A Study Into the Modelling and Analysis of Real-Time FPGA Based Systems

Irfan F. Mir
(PhD Student)

Supervisor:
Dr Alistair A. McEwan

University of Leicester

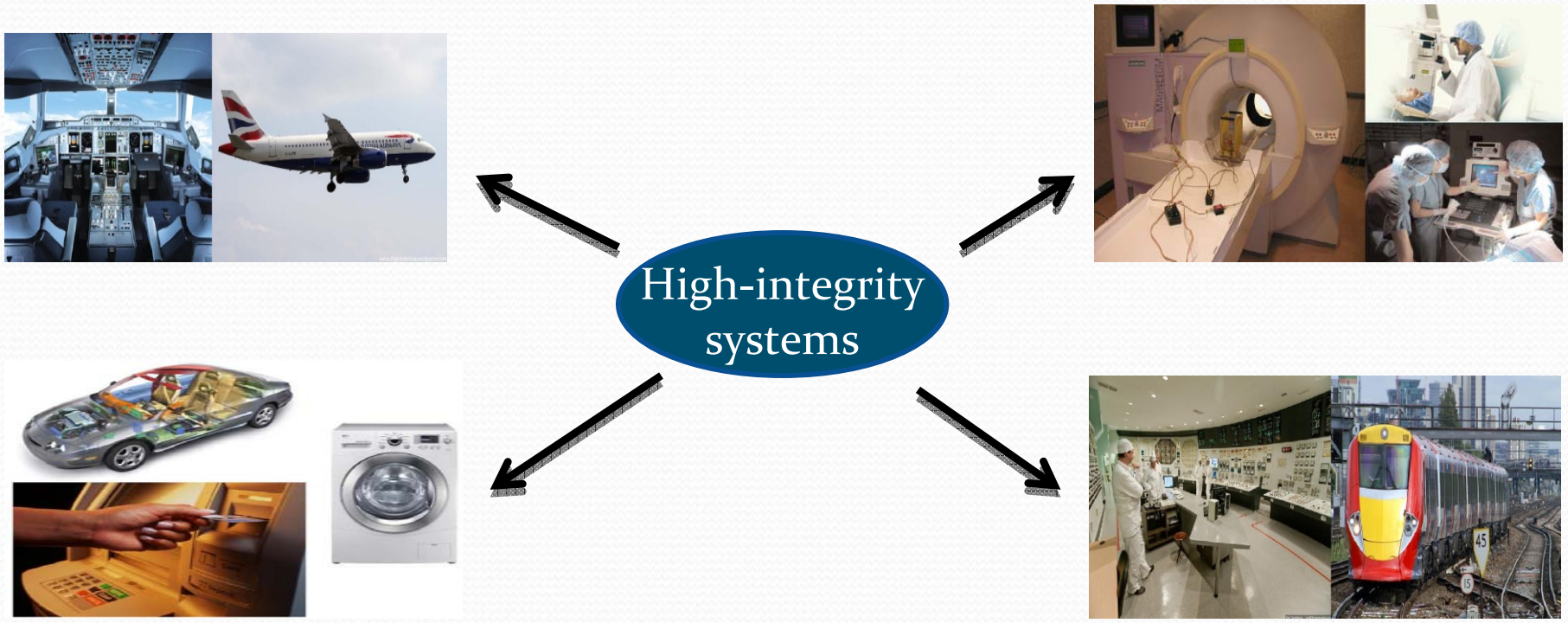
Embedded Systems Laboratory
(Applied Formal Methods)

Contents

- ***Motivation, aims, and scope***
- Formal techniques for high-integrity (FPGA) systems
- Real-time constraints in high level languages
- Embedding real-time constraints in Handel-C
- Case study – digital clock
- Conclusions and future work

Motivation

- High-integrity systems – detailed understanding of behaviours *and misbehaviours*!



