An Approach for the Formal Verification of DSP Designs Using Theorem Proving

Behzad Akbarpour and Sofiène Tahar, *Member, IEEE*

***Abstract*—This paper proposes a framework for the incorporation of formal methods in the design flow of digital signal processing (DSP) systems in a rigorous way. In the proposed approach, DSP descriptions were modeled and verified at different abstraction levels using higher order logic based on the higher order logic (HOL) theorem prover. This framework enables the formal verification of DSP designs that in the past could only be done partially using conventional simulation techniques. To this end, a shallow embedding of DSP descriptions in HOL at the floating-point (FP), fixed-point (FXP), behavioral, register transfer level (RTL), and netlist gate levels is provided. The paper made use of existing formalization of FP theory in HOL and a parallel one developed for FXP arithmetic. The high ability of abstraction in HOL allows a seamless hierarchical verification encompassing the whole DSP design path, starting from top-level FP and FXP algorithmic descriptions down to RTL, and gate level implementations. The paper illustrates the new verification framework on the fast Fourier transform (FFT) algorithm as a case study.**

***Index Terms*—Design automation, digital signal processors, error analysis, fast Fourier transforms, finite wordlength effects, formal verification, higher order logic, theorem proving.**

I. Introduction

DIGITAL system design is characterized by an ever increasing system complexity that has to be implemented within reduced time, resulting in minimum costs and short time-to-market. These characteristics call for a seamless design flow that allows to perform the design steps on the highest suitable level of abstraction. For most digital signal processing (DSP) systems, the design has to result in a fixed-point (FXP) implementation. This is due to the fact that these systems are sensitive to power consumption, chip size, and price per device. FXP realizations outperform floating-point (FP) realizations by far with regard to these criteria. Fig. 1 illustrates a general DSP design flow as used nowadays in leading industry design projects.

The design of DSP systems starts from an ideal real number specification. In the theoretical analysis of digital systems, we generally assume that signal values and system coefficients are represented in the real number system and expressed to infinite precision. This allows to ignore the effects of finite word lengths and fixed exponents and to abstract from all implementation details. When implemented in a special-purpose digital hardware or as a computer algorithm, we must represent signals and coefficients in some digital number system that must always be of finite precision. In this case, attention must be paid to the effects of using finite register lengths to represent all relevant design parameters [32]. Despite the advantages offered by digital networks, there is an inherent accuracy problem associated with DSP systems since the signals are represented by a finite number of bits and the arithmetic operations must be carried out with an accuracy limited by the finite word length of the number representation. Depending on the type of arithmetic used in the system algorithm, the type of quantization used to reduce the word length to a desired size, and the exact system structure used, one can generally estimate how system performance is affected by these finite precision effects. There are several types of arithmetics used in the implementation of digital systems. Among the most common are FP and FXP. At the FP and FXP levels, all operands are represented by a special format

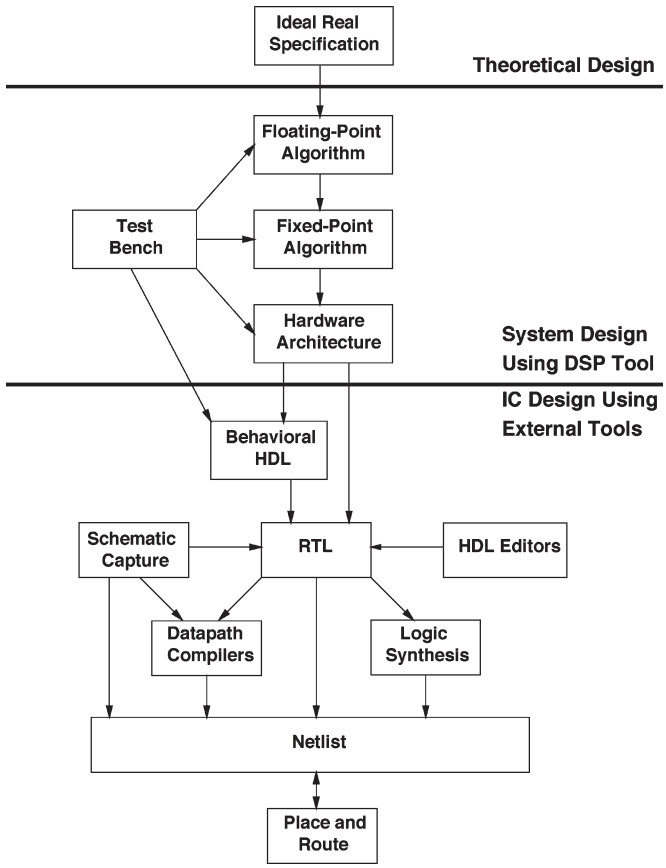

Fig. 1. DSP design flow.

Manuscript received December 1, 2004; revised May 25, 2005. This work was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) strategic research Grant STP234820. This paper is an extended and more generic version of an earlier publication in Formal Methods in Computer-Aided Design (FMCAD 2004) on FFT verification [2]. This paper was recommended by Associate Editor J. H. Kukula.

The authors are with the Department of Electrical and Computer Engineering, Concordia University, Montreal, QC H3G 1M8, Canada (e-mail: behzad@ece.concordia.ca; tahar@ece.concordia.ca).

Digital Object Identifier 10.1109/TCAD.2005.857314

or assigned a fixed word length and a fixed exponent, while the control structure and the operations of the ideal program remain unchanged. The transformation from real (numbers) to FP and FXP is quite tedious and error prone. On the implementation side, the FXP model of the algorithm has to be transformed/synthesized into the best-suited target description either using a hardware description language (HDL) or a programming language. Meeting the above (sometimes conflicting) requirements is a great challenge in any DSP design project.

The above design process can be aided by a number of specialized CAD tools such as Signal Processing WorkSystem (SPW, Cadence) [10], CoCentric (Synopsys) [11], Matlab-Simulink (Mathworks) [28], and FRIDGE (Aachen UT) [26]. The conformance of the FXP implementation with respect to descriptions in FP or real algorithm on one hand, and register transfer (RT) and gate levels on the other hand, is verified by simulation techniques. Simulation, however, is known to provide partial verification as it cannot cover all design errors, especially for large systems. On the other hand, adopting formal verification in system design generally means using methods of mathematical proof rather than simulation to ensure the quality of the design, to improve the robustness of a design, and to speed up the development. The overall aim of this paper is to propose a general methodology for the formalization and verification of DSP descriptions at different abstraction levels using higher order logic. To this end, we adopt a shallow embedding for DSP descriptions in which we translate the intended meaning of design blocks into higher order logic and then complete the formal proof in the HOL [17] theorem proving environment. To the best our knowledge, this is the first time that formal methods are applied to DSP modeling and verification in such a rigorous way.

The rest of the paper is organized as follows. Section II describes the proposed DSP formal verification methodology. Section III presents a case study verification of fast Fourier transform (FFT) algorithms in HOL from real number specification to RTL implementation. Section IV discusses related work. Finally, Section V concludes the paper and outlines future research directions.

II. Proposed DSP Verification Framework

In this paper, we propose a methodology for applying formal methods to the design flow of DSP systems in a rigorous way. The corresponding commutating diagram is shown in Fig. 2. Thereafter, we model the ideal real specification of the DSP algorithms and the corresponding FP and FXP representations as well as the RT and gate level implementations as predicates in higher order logic. The overall methodology for the formal specification and verification of DSP algorithms will be based on the idea of shallow embedding of languages [6] using the HOL theorem proving environment [17]. In the proposed approach, we first focus on the transition from real to FP and FXP levels. For this, we make use of existing theories in HOL on the construction of real [18] and complex [21] numbers, the formalization of IEEE-754 standard [24] based FP arithmetic [19], [20], and the formalization of FXP arithmetic [5]. We use valuation functions to find the real values of the FP and

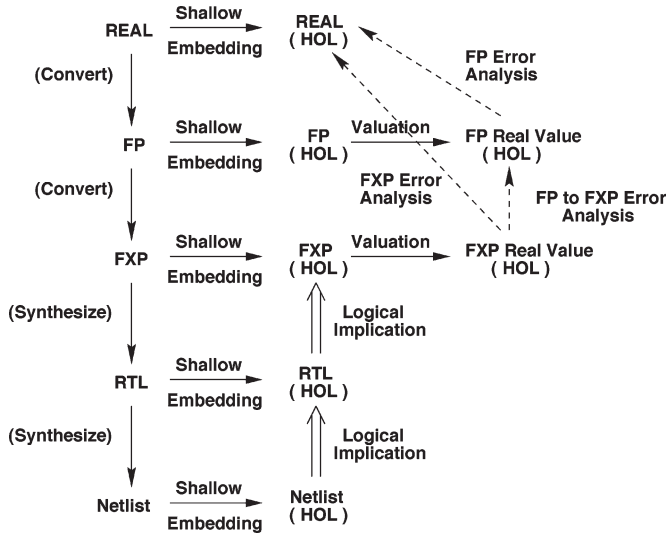

Fig. 2. Proposed DSP specification and verification approach.

FXP DSP outputs and define the error as the difference between these values and the corresponding output of the ideal real specification. Then, we establish fundamental lemmas on the error analysis of FP and FXP roundings and arithmetic operations against their abstract mathematical counterparts. Finally, based on these lemmas, we derive expressions for the accumulation of roundoff error in FP and FXP DSP algorithms using recursive definitions and initial conditions. While theoretical work on computing the errors due to finite precision effects in the realization of DSP algorithms with FP and FXP arithmetics has been extensively studied since the late 1960s [25], this paper contains the first formalization and proof of this analysis using a mechanical theorem prover, here HOL. The formal results are found to be in good agreement with the theoretical ones.

After handling the transition from real to FP and FXP levels, we turn to HDL representation. At this point, we use well-known techniques to model the DSP design at the RTL level within the HOL environment. The last step is to verify this level using a classical hierarchical proof approach in HOL [29]. In this way, we hierarchically prove that DSP RTL implementation implies high-level FXP algorithmic specification, which has already been related to the FP description and the ideal real specification through error analysis. The verification can be extended, following a similar manner, down to gate level netlist either in HOL or using other commercial verification tools as depicted in Fig. 2.

The process of specifying an HDL in higher order logic is commonly known as semantic embedding. There are two main approaches [6]: deep embedding and shallow embedding. In deep embedding, the abstract syntax of a design description is represented by terms, which are then interpreted by semantic functions defined in the logic that assign meaning to the design. With this method, it is possible to reason about classes of designs since one can quantify over the syntactic structures. However, setting up HOL types of abstract syntax and semantic functions can be very tedious. In a shallow embedding, on the other hand, the design is modeled directly by a formal specification of its functional behavior. This eliminates the

effort of defining abstract syntax and semantic functions, but it also limits the proofs to functional properties. In this paper, since our main concern is to check the correctness of the designs based on their functionality, we propose shallow embedding for DSP descriptions: translate the intended meaning of DSP block designs as described in its documentation into HOL and then complete the formal proof in the HOL theorem prover.

A. Application With SPW

In this section, we demonstrate how the proposed methodology can be used for the verification of an Integrator designed in SPW. The SPW of Cadence [10] is an integrated framework for developing DSP and communications products. It graphically represents a system as a network of functional blocks and comes with a vast library of DSP blocks, and users can also add their own blocks or build intellectual property (IP) blocks by composition of primitive blocks. SPW provides all the tools needed to interactively capture, simulate, test, and implement a broad range of DSP designs. Typical design applications include digital communication systems, image processing, multimedia, radar systems, control systems, digital audio, and high-definition television. SPW can be used to evaluate various architectural approaches to a design and to develop, simulate, and fine-tune algorithms. A design project in SPW typically consists of the same steps depicted in Fig. 1. More details about SPW design flow and the application of our methodology with it can be found in [4]. To briefly illustrate our approach, we show next the application of our methodology on a simple integrator designed in SPW.

A digital integrator is a discrete time system that transforms a sequence of input numbers into another sequence of output by means of a specific computational algorithm. To describe the general functionality of a digital integrator, let $\{x_t\}$, $\{w_t\}$, and a denote the input sequence, output sequence, and constant coefficient of the integrator, respectively. Then the integrator can be specified by the difference equation

$$w_t = x_{t-1} + aw_{t-1}. \tag{1}$$

Thereafter, the output sequence at time t is equal to the input sequence at time $t-1$, added to the output at time $t-1$ multiplied by the integrator coefficient.

