Signalbehandling med ekstreme rater i FPGA

FPGA-forum 2008

Helge Rustad

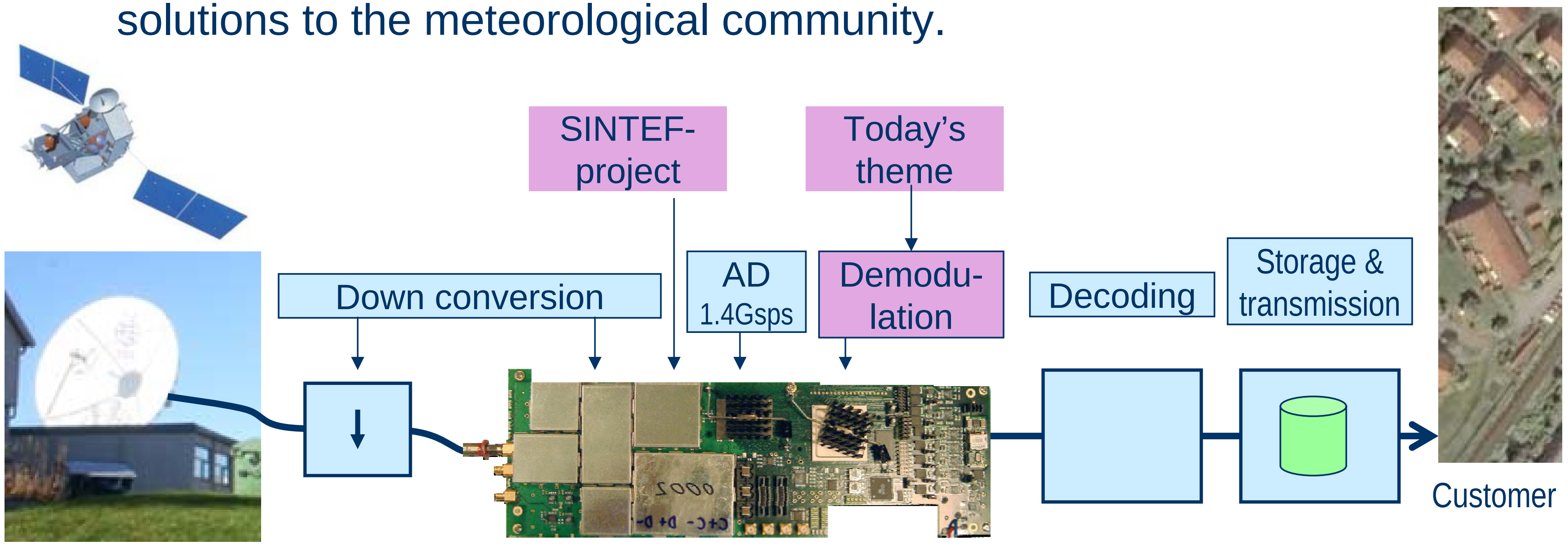
SINTEF IKT

SINTEF har utviklet en satellitt-demodulator for Kongsberg Spacetec. Denne arbeider med en innkommende sampelrate fra AD-konverter på 1,4 gigasampler/s og datarater opp til flere hundre Mbit/s. Med dette FPGA-designet som eksempel vil vi se på utfordringer og løsninger for implementasjon av filtre og andre signalbehandlings-elementer for meget høye datarater. Vi vil også se på løsninger som er brukt for testing av designet.

