Performance Overhead with High Level Waveform Development

Stefan Nagel, Michael Schwall,
Friedrich K. Jondral

2010 European Reconfigurable Radio Technologies Workshop

June 23-25, 2010

Mainz, Germany

Communications Engineering Lab
Prof. Dr.rer.nat. Friedrich K. Jondral

Motivation


HARRIS
RF-5720
HF MODEM
MODE
SETUP
ALPHA
PRGM
1 ABC
2 DEF
3 GHI
SEND
4 JKL
5 MNO
6 PQR
TEST
7 STU
8 VWX
9 YZ_
CLR
0
ENTR
PT
OFF
CT
LD
RMT
Z
VOL
DATA
RADIO

Motivation

Waveform

MIL-STD.exe
MIL-STD.out
MIL-STD.bin

Platform

Motivation

Waveform

MIL-STD.exe
MIL-STD.out
MIL-STD.bin

GSM.exe
GSM.out
GSM.bin

TETRA.exe
TETRA.out
TETRA.bin

W-LAN.exe
W-LAN.out
W-LAN.bin

Platform

Motivation

Waveform

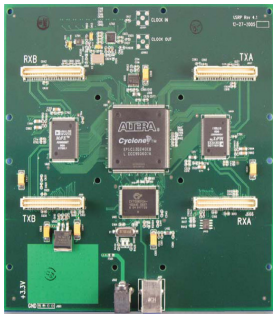
MIL-STD.exe
MIL-STD.out
MIL-STD.bin

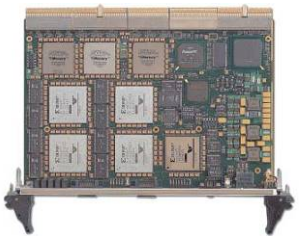
GSM.exe
GSM.out
GSM.bin

TETRA.exe
TETRA.out
TETRA.bin

W-LAN.exe
W-LAN.out
W-LAN.bin

Platform



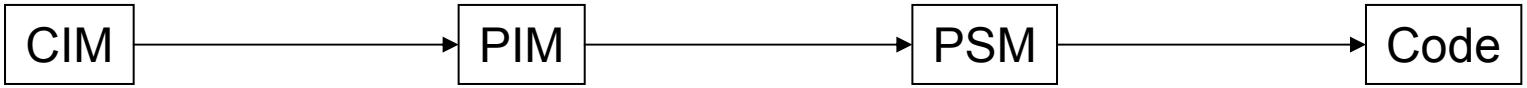


Question:

How can we build waveforms running on platforms, we (perhaps) don‘t know?

Waveform Development Process

- To provide portability we followed a design process based on the Model Driven Architecture



Computation Independent Model:	Platform Independent Model:	Platform Specific Model:	Code:
•Description of functionality •Specification of air interface	•Simulation of functionality •Without platform specific aspects	•Extension of PIM •With separation on different processing units •With usage of platform APIs	•Binary Code running on the platform

Waveform and Platforms

- Used Waveform: IEEE 802.11 a/g
- Key parameters:
  - OFDM system with 64 carrier
  - 16 MHz bandwidth → 4 µs symbol duration
  - Frame structure:
    - 16 µs Training Sequence (STS, LTS)
    - 4 µs signal information (BPSK, r = ½)
    - 16 µs data (QPSK, r = ½)
- Evaluation:
  - Is this possible with Model Based Design on different platforms?
  - If not, which bandwidths and hence data rates are possible?

Waveform and Platforms

- Used Platform 1: USRP

RF-Frontend ⟷ ADC/DAC ⟷ FPGA ⟷ GPP

- Flex400
- Direct Conversion

- ADC:
  - 64 MSPS
  - 12 bit
- DAC:
  - 128 MSPS
  - 14 bit Integrated Interpolation of 4

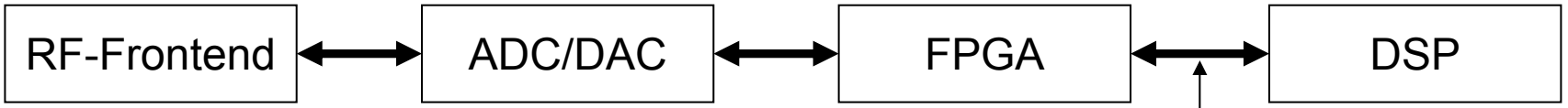
- Altera Cyclone 2
- 12060 LEs

- Intel Core 2 Duo
- 2x 2.26 GHz
- L2 cache: 3 MB

- USB 2.0
- Theoretical: 480 Mbps
- Real: ~ 228 Mbps
  → 7.11 MHz BW

Waveform and Platforms

- Used Platform 2: SFF SDR DP



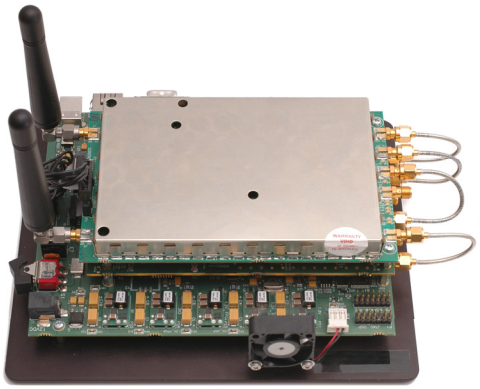
- Tunable RF
- Superhet Receiver

- ADC:
  - 125 MSPS
  - 14 bit
- DAC:
  - 500 MSPS
  - 16 bit
  - Integrated Interpolation of 2,4 or 8