Fig. 3 shows the SPW design of an integrator. The integrator is first designed and simulated using the SPW predefined FP blocks and parameters [Fig. 3(a)]. The design is composed of an adder $(M1)$, a multiplier by constant $(M2)$, and a delay $(M3)$ block, together with signal source $(M4)$ and sink $(M5)$ elements. The input signal, the output signal, and the output of the adder and multiplier blocks are labeled by IN′, OUT′, S1′, and S2′, respectively. Fig. 3(b) shows the converted FXP design in which each block is replaced with the corresponding FXP block $(M1', M2', M3', M4', M5')$. FXP blocks are shown by double circles and squares to distinguish them from FP blocks. The attributes of all FXP block outputs are set to (64, 31, t) to ensure that overflow and quantization do not affect system operation. The corresponding FXP signals are labeled by IN″, OUT″, S1″, and S2″.

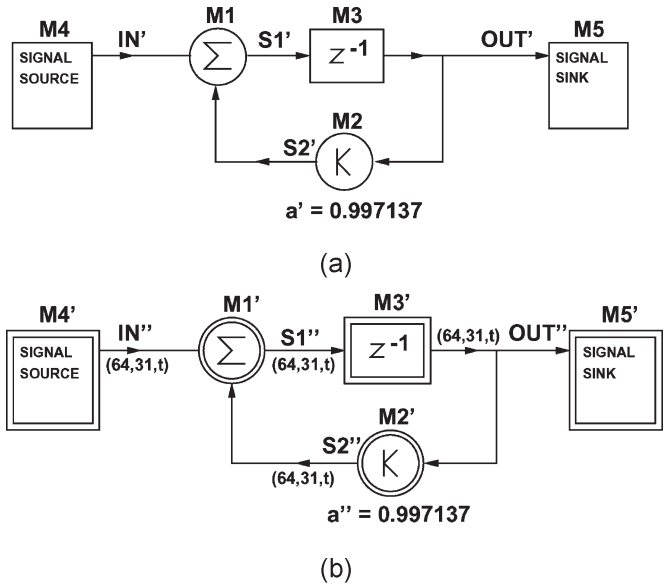

Fig. 3. SPW design of an integrator. (a) Floating-point design. (b) Fixed-point design.

In HOL, we first model the design at each level as predicates in higher order logic. The predicates corresponding to the FP design in IEEE 64-bit double precision format are

$$\begin{aligned}
\vdash_{def}\ & \texttt{Float_Gain_Block a' b' c'} \\
& = (\forall \texttt{t}.\ \texttt{c' t} = \texttt{a' t float_mul b'}) \\
\vdash_{def}\ & \texttt{Float_Delay_Block a' b'} \\
& = (\forall \texttt{t}.\ \texttt{b' t} = \texttt{a'}\ (\texttt{t} - 1)) \\
\vdash_{def}\ & \texttt{Float_Add_Block a' b' c'} \\
& = (\forall \texttt{t}.\ \texttt{c' t} = \texttt{a' t float_add b' t}) \\
\vdash_{def}\ & \texttt{Float_Integrator_Imp a' IN' OUT'} \\
& = \exists\, \texttt{S1' S2'}. \\
& \quad \texttt{Float_Add_Block IN' S2' S1'} \\
& \quad \wedge\ \texttt{Float_Delay_Block S1' OUT'} \\
& \quad \wedge\ \texttt{Float_Gain_Block OUT' a' S2'}.
\end{aligned}$$

The HOL description of the FXP implementation is

$$\begin{aligned}
\vdash_{def}\ & \texttt{Fxp_Gain_Block a'' b'' c''} \\
& = (\forall \texttt{t}.\ \texttt{c'' t} = \texttt{a'' t fxp_mul b''}) \\
\vdash_{def}\ & \texttt{Fxp_Delay_Block a'' b''} \\
& = (\forall \texttt{t}.\ \texttt{b'' t} = \texttt{a''}\ (\texttt{t} - 1)) \\
\vdash_{def}\ & \texttt{Fxp_Add_Block a'' b'' c''} \\
& = (\forall \texttt{t}.\ \texttt{c'' t} = \texttt{a'' t fxp_add b'' t}) \\
\vdash_{def}\ & \texttt{Fxp_Integrator_Imp a'' IN'' OUT''} \\
& = \exists\, \texttt{S1'' S2''}. \\
& \quad \texttt{Fxp_Add_Block IN'' S2'' S1''} \\
& \quad \wedge\ \texttt{Fxp_Delay_Block S1'' OUT''} \\
& \quad \wedge\ \texttt{Fxp_Gain_Block OUT'' a'' S2''}.
\end{aligned}$$

In the next step (shown at the bottom of the page), we describe each design as a difference equation relating the input and output samples according to (1).

The following theorems (Theorems 1 and 2) ensure that the implementation at each level satisfies the corresponding specification.

Theorem 1: FLOAT_INTEGRATOR_IMP_TO_SPEC_THM

$$\vdash \texttt{Float\_Integrator\_Imp}\ \texttt{a}'\ \texttt{IN}'\ \texttt{OUT}' \supset \texttt{Float\_Integrator\_Spec}\ \texttt{a}'\ \texttt{IN}'\ \texttt{OUT}'.$$

Theorem 2: FXP_INTEGRATOR_IMP_SPEC

$$\vdash \texttt{Fxp\_Integrator\_Imp}\ \texttt{a}''\ \texttt{IN}''\ \texttt{OUT}'' \supset \texttt{Fxp\_Integrator\_Spec}\ \texttt{a}''\ \texttt{IN}''\ \texttt{OUT}''.$$

Now we assume that the FP and FXP input sequences are the rounded versions of an infinite precision ideal input IN. We also make some other assumptions on finiteness and validity of FP and FXP inputs, coefficients, and intermediate results in order to have finite and valid final outputs. Using these assumptions, we proved the following theorem (Theorem 3) concerning the error between the real values of the FP and FXP precision integrator output samples.

Theorem 3: INTEGRATOR_FXP_TO_FLOAT_THM

$$\begin{aligned}\vdash\ & \texttt{Fxp\_Integrator\_Imp}\ \texttt{a}''\ \texttt{IN}''\ \texttt{OUT}'' \\ & \supset \texttt{Float\_Integrator\_Imp}\ \texttt{a}'\ \texttt{IN}'\ \texttt{OUT}' \\ & \wedge\ \texttt{Floaterror}\ \texttt{a}'\ \texttt{IN}'\ \texttt{OUT}' \\ & \wedge\ \texttt{Fxperror}\ \texttt{a}''\ \texttt{IN}''\ \texttt{OUT}''.\end{aligned}$$

According to this theorem, for a valid and finite set of input and output sequences at time $(t-1)$ to the integrator design at FP and FXP levels, we can have finite and valid outputs at time t, and the difference in real values corresponding to these output samples can be expressed as the difference in input and output values multiplied by the corresponding coefficients, taking into account the effects of finite precision in coefficients and arithmetic operations. The functions Floaterror and Fxperror represent the errors resulting from rounding the real operation results to a FXP and FP number, respectively. These errors are already quantified using the theorems mentioned in [5] for FXP arithmetic and the corresponding theorems for error analysis in the FP case [20].

Next, we generated with SPW the VHDL code corresponding to the filter design and used Synopsys to synthesize the gate level netlist. The resulting codes are then translated into HOL notation and the corresponding correctness theorems established as follows (Theorems 4 and 5).

Theorem 4: INTEGRATOR_Netlist_TO_RTL_THM

$$\vdash \texttt{Netlist\_Integrator\_Imp}\ \texttt{a}'''\ \texttt{IN}'''\ \texttt{OUT}''' \supset \texttt{RTL\_Integrator\_Imp}\ \texttt{a}'''\ \texttt{IN}'''\ \texttt{OUT}'''.$$

Theorem 5: INTEGRATOR_RTL_TO_FXP_THM

$$\begin{aligned}\vdash\ & \texttt{RTL\_Integrator\_Imp}\ \texttt{a}'''\ \texttt{IN}'''\ \texttt{OUT}''' \\ & \supset \texttt{Fxp\_Integrator\_Imp} \\ & \qquad \texttt{Fxp}\ (\texttt{a}''')\ \texttt{Fxp}\ (\texttt{IN}''')\ \texttt{Fxp}\ (\texttt{OUT}''').\end{aligned}$$

Here, the input and output signals IN‴ and OUT‴ are Boolean words. To relate them to the corresponding specifications in fixed-point and FP, we make use of the bijection functions Fxp [5] and Float [20], respectively. In the proof of these theorems, we used the modular behavior of the circuit so that we proved separate lemmas for different modules such as adder, multiplier, and delay, and then used these lemmas in the proof of the original theorems.

Finally, using the obtained Theorems 1–5, we can easily deduce our ultimate theorem (Theorem 6) proving the correctness of the FP specification from gate-level implementation, taking into account the error analysis computed beforehand.

Theorem 6: INTEGRATOR_Netlist_TO_FLOAT_THM

$$\begin{aligned}\vdash\ & \texttt{Netlist\_Integrator\_Imp}\ \texttt{a}'''\ \texttt{IN}'''\ \texttt{OUT}''' \\ & \supset \texttt{Float\_Integrator\_Spec} \\ & \qquad \texttt{Float}\ (\texttt{a}''')\ \texttt{Float}\ (\texttt{IN}''')\ \texttt{Float}(\texttt{OUT}''') \\ & \wedge\ \texttt{Floaterror}\ \texttt{a}'''\ \texttt{IN}'''\ \texttt{OUT}''' \\ & \wedge\ \texttt{Fxperror}\ \texttt{a}'''\ \texttt{IN}'''\ \texttt{OUT}'''.\end{aligned}$$

More details about this analysis to find the error bounds can be found in [5]. In the rest of the paper, we demonstrate in more detail how the error analysis and verification methodology presented in this section can be used for the verification of the FFT algorithms implemented in different canonical forms of realization. A similar discussion can be applied to other types of signal analysis algorithms.

III. Case Study: FFT Error Analysis and Verification

FFT [8], [12] is a highly efficient method for computing the discrete Fourier transform (DFT) coefficients of a finite sequence of complex data. Because of the substantial time saving over conventional methods [31], FFT has found important applications in a number of diverse fields such as spectrum analysis, speech and optical signal processing, and digital filter design. FFT algorithms are based on the fundamental principle of decomposing the computation of the DFT of a

---

$$\vdash_{def}\ \texttt{FLOAT\_Integrator\_Spec}\ \texttt{X}\ \texttt{a}'\ \texttt{IN}'\ \texttt{OUT}' = \forall \texttt{t}.\ \texttt{OUT}'\ \texttt{t} = (\texttt{IN}'\ (\texttt{t}-1)\ \texttt{float\_add}\ (\texttt{a}'\ \texttt{float\_mul}\ \texttt{OUT}'\ (\texttt{t}-1)))$$

$$\begin{aligned}\vdash_{def}\ & \texttt{FXP\_Integrator\_Spec}\ \texttt{X}'\ \texttt{o\_mode}\ \texttt{q\_mode}\ \texttt{n\_bits}\ \texttt{a}''\ \texttt{IN}''\ \texttt{OUT}'' = \forall \texttt{t}.\ \texttt{OUT}''\ \texttt{t} \\ & = (\texttt{IN}''\ (\texttt{t}-1)\ \texttt{fxp\_add}\ (\texttt{a}''\ \texttt{fxp\_mul}\ \texttt{OUT}''\ (\texttt{t}-1)))\end{aligned}$$

finite-length sequence of length N into successively smaller DFTs. The manner in which this principle is implemented leads to a variety of different algorithms, all with comparable improvements in computational speed. There are two basic classes of FFT algorithms for which the number of arithmetic multiplications and additions as a measure of computational complexity is proportional to $N \log N$ rather than N^2 as in conventional methods. The first, proposed by Cooley and Tukey [13], called decimation-in-time (DIT), derives its name from the fact that in the process of arranging the computation into smaller transformations the input sequence (generally thought of as a time sequence) is decomposed into successively smaller subsequences. In the second general class of algorithms proposed by Gentleman and Sande [16], the sequence of DFT coefficients is decomposed into smaller subsequences, hence its name decimation-in-frequency (DIF).

As our case study in this paper, we consider the formal verification of the DIT and DIF FFT algorithms. We used our methodology to derive expressions for the accumulation of roundoff error in FP and FXP FFT algorithms by recursive definitions and initial conditions, considering the effects of input quantization and inaccuracy in the coefficients. Based on the extensively studied theoretical work on computing the errors due to finite precision effects in the realization of FFT algorithms with FP and FXP arithmetics [25], we perform a similar analysis using the HOL theorem proving environment. The formal results are found to be in good agreement with the theoretical ones. Finally, we prove hierarchically that the FFT RTL implementation implies the high-level FXP algorithmic specification that has already been related to the FP description and ideal real specification through error analysis.

