DESIGN OF AN ARCHITECTURE AND INSTRUCTION-SET OF A RECONFIGURABLE ASIP FOR SOFTWARE-DEFINED RADIO

K. Solomon Raju$^{(1)}$, Chandra Shekhar$^{(2)}$ and R C Joshi$^{(3)}$

$^{(1)}$E&CE Department, Indian Institute of Technology Roorkee, Rookee-247667, India
and Email: kotasolomonraju@gmail.com
$^{(2)}$ Director, Central Electronics Engineering Research Institute (CEERI) Pilani-333031, India
and Email: chandra@ceeri.ernet.in
$^{(3)}$ "As (1) above but Email": joshifcc@iitr.ernet.in

Abstract

This paper presents a new approach that is complete functionality of SDR implemented in an Application Specific Instruction Processor (ASIP). In this paper, we discuss the details of design of architecture for an ASIP for SDR with a single bus and dedicated paths. We explain the instruction-set design for implementation of the SDR functionality and their instruction formats. We have used SystemC language for functional modeling and simulating architecture of an ASIP as well as its instruction-set. Simulation results for chosen instruction set are shown.
Keywords — ASIC, ASIP, SDR, special-Instructions and SystemC

1. INTRODUCTION

The future communication system has to support more than one standard and one of the implementation of this requirement is using Software-Defined Radio (SDR). One of the key target standards that can benefit from an SDR approach are the third and fourth generation wireless systems. So far, the implementation of SDR has been performed as follows: (i) Using Software method: In this approach all the functionality of the SDR implemented in SW, which run in a high-end microprocessor, and/or Digital Signal Processor and (ii) Embedded system: The functionality has divided into either SW component or hardware (HW) component and this HW component has been implemented on an ASIC as well as on a reconfigurable coprocessor [6]. The SW component has been implemented in DSP. This paper presents a new approach that is complete functionality of SDR implemented in an Application Specific Instruction Processor (ASIP).

ASIPs represent a hybrid approach between the paradigms of general-purpose processor and Application-Specific Integrated Circuits (ASICs) and achieve performance levels significantly higher than the performance obtained with ASIC and same flexibility that offered by general-purpose processors like Digital Signal Processors (DSPs), Microprocessors and Micro-controllers. ASIPs combine a software programmable processor and application-specific hardware component that can be used in a number of times in an instruction pipeline passion. Our instruction-set aims complete operation of SDR, which is the implementation of wireless network protocols, whose operation modes can be changed after design and development. This requires an inherent parallelism of functional HW.

This paper is organized as follows. In Section 2 discuses the design of architecture of the ASIP for SDR, Section 3 about the instruction-set design and their formats, Section 4 will discuss modeling and simulation of architecture and the instruction-set of an ASIP, in Section 5 we presented result and future work and concluding remarks are given in Section 6 and final section gives references.

2. DESIGN OF ARCHITECTURE OF ASIP FOR SDR

As the SDR functionality is real-time task, due to complexity involved in its algorithm and to improve flexibility and enhanceability to implement multiple and new coming standards an ASIP based approach has been chosen to give as flexibility as general-purpose processor and almost same performance as ASIC [1].

The architecture of the proposed block diagram is shown in Fig. 1. Our proposed architecture has single-bus architecture, for data and some dedicated paths used wherever necessary. The diagram shows the two kinds of special functional block diagrams: (i). Block diagrams calling them as base-band and Channel-coding sub-functionalities such as Modulation and Demodulation schemes, Channel coding and decoding, DUC, DDC, NOC, FIR/IIR/CIC, and FFT. These functional block diagrams are useful for the base-band coding and channel coding (ii). Special functions but general to any communication system are shown as EU in the architecture, such as COS/SINE, EXP, Random, SETP (set protocol type), FORS (for loop starting point), FORE (for loop ending point), Floating point divider, multiplier and format converters. (iii). Logic block, address generation, Program counter logic, PCL and other control logic blocks.

2.1 The functional definitions of sub-blocks in architectural diagram

- **Execution Unit (EU):** Assumes all the functional units it will implement other than mentioned in the architecture that is COS/SINE, EXP, FORS, FORE, COPY, FADS, etc. I/P reg. and O/P reg., may be clocked enable or direct enabled.