- We need verification techniques that ensure the reliability and understanding of these classes of systems

Aims and scope

- Aims
  - To develop techniques and a tool for verifying real-time constraints in high level languages for high-integrity systems
  - To propose a novel methodology using "**Timed CSP**" to ensure the temporal correctness of these systems

- Scope
  - FPGA-based high-integrity systems that may have soft or hard real-time constraints
  - Handel-C is used as a high level language for FPGA design

Contents

- About me
- Motivation, aims, and scope
- ***Formal techniques for high-integrity (FPGA) systems***
- Real-time constraints in high level languages
- Embedding real-time constraints in Handel-C
- Case study – digital clock
- Conclusions and future work

Formal techniques for high-integrity (FPGA) systems

- Mathematical modelling, applicable to all stages of systems development, for instance:
  - **CSP:** Communicating Sequential Processes
  - **ACL2:** Application Common Lisp, a computational logic
  - **Esterel:** Synchronous reactive programming
  - **HyTech:** Hybrid technology – an automatic tool for analysis of embedded systems
- CSP has been practically used in many industrial applications
- Timed CSP verifies timing as well as functional properties of the design, but Classic CSP does not!

Contents

- About me
- Motivation, aims, and scope
- Formal techniques for high-integrity (FPGA) systems
- ***Real-time constraints in high level languages***
- Embedding real-time constraints in Handel-C
- Case study – digital clock
- Conclusions and future work

Real-time constraints in high level languages

- High level languages for FPGAs
  - Handel-C, System-C, Mobius, Impuse-C, Streams-C, Ada95 and others…
  - No support for real-time constraints!
  - Ada95 is a language that has been used extensively in real-time systems
  - FPGAs are more suitable as compare to processors for real-time systems – no caches + predictable timing behaviour

Real-time constraints in high level languages

- Various methods have been proposed to add real-time constraints in high-level languages
- But… still there is no significant research into using Handel-C as a real-time language!
- Annotating real-time constraints in Handel-C may make it suitable for real-time systems.

Contents

- About me
- Motivation, aims, and scope
- Formal techniques for high-integrity (FPGA) systems
- Real-time constraints in high level languages
- ***Embedding real-time constraints in Handel-C***
- Case study – digital clock
- Conclusions and future work

Embedding real-time constraints in Handel-C

- Handel-C – High level language for FPGAs
  - Hybrid of CSP and C languages, designed to target FPGAs
  - Fully synchronous – each statement executes in one Handel-C clock cycle
    - So timing can be calculated by counting statements, but…
    - This is not a complete real-time analysis.
  - No explicit time constructs in Handel-C, but…
  - We can follow designs real-time constraints!

Embedding real-time constraints in Handel-C

- Meta-language style annotation
- Locate the code blocks for RT constraints
- Describe constraints in meta-language annotations
- Non-intrusive effect on source

- Real-time Preprocessor (RTCpreprocessor)
  - Development of a real-time pre-processor for Handel-C meta-language (*future work*…)

Embedding real-time constraints in Handel-C

Digital Clock

(Handel-C ver.1)

```
// @:- NET "ovflw_1sec" TIMESPEC = 1000 ms AFTER "Master_Enb" ## RTC-1
Cnt_sec (Master_Enb, PAL_ACTUAL_CLOCK_RATE, ovflw_1sec);

// @:- NET "ovflow_10sec" TIMESPEC = 10000 ms AFTER 10 "ovflw_1sec" ## RTC-2
Counter (ovflw_1sec, Master_Rst1, 0, 10, ovflow_10sec, sec_lo);
// @:- NET "ovflow_1min" TIMESPEC = 60000 ms AFTER 6 "ovflow_10sec" ## RTC-3
Counter (ovflow_10sec, Master_Rst2, 1, 6, ovflow_1min, sec_hi);

// @:- NET "ovflow_10min" TIMESPEC = 600000 ms AFTER 10 "ovflow_1min" ## RTC-4
Counter (ovflow_1min, Master_Rst3, 2, 10, ovflow_10min, min_lo);
// @:- NET "ovflow_1hrs" TIMESPEC = 3600000 ms AFTER 6 "ovflow_10min" ## RTC-5
Counter (ovflow_10min, Master_Rst4, 3, 6, ovflow_1hrs, min_hi);

// @:- NET "ovflow_10hrs" TIMESPEC = 36000000 ms AFTER 10 "ovflow_1hrs" ## RTC-6
Counter (ovflow_1hrs, Master_Rst5, 4, 10, ovflow_10hrs, hrs_lo);
Counter (ovflow_10hrs, Master_Rst6, 5, 6, ovflow_24hrs, hrs_hi);
```