A. Error Analysis of FFT Algorithms in HOL

In this section, the principal results for accumulation of error in FFT algorithms using HOL theorem proving are derived and summarized. For the most part, the following discussion is phrased in terms of the radix-2 algorithm. However, most of the ideas employed in the error analysis of radix-2 algorithms can be utilized in the analysis of other algorithms. In the following, we first analyze error in the DIF FFT algorithm. Then, we perform a similar analysis for the DIT FFT algorithm. In either case, we will first describe in detail the theory behind the analysis and then explain how this analysis is performed in HOL.

1) DIF FFT Algorithm: The DFT of a sequence $\{x(n)\}_{n=0}^{N-1}$ is defined as [31]

$$A(p) = \sum_{n=0}^{N-1} x(n)(W_N)^{np}, \qquad p = 0, 1, 2, \ldots, N-1 \quad (2)$$

where $W_N = e^{-j2\pi/N}$ and $j = \sqrt{-1}$. The multiplicative factors $(W_N)^{np}$ are called twiddle factors. For simplicity, our discussion is restricted to the radix-2 FFT algorithm, in which the number of points N to be Fourier transformed satisfies the relationship $N = 2^m$, where m is an integer value. The results can be extended to radices other than 2. By using the FFT method, the Fourier coefficients $\{A(p)\}_{p=0}^{N-1}$ can be calculated in $m = \log_2 N$ iterative steps. At each step, an array of N complex numbers is generated by using only the numbers in the previous array. To explain the FFT algorithm, let each integer p, $p = 0, 1, 2, \ldots, N-1$, be expanded into a binary form as

$$p = 2^{m-1}p_0 + 2^{m-2}p_1 + \cdots + 2p_{m-2} + p_{m-1}, \quad p_k = 0 \text{ or } 1 \quad (3)$$

and let p^* denote the number corresponding to the reverse bit sequences of p, i.e.,

$$p^* = 2^{m-1}p_{m-1} + 2^{m-2}p_{m-2} + \cdots + 2p_1 + p_0. \quad (4)$$

Let $\{A_k(p)\}_{p=0}^{N-1}$ denote the N complex numbers calculated at the kth step. Then, the DIF FFT algorithm [25] can be expressed as

$$A_{k+1}(p) = \begin{cases} A_k(p) + A_k(p + 2^{m-1-k}), & \text{if } p_k = 0 \\ \left[A_k(p - 2^{m-1-k}) - A_k(p)\right] w_k(p), & \text{if } p_k = 1 \end{cases} \quad (5)$$

where $w_k(p)$ is a power of W_N given by $w_k(p) = (W_N)^{z_k(p)}$ and

$$z_k(p) = 2^k(2^{m-1-k}p_k + 2^{m-2-k}p_{k+1} + \cdots + 2p_{m-2} + p_{m-1}) - 2^{m-1}p_k. \quad (6)$$

Equation (5) is carried out for $k = 0, 1, 2, \ldots, m-1$, with $A_0(p) = x(p)$. It can be shown [16] that at the last step $\{A_m(p)\}_{p=0}^{N-1}$ are the discrete Fourier coefficients in rearranged order. Specifically, $A_m(p) = A(p^*)$ with p and p^* expanded and defined as in (3) and (4), respectively. Fig. 4 shows the signal flowgraph of the actual computation for the case $N = 2^4$.

There are three common sources of errors associated with the FFT algorithms [25], namely:

1) input quantization: caused by the quantization of the input signal $\{x_n\}$ into a set of discrete levels;
2) coefficient accuracy: caused by the representation of the coefficients $\{w_k(p)\}$ by a finite word length;
3) roundoff accumulation: caused by the accumulation of roundoff errors at arithmetic operations.

Therefore, the actual array computed by using (5) is in general different from $\{A_k(p)\}_{p=0}^{N-1}$. We denote the actual FP and FXP computed arrays by $\{A'_k(p)\}_{p=0}^{N-1}$ and $\{A''_k(p)\}_{p=0}^{N-1}$, respectively. Then, we define the corresponding errors of the pth element at step k as

$$e_k(p) = A'_k(p) - A_k(p) \quad (7)$$

$$e'_k(p) = A''_k(p) - A_k(p) \quad (8)$$

$$e''_k(p) = A''_k(p) - A'_k(p) \quad (9)$$

where $e_k(p)$ and $e'_k(p)$ are the errors between the actual FP and FXP implementations and the ideal real specification, respectively. $e''_k(p)$ is the error in the transition from FP to FXP.

In analyzing the effect of FP roundoff, the effect of rounding will be represented multiplicatively. Let $*$ denote any of the

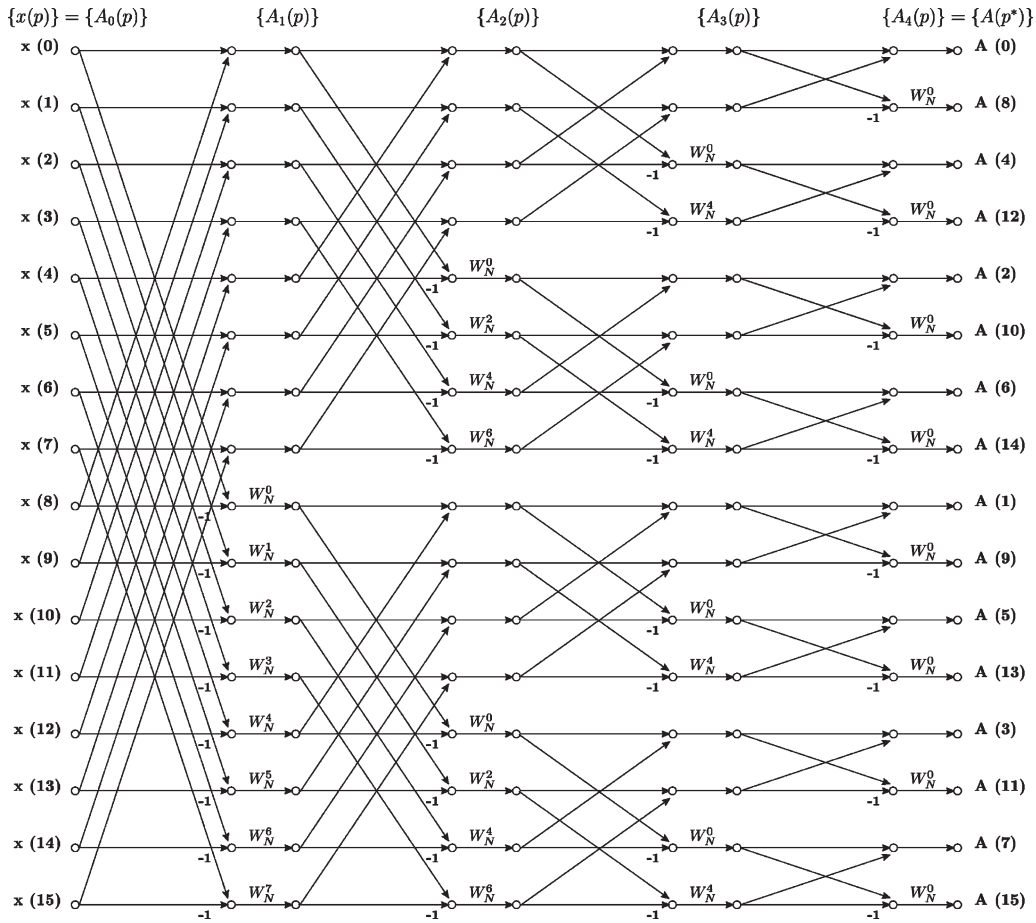

Fig. 4. Signal flowgraph of DIF FFT, $N = 2^4$.

arithmetic operations $+$, $-$, $\times$, /. It is known [14], [36] that if p represents the precision of the FP format, then

$$fl(x * y) = (x * y)(1 + \delta), \text{ where } |\delta| \leq 2^{-p}. \quad (10)$$

The notation $fl(.)$ is used to denote that the operation is performed using FP arithmetic. The above theorem relates the FP arithmetic operations such as addition, subtraction, multiplication, and division to their abstract mathematical counterparts according to the corresponding errors.

While the rounding error for FP arithmetic enters into the system multiplicatively, it is an additive component for FXP arithmetic. In this case, the fundamental error analysis theorem for FXP arithmetic operations against their abstract mathematical counterparts can be stated as

$$fxp(x * y) = (x * y) + \epsilon, \quad \text{where } |\epsilon| \leq 2^{-\text{fracbits}(X)} \quad (11)$$

and fracbits is the number of bits that are to the right of the binary point in the given FXP format X. The notation $fxp(.)$ is used to denote that the operation is performed using FXP arithmetic. We have proved (10) and (11) as theorems in higher order logic within HOL. These theorems are proved under the assumption that there is no overflow or underflow in the operation result. This means that the input values are scaled so that the actual value of the result is located in the ranges defined by the maximum and minimum representable values of the given FP and FXP formats. The details can be found in [3].

In (5), $\{A_k(p)\}$ are complex numbers, so their real and imaginary parts are calculated separately. Let

$$\begin{aligned} B_k(p) &= \text{Re}\,[A_k(p)] \\ C_k(p) &= \text{Im}\,[A_k(p)] \\ U_k(p) &= \text{Re}\,[w_k(p)] \\ V_k(p) &= \text{Im}\,[w_k(p)] \end{aligned} \quad (12)$$

where the notations Re[.] and Im[.] denote, respectively, real and imaginary parts of the quantity inside the bracket [.]. Equation (5) can be rewritten as

$$\left.\begin{aligned} B_{k+1}(p) &= B_k(p) + B_k(q) \\ C_{k+1}(p) &= C_k(p) + C_k(q) \end{aligned}\right\}, \quad \text{if } p_k = 0$$

$$\left.\begin{aligned} B_{k+1}(p) &= [B_k(r) - B_k(p)]\,U_k(p) \\ &\quad - [C_k(r) - C_k(p)]\,V_k(p) \\ C_{k+1}(p) &= [C_k(r) - C_k(p)]\,U_k(p) \\ &\quad + [B_k(r) - B_k(p)]\,V_k(p) \end{aligned}\right\}, \quad \text{if } p_k = 1 \quad (13)$$

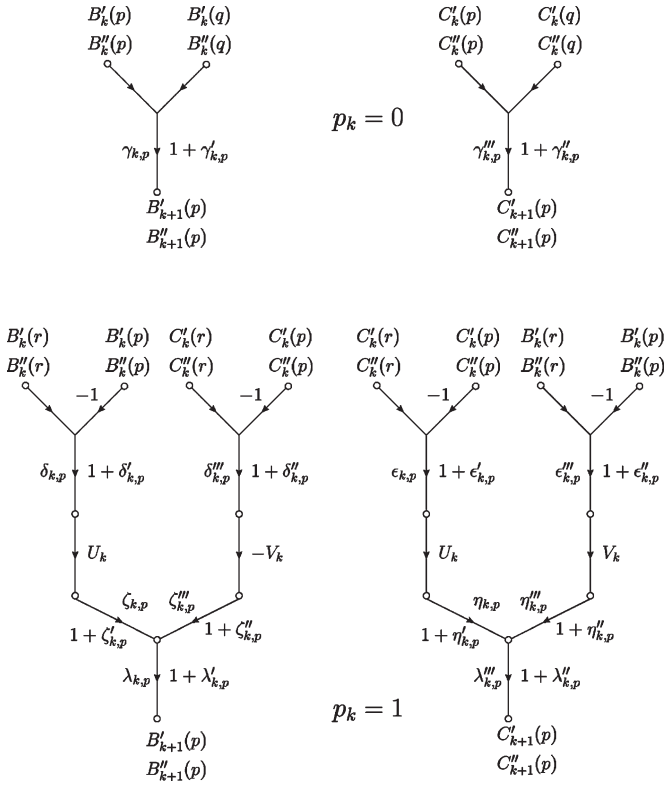

Fig. 5. Error flowgraph for DIF FFT.

where $q = p + 2^{m-1-k}$ and $r = p - 2^{m-1-k}$. On using prime and double prime to denote the calculated FP and FXP results, the real and imaginary parts of $A'_{k+1}(p)$ and $A''_{k+1}(p)$ are given, respectively, by

$$\left.\begin{aligned} B'_{k+1}(p) &= fl\left\{B'_k(p) + B'_k(q)\right\} \\ C'_{k+1}(p) &= fl\left\{C'_k(p) + C'_k(q)\right\} \end{aligned}\right\}, \quad \text{if } p_k = 0$$

$$\left.\begin{aligned} B'_{k+1}(p) &= fl\left\{\left[B'_k(r) - B'_k(p)\right] U_k(p) \right. \\ &\qquad \left. - \left[C'_k(r) - C'_k(p)\right] V_k(p)\right\} \\ C'_{k+1}(p) &= fl\left\{\left[C'_k(r) - C'_k(p)\right] U_k(p) \right. \\ &\qquad \left. + \left[B'_k(r) - B'_k(p)\right] V_k(p)\right\} \end{aligned}\right\}, \quad \text{if } p_k = 1 \tag{14}$$

$$\left.\begin{aligned} B''_{k+1}(p) &= fxp\left\{B''_k(p) + B''_k(q)\right\} \\ C''_{k+1}(p) &= fxp\left\{C''_k(p) + C''_k(q)\right\} \end{aligned}\right\}, \quad \text{if } p_k = 0$$

$$\left.\begin{aligned} B''_{k+1}(p) &= fxp\left\{\left[B''_k(r) - B''_k(p)\right] U_k(p) \right. \\ &\qquad \left. - \left[C''_k(r) - C''_k(p)\right] V_k(p)\right\} \\ C''_{k+1}(p) &= fxp\left\{\left[C''_k(r) - C''_k(p)\right] U_k(p) \right. \\ &\qquad \left. + \left[B''_k(r) - B''_k(p)\right] V_k(p)\right\} \end{aligned}\right\}, \quad \text{if } p_k = 1. \tag{15}$$

The corresponding error flowgraph showing the effect of roundoff error using the fundamental FP and FXP error analysis theorems according to (10) and (11), respectively, is given in Fig. 5, which also indicates the order of calculation.