- Xilinx Virtex-4
- SX-35
- 15360 Slices
- 192 Multplier

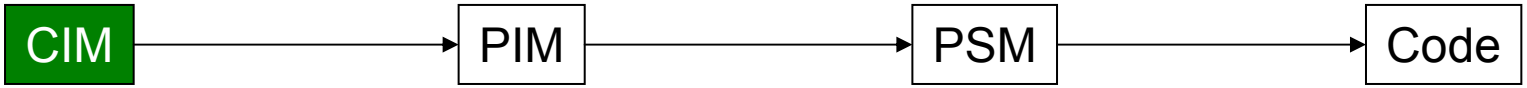
- TI C64x+
- 594 MHz
- L1 cache: 112 kB
- L2 cache: 64 kB

- VPSS
- Theoretical: 160 Mbps
- Real: ~ 52 Mbps
  → 1.5 MHz BW



Computation Independent Model

- Take or write documents of the specification



Computation Independent Model:

- Description of functionality
- Specification of air interface

Platform Independent Model:

- Simulation of functionality
- Without platform specific aspects

Platform Specific Model:

- Extension of PIM
- With separation on different processing units
- With usage of platform APIs

Code:

- Binary Code running on the platform

Computation Independent Model

IEEE Std 802.11a-1999(R2003)
(Supplement to IEEE Std 802.11-1999)

[Adopted by ISO/IEC and redesignated as
ISO/IEC 8802-11:1999/Amd 1:2000(E)]

Supplement to IEEE Standard for Information technology—

Telecommunications and information exchange between systems—

Local and metropolitan area networks—

Specific requirements

Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications

High-speed Physical Layer in the 5 GHz Band

Adopted by the ISO/IEC and redesignated as ISO/IEC 8802-11:1999/Amd 1:2000(E)

Sponsor

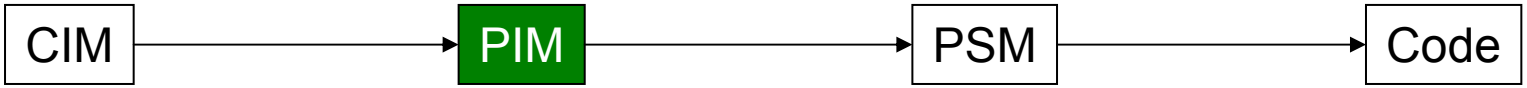
LAN/MAN Standards Committee
of the
IEEE Computer Society

Reaffirmed 12 June 2003

IEEE-SA Standards Board

Platform Independent Model

- Simulate air interface functionality



Computation Independent Model:

- Description of functionality
- Specification of air interface

Platform Independent Model:

- Simulation of functionality
- Without platform specific aspects

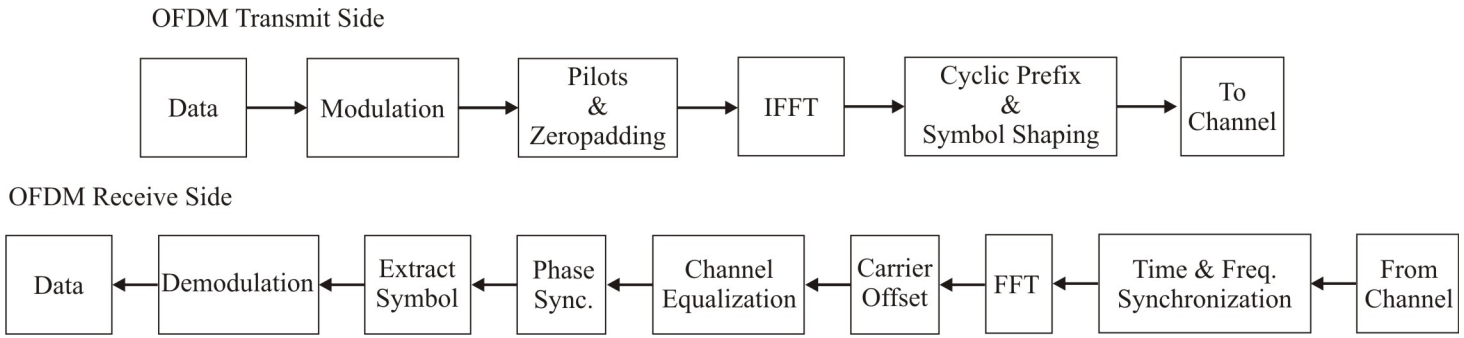
Platform Specific Model:

- Extension of PIM
- With separation on different processing units
- With usage of platform APIs