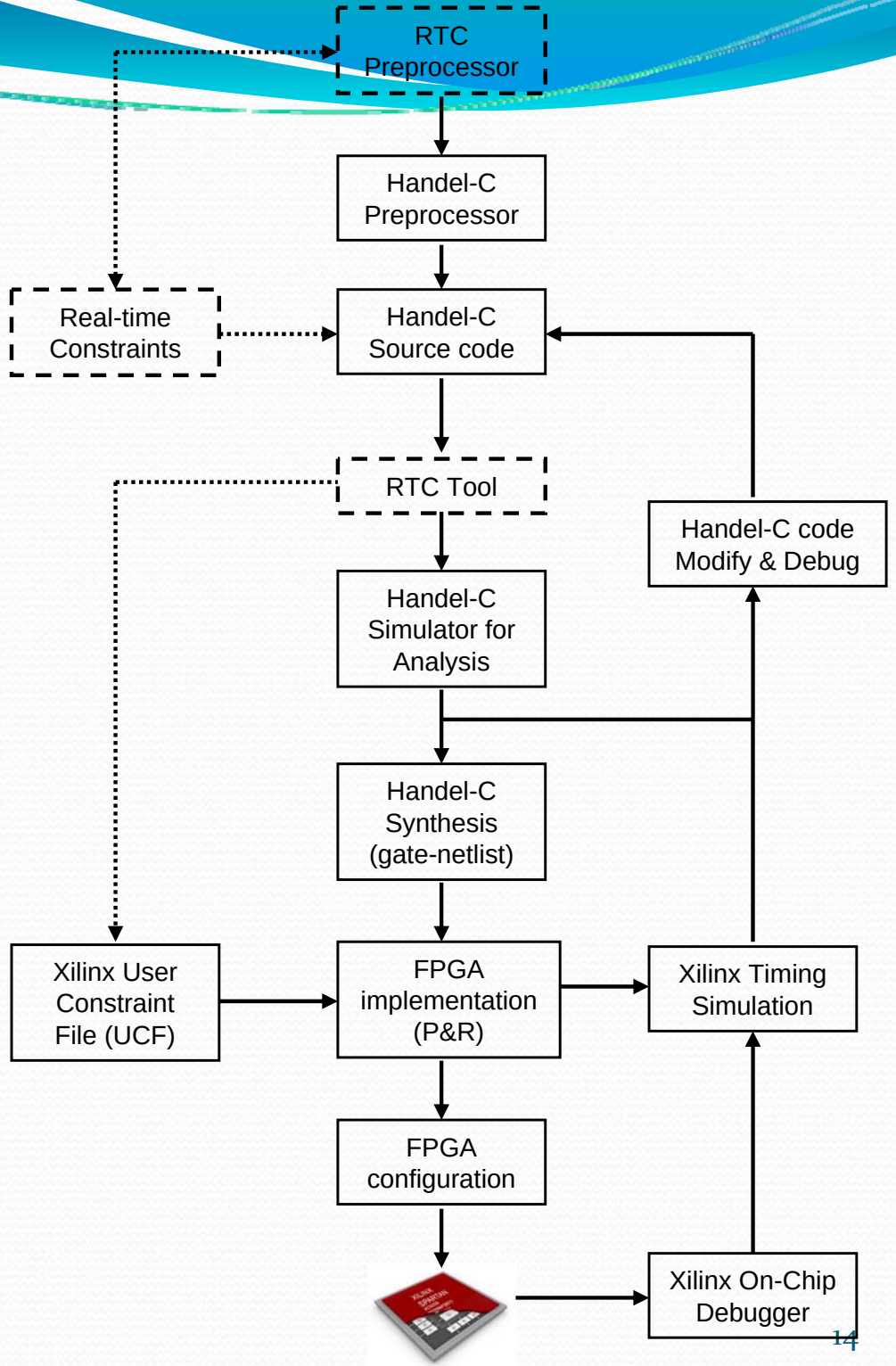
Digital Clock

(Handel-C ver.2)

```
milli_counter (Master_Enb, count_ch1_msec, count_ch_msec, msec_cnt);

// @:- BUS "sec_cnt" n:{1to59} TIMESPEC = 1000 ms AFTER "Master_Enb" ## RTC-1
counter1 (msec_cnt, count_ch_sec, sec_cnt);

// @:- BUS "min_cnt" n:{1to59} TIMESPEC = 60000 ms AFTER "Master_Enb" ## RTC-2
counter2 (count_ch_msec, sec_cnt, min_cnt);

// @:- BUS "hrs_cnt" n:{1to23} TIMESPEC = 3600000 ms AFTER "Master_Enb" ## RTC-3
counter3 (count_ch1_msec, count_ch_sec, min_cnt, hrs_cnt);
```