The quantities $\gamma'_{k,p}$, $\gamma''_{k,p}$, $\delta'_{k,p}$, $\delta''_{k,p}$, $\epsilon'_{k,p}$, $\epsilon''_{k,p}$, $\zeta'_{k,p}$, $\zeta''_{k,p}$, $\eta'_{k,p}$, $\eta''_{k,p}$, $\lambda'_{k,p}$, and $\lambda''_{k,p}$ in Fig. 5 are errors caused by FP roundoff at each arithmetic step. The corresponding error quantities for FXP roundoff are $\gamma_{k,p}$, $\gamma'''_{k,p}$, $\delta_{k,p}$, $\delta'''_{k,p}$, $\epsilon_{k,p}$, $\epsilon'''_{k,p}$, $\zeta_{k,p}$, $\zeta'''_{k,p}$, $\eta_{k,p}$, $\eta'''_{k,p}$, $\lambda_{k,p}$, and $\lambda'''_{k,p}$. Thereafter, the actual real and imaginary parts of the FP and FXP outputs $A'_{k+1}(p)$ and $A''_{k+1}(p)$, respectively, can be given explicitly by

$$\left.\begin{aligned} B'_{k+1}(p) &= \left[B'_k(p) + B'_k(q)\right]\left(1+\gamma'_{k,p}\right) \\ C'_{k+1}(p) &= \left[C'_k(p) + C'_k(q)\right]\left(1+\gamma''_{k,p}\right) \end{aligned}\right\}, \quad \text{if } p_k = 0$$

$$\left.\begin{aligned} B'_{k+1}(p) &= \left[B'_k(r) - B'_k(p)\right] U_k(p) \\ &\quad \times \left(1+\delta'_{k,p}\right)\left(1+\zeta'_{k,p}\right)\left(1+\lambda'_{k,p}\right) \\ &\quad - \left[C'_k(r) - C'_k(p)\right] V_k(p) \\ &\quad \times \left(1+\delta''_{k,p}\right)\left(1+\zeta''_{k,p}\right)\left(1+\lambda'_{k,p}\right) \\ C'_{k+1}(p) &= \left[C'_k(r) - C'_k(p)\right] U_k(p) \\ &\quad \times \left(1+\epsilon'_{k,p}\right)\left(1+\eta'_{k,p}\right)\left(1+\lambda''_{k,p}\right) \\ &\quad + \left[B'_k(r) - B'_k(p)\right] V_k(p) \\ &\quad \times \left(1+\epsilon''_{k,p}\right)\left(1+\eta''_{k,p}\right)\left(1+\lambda''_{k,p}\right) \end{aligned}\right\}, \quad \text{if } p_k = 1 \tag{16}$$

and

$$\left.\begin{aligned} B''_{k+1}(p) &= \left[B''_k(p) + B''_k(q)\right] + \gamma_{k,p} \\ C''_{k+1}(p) &= \left[C''_k(p) + C''_k(q)\right] + \gamma'''_{k,p} \end{aligned}\right\}, \quad \text{if } p_k = 0$$

$$\left.\begin{aligned} B''_{k+1}(p) &= \left[B''_k(r) - B''_k(p) + \delta_{k,p}\right] \\ &\quad \times U_k(p) + \zeta_{k,p} \\ &\quad - \left(\left[C''_k(r) - C''_k(p) + \delta'''_{k,p}\right]\right. \\ &\qquad \left. \times V_k(p) + \zeta'''_{k,p}\right) + \lambda_{k,p} \\ C''_{k+1}(p) &= \left[C''_k(r) - C''_k(p) + \epsilon_{k,p}\right] \\ &\quad \times U_k(p) + \eta_{k,p} \\ &\quad + \left(\left[B''_k(r) - B''_k(p) + \epsilon'''_{k,p}\right]\right. \\ &\qquad \left. \times V_k(p) + \eta'''_{k,p}\right) + \lambda'''_{k,p} \end{aligned}\right\}, \quad \text{if } p_k = 1. \tag{17}$$

The errors $e_k(p)$, $e'_k(p)$, and $e''_k(p)$ defined in (7)–(9) are complex and can be rewritten as

$$e_k(p) = B'_k(p) - B_k(p) + j\left[C'_k(p) - C_k(p)\right] \tag{18}$$

$$e'_k(p) = B''_k(p) - B_k(p) + j\left[C''_k(p) - C_k(p)\right] \tag{19}$$

$$e''_k(p) = B''_k(p) - B'_k(p) + j\left[C''_k(p) - C'_k(p)\right] \tag{20}$$

$$k = 1, 2, \ldots, m, \quad p = 0, 1, \ldots, N-1$$

with

$$e_0(p) = e'_0(p) = e''_0(p) = 0, \qquad p = 0, 1, \ldots, N-1. \tag{21}$$

From (13) and (16)–(20), we derive the following error analysis cases.

1) DIF FFT real to FP

$$e_{k+1}(p) = \begin{cases} e_k(p) + e_k(q) + f_k(p), & \text{if } p_k = 0 \\ \left[e_k(r) - e_k(p)\right] w_k(p) + f_k(p), & \text{if } p_k = 1 \end{cases} \tag{22}$$

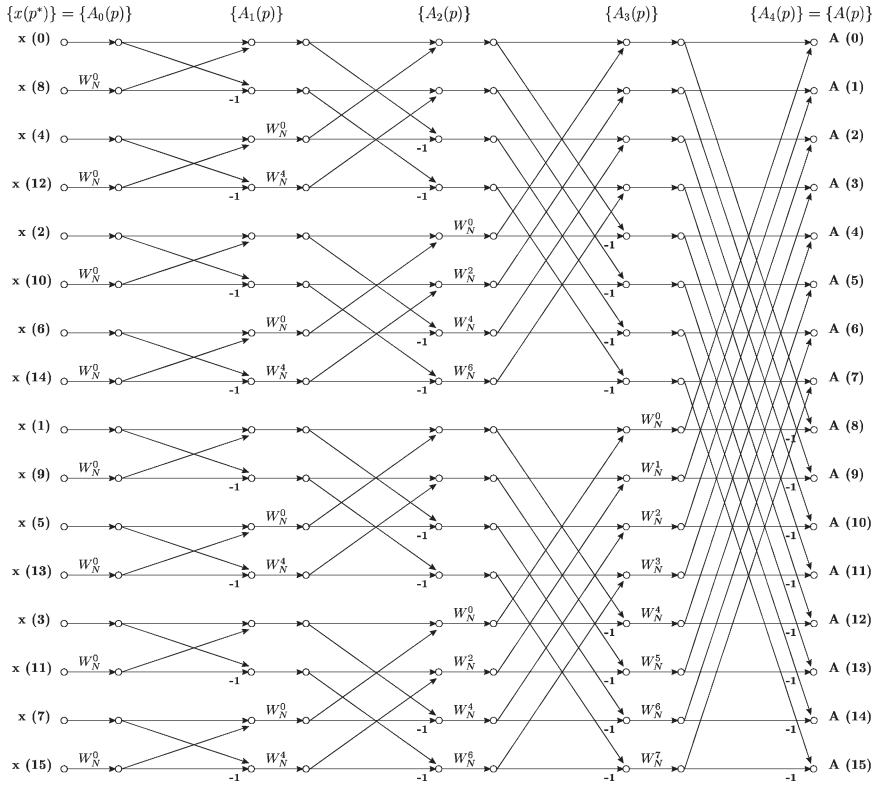

Fig. 6. Signal flowgraph of DIT FFT, $N = 2^4$.

where $f_k(p)$ is given by (23), shown at the bottom of the page.

2) DIF FFT real to FXP

$$e'_{k+1}(p) = \begin{cases} e'_k(p) + e'_k(q) + f'_k(p), & \text{if } p_k = 0 \\ [e'_k(r) - e'_k(p)]\, w_k(p) + f'_k(p), & \text{if } p_k = 1 \end{cases} \tag{24}$$

where $f'_k(p)$ is given by (25), shown at the bottom of the next page.

3) DIF FFT FP to FXP

$$e''_{k+1}(p) = \begin{cases} e''_k(p) + e''_k(q) + f'_k(p) - f_k(p), & \text{if } p_k = 0 \\ [e''_k(r) - e''_k(p)]\, w_k(p) \\ \quad + f'_k(p) - f_k(p), & \text{if } p_k = 1 \end{cases} \tag{26}$$

where $f_k(p)$ and $f'_k(p)$ are given by (23) and (25).

The accumulation of roundoff error is determined by the recursive equations (22)–(26), with initial conditions given by (21).

2) DIT FFT Algorithm: Let $\{A_k(p)\}_{p=0}^{N-1}$ denote the N complex numbers calculated at the kth step. Then the DIT FFT algorithm [27] can be expressed as

$$A_{k+1}(p) = \begin{cases} A_k(p) + w_k(p) A_k(p + 2^k), & \text{if } p_{m-1-k} = 0 \\ A_k(p - 2^k) - w_k(p) A_k(p), & \text{if } p_{m-1-k} = 1 \end{cases} \tag{27}$$

where $w_k(p)$ is a power of W_N given by $w_k(p) = (W_N)^{z_k(p)}$, where

$$z_k(p) = 2^{m-1-k}(2^k p_{m-1-k} + 2^{k-1} p_{m-k} + \cdots + 2p_{m-2} + p_{m-1}) - 2^{m-1} p_{m-1-k}. \tag{28}$$

Equation (27) is carried out for $k = 0, 1, 2, \ldots, m-1$, with $A_0(p) = x(p^*)$, where p and p^* are expanded and defined as in (7) and (8), respectively. It can be shown [13] that at the last step $\{A_m(p)\}_{p=0}^{N-1}$ are the discrete Fourier coefficients in normal order. Specifically, $A_m(p) = A(p)$. Fig. 6 shows the signal flowgraph of the actual computation for the case $N = 2^4$.

$$f_k(p) = \begin{cases} \gamma'_{k,p}\,[B'_k(p) + B'_k(q)] + j\gamma''_{k,p}\,[C'_k(p) + C'_k(q)]\,, & \text{if } p_k = 0 \\ \left[\left(1 + \delta'_{k,p}\right)\left(1 + \zeta'_{k,p}\right)(1 + \lambda_{k,p}) - 1\right][B'_k(r) - B'_k(p)]\, U_k(p) \\ -\left[\left(1 + \delta''_{k,p}\right)\left(1 + \zeta''_{k,p}\right)\left(1 + \lambda'_{k,p}\right) - 1\right][C'_k(r) - C'_k(p)]\, V_k(p) \\ + j\left[\left(1 + \epsilon'_{k,p}\right)\left(1 + \eta'_{k,p}\right)\left(1 + \lambda''_{k,p}\right) - 1\right][C'_k(r) - C'_k(p)]\, U_k(p) \\ + j\left[\left(1 + \epsilon''_{k,p}\right)\left(1 + \eta''_{k,p}\right)\left(1 + \lambda''_{k,p}\right) - 1\right][B'_k(r) - B'_k(p)]\, V_k(p), & \text{if } p_k = 1 \end{cases} \tag{23}$$