- **Address Generation Unit. (AGU) and Memory Units**

Seven separate memory units are proposed; out of which four of 256 X 32-bit size memories are for program memory floating-point look-up table, dynamic data and constants storage. Three memory units of size 256 X 16-bit for integers look-up table, dynamic data and constants storage. Interface to and from other I/O blocks are either doing the computations or to store the I/O and computation results. All the memory unit's addresses and their corresponding data should be display in Binary format only. The reason behind this condition is compatible to any kind of crosschecking and independent of the instruction-set design.

2.1.1 The following units are included their IP/OP registers in the block itself.

- **Protocol Stack Engine**:

It will run any one of chosen protocol depending on the IP of the ASIP and decided by Reset and Program start pointer. That is it will load one protocol lib. Among the available based on its IP. Ex: CDMA-2000, GSM, and so on.

- **Channel Coding and decoding:** depending upon the protocol chosen, any one or combination of the channel coding will be used. 1.Viterbi Encoding and decoding 2. Turbo Encoding and decoding 3. Reed-Solomon (RS) Encoding and decoding 4. Convolution Encoding and decoding and 5. Trellis-coded encoding and decoding
- **Modulation and demodulation schemes**: Ex: GMSK, 64-QAM

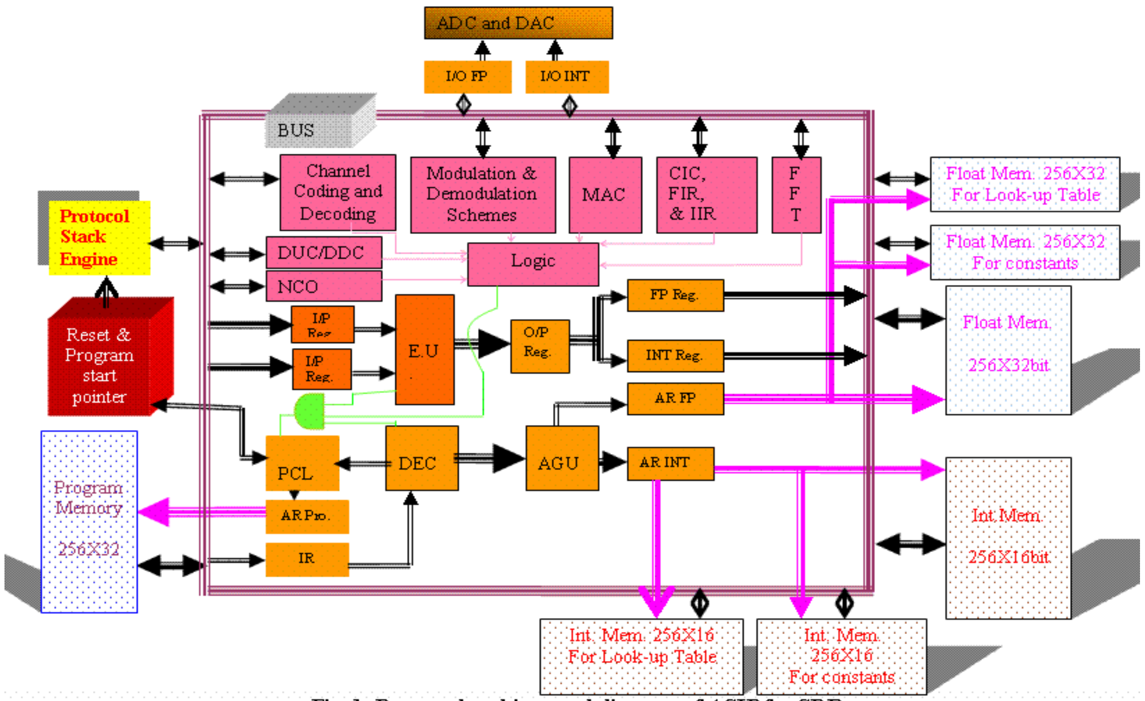
ADC and DAC
I/O FP
I/O INT
BUS
Channel Coding and Decoding
Modulation & Demodulation Schemes
MAC
CIC, FIR, & IIR
F F T
Float Mem. 256X32 For Look-up Table
Float Mem. 256X32 For constants
Protocol Stack Engine
DUC/DDC
NCO
Logic
I/P Reg
I/P Reg.
E.U
O/P Reg.
FP Reg.
INT Reg.
AR FP
Float Mem. 256X32bit
Reset & Program start pointer
PCL
DEC
AGU
AR INT
AR P10.
IR
Program Memory 256X32
Int.Mem. 256X16bit
Int. Mem. 256X16 For Look-up Table
Int. Mem. 256X16 For constants

