Digital Signal Processing on FPGAs

Philipp Huebner[1]

[1]Institute of Networked and Embedded Systems
Alpen-Adria Universitaet Klagenfurt

January 14, 2015

Introduction

Digital Signal Processing

1965	Discrete Fourier Transform (Cooley and Tuckey)
late 1970s	Introduction of programmable DSP
mid 1990s	Multi-core DSPs
ca. 2001	GPGPU (General purpose computing on GPUs)

Introduction

Digital Signal Processing

1965	Discrete Fourier Transform (Cooley and Tuckey)
late 1970s	Introduction of programmable DSP
mid 1990s	Multi-core DSPs
ca. 2001	GPGPU (General purpose computing on GPUs)

Field Programmable Gate Arrays

1985	First commercially available FPGA
ca. 2000	Soft microprocessors.
ca. 2010	Intel Atom + FPGA in one package (E600C)
ca. 2011	FPGA-centric System on Chip (SoC)

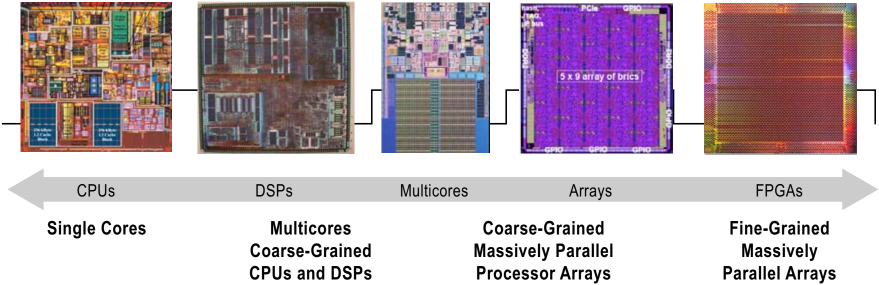
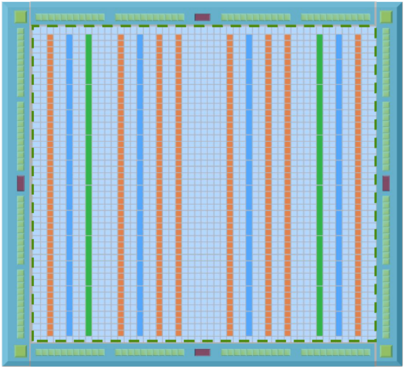

5 x 9 array of brics
CPUs
DSPs
Multicores
Arrays
FPGAs
Single Cores
Multicores Coarse-Grained CPUs and DSPs
Coarse-Grained Massively Parallel Processor Arrays
Fine-Grained Massively Parallel Arrays

Figure: Recent Trends of Programmable and Parallel Technologies

Modern FPGAs



- Over a million logic elements
- Thousands of 20-Kb memory blocks
- Thousands of DSP blocks
- High-speed transceivers

FPGA-based SoC

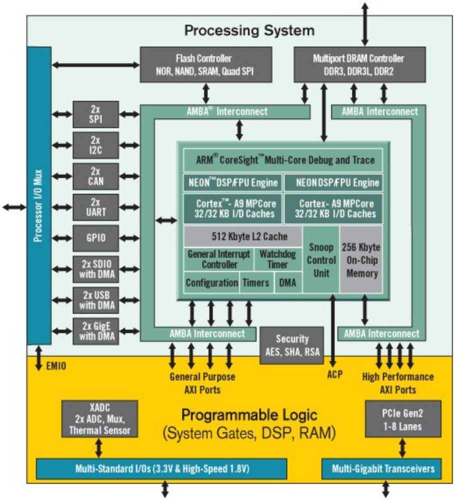


Figure: Xilinx Zynq-7000 SoC

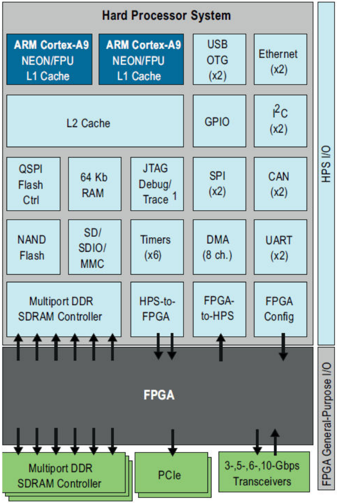


Figure: Altera Cyclone V SoC

ARM Cortex-A9

- 800 MHz dual core processor
- Superscalar pipeline architecture with 2.5 DMIPS per MHz
- 32 KB instruction/32 KB data L1 cache (4-way set-associative)
- Shared 512 KB L2 cache (8-way associate)