Design flow for real-time Handel-C

- Design methodology
  - Annotated real-time constraints without changing the actual design timing
  - Add RTCpreprocessor that have real-time constraints' definitions
  - Analyse timing constraints using debugger of DK suite
  - Synthesis design with DK
  - Implement design with FPGA tool
  - Timing simulation with ModelSim
  - FPGA configuration

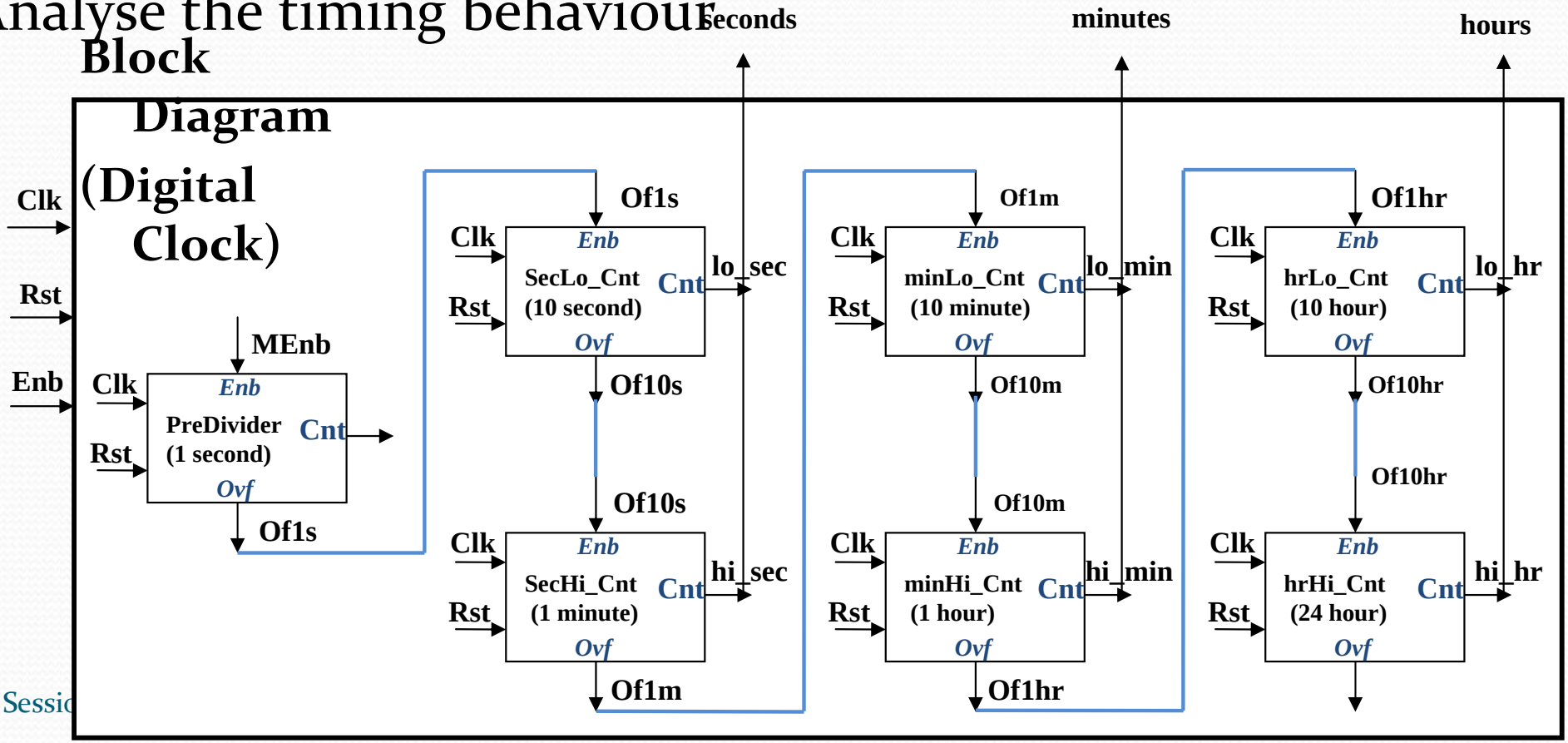


Contents

- About me
- Motivation, aims, and scope
- Formal techniques for high-integrity (FPGA) systems
- Real-time constraints in high level languages
- Embedding real-time constraints in Handel-C
- ***Case study – digital clock***
- Conclusions and future work

