PyGears: A Functional Approach to Hardware Design

Bogdan Vukobratović
Andrea Erdeljan
Damjan Rakanović

Contents

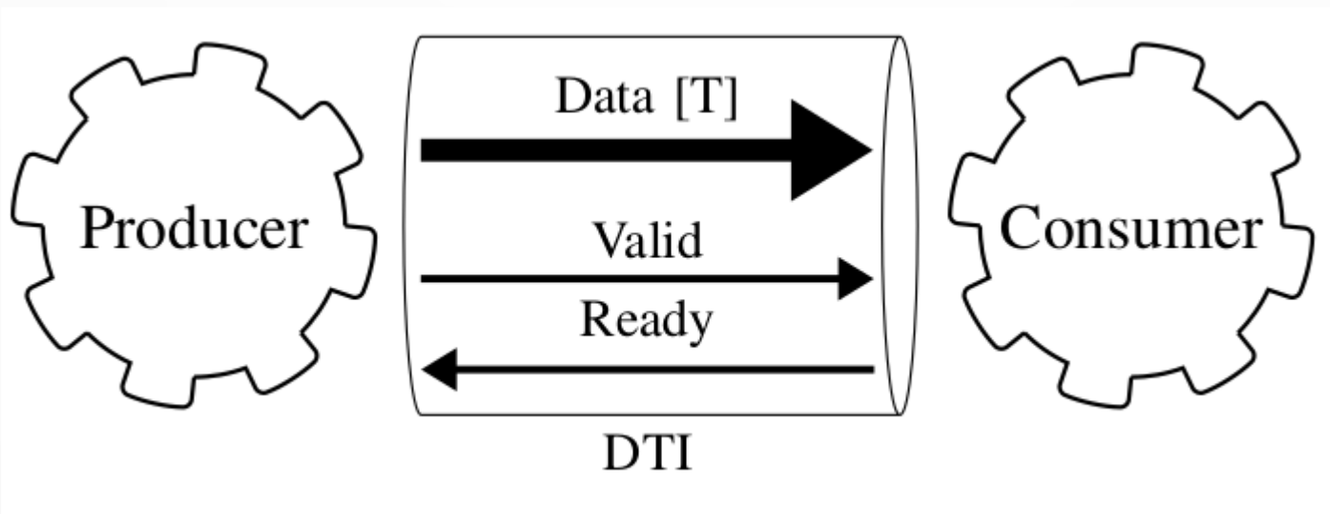
- Introduction
- Gears Methodology
- PyGears
- Results
- Future Work
- Conclusion

Introduction

- Motivation
  - Non-standard, ad hoc interfaces
  - Centralized control logic as complex FSMs
  - Difficulties with module synchronization
  - Reuse, reuse, reuse!
- Proposed solution: Gears methodology
- PyGears framework

