProgDev`

cont...

The main steps of our design are:

1. Implement the `ReconDev`
    1. Describe the hardware without partitions
    2. Identify the parts to be partitioned
    3. Define the logical area and physical area for the identified partitions
    4. Implement and test each `prosopon`
2. Generate the bitstream for FR and for each `prosopon`
3. Implement the `ReMan` in the `ProgDev`

cont...

The main steps of our design are:

1. Implement the `ReconDev`
   1. Describe the hardware without partitions
   2. Identify the parts to be partitioned
   3. Define the logical area and physical area for the identified partitions
   4. Implement and test each `prosopon`
2. Generate the bitstream for FR and for each `prosopon`
3. Implement the `ReMan` in the `ProgDev`

cont...

The main steps of our design are:

1. Implement the `ReconDev`
   1. Describe the hardware without partitions
   2. Identify the parts to be partitioned
   3. Define the logical area and physical area for the identified partitions
   4. Implement and test each `prosopon`
2. Generate the bitstream for FR and for each `prosopon`
3. Implement the `ReMan` in the `ProgDev`

cont...

The main steps of our design are:

1. Implement the `ReconDev`
   1. Describe the hardware without partitions
   2. Identify the parts to be partitioned
   3. Define the logical area and physical area for the identified partitions
   4. Implement and test each `prosopon`
2. Generate the bitstream for FR and for each `prosopon`
3. Implement the `ReMan` in the `ProgDev`

cont...

The main steps of our design are:

1. Implement the `ReconDev`
    1. Describe the hardware without partitions
    2. Identify the parts to be partitioned
    3. Define the logical area and physical area for the identified partitions
    4. Implement and test each `prosopon`
2. Generate the bitstream for FR and for each `prosopon`
3. Implement the `ReMan` in the `ProgDev`

cont...

The main steps of our design are:

1. Implement the `ReconDev`
   1. Describe the hardware without partitions
   2. Identify the parts to be partitioned
   3. Define the logical area and physical area for the identified partitions
   4. Implement and test each `prosopon`
2. Generate the bitstream for FR and for each `prosopon`
3. Implement the `ReMan` in the `ProgDev`

Case Study

Implemented by using the Cyclone V SX SoC

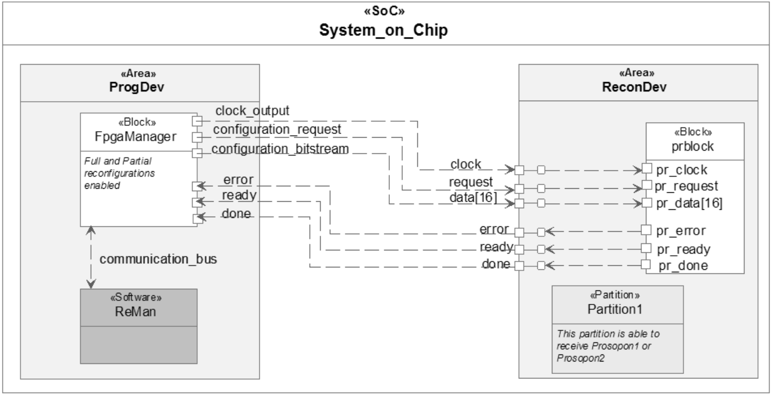

Figure: Implemented design overview showing the `ProgDev` and `ReconDev` areas, `ReMan` software part inside `ProgDev`, `prblock` and `Partition1` inside the `ReconDev` area, and the communication signals between `ProgDev` and `ReconDev`

Results

The average power consumption of `ProgDev` and `ReconDev` (regular operation)

Table: Average power consumption

Area	**Average Power [Watt]**
`ProgDev`	1.170
`ReconDev`	0.351

cont...

Table: Time to complete FR

Bitstream	Time [mili second]
FR bitstream with DC	32.40
FR bitstream without DC	29.33

Table: Average power for FR

Bitstream	Average Power [mili Watt]
FR bitstream with DC	170.03
FR bitstream without DC	42.29

DC stands for data compression

cont...

Table: Time to complete FR

Bitstream	Time [mili second]
FR bitstream with DC	32.40
FR bitstream without DC	29.33

Table: Average power for FR

Bitstream	Average Power [mili Watt]
FR bitstream with DC	170.03
FR bitstream without DC	42.29

DC stands for data compression

cont...

Table: Time to complete PR

Prosopon	Time [mili second]
Prosopon1 - AO	2.73
Prosopon1 - S	2.23
Prosopon1 - SCS	4.29
Prosopon2 - AO	2.76
Prosopon2 - S	2.22
Prosopon2 - SCS	4.28

Table: Average power for PR

Prosopon	Average Power [mili Watt]
Prosopon1 - AO	22.23
Prosopon1 - S	30.89
Prosopon1 - SCS	75.63
Prosopon2 - AO	19.05
Prosopon2 - S	40.41
Prosopon2 - SCS	73.32

cont...

Table: Time to complete PR

Prosopon	Time [mili second]
Prosopon1 - AO	2.73
Prosopon1 - S	2.23
Prosopon1 - SCS	4.29
Prosopon2 - AO	2.76
Prosopon2 - S	2.22
Prosopon2 - SCS	4.28

Table: Average power for PR

Prosopon	Average Power [mili Watt]
Prosopon1 - AO	22.23
Prosopon1 - S	30.89
Prosopon1 - SCS	75.63
Prosopon2 - AO	19.05
Prosopon2 - S	40.41
Prosopon2 - SCS	73.32

cont...