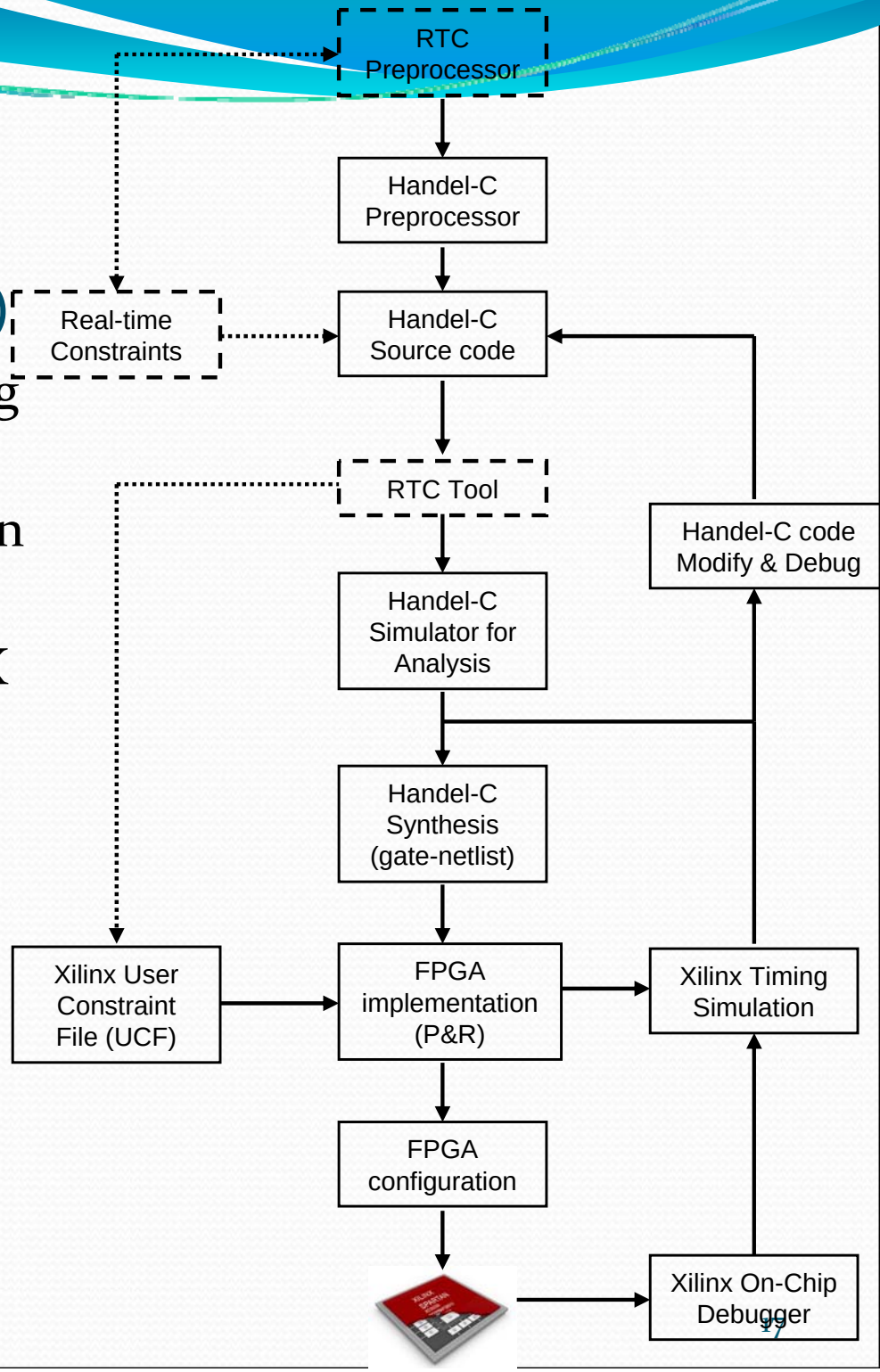
Case study – digital clock

- Digital clock is a simple real-time system
- Implementation in Handel-C using channel communication
- Analyse the timing behaviour