Fig. 1. Proposed architectural diagram of ASIP for SDR

3. INSTRUCTION-SET DESIGN

SDR is a device which will support multiple standards and applications to the user in a simple and intelligent manner. The general architecture [3] can be viewed as a different-tire algorithms implementation to achieve the complete functionality. Hence we are providing two-tier functionality implementation of SDR algorithms. The top-tier is the complete implementation of the SDR functionality includes transmission, reception and corresponding protocol functionality and individual subroutines to obtain the top-tier functionality that is mathematical operations [7], protocol dependent base-band and channel coding methods and functions. We have decided to embed the second tier algorithms in the functional units (the architecture as mentioned in the above) as separate instructions along with other instructions

Our instruction-set design follows the hybrid approach to reduce the control bits. We are also assuming that each instruction will take one instruction cycle (three clock cycles), this assumption makes Instruction-set designer independent of the architecture design. Based on previous experience and requirements of the SDR implementation we have decided to have the following different instruction formats for designing the instruction-set

3.1 Different Instruction Formats that used

For Arithmetic/ copy, move etc. (Integer and Floating point): FMT-1

(6-bits) Op-code	(10-bits) ptr.1 Destination address	10-bit Source address	6-bits This field Not used.

For Integer Arithmetic/store with constant: FMT-2

(6-bits) Op code	(10-bits) ptr.1 Destination address	(16-bits) Integer constant

For floating-point Arithmetic/store with constant: FMT-3

(6-bits) Op code	(10-bits) ptr1 Destination address	10-bit Source address	6-bits This field Not used.
← (32-bits) floating-point constant →			

For integer branch: FMT-4

Op-code (6-bits)	(10 bits) ptr.1 Destination address	(4-bits) BC	(10-bits) BT	(2-bits) Not used
← (16-bits)Not used →		← (16-bits) Integer constant →		

For floating-point branch: FMT-5

Op-code (6-bits)	(10 bits) ptr 1 Destination address	(4-bits) BC	(10-bits) BT	Not used
← (32-bits) Floating-point constant →				

For Base-band and Channel coding instructions: FMT6.

Op- code (6-bit)	10-bit ptr.1 Destination address	10-bit Source address	6-bits to decide type.
← (32-bits) Extended mode →			

3.2 Total instructions and their category

3.2.1 Data move and program control (13): Total instructions in this category are 13 and are shown in table-1
3.2.2 Arithmetic (14): Total instructions in this category are 14 and are shown in table-1
3.2.3 Application Specific but general to communication system modeling (14): Total instructions in this category are 14 and are shown in table-1
3.2.4 Application Specific to base-band and channel coding (11): Total instructions in this category are 11 and are shown in table-1

4. INSTRUCTION-SET MODELING

To model and simulation of the architecture and instruction-set behavior, we have used SystemC language [2]. This will be used for modeling of hardware (HW) as well as software (SW) more details are given in [8].

5. RESULTS

To design instruction-set, we have chosen six different instruction formats for simple design and understanding and four classifications of the instruction groups have been finalized. They will be covering all the 52 instructions. The different formats are shown in section-3. The summery of the encoding of the op-code is given in table-2.

5.1 OP-code encoding

Table-1 shows that complete instruction-set encoding at the op-code level. Our encoding technique is simple and requires less complex circuitry to decode the instructions.