Similar to the discussion of error analysis of DIF FFT, we first rewrite (27) using real and imaginary parts as

$$\left.\begin{aligned} B_{k+1}(p) &= B_k(p) + U_k(p)B_k(q) \\ &\quad - V_k(p)C_k(q) \\ C_{k+1}(p) &= C_k(p) + U_k(p)C_k(q) \\ &\quad + V_k(p)B_k(q) \end{aligned}\right\}, \quad \text{if } p_{m-1-k} = 0$$
$$\left.\begin{aligned} B_{k+1}(p) &= B_k(r) - U_k(p)B_k(p) \\ &\quad + V_k(p)B_k(q) \\ C_{k+1}(p) &= C_k(r) - U_k(p)C_k(p) \\ &\quad - V_k(p)B_k(p) \end{aligned}\right\}, \quad \text{if } p_{m-1-k} = 1 \tag{29}$$

where $q = p + 2^k$ and $r = p - 2^k$. We also use prime and double prime to denote the calculated FP and FXP results as $A'_{k+1}(p)$ and $A''_{k+1}(p)$. Similarly, we can express the real and imaginary parts of $A'_{k+1}(p)$, $B'_{k+1}(p)$, and $C'_{k+1}(p)$, and $A''_{k+1}(p)$, $B''_{k+1}(p)$, and $C''_{k+1}(p)$, using the FP and FXP operations, respectively, i.e.,

$$\left.\begin{aligned} B'_{k+1}(p) &= fl\{B'_k(p) + U_k(p)B'_k(q) \\ &\quad - V_k(p)C'_k(q)\} \\ C'_{k+1}(p) &= fl\{C'_k(p) + U_k(p)C'_k(q) \\ &\quad + V_k(p)B'_k(q)\} \end{aligned}\right\}, \quad \text{if } p_{m-1-k} = 0$$
$$\left.\begin{aligned} B'_{k+1}(p) &= fl\{B'_k(r) - U_k(p)B'_k(p) \\ &\quad + V_k(p)B'_k(q)\} \\ C'_{k+1}(p) &= fl\{C'_k(r) - U_k(p)C'_k(p) \\ &\quad - V_k(p)B'_k(p)\} \end{aligned}\right\}, \quad \text{if } p_{m-1-k} = 1 \tag{30}$$

$$\left.\begin{aligned} B''_{k+1}(p) &= fxp\{B''_k(p) + U_k(p)B''_k(q) \\ &\quad - V_k(p)C''_k(q)\} \\ C''_{k+1}(p) &= fxp\{C''_k(p) + U_k(p)C''_k(q) \\ &\quad + V_k(p)B''_k(q)\} \end{aligned}\right\}, \quad \text{if } p_{m-1-k} = 0$$
$$\left.\begin{aligned} B''_{k+1}(p) &= fxp\{B''_k(r) - U_k(p)B''_k(p) \\ &\quad + V_k(p)B''_k(q)\} \\ C''_{k+1}(p) &= fxp\{C''_k(r) - U_k(p)C''_k(p) \\ &\quad - V_k(p)B''_k(p)\} \end{aligned}\right\}, \quad \text{if } p_{m-1-k} = 1. \tag{31}$$

The corresponding error flowgraph showing the effect of roundoff error using the fundamental FP and FXP error analysis theorems according to (10) and (11), respectively, is given in Fig. 7, which also indicates the order of calculation.

The quantities $\gamma'_{k,p}$, $\gamma''_{k,p}$, $\delta'_{k,p}$, $\delta''_{k,p}$, $\epsilon'_{k,p}$, $\epsilon''_{k,p}$, $\zeta'_{k,p}$, $\zeta''_{k,p}$, $\eta'_{k,p}$, $\eta''_{k,p}$, $\lambda'_{k,p}$, $\lambda''_{k,p}$, $\alpha'_{k,p}$, $\alpha''_{k,p}$, $\beta'_{k,p}$, and $\beta''_{k,p}$ in Fig. 7

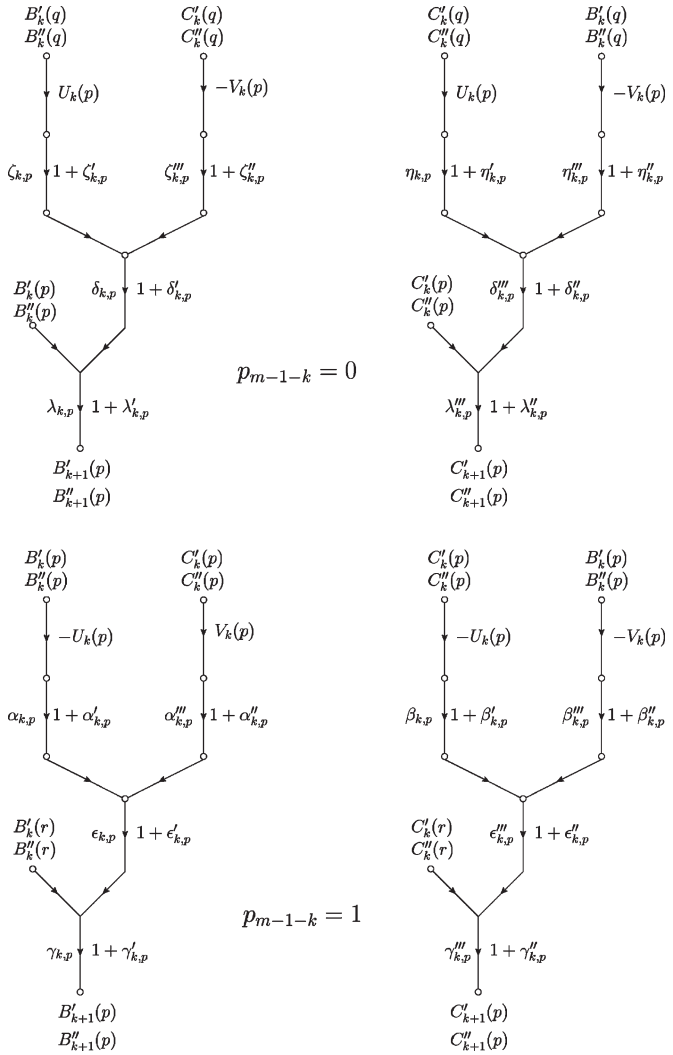

Fig. 7. Error flowgraph for DIT FFT.

are errors caused by FP roundoff at each arithmetic step. The corresponding error quantities for FXP roundoff are $\gamma_{k,p}$, $\gamma'''_{k,p}$, $\delta_{k,p}$, $\delta'''_{k,p}$, $\epsilon_{k,p}$, $\epsilon'''_{k,p}$, $\zeta_{k,p}$, $\zeta'''_{k,p}$, $\eta_{k,p}$, $\eta'''_{k,p}$, $\lambda_{k,p}$, $\lambda'''_{k,p}$, $\alpha_{k,p}$, $\alpha'''_{k,p}$, $\beta_{k,p}$, and $\beta'''_{k,p}$. Thereafter, the actual real and imaginary parts of the FP and FXP outputs $A'_{k+1}(p)$ and $A''_{k+1}(p)$, respectively, can be given explicitly by (32) and (33), shown at the bottom of the next page.

From (29), (32), and (33), we derive the following error analysis cases.

1) DIT FFT real to FP

$$e_{k+1}(p) = \begin{cases} e_k(p) + e_k(q)w_k(p) + f_k(p), & \text{if } p_{m-1-k} = 0 \\ e_k(r) - e_k(p)w_k(p) + f_k(p), & \text{if } p_{m-1-k} = 1 \end{cases} \tag{34}$$

$$f'_k(p) = \begin{cases} \gamma_{k,p} + j\gamma''_{k,p}, & \text{if } p_k = 0 \\ \delta_{k,p}U_k(p) + \zeta_{k,p} - \delta'''_{k,p}V_k(p) - \zeta'''_{k,p} + \lambda_{k,p} & \\ \quad + j\left(\epsilon_{k,p}U_k(p) + \eta_{k,p} + \epsilon'''_{k,p}V_k(p) + \eta'''_{k,p} + \lambda'''_{k,p}\right), & \text{if } p_k = 1 \end{cases} \tag{25}$$

where $f_k(p)$ is given by (35), shown at the bottom of the page.

2) DIT FFT real to FXP

$$e'_{k+1}(p) = \begin{cases} e'_k(p) + e'_k(q)w_k(p) + f'_k(p), & \text{if } p_{m-1-k} = 0 \\ e'_k(r) - e'_k(p)w_k(p) + f'_k(p), & \text{if } p_{m-1-k} = 1 \end{cases} \tag{36}$$

where $f'_k(p)$ is given by

$$f'_k(p) = \begin{cases} \zeta_{k,p}+\zeta'''_{k,p}+\delta_{k,p}+\lambda_{k,p} \\ + j\left(\eta_{k,p}+\eta'''_{k,p}+\delta'''_{k,p}+\lambda'''_{k,p}\right), & \text{if } p_{m-1-k} = 0 \\ \alpha_{k,p}+\alpha'''_{k,p}+\epsilon_{k,p}+\gamma_{k,p} \\ + j\left(\beta_{k,p}+\beta'''_{k,p}+\epsilon'''_{k,p}+\gamma'''_{k,p}\right), & \text{if } p_{m-1-k} = 1. \end{cases} \tag{37}$$

3) DIT FFT FP to FXP

$$e''_{k+1}(p) = \begin{cases} e''_k(p) + e''_k(q)w_k(p) \\ + f'_k(p) - f_k(p), & \text{if } p_{m-1-k} = 0 \\ e''_k(r) - e''_k(p)w_k(p) \\ + f'_k(p) - f_k(p), & \text{if } p_{m-1-k} = 1 \end{cases} \tag{38}$$

where $f_k(p)$ and $f'_k(p)$ are given by (35) and (37), respectively.

The accumulation of roundoff error is determined by the recursive equations (34)–(38), with the initial conditions given by (21).

3) Effects of Input Quantization and Coefficient Inaccuracy: The discussion presented in previous sections concerns only the roundoff accumulation effect. As mentioned before, there are two other common causes of error due to the finite word length in computing Fourier coefficients. They are the quantization of

$$\left.\begin{aligned} B'_{k+1}(p) &= \left[\left[B'_k(q)U_k(p)\left(i+\zeta'_{k,p}\right) - C'_k(q)V_k(p)\left(1+\zeta''_{k,p}\right)\right]\right. \\ &\quad \left.\times\left(1+\delta'_{k,p}\right) + B'_k(p)\right]\left(1+\lambda'_{k,p}\right) \\ C'_{k+1}(p) &= \left[\left[C'_k(p)U_k(p)\left(1+\eta'_{k,p}\right) + B'_k(q)V_k(p)\left(1+\eta''_{k,p}\right)\right]\right. \\ &\quad \left.\times\left(1+\delta''_{k,p}\right) + C'_k(p)\right]\left(1+\lambda''_{k,p}\right) \end{aligned}\right\}, \quad \text{if } p_{m-1-k} = 0$$

$$\left.\begin{aligned} B'_{k+1}(p) &= \left[\left[B'_k(p)\left(-U_k(p)\right)\left(1+\alpha'_{k,p}\right) + C'_k(p)V_k(p)\left(1+\alpha''_{k,p}\right)\right]\right. \\ &\quad \left.\times\left(1+\epsilon'_{k,p}\right) + B'_k(r)\right]\left(1+\gamma'_{k,p}\right) \\ C'_{k+1}(p) &= \left[\left[C'_k(p)\left(-U_k(p)\right)\left(1+\beta'_{k,p}\right) + B'_k(p)\left(-V_k(p)\right)\left(1+\beta'_{k,p}\right)\right]\right. \\ &\quad \left.\times\left(1+\epsilon''_{k,p}\right) + C'_k(r)\right]\left(1+\gamma''_{k,p}\right) \end{aligned}\right\}, \quad \text{if } p_{m-1-k} = 1 \tag{32}$$

and

$$\left.\begin{aligned} B''_{k+1}(p) &= B''_k(q)U_k(p) - C''_k(q)V_k(p) + B''_k(p) + \zeta_{k,p} + \zeta'''_{k,p} + \delta_{k,p} + \lambda_{k,p} \\ C''_{k+1}(p) &= C''_k(q)U_k(p) + B''_k(q)V_k(p) + C''_k(p) + \eta_{k,p} + \eta'''_{k,p} + \delta'''_{k,p} + \lambda'''_{k,p} \end{aligned}\right\}, \quad \text{if } p_{m-1-k} = 0$$

$$\left.\begin{aligned} B''_{k+1}(p) &= B''_k(p)\left(-U_k(p)\right) + C''(p)V_k(p) + B''_k(r) + \alpha_{k,p} + \alpha'''_{k,p} + \epsilon_{k,p} + \delta_{k,p} \\ C''_{k+1}(p) &= C''_k(p)\left(-U_k(p)\right) + B''_k(p)\left(-V_k(p)\right) + C''_k(r) + \beta_{k,p} + \beta'''_{k,p} + \epsilon'''_{k,p} + \gamma'''_{k,p} \end{aligned}\right\}, \quad \text{if } p_{m-1-k} = 1 \tag{33}$$