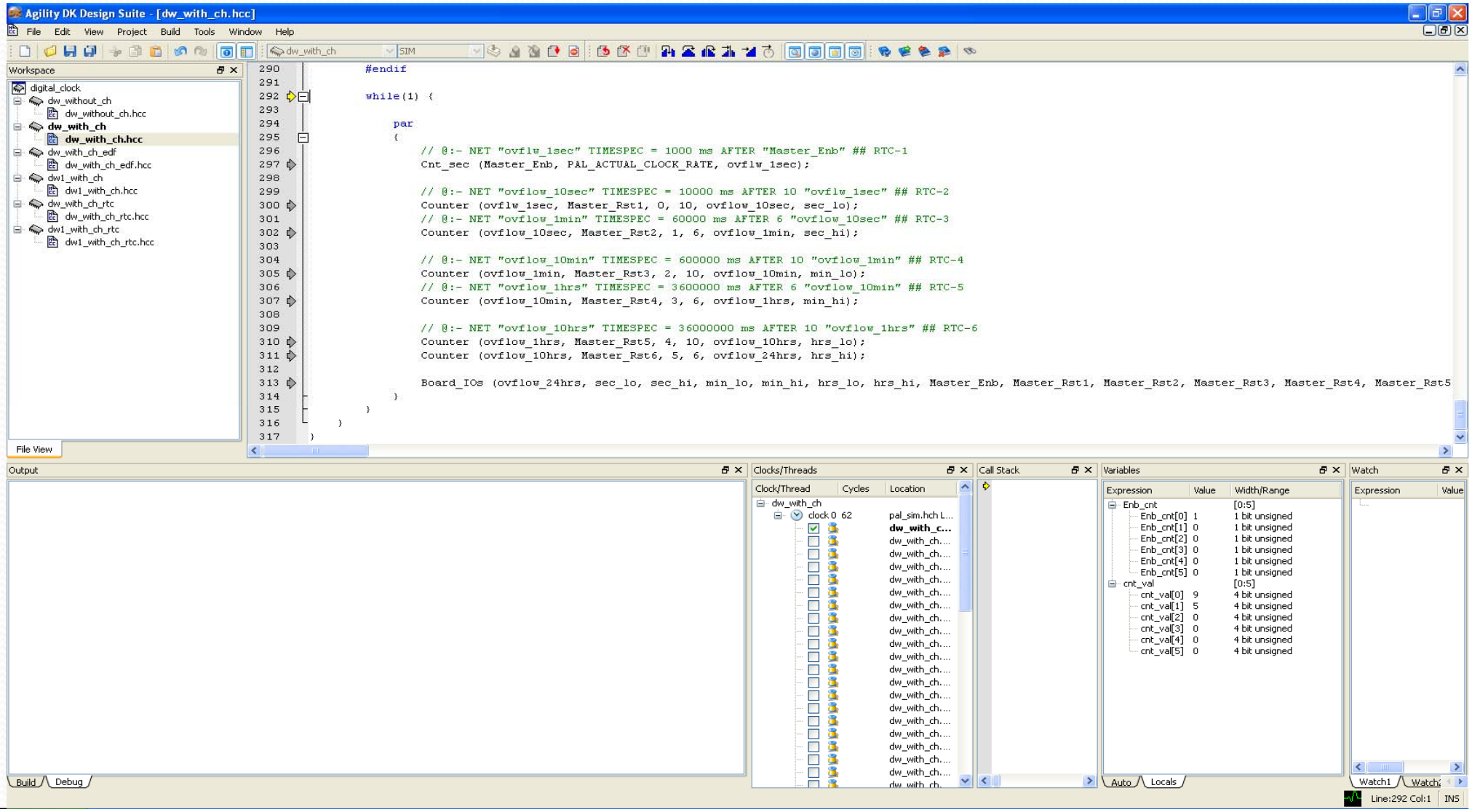
Design Flow for Digital Clock

- Phase 1: Design in Handel-C (HC)
  - Design digital clock in DK suite using channel communication
  - Embed real-time constraints (RTC) in HC code
  - Simulate and verify the RTC with DK debugger
- Phase 2: Synthesis & Implement
  - DK directly compile HC blocks to EDIF
  - Xilinx P&R tool for Sparatn-3A target platform
- Phase 3: Timing simulation
  - Simulate and verify the RTC of P&R design model with ModelSim