- 32-bit timer and watchdog
- Dynamic branch prediction
- NEON media processing accelerator (128-bit SIMD)
- single and double precision floating-point support
- MMU that works with L1 and L2 to ensure coherent data
- Support for many I/O standards (CAN, I^{2}C, USB, Ethernet, SPI & JTAG)
- configurable 32-, 64- or 128-bit AMBA AXI interface. (Advanced Microcontroller Bus Architecture Advanced eXtensible Interface)

DSP Applications

General :
filtering, detection, correlation, FFT, ...

Audio processing :
coding/decoding, noise cancellation, EQ, ...

Image processing :
compression/decompression, rotation, image recognition, image enhancement, ...

Control and instrumentation :
servo/engine control, guidance/navigation, ...

Information systems :
modulation/demodulation, encryption/decryption, waveform generation, beamforming, ...

Requirements of DSP systems

- High throughput (real-time)
- Reduction in power-dissipation
- Reduction in size and weight

Requirements of DSP systems

- High throughput (real-time)
- Reduction in power-dissipation
- Reduction in size and weight

Efficiency

- Power efficiency: $MOPS/mW$
- Silicon area efficiency: $MOPS/mm^2$

Requirements of DSP systems

- High throughput (real-time)
- Reduction in power-dissipation
- Reduction in size and weight

Efficiency

- Power efficiency: $MOPS/mW$
- Silicon area efficiency: $MOPS/mm^2$

- Reduced cost
  - Device cost
  - Development time
  - Cost for testing

Efficiency

- Cost effectiveness: $MOPS/\$$

Why Signal Processing on FPGA?

- Most algorithms are multiply and accumulate (MAC) intensive
- DSPs use multi-stage pipelining (MAC rates limited by multiplier)
- Parallelism can only be achieved by replicating the same generic computation hardware multiple times. (Programmers must explicitly code their application in a parallel fashion)

Example

Finite Impulse Response (FIR) Filter; length L

$$y[n] = x[n] * f[n] = \sum_{k=0}^{L-1} f[k]x[n-k]$$

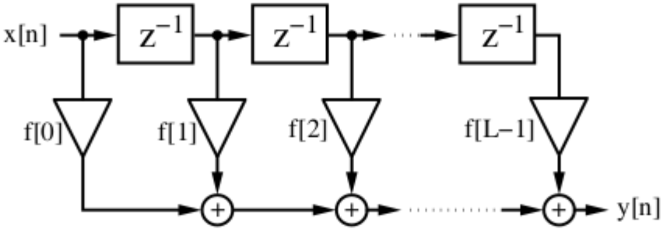
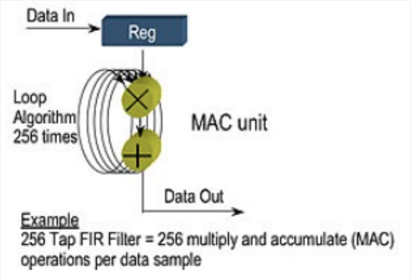


Figure: FIR Filter; length L

Example: 256-tap FIR Filter

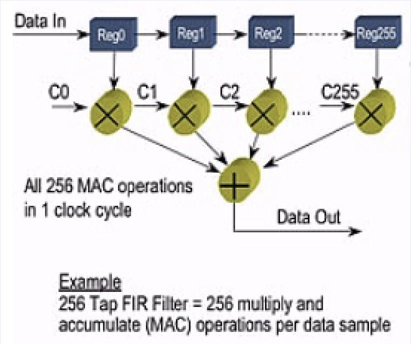
Conventional DSPs



Example

256 Tap FIR Filter = 256 multiply and accumulate (MAC) operations per data sample

FPGAs



All 256 MAC operations in 1 clock cycle

Example

256 Tap FIR Filter = 256 multiply and accumulate (MAC) operations per data sample

Why Signal Processing on FPGA?