Labels related to partial reconfiguration (PR) results:

- **AO** stands for and/or mode
- **S** stands for scrub mode
- **SCS** stands for scrub clear/set mode

Conclusion

We introduced a **runtime reconfiguration design** mainly concerning **performance** and **power consumption** aspects, which are considered key points to observe in real-time embedded systems design

Also, according to the literature, **reconfiguration capabilities** are pointed out as a **present challenge for next generation of avionics** architectures

Our design addressed those aspects and we showed through a case study that is possible to perform both **full and partial runtime reconfiguration** in tens of mili seconds. The average power spent in the reconfiguration was tens of mili Watt

Based on our results, we consider that our design is suitable to be used in **new generations of avionics systems** that will take into account hardware runtime reconfiguration

Conclusion

We introduced a **runtime reconfiguration design** mainly concerning **performance** and **power consumption** aspects, which are considered key points to observe in real-time embedded systems design

Also, according to the literature, **reconfiguration capabilities** are pointed out as a **present challenge for next generation of avionics** architectures

Our design addressed those aspects and we showed through a case study that is possible to perform both **full and partial runtime reconfiguration** in tens of mili seconds. The average power spent in the reconfiguration was tens of mili Watt

Based on our results, we consider that our design is suitable to be used in **new generations of avionics systems** that will take into account hardware runtime reconfiguration

Conclusion

We introduced a **runtime reconfiguration design** mainly concerning **performance** and **power consumption** aspects, which are considered key points to observe in real-time embedded systems design

Also, according to the literature, **reconfiguration capabilities** are pointed out as a **present challenge for next generation of avionics** architectures

Our design addressed those aspects and we showed through a case study that is possible to perform both **full and partial runtime reconfiguration** in tens of mili seconds. The average power spent in the reconfiguration was tens of mili Watt

Based on our results, we consider that our design is suitable to be used in **new generations of avionics systems** that will take into account hardware runtime reconfiguration

Conclusion

We introduced a **runtime reconfiguration design** mainly concerning **performance** and **power consumption** aspects, which are considered key points to observe in real-time embedded systems design

Also, according to the literature, **reconfiguration capabilities** are pointed out as a **present challenge for next generation of avionics** architectures

Our design addressed those aspects and we showed through a case study that is possible to perform both **full and partial runtime reconfiguration** in tens of mili seconds. The average power spent in the reconfiguration was tens of mili Watt

Based on our results, we consider that our design is suitable to be used in **new generations of avionics systems** that will take into account hardware runtime reconfiguration

cont...

Other highlights: one may achieve a **reduced circuit area** since just the functionality/hardware in use will take place in a given time frame (contributes for the minimization of SEU susceptibility, once the circuit area will be smaller)

Considering that FR generally takes more time to complete than PR, one may **speed up the system start** by just placing the essential parts in FR and then placing the other functionality after with PR

On-chip **resources wasting minimization** and the trade-off between performance and power consumption may also be handled in our design

Finally, our proposal opens up the possibility to **optimize a system towards adaptability** to comply with runtime constraints changing regarding performance and power consumption

cont...

Other highlights: one may achieve a **reduced circuit area** since just the functionality/hardware in use will take place in a given time frame (contributes for the minimization of SEU susceptibility, once the circuit area will be smaller)

Considering that FR generally takes more time to complete than PR, one may **speed up the system start** by just placing the essential parts in FR and then placing the other functionality after with PR

On-chip **resources wasting minimization** and the trade-off between performance and power consumption may also be handled in our design

Finally, our proposal opens up the possibility to **optimize a system towards adaptability** to comply with runtime constraints changing regarding performance and power consumption

cont...

Other highlights: one may achieve a **reduced circuit area** since just the functionality/hardware in use will take place in a given time frame (contributes for the minimization of SEU susceptibility, once the circuit area will be smaller)

Considering that FR generally takes more time to complete than PR, one may **speed up the system start** by just placing the essential parts in FR and then placing the other functionality after with PR

On-chip **resources wasting minimization** and the trade-off between performance and power consumption may also be handled in our design

Finally, our proposal opens up the possibility to **optimize a system towards adaptability** to comply with runtime constraints changing regarding performance and power consumption

cont...

Other highlights: one may achieve a **reduced circuit area** since just the functionality/hardware in use will take place in a given time frame (contributes for the minimization of SEU susceptibility, once the circuit area will be smaller)

Considering that FR generally takes more time to complete than PR, one may **speed up the system start** by just placing the essential parts in FR and then placing the other functionality after with PR

On-chip **resources wasting minimization** and the trade-off between performance and power consumption may also be handled in our design

Finally, our proposal opens up the possibility to **optimize a system towards adaptability** to comply with runtime constraints changing regarding performance and power consumption

Acknowledgments

This research work is supported by:

- Regular Research Awards grant #2014/24855-8
  São Paulo Research Foundation - FAPESP
- Altera University Program (partially)

Thank you for your attention!

Questions & Answers