Project for Kongsberg Spacetec, 2005-2008

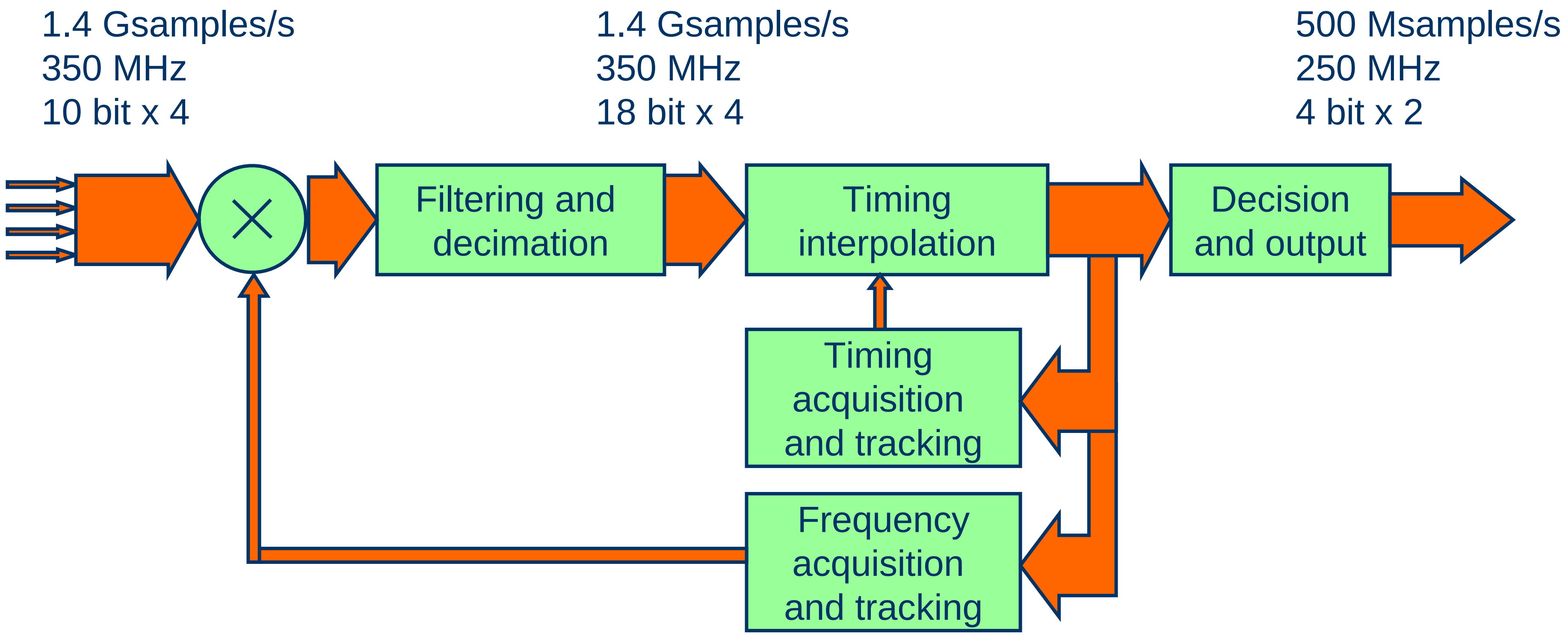
Kongsberg Spacetec

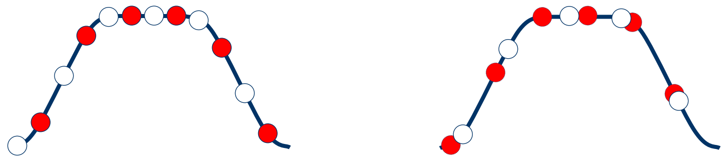
- Provider of Satellite Ground station systems for Earth observation satellites.
- Kongsberg Spacetec is the world leading provider of turnkey satellite ground stations. Our systems include solutions for the most known SAR and optical satellite sensors. We are also providing specialized solutions to the meteorological community.



- Introduction
- Very high rate filter design
- General experiences and solutions
- Test and verification

Very simplified block diagram





What is the difference between very high rate DSP design and other designs?

- High data rate (>1 sample per clock)
  - Parallelization, more hardware
  - Discard data (work on every n-th sample)
  - Modify algorithms
- High clock frequencies
  - Limits “freedom of design”
- High power consumption
  - Cooling becomes important

What are the design challenges

- Large and complex design with many different modules operating together
- Extreme data rates for signal processing
  - Four AD samples per clock cycle (1.4 Gs/s, 350 MHz)
  - Wide data buses
- High clock rates
  - Stressing the FPGA
- Large design and “slow” synchronization loops
  - VHDL simulation of real scenarios not realistic
- “Non-standard” design
  - Core-generator and IP modules not easy to use

FIR filter basics (Finite Impulse Response filter)

Viewed as an Equation

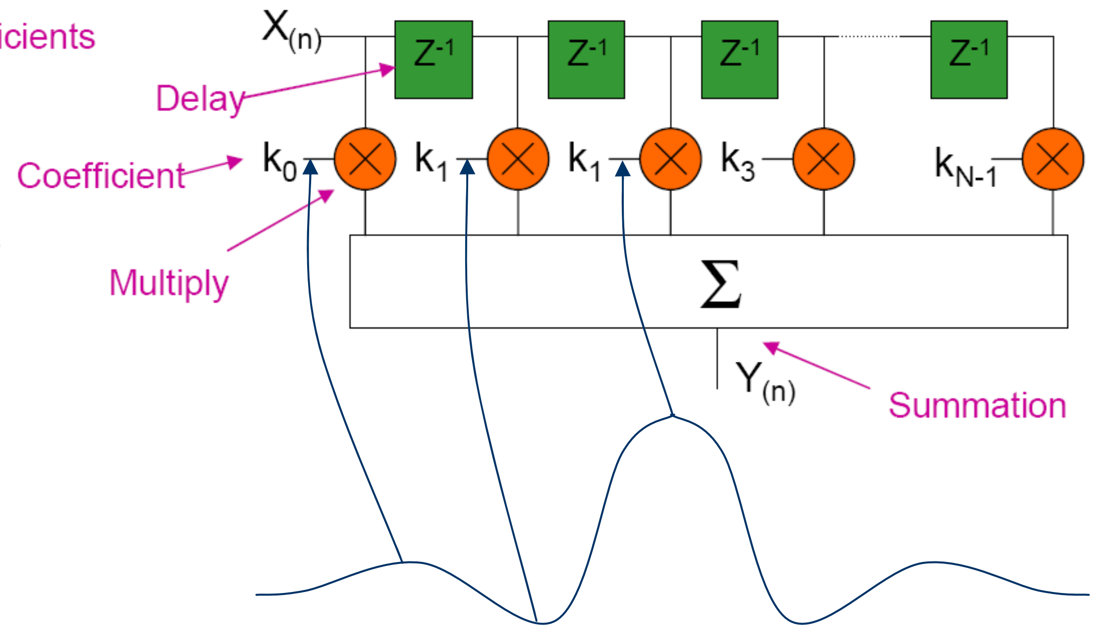
$$Y_{(n)} = \sum_{i=0}^{i=N-1} k_i . X_{(n-i)}$$

Coefficients

Accumulate N times

Multiply

Viewed as a Diagram

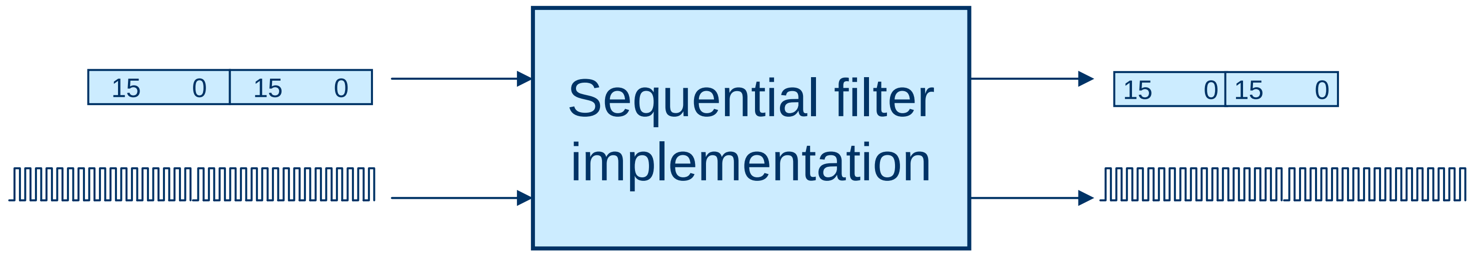


Filter impulse response

Figures and examples adapted from:

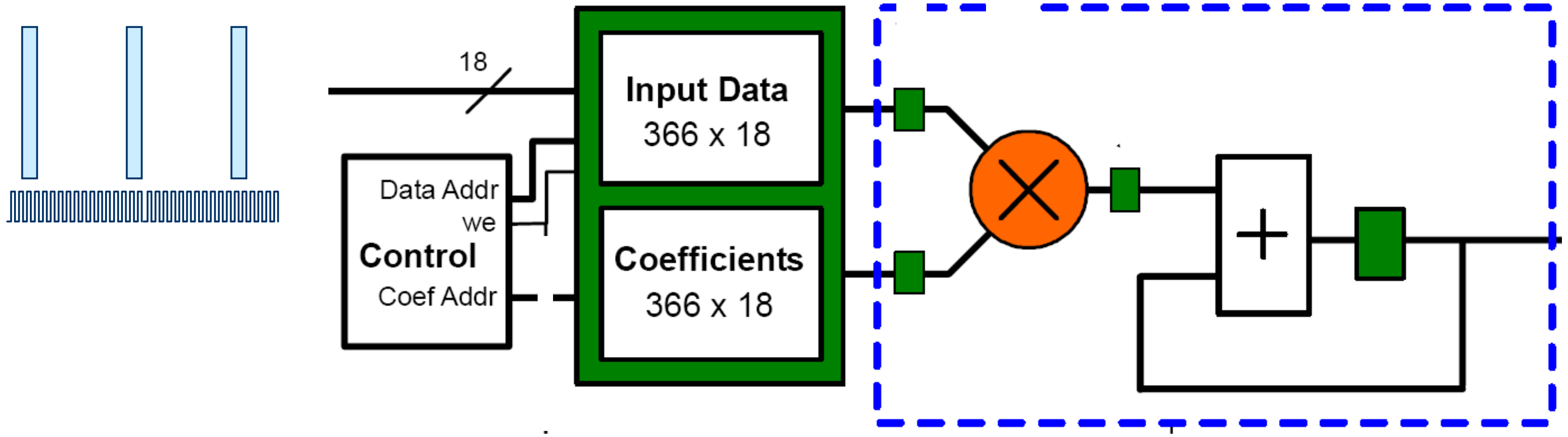
DSP Implementation Techniques,
Virtex-4 and Xtreme DSP Slice,
Niall Battson, www.xilinx.com/dsp

FIR filters for very low data rates



Optimal for low rates (<<= 1 bit per clock)

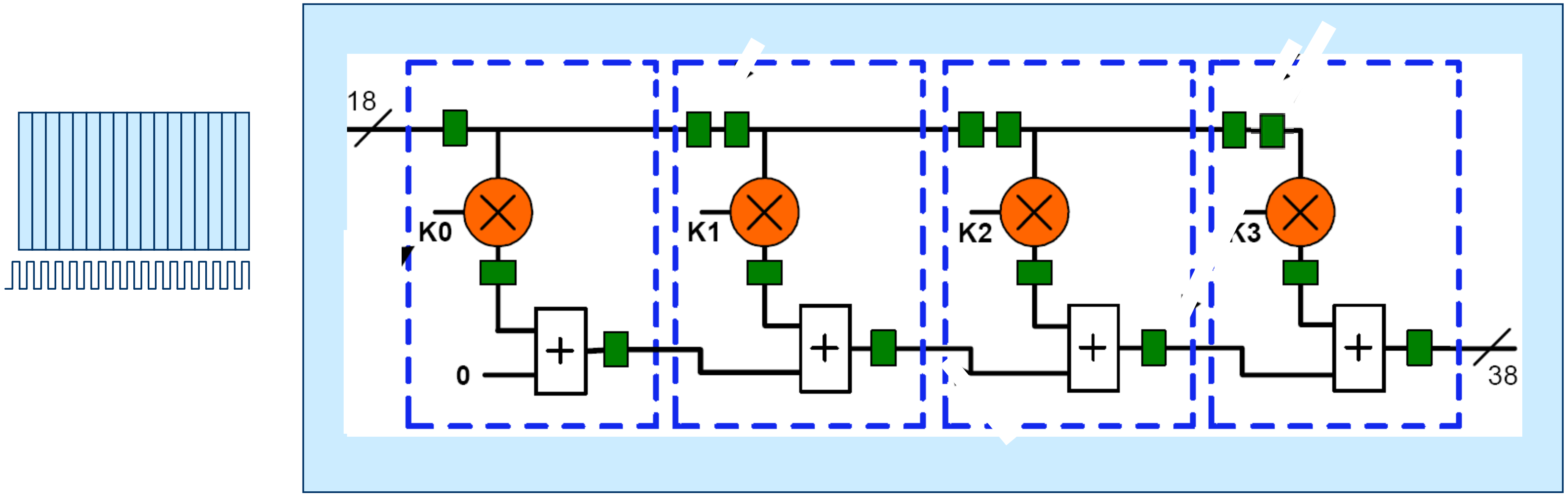
Medium data rates - Single Multiplier FIR filter



Reuse of multiplier and adder (Xilinx DSP module)