$$f_k(p) = \begin{cases} B'_k(p)\lambda'_{k,p} + B'_k(q)U_k(p)\left[\left(1+\zeta'_{k,p}\left(1+\delta'_{k,p}\right)\left(1+\lambda'_{k,p}\right)-1\right]\right. \\ - C'_k(q)V_k(p)\left[\left(1+\zeta''_{k,p}\right)\left(1+\delta_{k,p}\left(1+\lambda'_{k,p}\right)-1\right] + j\left[C'_k(p)\lambda'_{k,p}\right.\right. \\ + C'_k(q)U_k(p)\left[\left(1+\zeta'_{k,p}\right)\left(1+\delta''_{k,p}\right)\left(1+\lambda''_{k,p}\right)-1\right]\right. \\ + B'_k(q)V_k(p)\left[\left(1+\zeta''_{k,p}\right)\left(1+\delta''_{k,p}\right)\left(1+\lambda''_{k,p}\right)-1\right], & \text{if } p_{m-1-k} = 0 \\ B'_k(r)\gamma'_{k,p} - B'_k(p)U_k(p)\left[\left(1+\alpha'_{k,p}\left(1+\epsilon'_{k,p}\right)\left(1+\gamma'_{k,p}\right)-1\right]\right. \\ - C'_k(p)V_k(p)\left[\left(1+\alpha''_{k,p}\right)\left(1+\epsilon'_{k,p}\left(1+\gamma'_{k,p}\right)-1\right] + j\left[C'_k(r)\gamma''_{k,p}\right.\right. \\ - C'_k(p)U_k(p)\left[\left(1+\beta'_{k,p}\right)\left(1+\epsilon''_{k,p}\right)\left(1+\gamma''_{k,p}\right)-1\right]\right. \\ \left. + B'_k(q)V_k(p)\left[\left(1+\zeta''_{k,p}\right)\left(1+\gamma''_{k,p}\right)\left(1+\lambda''_{k,p}\right)-1\right], \right. & \text{if } p_{m-1-k} = 1 \end{cases} \tag{35}$$

the input data $x(n)$ and the inaccuracy of the coefficients $w_k(p)$. The effect of the quantization of $x(n)$ can be treated as follows. Let $x'(n)$ and $x''(n)$ be the FP and FXP quantized versions of $x(n)$, respectively. Then, from the discussion in Section III-A1, we can write

$$\begin{aligned}\mathrm{Re}\,[x'(n)] &= (1+\theta_n)\mathrm{Re}\,[x(n)]\\ \mathrm{Im}\,[x'(n)] &= (1+\xi_n)\mathrm{Im}\,[x(n)] \qquad (39)\\ \mathrm{Re}\,[x''(n)] &= \mathrm{Re}\,[x(n)] + \theta'_n\\ \mathrm{Im}\,[x''(n)] &= \mathrm{Im}\,[x(n)] + \xi'_n \qquad (40)\end{aligned}$$

where θ_n and ξ_n are the errors caused by FP quantization, and θ'_n and ξ'_n are the errors caused by FXP quantization in the input signal. The effect of (39) and (40) modifies the initial conditions as described in (21) to

$$\begin{aligned}e_0(n) &= \theta_n\mathrm{Re}\,[x(n)] + j\xi_n\mathrm{Im}\,[x(n)] \qquad (41)\\ e'_0(n) &= \theta'_n + j\xi'_n \qquad (42)\\ e''_0(n) &= e'_0(n) - e_0(n). \qquad (43)\end{aligned}$$

It can be shown that with these modifications the final results of the mean square errors remain the same except for an addition term that is independent of p [25].

Another cause for error that has been neglected in the treatment of the previous sections is the fact that the coefficients $w_k(p)$ can only be represented in finite accuracy. It is possible to analyze the effect of the inaccuracy of $w_k(p)$ as follows. Let $U_k(p)$ and $V_k(p)$ be the real and imaginary parts of $w_k(p)$ as defined in (13), respectively. Also, let $U'_k(p)$ and $U''_k(p)$ be the FP and FXP quantized version of $U_k(p)$, and $V'_k(p)$ and $V''_k(p)$ be the FP and FXP quantized version of $V_k(p)$. Then, from the discussion in Section III-A1, we can write

$$\begin{aligned}U'_k(p) &= (1+\varphi_{k,p})\,U_k(p)\\ V'_k(p) &= (1+\psi_{k,p})\,V_k(p) \qquad (44)\\ U''_k(p) &= U_k(p) + \varphi'_{k,p}\\ V''_k(p) &= V_k(p) + \psi'_{k,p} \qquad (45)\end{aligned}$$

where $\varphi_{k,p}$ and $\psi_{k,p}$ are the errors caused by FP quantizations, and $\varphi'_{k,p}$ and $\psi'_{k,p}$ are the errors caused by FXP quantizations in the coefficients. One may now proceed with the analysis of Sections III-A1 and III-A2 by adding the factors $(1+\varphi_{k,p})$, $(1+\psi_{k,p})$, $\varphi'_{k,p}$, and $\psi'_{k,p}$ in appropriate places in (23), (25), (35), and (37).

4) Error Analysis in HOL: In HOL, we first constructed complex numbers on reals similar to [21]. We defined in HOL a new type for complex numbers to be in bijection with $\mathbb{R}\times\mathbb{R}$. The bijections are written in HOL as complex: $\mathbb{R}^2\to\mathbb{C}$ and coords: $\mathbb{C}\to\mathbb{R}^2$. We used convenient abbreviations for the real (Re) and imaginary (Im) parts of a complex number. We also defined arithmetic operations such as addition, subtraction, and multiplication on complex numbers. We overloaded the usual symbols $(+,-,\times)$ for $\mathbb{C}$ and $\mathbb{R}$. Furthermore, we defined, using recursive definition in HOL, expressions for the finite summation on complex numbers. Similarly, we constructed complex numbers on FP and FXP variables. We also defined rounding and valuation functions for FP and FXP complex numbers. Then, we defined the principal N-roots on unity $(e^{-j2\pi n/N} = \cos(2\pi n/N) - j\sin(2\pi n/N))$ and its powers as a complex number using the sine and cosine functions available in the transcendental theory of HOL reals library [18]. We specified expressions in HOL for expansion of a natural number into a binary form in normal and rearranged order according to (3), (4), (6), and (28). The above enables us to specify the FFT algorithms in real, FP, and FXP abstraction levels using recursive definitions in HOL as described in (5) and (27). Then, we define the real and imaginary parts of the FFT algorithm, and powers of the principal N-roots on unity according to (12). Later, we proved in separate lemmas that the real and imaginary parts of the FFT algorithm in real, FP, and FXP levels can be expanded as in (13) and (29). Then, we proved lemmas to introduce an error in each of the arithmetic steps in real and imaginary parts of the FP and FXP FFT algorithms according to (16), (17), (32), and (33). We proved these lemmas using fundamental error analysis lemmas for basic arithmetic operations [3] according to (10) and (11). Then, we defined in HOL the error of the pth element of the FP and FXP FFT algorithms at step k, and the corresponding error in transition from FP to FXP, according to (7)–(9). Thereafter, we proved lemmas to rewrite the errors as complex numbers using the real and imaginary parts according to (18)–(20). Finally, we proved a set of lemmas to determine the accumulation of roundoff error in FP and FXP FFT algorithms by recursive equations and initial conditions according to (21)–(26) for DIF, and (34)–(38) for DIT FFT. A complete list of the derived HOL definitions and theorems can be found in [1].

B. Radix-4 64-Point FFT Design Verification

In this section, we describe the application of the proposed approach for the verification in HOL of the transition from real, FP, and FXP specifications to RTL implementation of an FFT algorithm. We have chosen the case study of a radix-4 pipelined 64-point complex FFT core available as VHDL RTL model in the Xilinx Coregen library [37]. All proofs have been conducted in HOL, hence establishing correctness of the FFT design implementation with respect to its high-level algorithmic specifications. Fig. 8 shows the overall block diagram of the Radix-4 64-point pipelined FFT design. The basic elements are memories, delays, multiplexers, and dragonflies. In general, the 64-point pipelined FFT requires the calculation of three radix-4 dragonfly ranks. Each radix-4 dragonfly is a successive combination of a radix-4 butterfly with four twiddle factor multipliers. The FFT core accepts naturally ordered data on the input buses in a continuous stream, performs a complex FFT, and streams out DFT samples on the output buses in a natural order. These buses are, respectively, the real and imaginary components of the input and output sequences. An internal input data memory controller orders the data into blocks to be presented to the FFT processor. The twiddle factors are stored in coefficient memories. The real and imaginary components of complex input and output samples and the phase factors are represented as 16-bit 2's complement numbers. The unscrambling operation is performed using output bit-reversing buffer.

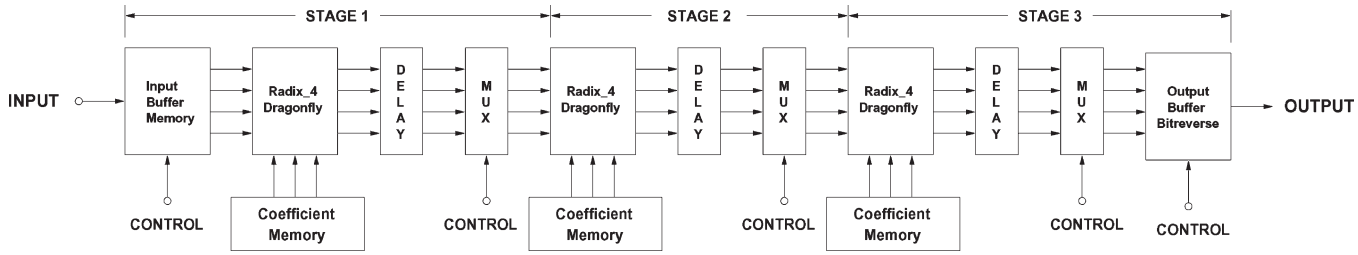

Fig. 8. Radix-4 64-point pipelined FFT implementation.

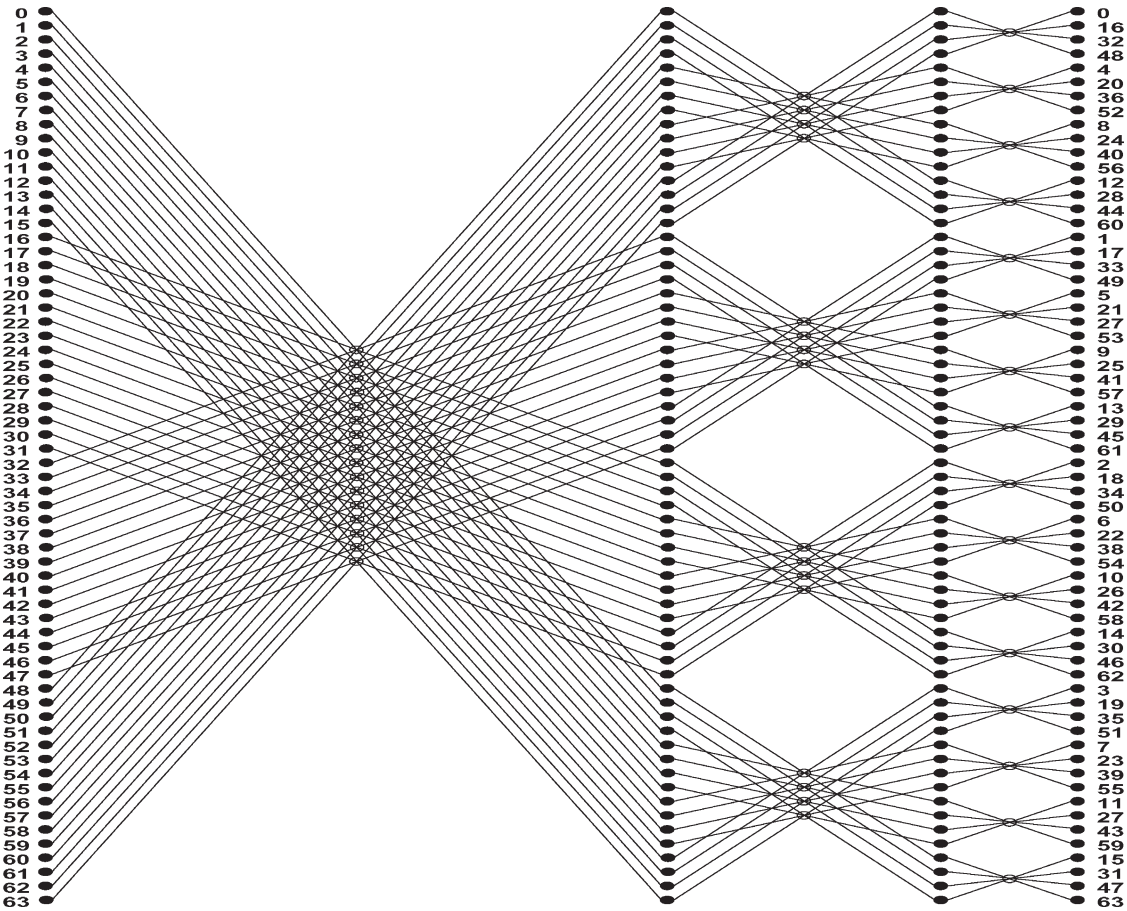

Fig. 9. Signal flowgraph of radix-4 64-point FFT.