- Most algorithms are multiply and accumulate (MAC) intensive
- DSPs use multi-stage pipelining (MAC rates limited by multiplier)
- Parallelism can only be achieved by replicating the same generic computation hardware multiple times. (Programmers must explicitly code their application in a parallel fashion)

Why Signal Processing on FPGA?

- Most algorithms are multiply and accumulate (MAC) intensive
- DSPs use multi-stage pipelining (MAC rates limited by multiplier)
- Parallelism can only be achieved by replicating the same generic computation hardware multiple times. (Programmers must explicitly code their application in a parallel fashion)

Pro's & Con's of FPGAs

Pro's

- Full parallelism
- Low latency
- High throughput
- Hard real-time behaviour
- Full control over the actual design implementation

Con's

- Difficult to design complex algorithms
- Difficult to test
- High initial cost

DSP-Benchmarks for FPGAs

- Berkley Design Technology Inc. (BDTi): Single algorithms not well suited to compare massively parallel chips.[1]
- Full System benchmarking
- Orthogonal Frequency Division Multiplex (OFDM)

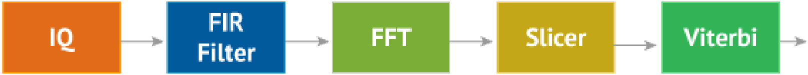


Figure: Simplified Block Diagram of the BDTI OFDM Receiver Benchmark

Representative?

[1] "The Art of Processor Benchmarking," Berkley Design Technology Inc., Tech. Rep., 2006.