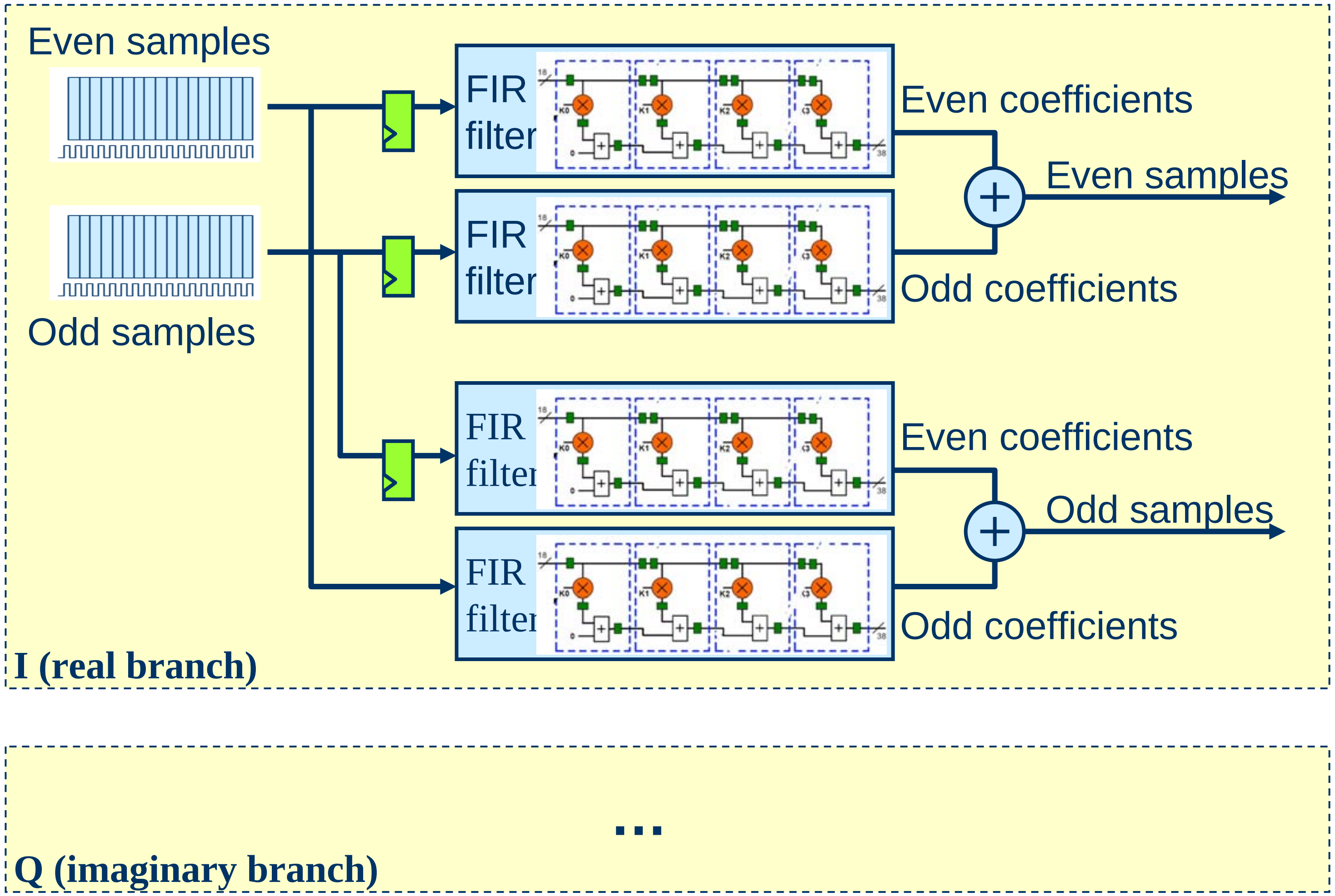
- Parallel data, many clocks per sample.
- Fits well with FPGA architectures with RAMs, ROMs and built-in multipliers and accumulators.

High data rates - Parallel FIR filter - “systolic”

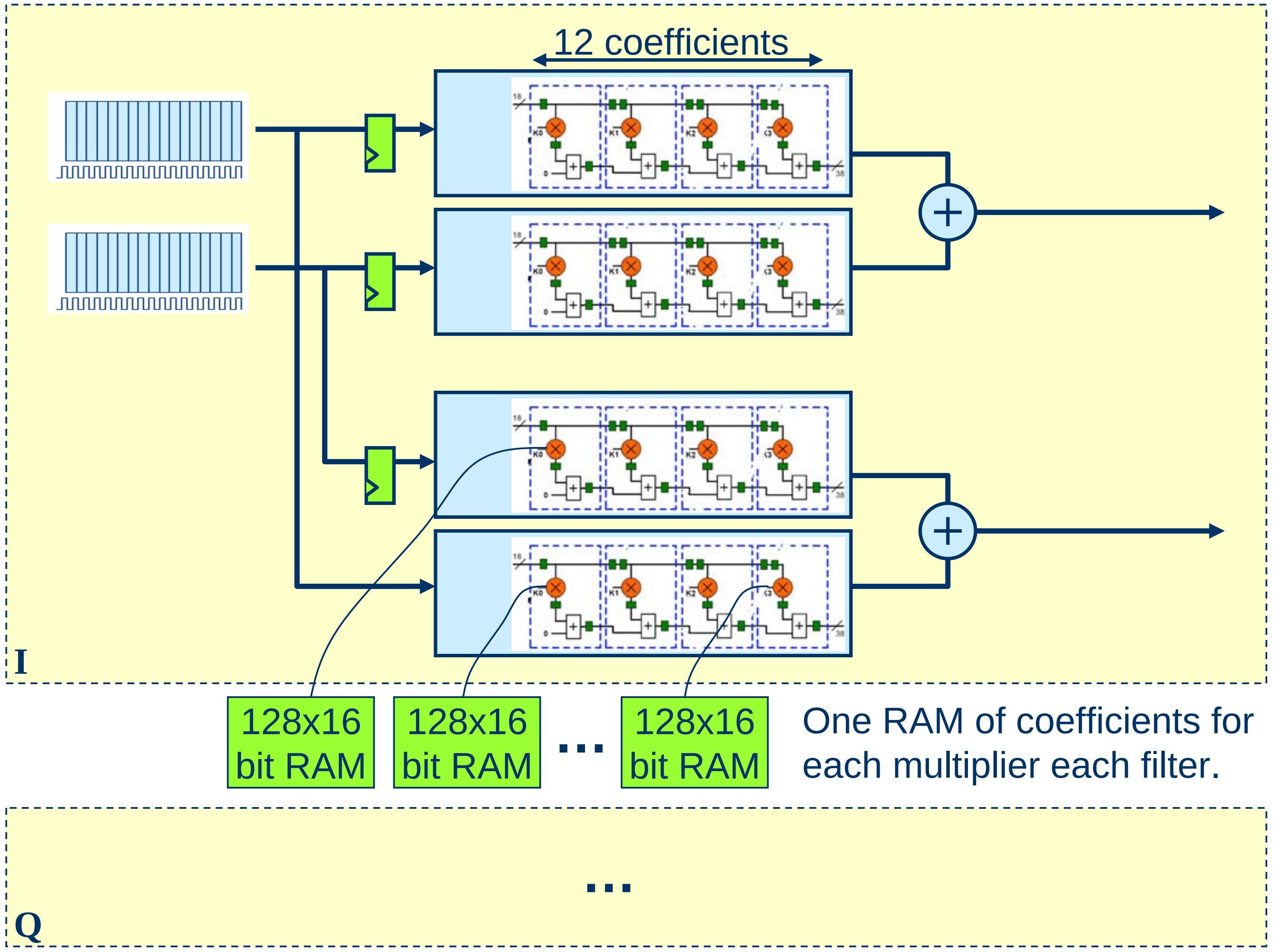


- Parallel data, one clock per sample.
- Fits well with FPGA architectures with fast built-in multipliers and accumulators.