Code:

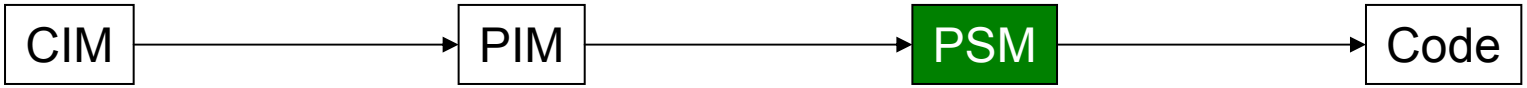
- Binary Code running on the platform

Platform Independent Model



Platform Specific Model

- Extend PIM for use on USRP



Computation Independent Model:

- Description of functionality
- Specification of air interface

Platform Independent Model:

- Simulation of functionality
- Without platform specific aspects

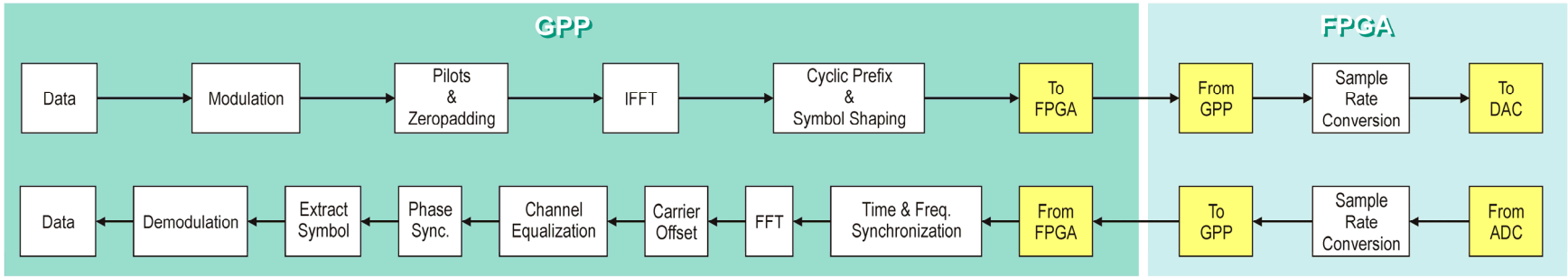
Platform Specific Model:

- Extension of PIM
- With separation on different processing units
- With usage of platform APIs

Code:

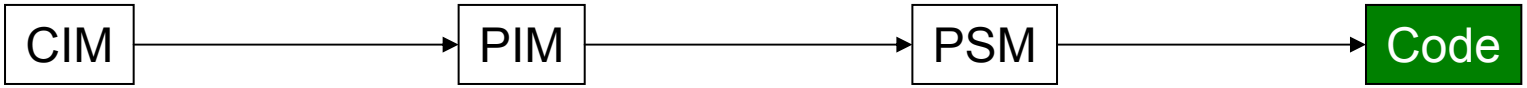
- Binary Code running on the platform

Platform Specific Model



Code

- Generate the code



Computation Independent Model:

- Description of functionality
- Specification of air interface

Platform Independent Model:

- Simulation of functionality
- Without platform specific aspects

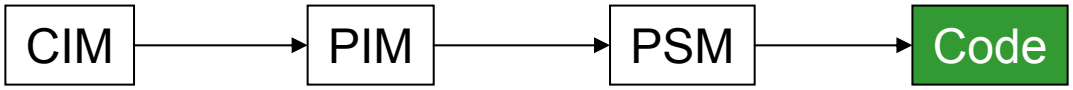
Platform Specific Model:

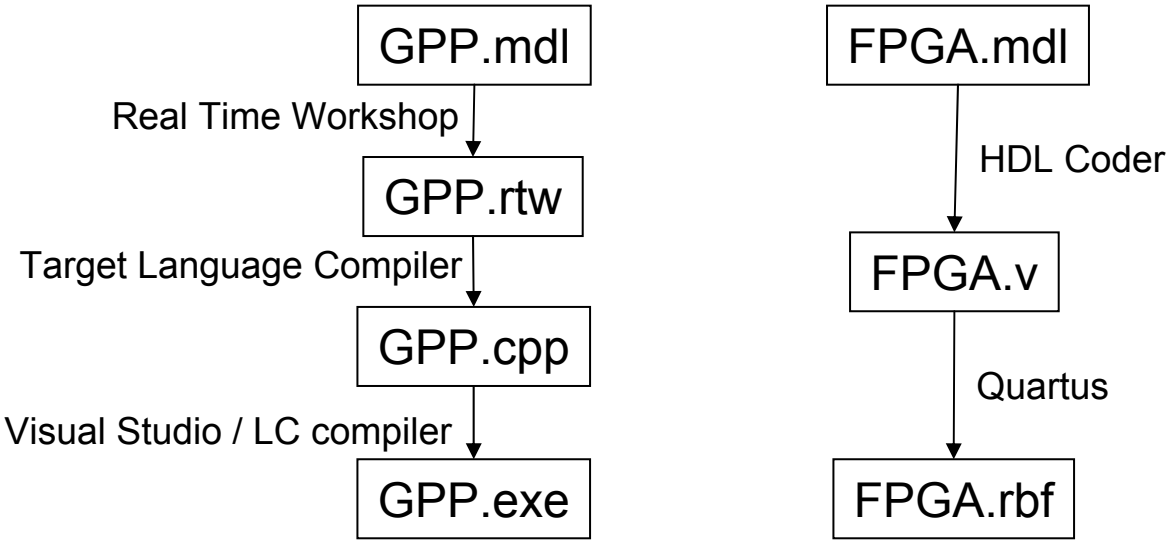
- Extension of PIM
- With separation on different processing units
- With usage of platform APIs