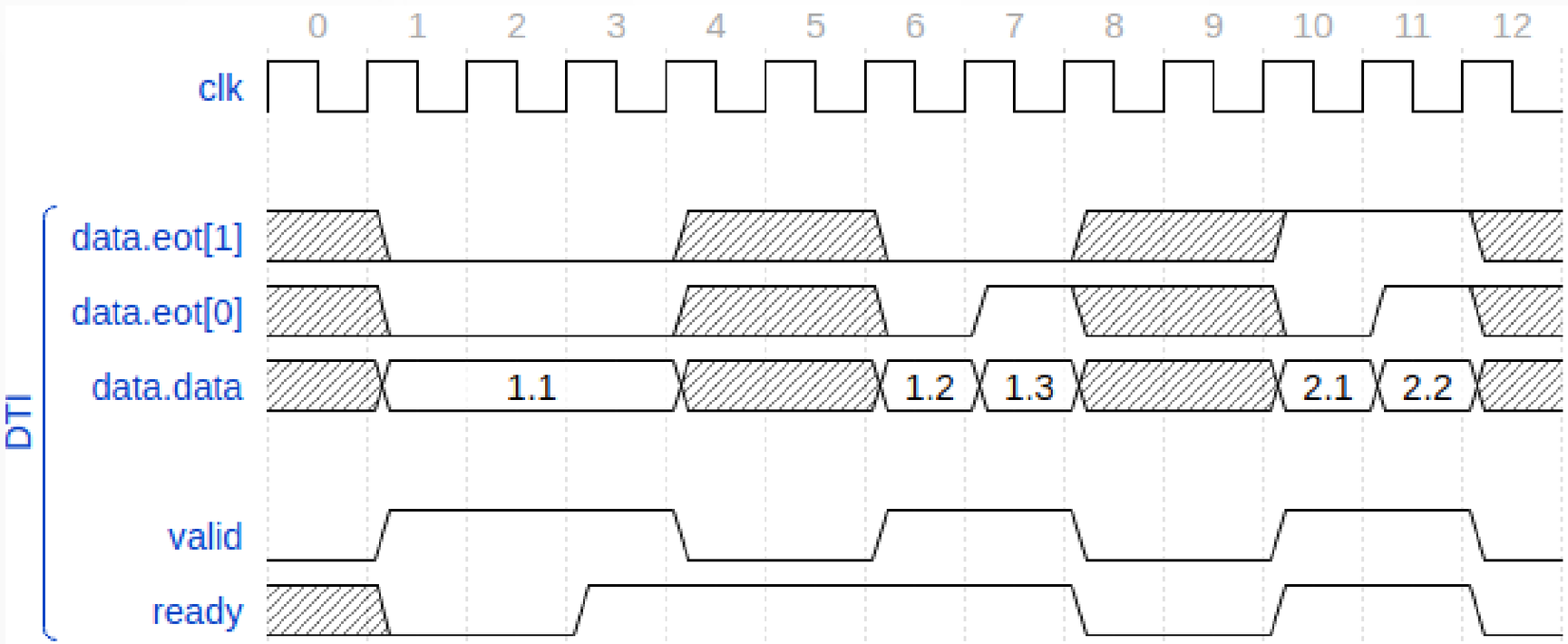
Gears Methodology - DTI

- Data Transfer Interface



Gears Methodology – Data Types

- Queue, Array, Tuple, Union, Uint, Int, Unit...



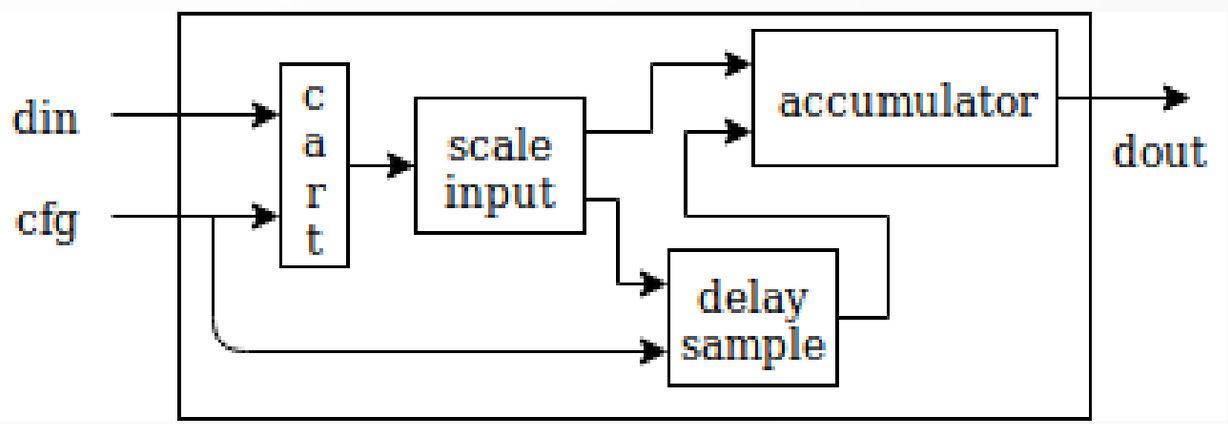

```
example_t = Queue[Uint[8], 2]
```

PyGears – Gear Definition

```
@gear
def moving_average(
    cfg: Tuple[{'avg_coef'   : Int['W'],
                'avr_window': Uint['W']}],
    din: Queue[Int['W']],
    *,
    shamt=15, max_filter_ord=1024):
```

PyGears – Gear Instantiation

```
@gear
def moving_average(...):
    scaled_sample = cart(cfg['avg_coef'], din) \
        | fmap(f=scale_input(shamt=shamt, W=W),
               fcat=czip)
    delayed_din = delay_sample(
        scaled_sample,
        cfg['avr_window']
        W=W,
        max_filter_ord=max_filter_ord)
    return accumulator(
               scaled_sample, delayed_din, W=W)
```



Results

- Parameterization: W=16 (16 bit samples)
- Xilinx's Vivado 2018.2 tool, for Zynq-7020