Very high data rates
Example: Complex FIR filter at 4 x clock rate



Complex interpolating FIR filter at 4 x clock rate



One RAM of coefficients for each multiplier each filter.

Instantiation, inference and core generator

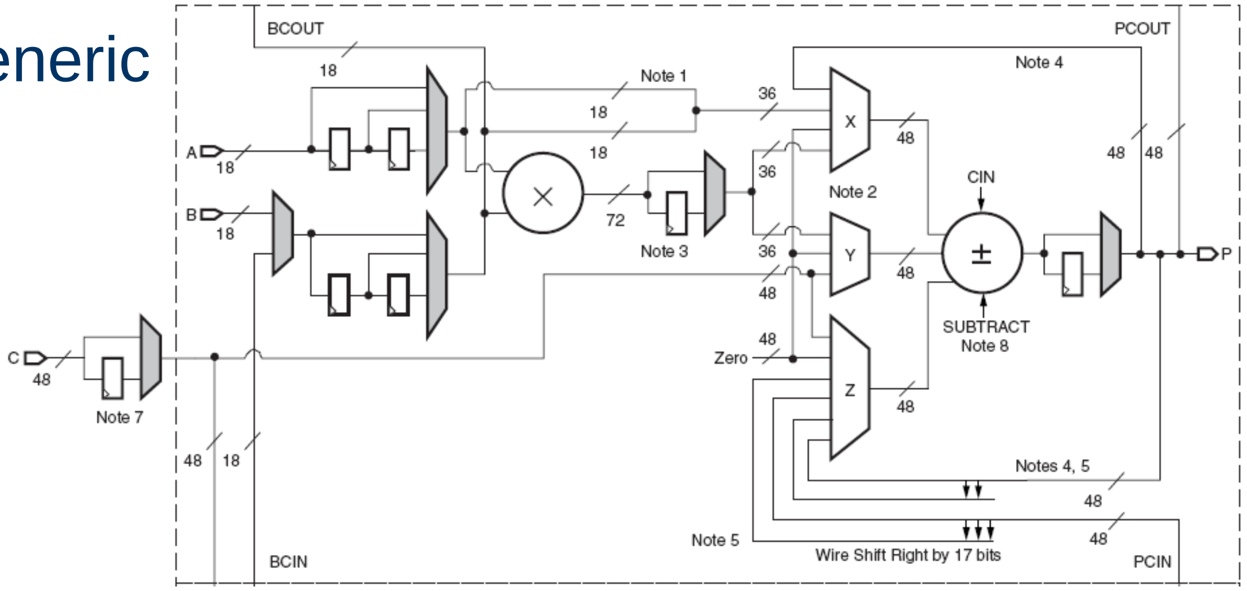
- RAMs and ROMs, single and dual port are generally easy to describe in VHDL and have the tools infer them.
- Only fully synchronous RAMs were fast enough.
- All our filter coefficient RAMs are dual port.
  - Slow clock port (44 MHz) for coefficient loading.
  - Fast clock port (350 MHz) for coefficient use.

```
type t_ram is array (1023 downto 0) of std_logic_vector (17 downto 0);
signal ram: t_ram;

process (clk) begin
   if rising_edge(clk) then
      if ram_write_enable='1' then ram(write_address) <= new_data;
      end if;
      if ram_read_enable='1'  then ram_out <= ram(read_address);
      end if;
   end if;
end process;
```

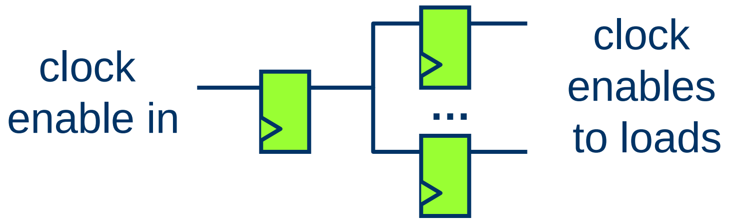
Instantiation, inference and core generator

- Our FIR filters were outside the core generator range.
- Describing DSP algorithms in VHDL so that the tools infer use of Xilinx DSP modules was (became) possible with limitations.
- Using the most advanced features of modules like the Xilinx DSPs generally requires instantiation.
- We designed generic solutions from scratch using instantiation of Xilinx DSP modules.

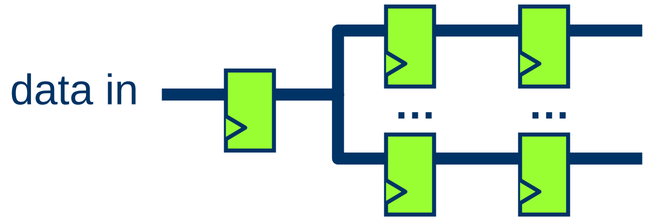


Manual pipelining and replication required

- Turn off automatic removal of equivalent signals and registers
- Do manual replication of clock enables and similar control signals
- Registers are abundant in modern FPGAs, better to replicate generously than to have routing problems all the time.
- Simple rule: **At least two** pipelining registers between modules.



- The problem is not fan-out but physical distance
  - between loads
  - between source and loads



- Be careful when using register chains like this for timing reasons. The tools like to convert them to shift registers and then the effect is lost. Use the necessary constraints / attributes.