Table-1 Encoding of op-code field in the first, second and third group instructions

Group code	*Cat. code*	*Mem. code*	*Instruction*	*Group code*	*Cat. code*	*Mem. code*	*Instruction*
	0	000	COPY I,I		0	000	ADD I,I
	0	001	COPY I,C		0	001	ADD I,C
	0	010	NEG I		0	010	SUB I,I
	0	011	BRANCH I,I		0	011	SUB I,C

00 Data move and program control instructions	0	100	BRANCH I,C	01 Arithmetic instructions	0	100	MULT I, I
	0	101	COMPR I,I		0	101	MULT I,C
	0	110	CALL ( )		1	000	ADD F,F
	0	111	RETURN ( )		1	001	ADD F,C
	1	000	COPY F,F		1	010	SUB F,F
	1	001	COPY F,C		1	011	SUB F,C
	1	010	NEG F		1	100	MULT F,F
	1	011	Not defined		1	101	MULT F,C
	1	100	BRANCH I,C		1	110	DIVD F,F
	1	101	COMPR I,I		1	111	DIVD F,C
	0	000	COPY I,I		0	000	ADD I,I
					0	001	ADD I,C
					0	010	SUB I,I

Group code	***Cat. code***	***Mem. code***	***Instruction***	Group code	Member code	Instruction
10 Application specific but general to communication system design instructions	0	000	RND I,F	11 Application specific instructions for base band and channel coding	0000	CH_CODE( )
	0	001	TRUNC I,F		0001	CH_DECOD( )
	0	010	FOR_S( )		0010	DDC( )
	0	011	FOR_E( )		0011	DUC( )
	0	100	CONVT I,F		0100	NOC( )
	0	101	SETP( )		0101	FIR( )
	1	000	Fn_EXP F,F		0110	IIR( )
	1	001	Fn_COS F,F		0111	CIC( )
	1	010	Fn_SINE F,F		1000	FFT( )
	1	011	MAC F,F		1001	MOD( )
	1	100	DBLINSC( )		1010	DEMOD( )
	1	101	DOWNLOAD( )			
	1	110	GEN_NOISE ( )			
	0	111	CONVT F,I			

5.2. Simulation results

The complete architecture behavior (without actual timings) has been simulated. We have implemented most of the instructions of category 1, 2, and 3. The fourth category instructions are yet to be implemented. For modeling and implementation of architecture as well as instruction-set, we have chosen SystemC language. The results are showing correct logic and behavior. More details are given about instruction-set simulation in [8].

6. CONCLUDING REMARKS

In this paper we have discussed about the architecture design and instruction-set of an ASIP for SDR. We have given complete encoding of the op-code of the total instructions. Due to space limitation, this paper does not provides all the implemented instructions wave forms in ".vcd" format and results. So far we have modeled the entire data move, arithmetic instructions and some application specific instructions. In future we are going to incorporate these concepts to Reconfigurable ASIP (RASIP) for SDR.

7. REFERENCES

[1] K. Solomon Raju, "Architecture and design of processor for speech synthesis," Master's. Dissertation, Birla Institute of Technology and Science (BITS), Pilani, April-2003

[2] www.systemc.org

[3] Joe Mitola, "The software radio architecture", in IEEE Communications Magazine, Vol.33, No.5, pp26-37, May 1995.

[4] SystemC 2.0 user guide available at www.systemc.org

[5] J. Mitola III, "Software Radios: Survey, Critical Evaluation and Future Directions", in IEEE National Telesystems Conference, pp.13-15-13-23, 1992

[6] Walter H.W.Tutlebee, "Software-Defined Radio: Facets of a developing technology", from IEEE Personnel Communications Magazine, Vol.6, No.2, pp. 38-44, April-1999

[7] Joseph Motila III, "Software Radio Architecture: A Mathematical Perspective", in IEEE Journal on Selected Areas of Communications, Vol.17, No.4, pp.514-538, April 1999.

[8] K. Solomon Raju, Chandra Shekhar and R. C. Joshi "Behavioral Modeling and Simulation of Instruction-set of an ASIP for Software-Defined Radio", in National Conference on Issues and Trends in Wireless Networks (IT-WiNS), Patiala, pp.130-136, December-2004.