To define the radix-4 64-point FFT algorithm [8], [31], we represent the indices p and n in (2) in base 4 (quaternary number system) as

$$p = 16p_2 + 4p_1 + p_0, \quad p_2, p_1, p_0 = 0, 1, 2, 3 \tag{46}$$
$$n = 16n_2 + 4n_1 + n_0, \quad n_2, n_1, n_0 = 0, 1, 2, 3. \tag{47}$$

It is easy to verify that as n_0, n_1, and n_2 take on all possible values in the range indicated, n goes through all possible values from 0 to 63 with no values repeated. This is also true for the frequency index p. Using these index mappings, we can express the radix-4 64-point FFT algorithm recursively as

$$A_1(p_0, n_1, n_0) = \sum_{n_2=0}^{3} x(n_2, n_1, n_0)(W_{64})^{16p_0n_2} \tag{48}$$

$$A_2(p_0, p_1, n_0) = \sum_{n_1=0}^{3} A_1(p_0, n_1, n_0)(W_{64})^{(4p_1+p_0)4n_1} \tag{49}$$

$$A_3(p_0, p_1, p_2) = \sum_{n_0=0}^{3} A_2(p_0, p_1, n_0)(W_{64})^{(16p_2+4p_1+p_0)n_0}. \tag{50}$$

The final result can be written as

$$A(p_2, p_1, p_0) = A_3(p_0, p_1, p_2). \tag{51}$$

Thus, as in the radix-2 algorithm, the results are in reversed order. Based on (48)–(50), and (51), we can develop a signal flowgraph for the radix-4 64-point FFT algorithm as shown in Fig. 9, which is an expanded version of the pipelined implementation in Fig. 8. The graph is composed of three

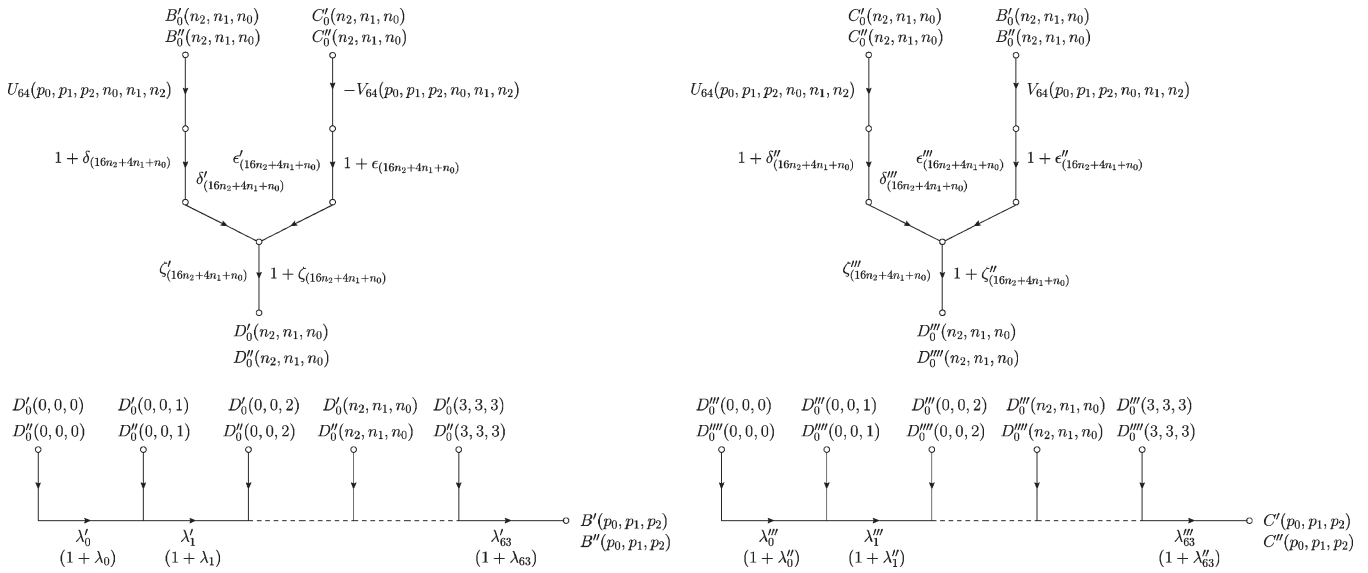

Fig. 10. Error flowgraph for radix-4 64-point FFT.

successive radix-4 dragonfly stages, with each stage comprising 16 dragonflies.

From (48)–(51), we can express the input–output relationship of a radix-4 64-point FFT as

$$A(p_2,p_1,p_0)=\sum_{n_2=0}^{3}\sum_{n_1=0}^{3}\sum_{n_0=0}^{3}x(n_2,n_1,n_0)\times(W_{64})^{16p_0n_2+4(4p_1+p_0)n1+(16p_2+4p_1+p_0)n_0}. \quad (52)$$

Equation (52) can be rewritten using real and imaginary parts as (53), shown at the bottom of the page.

The corresponding error flowgraph is given in Fig. 10. Therefore, the actual real and imaginary parts of the FP and FXP outputs can be given as that shown in (54)–(57), shown at the bottom of the next page.

From (53)–(57), we derive the following error analysis cases.

1) Radix-4 64-point FFT real to FP [(58) shown at the bottom of the next page].
2) Radix-4 64-point FFT real to FXP [(59) shown at the bottom of the next page].
3) Radix-4 64-point FFT FP to FXP [(60) shown at the bottom of the next page].

where f and f' are the error functions that can be derived based on Fig. 10.

1) Verification in HOL: In HOL, we first modeled the RTL description of a radix-4 butterfly as a predicate in higher order logic. The block takes a vector of four complex input data and performs the operations to generate a vector of four complex output signals. The real and imaginary parts of the input and output signals are represented as 16-bit Boolean words. We defined separate functions in HOL for arithmetic operations such as addition, subtraction, and multiplication on complex 2's complement 16-bit Boolean words. Then, we built the complete butterfly structure using a proper combination of these primitive operations.

Thereafter, we described a radix-4 dragonfly block as a conjunction of a radix-4 butterfly and four 16-bit twiddle factor complex multipliers. Finally, we modeled the complete RTL description of the radix-4 64-point structure in HOL. The FFT block is defined as a conjunction of 48 instantiations of radix-4 dragonfly blocks. Proper time instances of the input and output signals are applied to each block, according to Fig. 9.

Following similar steps, we described the radix-4 64-point FFT structures as FXP, FP, and real domains in HOL using the corresponding complex data types and arithmetic operations.

$$\begin{aligned}
&B(p_2,p_1,p_0)\\
&\quad=\sum_{n_2=0}^{3}\sum_{n_1=0}^{3}\sum_{n_0=0}^{3}B_0(n_2,n_1,n_0)U_{64}\left(16p_0n_2+4(4p_1+p_0)n_1+(16p_2+4p_1+p_0)n_0\right)\\
&\qquad-C_0(n_2,n_1,n_0)V_{64}\left(16p_0n_2+4(4p_1+p_0)n_1+(16p_2+4p_1+p_0)n_0\right)\\
&C(p_2,p_1,p_0)\\
&\quad=\sum_{n_2=0}^{3}\sum_{n_1=0}^{3}\sum_{n_0=0}^{3}C_0(n_2,n_1,n_0)U_{64}\left(16p_0n_2+4(4p_1+p_0)n_1+(16p_2+4p_1+p_0)n_0\right)\\
&\qquad+B_0(n_2,n_1,n_0)V_{64}\left(16p_0n_2+4(4p_1+p_0)n_1+(16p_2+4p_1+p_0)n_0\right) \quad (53)
\end{aligned}$$

The formal verification of the radix-4 DIF FFT algorithm case study was performed based on the commutating diagram in Fig. 2, in that we proved hierarchically that the FFT netlist implies the FFT RTL, and then proved that the FFT RTL description implies the corresponding FXP model. The proof of the FFT block is then broken down into the corresponding proof of the dragonfly block, which itself is broken down into the proofs of butterfly and primitive arithmetic operations. We

$$\begin{aligned} &B'(p_2,p_1,p_0) \\ &\quad=\sum_{n_2=0}^{3}\sum_{n_1=0}^{3}\sum_{n_0=0}^{3}\big(B'_0(n_2,n_1,n_0)U_{64}\left(16p_0n_2+4(4p_1+p_0)n_1+(16p_2+4p_1+p_0)n_0\right) \\ &\qquad\times\left(1+\delta_{(16n_2+4n_1+n_0)}\right)-C'_0(n_2,n_1,n_0)V_{64}\left(16p_0n_2+4(4p_1+p_0)n_1+(16p_2+4p_1+p_0)n_0\right) \\ &\qquad\times\left(1+\epsilon_{(16n_2+4n_1+n_0)}\right)\big)\left(1+\zeta_{(16n_2+4n_1+n_0)}\right)\prod_{i=16n_2+4n_1+n_0}^{63}(1+\lambda_i) \end{aligned} \tag{54}$$

$$\begin{aligned} &C'(p_2,p_1,p_0) \\ &\quad=\sum_{n_2=0}^{3}\sum_{n_1=0}^{3}\sum_{n_0=0}^{3}\Big(C'_0(n_2,n_1,n_0)U_{64}\left(16p_0n_2+4(4p_1+p_0)n_1+(16p_2+4p_1+p_0)n_0\right) \\ &\qquad\times\left(1+\delta''_{(16n_2+4n_1+n_0)}\right)+B'_0(n_2,n_1,n_0)V_{64}\left(16p_0n_2+4(4p_1+p_0)n_1+(16p_2+4p_1+p_0)n_0\right) \\ &\qquad\times\left(1+\epsilon''_{(16n_2+4n_1+n_0)}\right)\Big)\left(1+\zeta''_{(16n_2+4n_1+n_0)}\right)\prod_{i=16n_2+4n_1+n_0}^{63}(1+\lambda''_i) \end{aligned} \tag{55}$$

$$\begin{aligned} &B''(p_2,p_1,p_0) \\ &\quad=\sum_{n_2=0}^{3}\sum_{n_1=0}^{3}\sum_{n_0=0}^{3}B''_0(n_2,n_1,n_0)U_{64}\left(16p_0n_2+4(4p_1+p_0)n_1+(16p_2+4p_1+p_0)n_0\right) \\ &\qquad-C''_0(n_2,n_1,n_0)V_{64}\left(16p_0n_2+4(4p_1+p_0)n_1+(16p_2+4p_1+p_0)n_0\right) \\ &\qquad+\delta'_{(16n_2+4n_1+n_0)}+\epsilon'_{(16n_2,4n_1+n_0)}+\zeta'_{(16n_2+4n_1+n_0)}+\sum_{i=16n_2+4n_1+n_0}^{63}\lambda'_i \end{aligned} \tag{56}$$

$$\begin{aligned} &C''(p_2,p_1,p_0) \\ &\quad=\sum_{n_2=0}^{3}\sum_{n_1=0}^{3}\sum_{n_0=0}^{3}C'''_0(n_2,n_1,n_0)U_{64}\left(16p_0n_2+4(4p_1+p_0)n_1+(16p_2+4p_1+p_0)n_0\right) \\ &\qquad+B''_0(n_2,n_1,n_0)V_{64}\left(16p_0n_2+4(4p_1+p_0)n_1+(16p_2+4p_1+p_0)n_0\right) \\ &\qquad+\delta'''_{(16n_2+4n_1+n_0)}+\epsilon'''_{(16n_2,4n_1+n_0)}+\zeta'''_{(16n_2+4n_1+n_0)}+\sum_{i=16n_2+4n_1+n_0}^{63}\lambda'''_i \end{aligned} \tag{57}$$

$$e(p_2,p_1,p_0)=\sum_{n_2=0}^{3}\sum_{n_1=0}^{3}\sum_{n_0=0}^{3}\left(e_0(n_2,n_1,n_0)(W_{64})^{(16p_0n_2+4(4p_1+p_0)n_1+(16p_2+4p_1+p_0)n_0)}\right)+f(p_0,p_1,p_2) \tag{58}$$

$$e'(p_2,p_1,p_0)=\sum_{n_2=0}^{3}\sum_{n_1=0}^{3}\sum_{n_0=0}^{3}\left(e'_0(n_2,n_1,n_0)(W_{64})^{(16p_0n_2+4(4p_1+p_0)n_1+(16p_2+4p_1+p_0)n_0)}\right)+f'(p_0,p_1,p_2) \tag{59}$$

$$\begin{aligned} e''(p_2,p_1,p_0)=&\sum_{n_2=0}^{3}\sum_{n_1=0}^{3}\sum_{n_0=0}^{3}\left(e''_0(n_2,n_1,n_0)\times(W_{64})^{(16p_0n_2+4(4p_1+p_0)n_1+(16p_2+4p_1+p_0)n_0)}\right) \\ &+f'(p_0,p_1,p_2)-f(p_0,p_1,p_2) \end{aligned} \tag{60}$$

used the data abstraction functions described in Section II-A to convert a complex vector of 16-bit 2's complement Boolean words into the corresponding FXP vector.