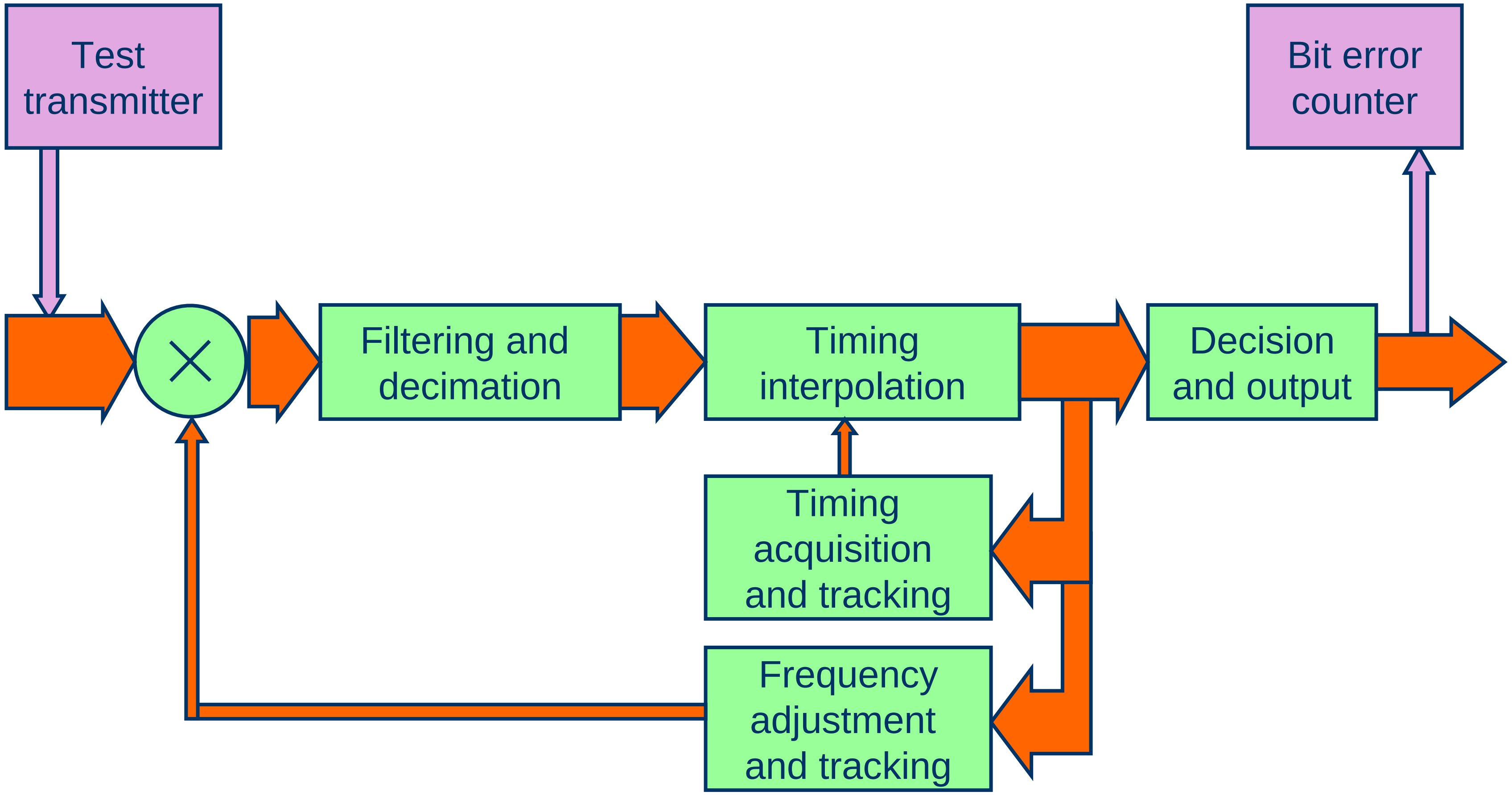
Simulation

- Synchronization algorithms can be slow:
  - 0,1s == 35 000 000 clock periods.
  - VHDL simulation of real scenarios unrealistic.
  - Bit-correct Matlab simulation unrealistic.

- Solution
  - Algorithms were verified in shortened Matlab simulations.
  - Modules- and subsystems were verified in VHDL.
  - Integrated system was verified in target with data from built-in test transmitter.

- Alternative solution
  - Use of behavioural modules for simulation speedup.