Code:

- Binary Code running on the platform

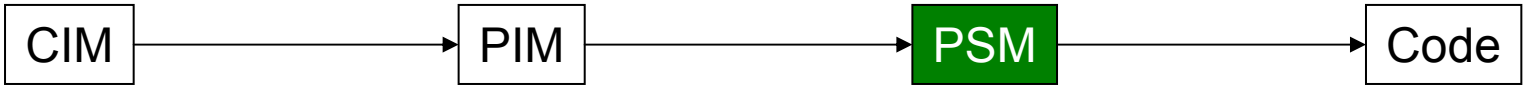
Code


CIM
PIM
PSM
Code


GPP.mdl
Real Time Workshop
GPP.rtw
Target Language Compiler
GPP.cpp
Visual Studio / LC compiler
GPP.exe
FPGA.mdl
HDL Coder
FPGA.v
Quartus
FPGA.rbf

Platform Specific Model

- Extend PSM for use on SFF



Computation Independent Model:

- Description of functionality
- Specification of air interface

Platform Independent Model:

- Simulation of functionality
- Without platform specific aspects

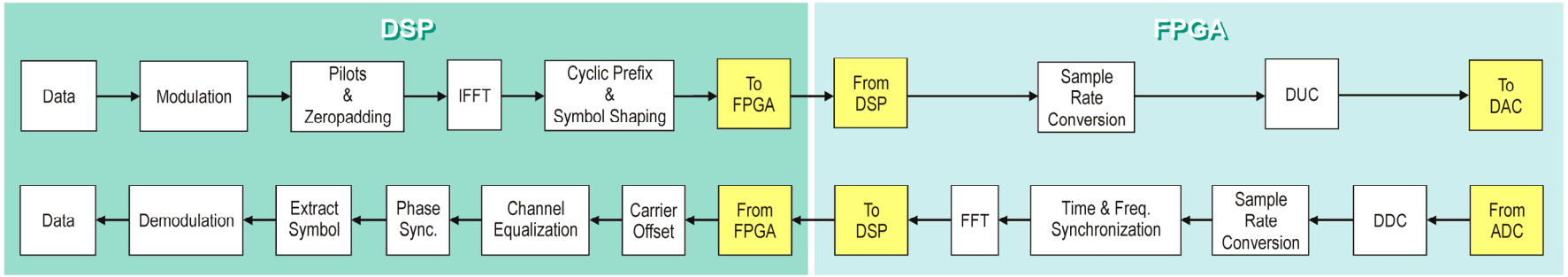
Platform Specific Model:

- Extension of PIM
- With separation on different processing units
- With usage of platform APIs

Code:

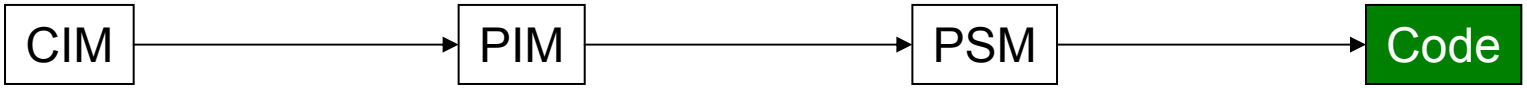
- Binary Code running on the platform

Platform Specific Model



Code

- Generate the code



Computation Independent Model:

- Description of functionality
- Specification of air interface

Platform Independent Model:

- Simulation of functionality
- Without platform specific aspects

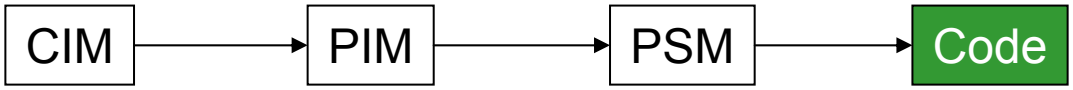
Platform Specific Model:

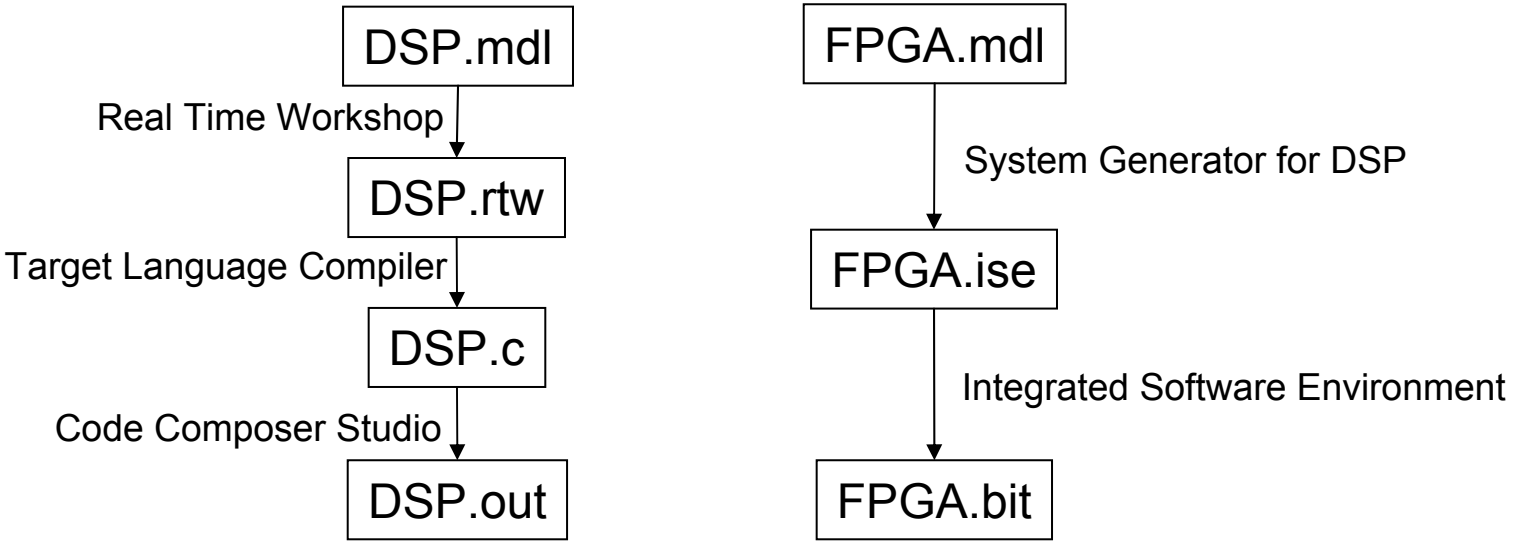
- Extension of PIM
- With separation on different processing units
- With usage of platform APIs

Code:

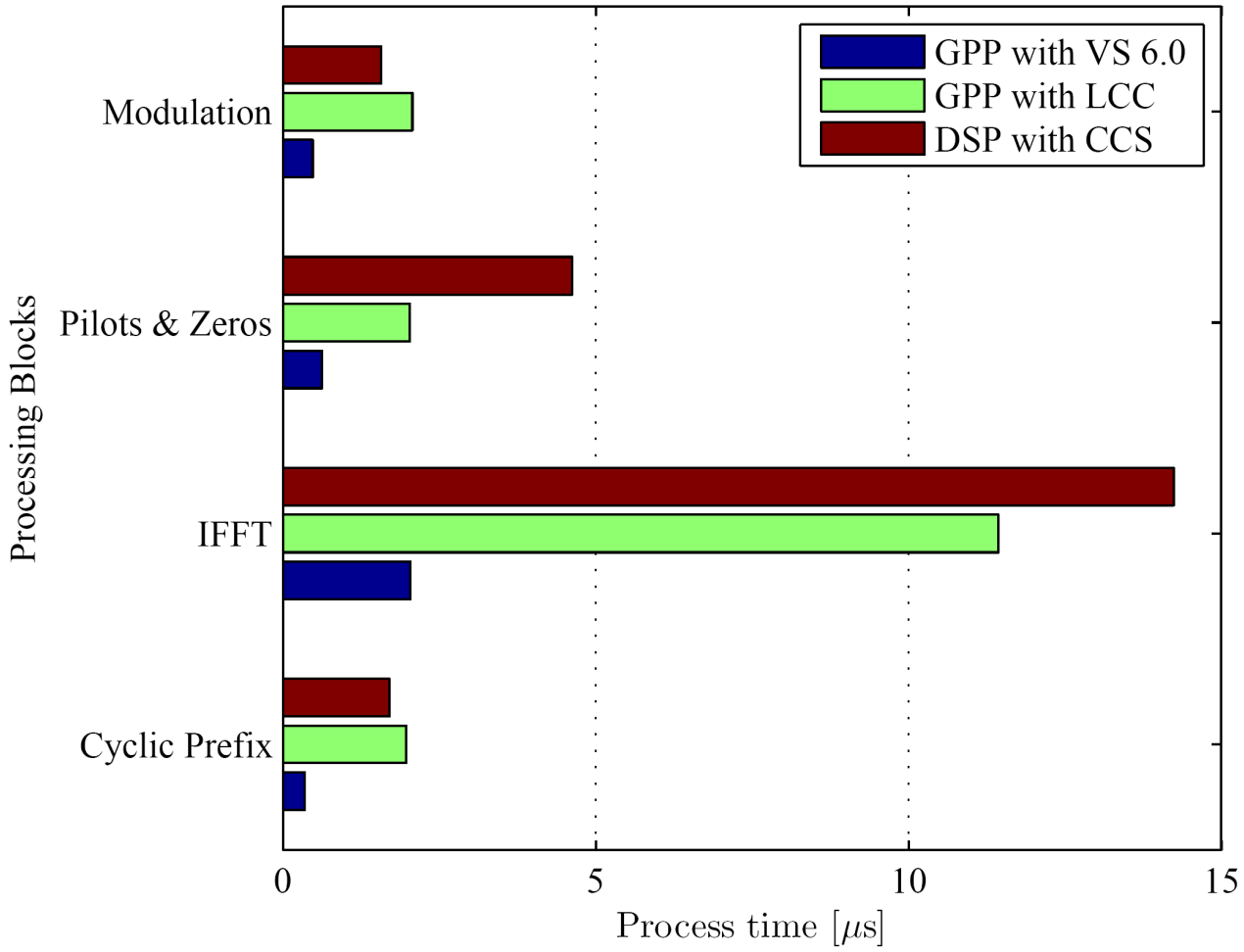
- Binary Code running on the platform

Code

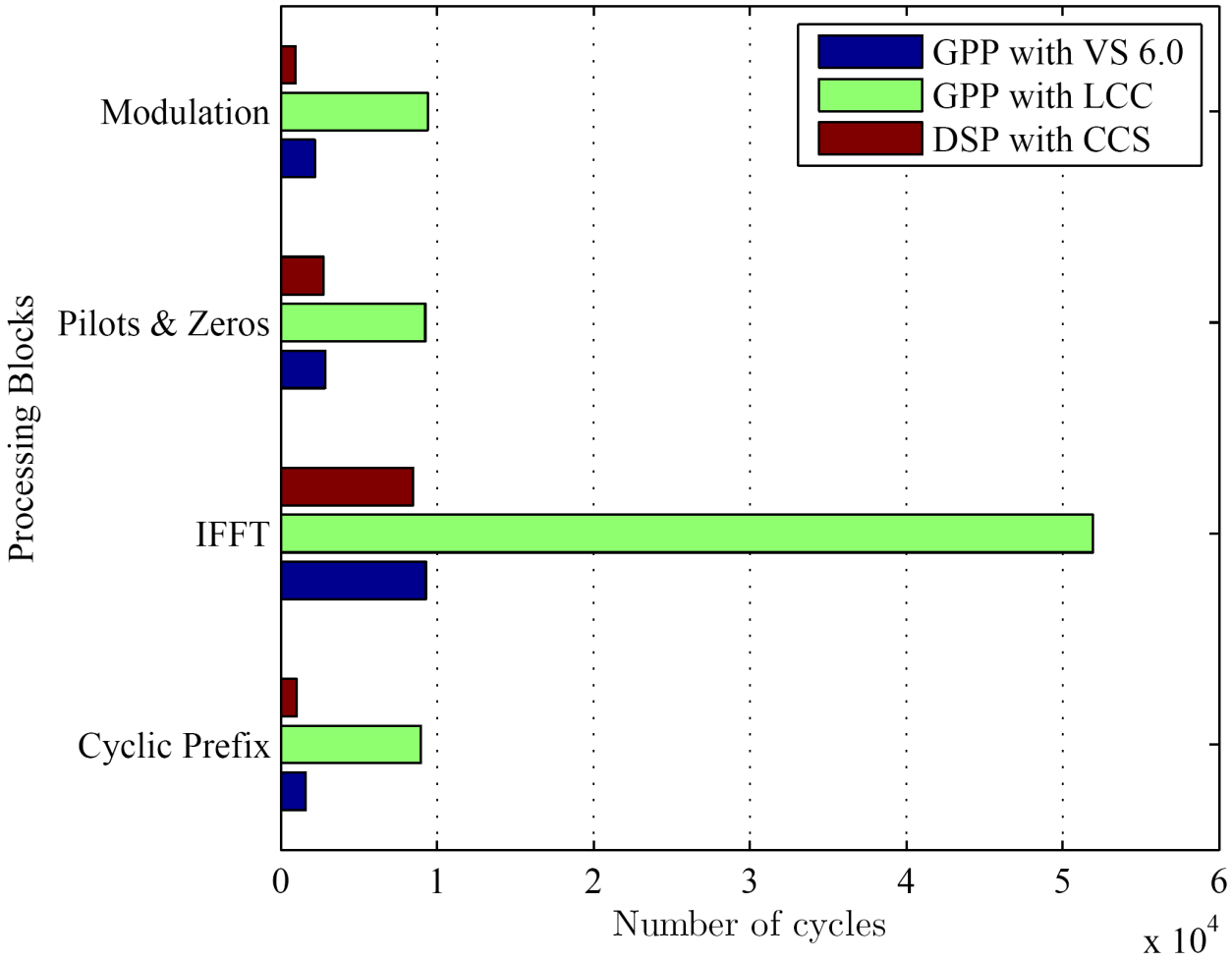

CIM
PIM
PSM
Code


DSP.mdl
Real Time Workshop
DSP.rtw
Target Language Compiler
DSP.c
Code Composer Studio
DSP.out
FPGA.mdl
System Generator for DSP
FPGA.ise
Integrated Software Environment
FPGA.bit