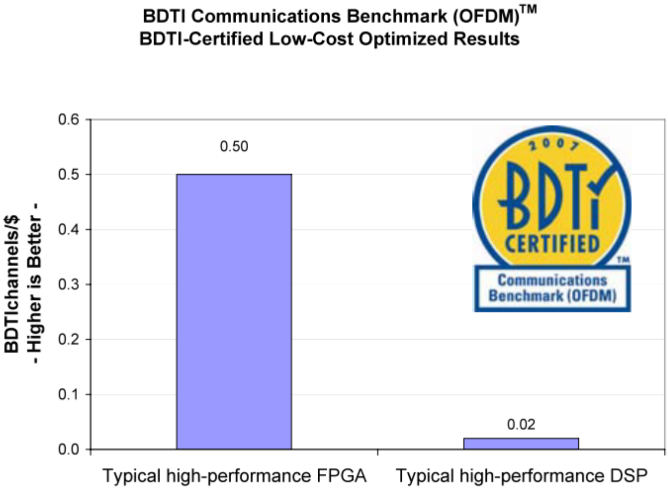


Figure: BDTI Benchmark Results on OFDM System

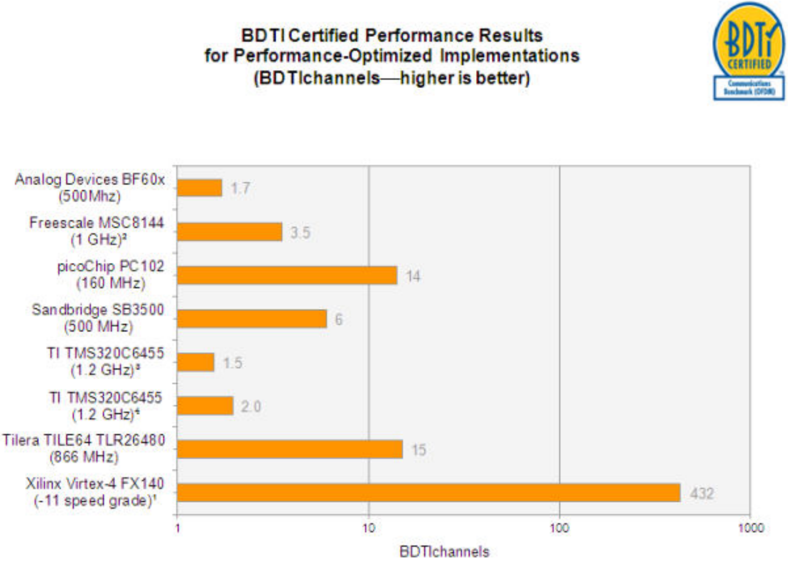


Figure: Maximum number of supported BDTIchannels on the device

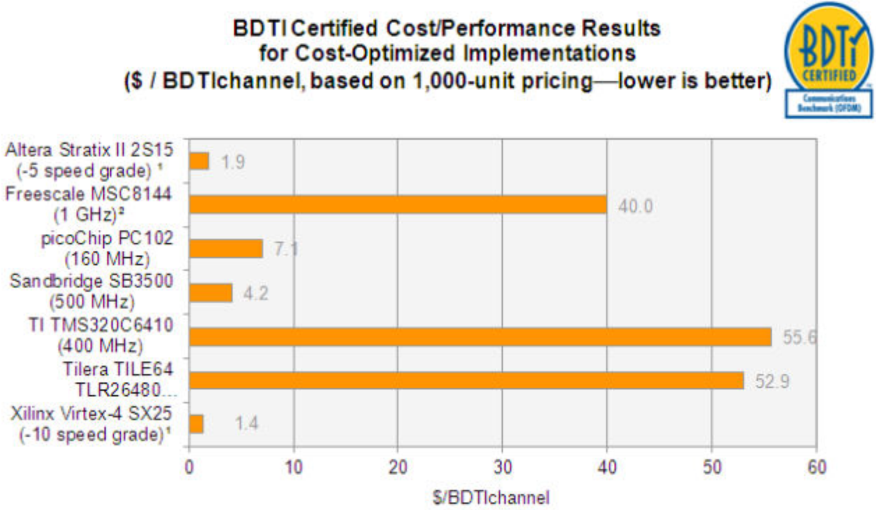


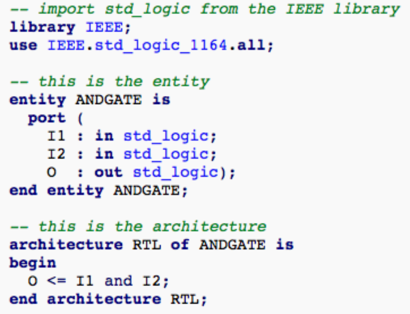
Figure: $/BDTIchannel

Programming Models

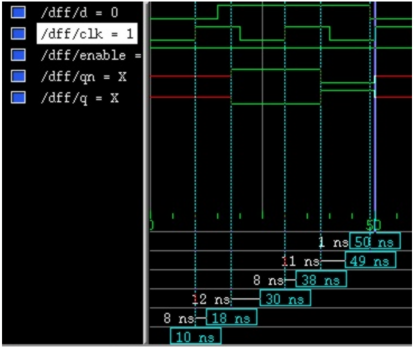
since 1987 Register transfer level (RTL) design using Hardware Description Languages (HDLs).

Hardware Description Languages (VHDL, Verilog)

- Specialized computer language to describe structure, design and operation of digital logic circuits.



```
-- import std_logic from the IEEE library
library IEEE;
use IEEE.std_logic_1164.all;

-- this is the entity
entity ANDGATE is
  port (
    I1 : in std_logic;
    I2 : in std_logic;
    O  : out std_logic);
end entity ANDGATE;

-- this is the architecture
architecture RTL of ANDGATE is
begin
  O <= I1 and I2;
end architecture RTL;
```

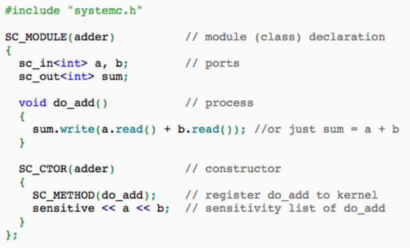


Programming Models

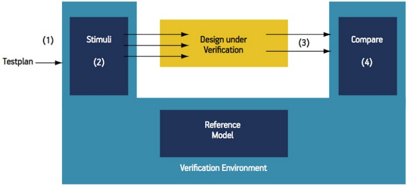
since 2000 Electronic system-level (ESL) and Transaction-level modeling (TLM). SystemC or SystemVerilog.