Design verification with built-in tester

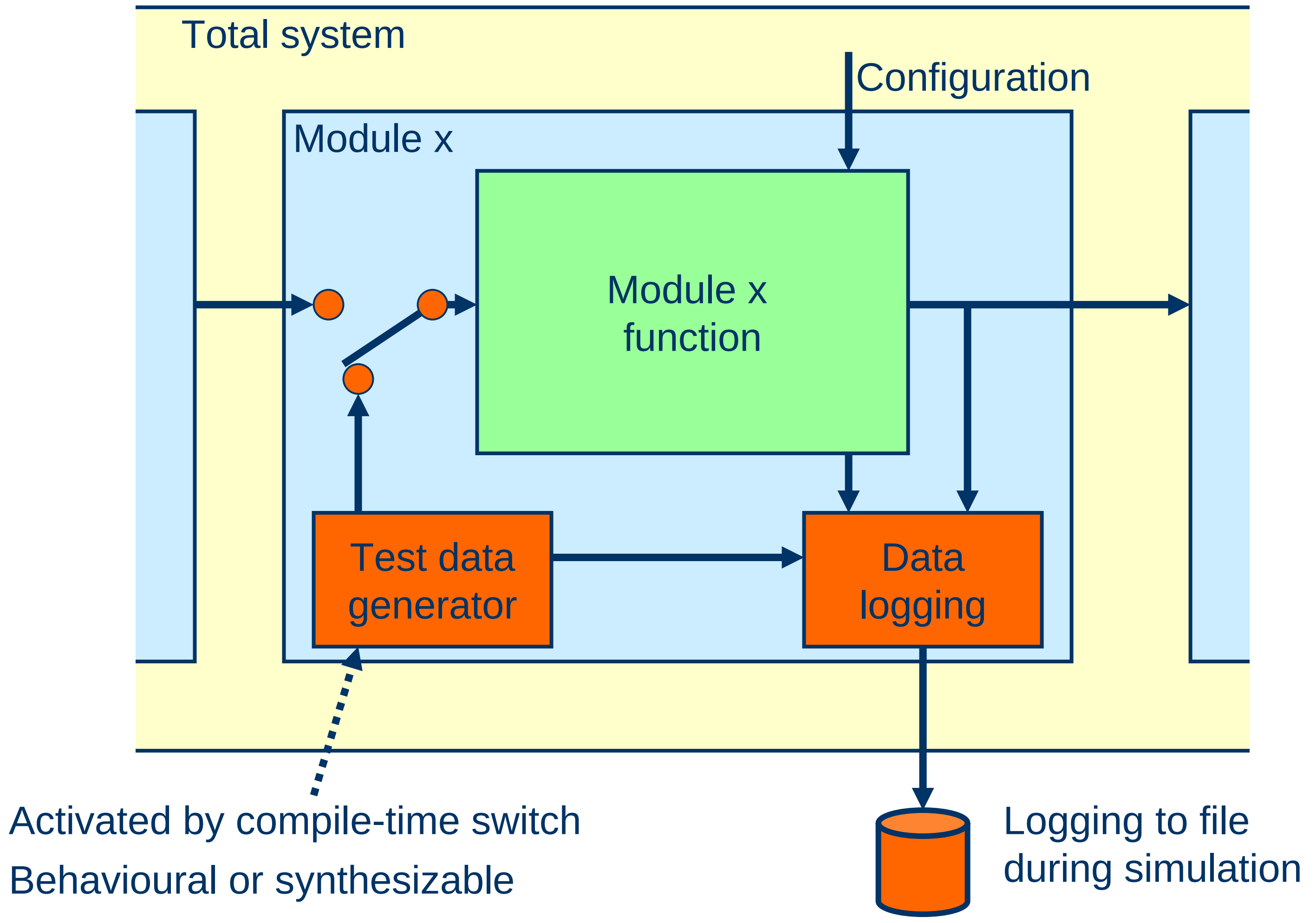


How to avoid large test benches

- Large modules with many inputs and outputs
  - Test benches become large.
  - Small modifications are often introduced without full retesting.
  - Test benches tend to be out of date after a while out of use.

- Solution
  - Modules are tested within the full design.
  - Normal routines for configuration can be used.
  - Internal test data generation in each module activated by local compile-time switch.
  - VHDL logging to file from code in each module being tested.

Test bench within each module



Activated by compile-time switch

Behavioural or synthesizable

Logging to file during simulation

Avoid using high clock rate when possible

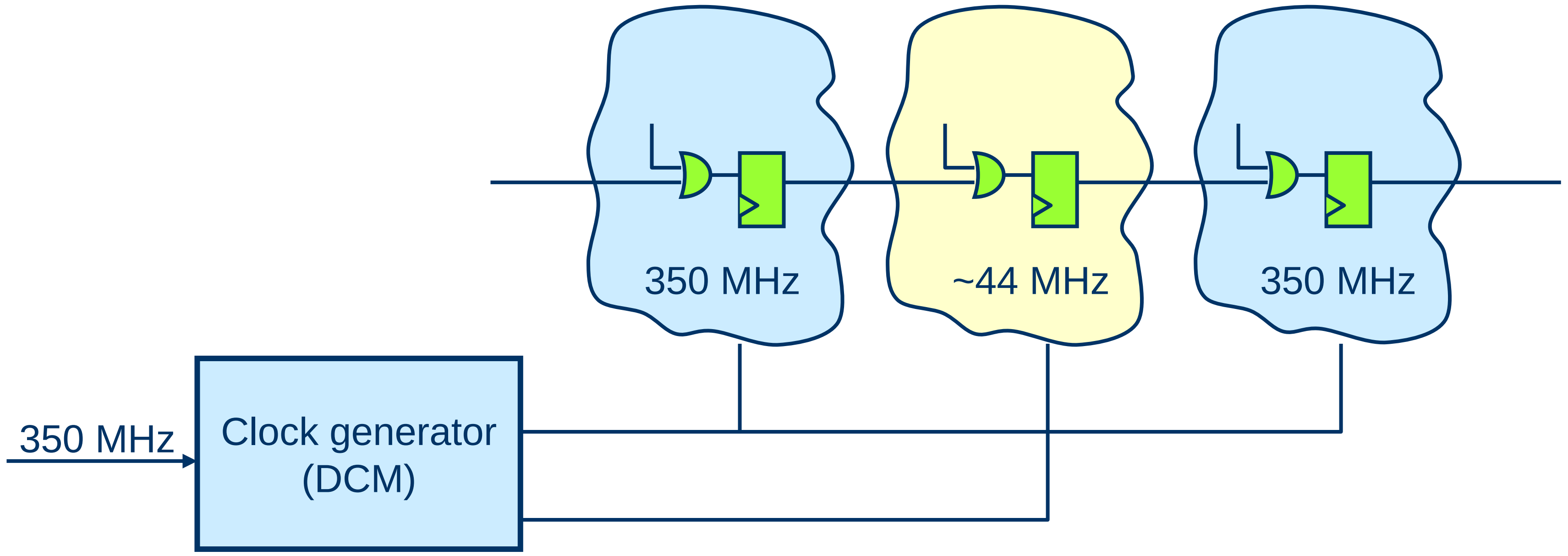
- The data clock is 350 MHz
  - 350 MHz stresses placement and routing and should not be used unless necessary
  - All our data buses, many 4 x 18 bit run at 350 MHz

- Solution
  - We used a 350/8=~44 MHz clock for less time critical functions
    - Configuration data paths – more than 100 addressable RAM blocks
    - Control signals
    - Parts of the frequency and timing acquisition loops