TABLE I
FPGA RESOURCES REQUIRED TO IMPLEMENT THE MOVING AVERAGE CORE

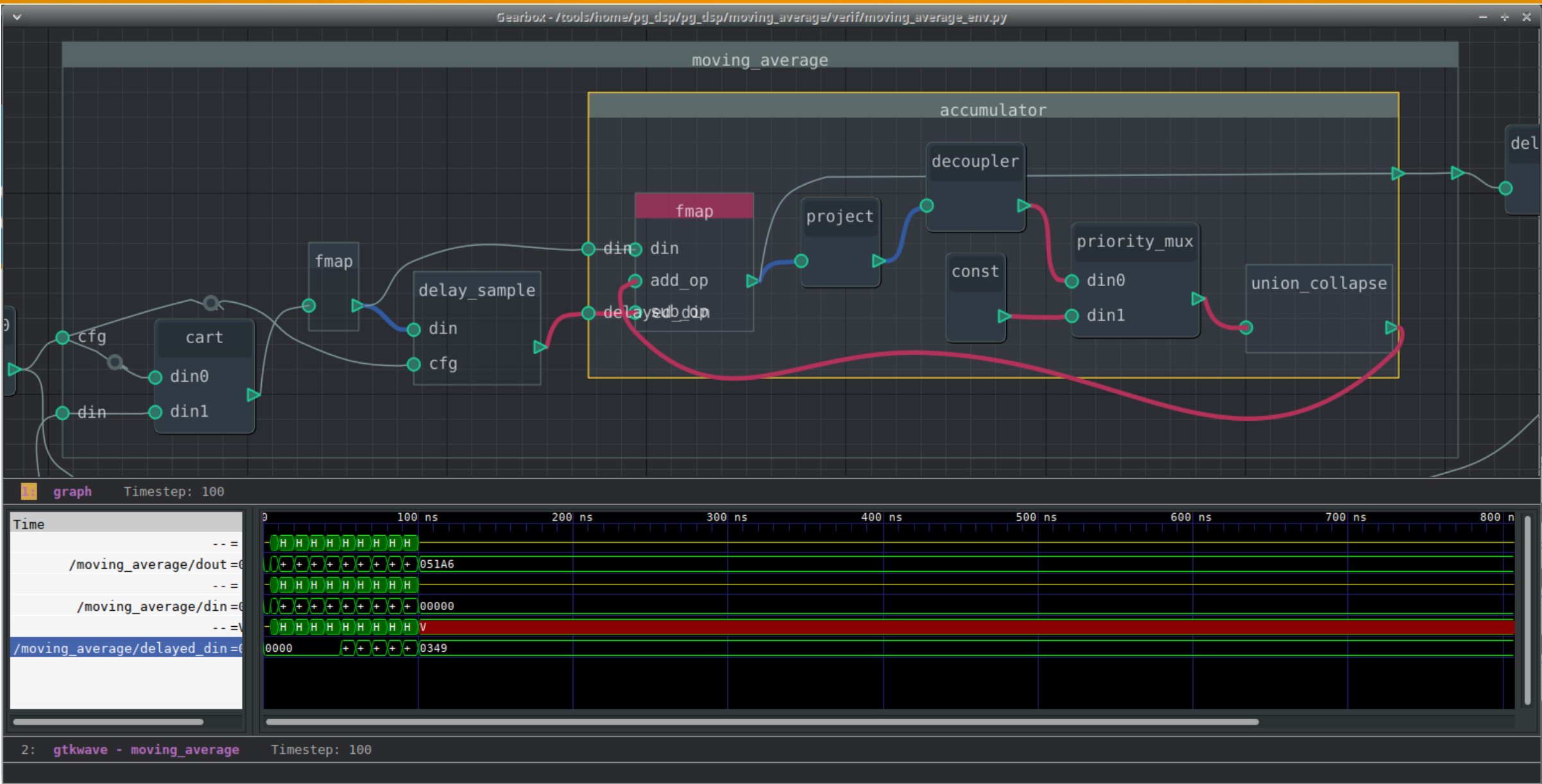
Implementation	LUTs	FFs	BRAMs	DSPs	Fmax [MHz]
PyGears	102	91	0.5	1	168.60
RTL	63	58	0.5	1	155.95
Vivado HLS	248	183	0.5	1	181.79

Ongoing work - HLS

- Abstract Python description compiled to RTL
  - Faster development
  - DTI protocol compliance
  - Improved quality of results
  - Accelerated functional verification
  - Formal verification using SymbiYosys

```
async def eq(din: Tuple[Any, Any]) -> Bool:
    async with din as data:
        yield data[0] == data[1]
```

Ongoing work - GUI



Thank you

bogdan.vukobratovic@gmail.com