SystemC, SystemVerilog

- Set of C++ classes and macros to provide an event-driven simulation interface.
- Applied to system-level modelling, functional verification and high-level synthesis

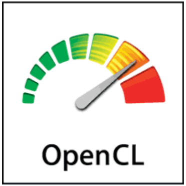


```
#include "systemc.h"

SC_MODULE(adder)             // module (class) declaration
{
  sc_in<int> a, b;           // ports
  sc_out<int> sum;

  void do_add()              // process
  {
    sum.write(a.read() + b.read()); //or just sum = a + b
  }

  SC_CTOR(adder)             // constructor
  {
    SC_METHOD(do_add);       // register do_add to kernel
    sensitive << a << b;     // sensitivity list of do_add
  }
};
```



OpenCL

Open Computing Language (OpenCL)

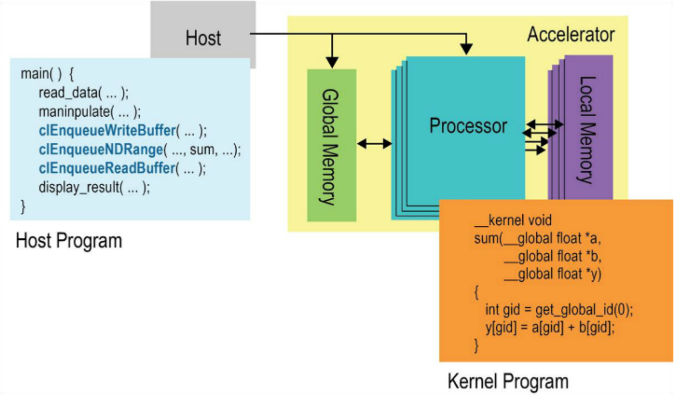


Framework for parallel heterogeneous computing.

- Designed by Apple (2008); Now developed by Khronos (www.khronos.org)
- Adopted by Intel, Qualcomm, AMD, Nvidia, Samsung, etc.
- since May 2013: SDK for Altera FPGAs
- since Nov 2014: Xilinx SDAccel Dev Environment.

OpenCL

- Includes a language based on C99 (OpenCL C).
- Provides application programming interface (API) for data-based and task-based parallelism.



```
main( ) {
    read_data( ... );
    maninpulate( ... );
    clEnqueueWriteBuffer( ... );
    clEnqueueNDRange( ..., sum, ...);
    clEnqueueReadBuffer( ... );
    display_result( ... );
}
```

```
__kernel void
sum(__global float *a,
    __global float *b,
    __global float *y)
{
  int gid = get_global_id(0);
  y[gid] = a[gid] + b[gid];
}
```

Figure: OpenCL Programming Model

OpenCL

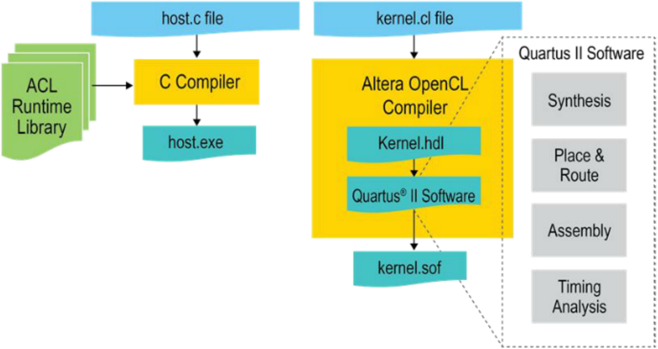


Figure: OpenCL Design Flow

Towards Resource Optimization in Parallel Heterogeneous Computing

Open Research Questions

- How can we use SoC and OpenCL to optimize the resource allocation...
  - ...on a single parallel heterogeneous computing platform? (e.g. Smart Camera)
  - ...on connected devices? (e.g. dedicated FPGAs in WSN)

References

[1] U. Meyer-Baese, *Digital Signal Processing with Field Programmable Gate Arrays*. Springer, 2014.

[2] "The Evolving Role of FPGAs in DSP Applications," Berkley Design Technology Inc., Tech. Rep., 2007.

[3] "The Art of Processor Benchmarking," Berkley Design Technology Inc., Tech. Rep., 2006.

[4] R. Njuguna. (Nov. 2008). A Survey of FPGA Benchmarks, [Online]. Available: `http://www.cse.wustl.edu/~jain/cse567-08/ftp/fpga/`.