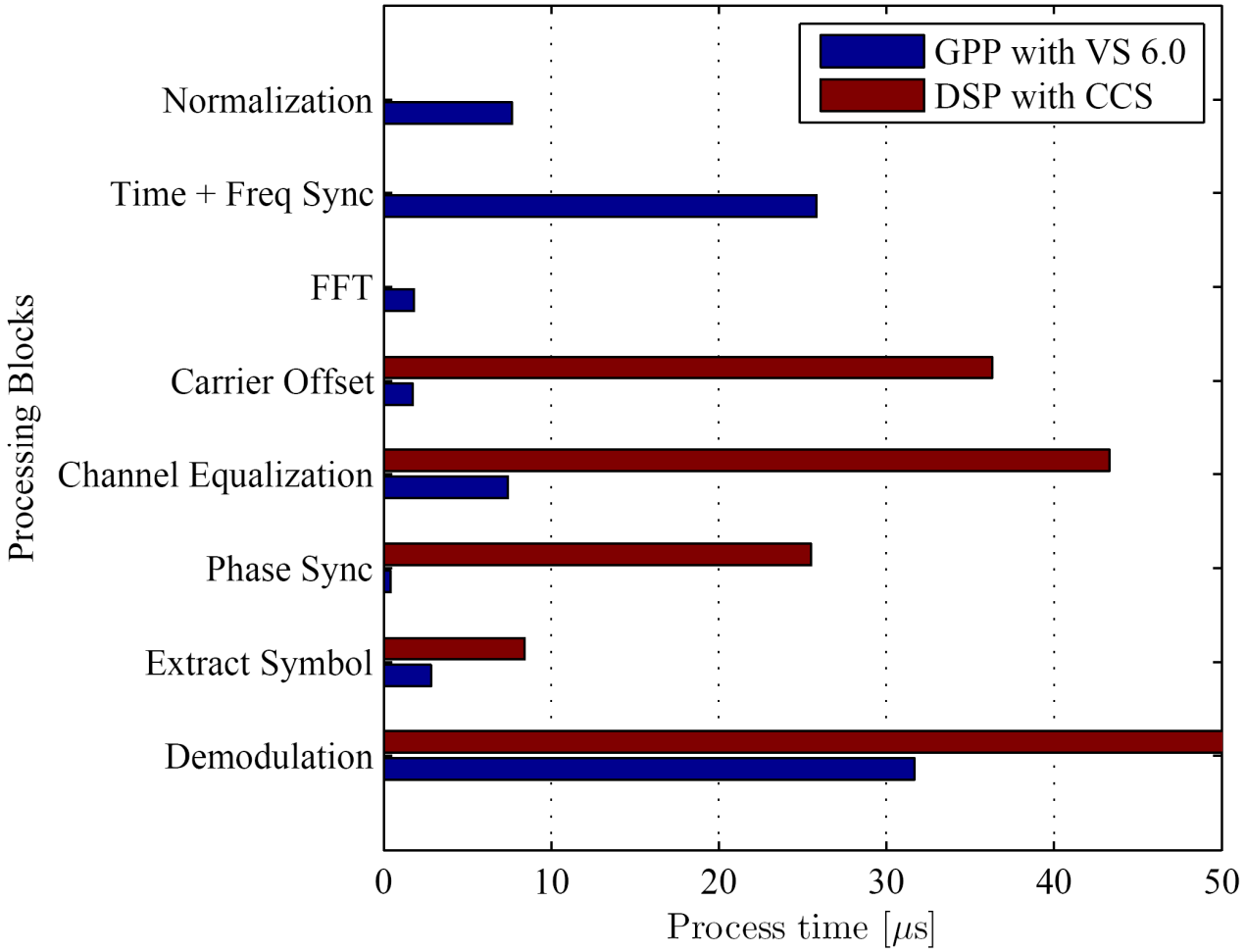
Results:


GPP with VS 6.0
GPP with LCC
DSP with CCS
Modulation
Pilots & Zeros
IFFT
Cyclic Prefix
Processing Blocks
0
5
10
15
Process time [μs]