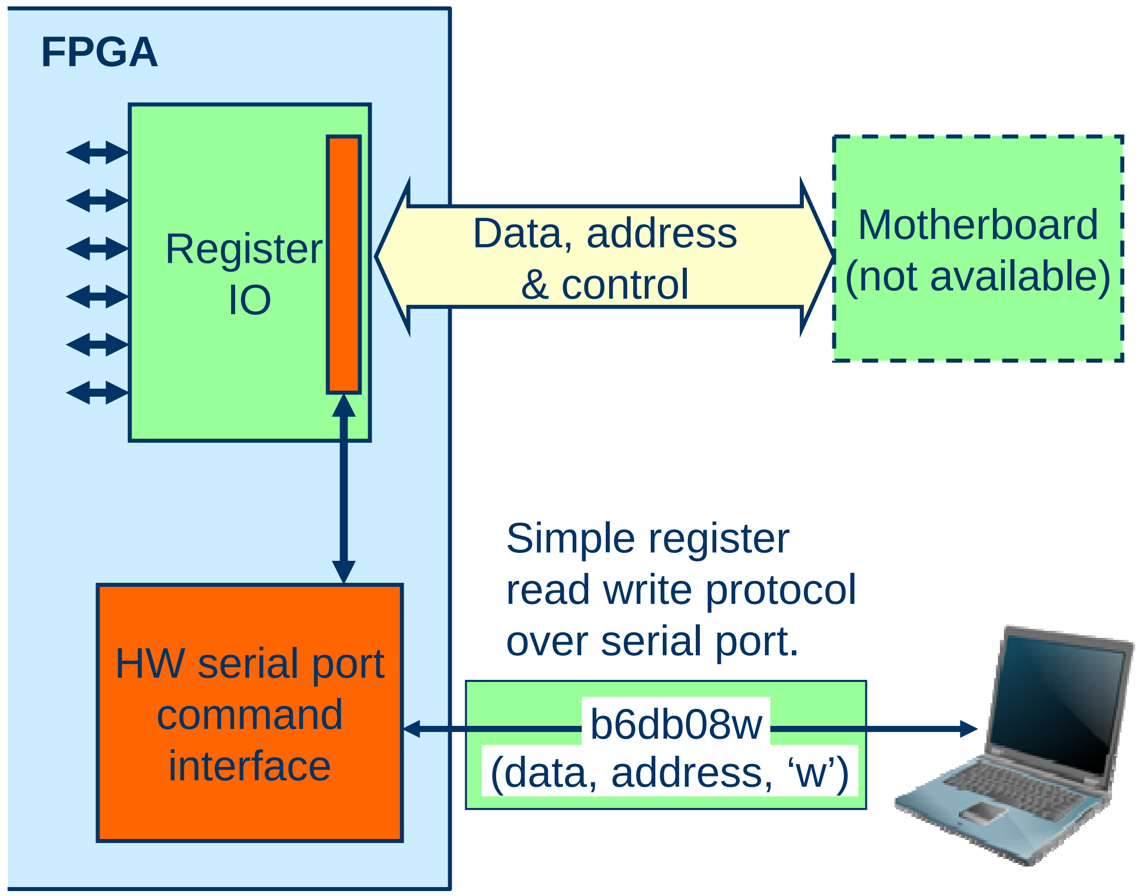
  - The two clocks are generated from the same clock manager
    - The tools guarantee low skew between them.
    - The tools constrain signals passing between the domain correctly.
    - Signals can pass between the two clock domains without bothering with the normal procedures for clock domain crossing.

Clock domain crossings with aligned clocks

- No special actions are necessary



Environment was missing - motherboard emulation



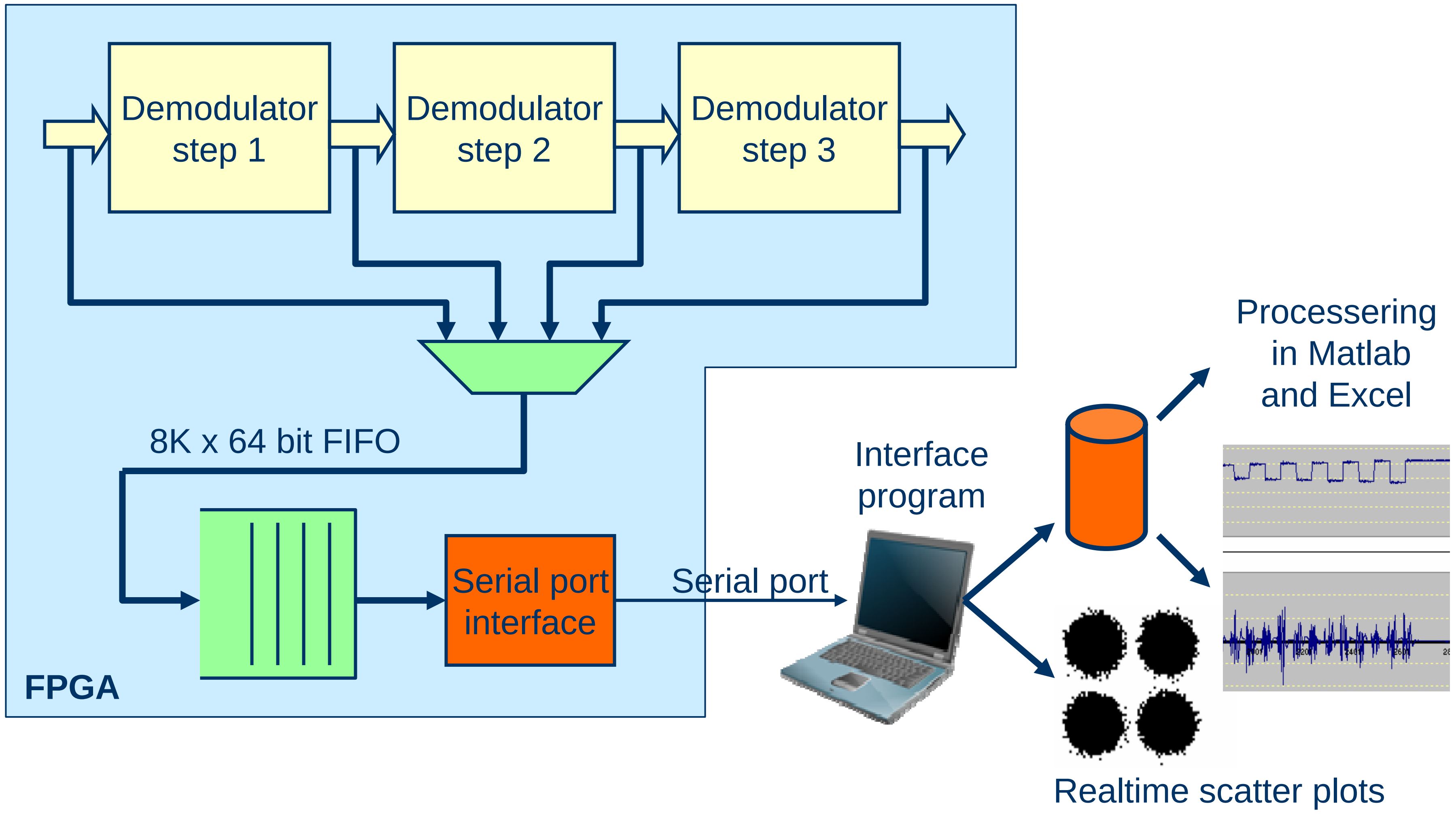
Simple register read write protocol over serial port.

The motherboard was not available during development.

Script file interpreter program on PC emulates configuration program.

```
;##############################
; Configure interpolation filter
w 8 xB6DB  ;-MSB
```

Data logging



Conclusion

- Successful project.
- Made possible by large FPGAs with built-in DSP functionality like high speed multipliers and accumulators.
- Some frustrations and tool problems on the way, but none serious.