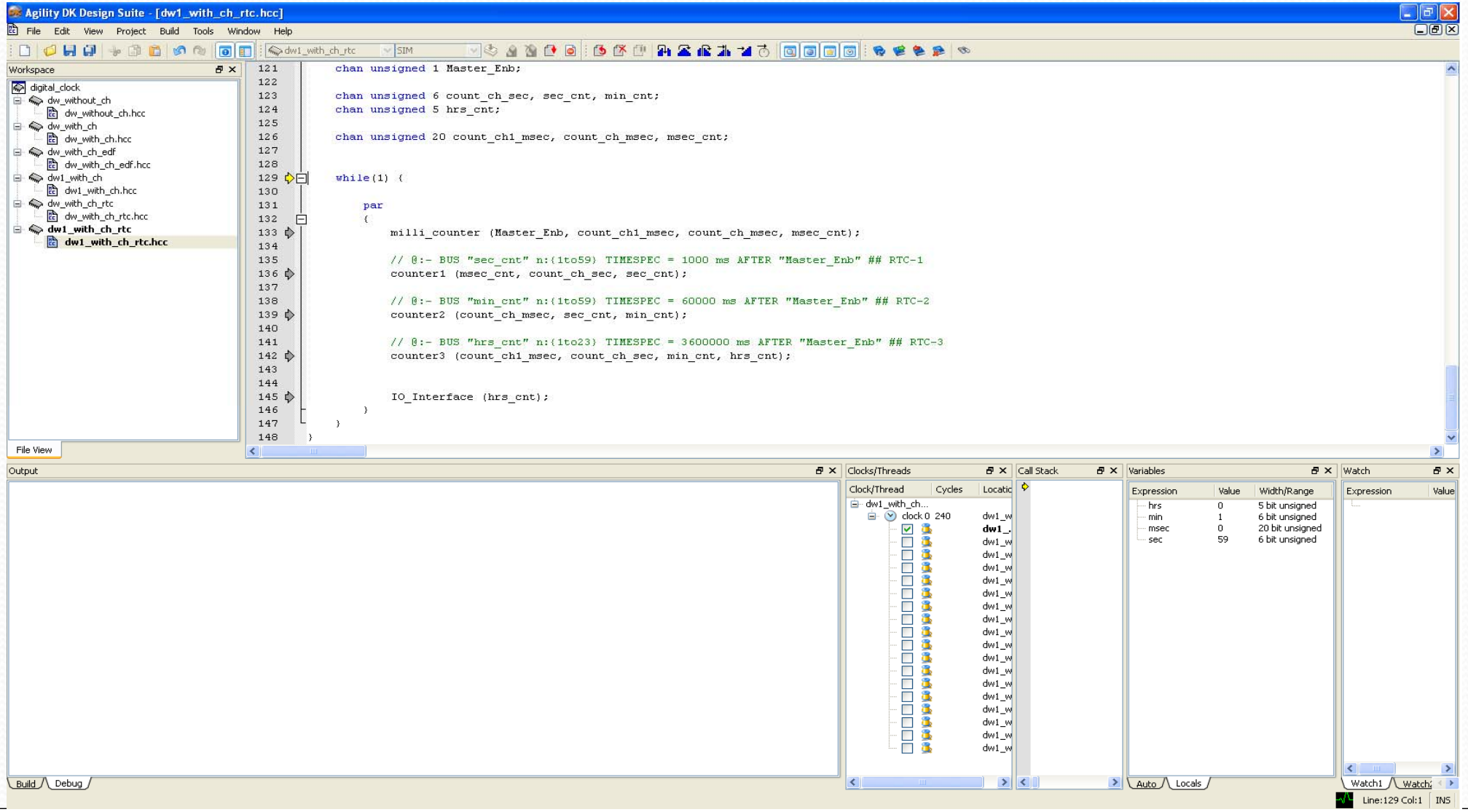
Digital Clock – Experiment

- Handel-C code – First version



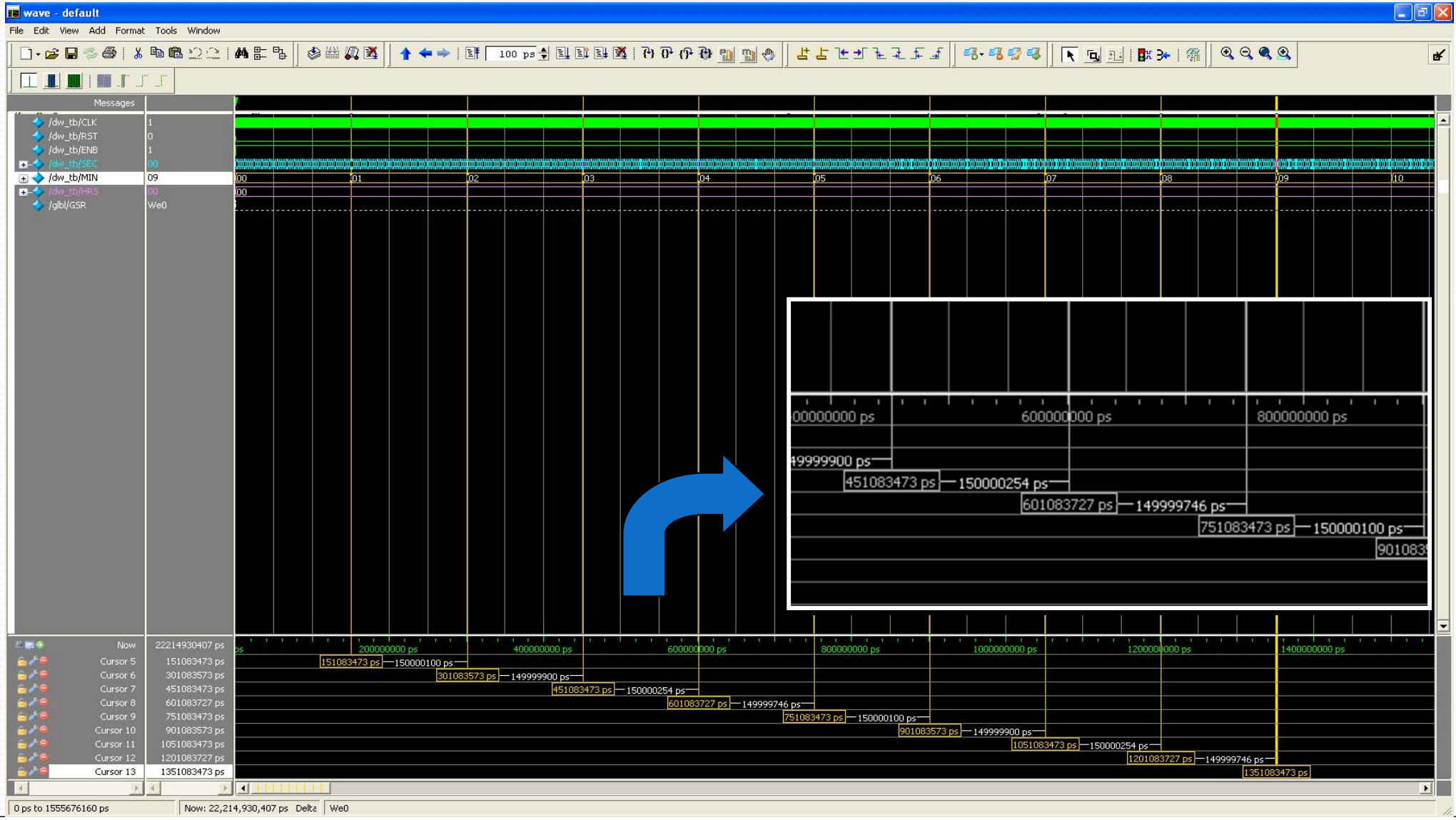
Digital Clock – Experiment

- Handel-C code – Second version



Digital Clock – Experiment

- Handel-C code – Timing simulation



Digital Clock – Case study results

- In the first version, timing analysis revealed a clock cycle drift on every tick of the digital clock.
- This means that the real-time constraints were not met!

- Timing analysis of the second version shows this clock cycle drift does not exist!
- This is a very subtle error that a constraint verifier could have revealed.

Contents

- About me
- Motivation, aims, and scope
- Formal techniques for high-integrity (FPGA) systems
- Real-time constraints in high level languages
- Embedding real-time constraints in Handel-C
- Case study – digital clock
- ***Conclusions and future work***

Conclusions and future work

- Conclusions
  - With suitable amendments, Handel-C can be used in some real-time high integrity system development
  - We propose a constraint meta-language and design flow to improve the timing analysis and verification of these systems

- Future work
  - Design the constraint meta-language and implement a tool which automates the analysis and verification process.
  - Investigate the implementation of Timed CSP in Handel-C, augmented with the constraint meta-language.


SLOW