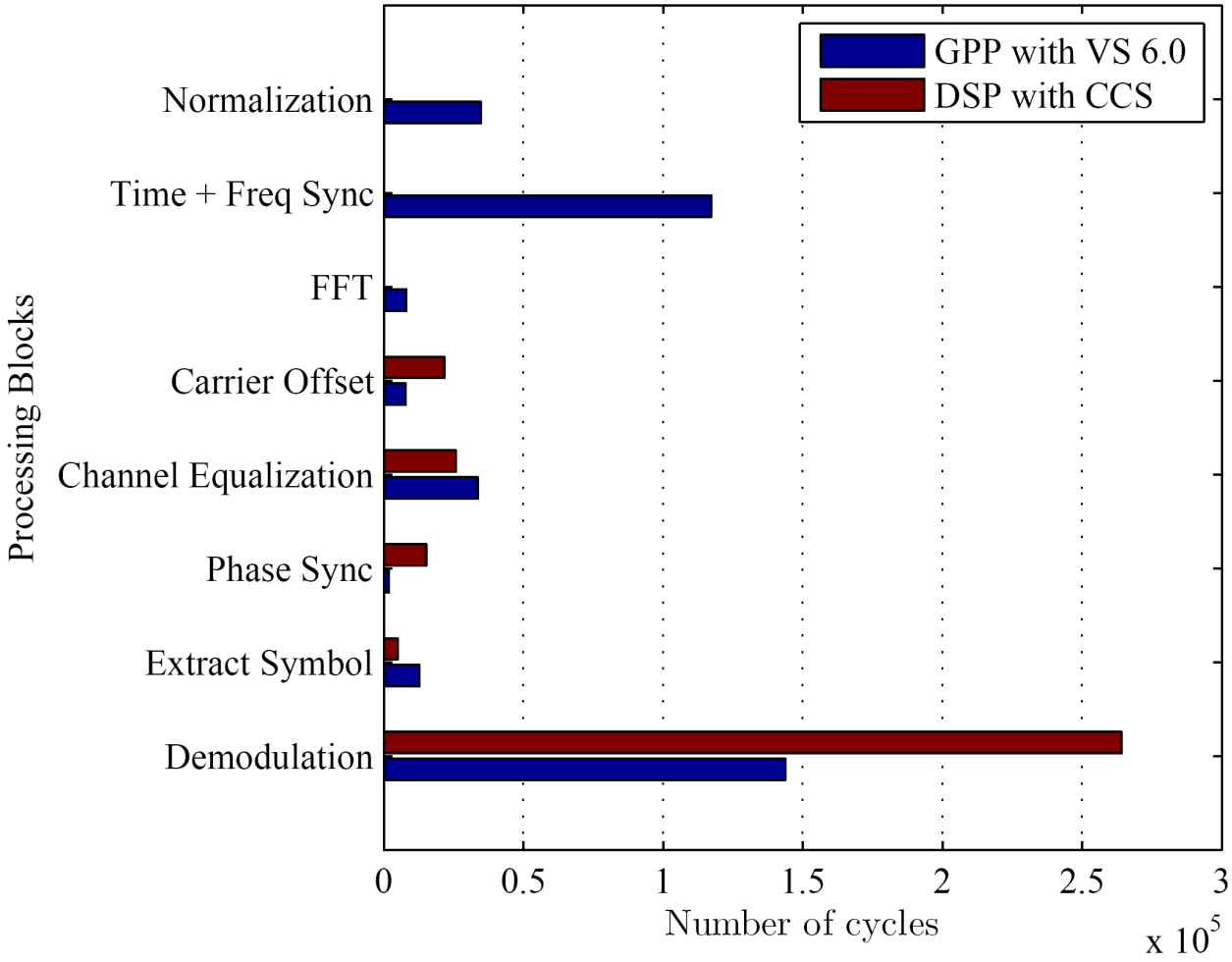
Results:


GPP with VS 6.0
GPP with LCC
DSP with CCS
Modulation
Pilots & Zeros
IFFT
Cyclic Prefix
Processing Blocks
0
1
2
3
4
5
6
Number of cycles
x 10^4

Results:


GPP with VS 6.0
DSP with CCS
Normalization
Time + Freq Sync
FFT
Carrier Offset
Channel Equalization
Phase Sync
Extract Symbol
Demodulation
Processing Blocks
0
10
20
30
40
50
Process time [μs]

Results:


GPP with VS 6.0
DSP with CCS
Normalization
Time + Freq Sync
FFT
Carrier Offset
Channel Equalization
Phase Sync
Extract Symbol
Demodulation
Processing Blocks
0
0.5
1
1.5
2
2.5
3
Number of cycles
x 10^5

Results:

Blocks		Cyclone II on USRP		Virtex-4 on SFF SDR			
		Logic Elements		Slices		DSP48	
Tx	APIs	1214	10,07%	1907	12,42%	5	2,60%
	Sample Rate Conversion	1523	12,63%	687	4,57%	10	8,33%
Rx	APIs	3448	28,59%	1981	12,90%	7	3,65%
	Sample Rate Conversion	1587	13,16%	658	4,38%	10	8,33%
	Synchronization	x	x	5880	38,28%	63	32,81%
	FFT	x	x	4016	26,15%	16	8,33%

Conclusions:

- Portable Waveform Development is possible with Model Based Design
- Partition of the waveform depends on the processing resources but also on the buses between them
- Code Generation for GPPs is straight forward
- Code Generation for DSPs need fix point algorithms for lightweight code
- Code Generation for FPGAs is on a starting point but leads to good HDL code

Performance Overhead with High Level Waveform Development