Then, we proved three theorems encompassing the error analysis of the radix-4 DIF FFT algorithm, as discussed in Section III. The first lemma represents the error between the real number specification and the FP specification. The second lemma represents the error between the real number and the FXP specifications. The third lemma represents the error between FP and FXP specifications. According to these lemmas, the FP and FXP implementations and the real specification of a radix-4 DIF FFT algorithm are related to each other based on the corresponding data abstraction and error analysis functions.

Finally, using the obtained theorems, we easily deduced our ultimate theorem proving the correctness of the real specification from the RTL implementation, taking into account the error analysis computed beforehand. A complete list of the derived HOL definitions and theorems can be found in [1].

IV. Related Work

Work related to our project can be classified in three groups, namely, using formal methods for error analysis, paper-and-pencil error analysis of FFT algorithms, and formal verification of FFT designs.

A. Error Analysis in Formal Verification

Previous work on error analysis in formal verification was done by Harrison [20], who verified the FP algorithms such as the exponential function against their abstract mathematical counterparts using the HOL light theorem prover. As the main theorem, he proved that the FP exponential function has a correct overflow behavior, and in the absence of overflow the error in the result is bounded to a certain amount. He also reported on an error in the hand proof mostly related to forgetting some special cases in the analysis. This error analysis is very similar to the type of analysis performed for DSP algorithms. The major difference, however, is the use of statistical methods and mean square error analysis for DSP algorithms, which is not covered in the error analysis of the mathematical functions used by Harrison. In this method, the error quantities are treated as independent random variables uniformly distributed over a specific interval depending on the type of arithmetic and the rounding mode. Then, error analysis is performed to derive expressions for the variance and mean square error. To perform such an analysis in HOL, we need to develop a mechanized theory on the properties of random variables and random processes. This type of analysis is not addressed in this paper and is a part of our future work.

Huhn *et al.* [23] proposed a hybrid formal verification method combining different state-of-the-art techniques to guide the complete design flow of imprecisely working arithmetic circuits starting at the algorithmic level down to the RT level. The usefulness of the method is illustrated with the example of discrete cosine transform algorithms. In particular, the authors have shown the use of computer algebra systems like Mathematica or Maple at the algorithmic level to reason about real numbers and to determine certain error bounds for the results of numerical operations. In contrast to [23], we proposed an error analysis for DSP designs using the HOL theorem prover. Although computer algebraic systems such as Maple or Mathematica are much more popular and have many powerful decision procedures and heuristics, theorem provers are more expressive, more precise, and more reliable [22]. One option is to combine the rigour of the theorem provers with the power of computer algebraic systems as proposed in [22].

B. Error Analysis of FFT Algorithms

Analysis of errors in FFT realizations due to finite precision effects has traditionally relied on paper-and-pencil proofs and simulation techniques. The roundoff error in using FFT algorithms depends on the algorithm, type of arithmetic, word length, and radix. For FFT algorithms realized with FXP arithmetic, the error problems have been studied extensively. For instance, Welch [35] presented an analysis of the FXP accuracy of the radix-2 DIT FFT algorithm. Thong and Liu [33] presented a general approach to the error analysis of various versions of the FFT algorithm when FXP arithmetic is used. While the roundoff noise for FXP arithmetic enters into the system additively, it is a multiplicative component in the case of FP arithmetic. This problem is analyzed first by Gentleman and Sande [16], who presented an upper bound on the mean-squared error for FP DIF FFT algorithm. Weinstein [34] presented a statistical model for roundoff errors of the FP FFT. Kaneko and Liu [25] presented a detailed analysis of roundoff error in the FFT DIF algorithm using FP arithmetic. This analysis is later extended by the same authors to the FFT DIT algorithm [27]. Oppenheim and Weinstein [32] discussed in some detail the effects of finite register length on the implementations of digital filters and FFT algorithms.

In order to validate the error analysis, most of the above works compare the theoretical results with experimental simulation. In this paper, we showed how the above error analyses for the FFT algorithms can be mechanically performed using the HOL theorem prover, providing a superior approach to validation by simulation. Our focus was on the process of translating the hand proofs done in the 1960s and 1970s into equivalent proofs in HOL. The analysis we developed is mainly inspired by the work done by Kaneko and Liu [25], who proposed a general approach to the error analysis problem of DIF FFT algorithm using FP arithmetic. Following a similar idea, we have extended this theoretical analysis for the DIT and FXP FFT algorithms. In all cases, good agreements between formal and theoretical results were obtained.

C. Formalization and Verification of FFT Algorithms

Related work on the formalization and mechanical verification of the FFT algorithm was done by Gamboa [15] using the ACL2 theorem prover. The author formalized the FFT as a recursive data-parallel algorithm using the powerlist data structure. He also presented an ACL2 proof of the correctness of the FFT algorithm by translating the hand proof taken from Misra's seminal paper on powerlists [30] into a mechanical

proof in ACL2. In the same line, Capretta [9] presented the formalization of the FFT using the type theory proof tool Coq. To facilitate the definition of the transform by structural recursion, Capretta used the structure of polynomial trees, which is similar to the data structure of powerlists introduced by Misra. Finally, he proved its correctness and the correctness of the inverse Fourier transform (IFT). In another related work, Bjesse [7] described the verification of FFT hardware at the netlist level with an automatic combination of symbolic simulation and theorem proving using the Lava hardware development platform. He proved that the sequential pipelined implementation of the radix-4 DIT FFT is equivalent to the corresponding combinational circuit. He also proved that the abstract implementations of the radix-2 and radix-4 FFTs are equivalent for sizes that are an exponent of four.

While Gamboa [15] and Capretta [9] prove the correctness of the high-level FFT algorithm against the DFT, the verification of Bjesse [7] is performed at the netlist level. In contrast, our work tried to close this gap by formally specifying and verifying the FFT algorithm realizations at different levels of abstraction based on different data types. Besides, the definition used for the FFT in [15] and [9] is based on the radix-2 DIT algorithm. We cover both DIT and DIF algorithms and radices other than 2. The methodology we proposed in this paper is, to the best of our knowledge, the first project of its kind that covers the formal specification and verification of integrated FFT algorithms at different abstraction levels starting from real specification to FP and FXP algorithmic descriptions, down to RT and netlist gate levels.

V. Conclusion

In this paper, we described a methodology for the formal specification and verification of DSP system designs at different abstraction levels. We proposed shallow embedding of DSP descriptions at different levels in HOL. For the verification of the transition from FP to FXP levels, we proposed an error analysis approach in which we consider the effects of finite precision in the implementation of DSP systems. These include errors due to the quantization of input samples and system coefficients, and also roundoff accumulation in arithmetic operations. The verification from FXP to RTL and netlist levels is performed using traditional hierarchical verification in HOL. In this paper, we demonstrated our methodology using the case study of FFT algorithms. The approach covers the two canonical forms (DIT and DIF) of realization of the FFT algorithm using real, FP, and FXP arithmetic as well as their RT implementations, each entirely specified in HOL. We proved lemmas to derive expressions for the accumulation of roundoff error in FP and FXP designs compared to the ideal real specification. Then, we proved that FFT RTL implementation implies the corresponding specification at FXP level using classical hierarchical verification in HOL, hence bridging the gap between hardware implementation and high levels of mathematical specification. In this work, we have also contributed to the upgrade and application of established real, complex real, FP, and FXP theories in HOL to the analysis of errors due to finite precision effects, and applied them on the realization of FFT algorithms. Error analyses using theoretical paper-and-pencil proofs did exist since the late 1960s while design verification is exclusively done by simulation techniques. We believe this is the first time that a complete formal framework has been proposed for the specification and verification of DSP algorithms at different levels of abstraction. The methodology presented in this paper opens new avenues in using formal methods for the verification of DSP systems as complement to traditional theoretical (analytical) and simulation techniques. We are currently investigating the verification of complex wired and wireless communication systems whose building blocks heavily make use of several instances of the FFT algorithms. As a future work, we also plan to extend the error analyses to cover worst case, average, and variance errors. Finally, we plan to link HOL with computer algebra systems to create a sound, reliable, and powerful system for the verification of DSP systems.

Acknowledgment

The experiments were carried out with CAD tools provided by the Canadian Microelectronics Corporation.

References

[1] B. Akbarpour, "Modeling and verification of DSP designs in HOL," Ph.D. dissertation, Dept. Elect. Comput. Eng., Concordia Univ., Montreal, QC, Canada, Mar. 2005.

[2] B. Akbarpour and S. Tahar, "A methodology for the formal verification of FFT algorithms in HOL," in *Formal Methods in Computer-Aided Design*, vol. 3312. Berlin, Germany: Springer-Verlag, 2004, pp. 37–51.

[3] ——, "Error analysis of digital filters using theorem proving," in *Theorem Proving in Higher Order Logics*, vol. 3223. Berlin, Germany: Springer-Verlag, 2004, pp. 1–16.

[4] ——, "The application of formal verification to SPW designs," in *Proc. Euromicro Symp. Digital System Design*, Belek-Antalya, Turkey, Sep. 2003, pp. 325–332.

[5] B. Akbarpour, S. Tahar, and A. Dekdouk, "Formalization of fixed-point arithmetic in HOL," *Form. Methods Syst. Des.*, vol. 27, no. 1/2, pp. 173–200, Sep. 2005.

[6] R. Boulton, A. Gordon, M. Gordon, J. Harrison, J. Herbert, and J. Van-Tassel, "Experience with embedding hardware description languages in HOL," in *Theorem Provers in Circuit Design*. Amsterdam, The Netherlands: North Holland, 1992, pp. 129–156.

[7] P. Bjesse, "Automatic verification of combinational and pipelined FFT circuits," in *Computer Aided Verification*, vol. 1633. Berlin, Germany: Springer-Verlag, 1999, pp. 380–393.

[8] E. O. Brigham, *The Fast Fourier Transform*. Englewood Cliffs, NJ: Prentice-Hall, 1974.

[9] V. Capretta, "Certifying the fast Fourier transform with Coq," in *Theorem Proving in Higher Order Logics*, vol. 2152. Berlin, Germany: Springer-Verlag, 2001, pp. 154–168.

[10] Cadence Design Systems, Inc., *Signal Processing WorkSystem (SPW) User's Guide*, Jul. 1999.

[11] Synopsys, Inc., *CoCentric System Studio User's Guide*, Aug. 2001.

[12] W. T. Cochran *et al.*, "What is the fast Fourier transform," *IEEE Trans. Audio Electroacoust.*, vol. AU-15, no. 2, pp. 45–55, Jun. 1967.

[13] J. W. Cooley and J. W. Tukey, "An algorithm for machine calculation of complex Fourier series," *Math. Comput.*, vol. 19, no. 90, pp. 297–301, Apr. 1965.

[14] G. Forsythe and C. B. Moler, *Computer Solution of Linear Algebraic Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1967.

[15] R. A. Gamboa, "The correctness of the fast Fourier transform: A structural proof in ACL2," *Form. Methods Syst. Des., Special Issue on UNITY*, vol. 20, no. 1, pp. 91–106, Jan. 2002.

[16] W. M. Gentleman and G. Sande, "Fast Fourier transforms—For fun and profit," in *AFIPS Fall Joint Computer Conf.*, San Francisco, CA, 1966, vol. 29, pp. 563–578.

[17] M. J. C. Gordon and T. F. Melham, *Introduction to HOL: A Theorem Proving Environment for Higher Order Logic*. Cambridge, U.K.: Cambridge Univ. Press, 1993.

[18] J. R. Harrison, "Constructing the real numbers in HOL," *Form. Methods Syst. Des.*, vol. 5, no. 1/2, pp. 35–59, 1994.
[19] ——, "A machine-checked theory of floating-point arithmetic," in *Theorem Proving in Higher Order Logics*, vol. 1690. Berlin, Germany: Springer-Verlag, 1999, pp. 113–130.
[20] ——, "Floating-point verification in HOL light: The exponential function," *Form. Methods Syst. Des.*, vol. 16, no. 3, pp. 271–305, 2000.
[21] ——, "Complex quantifier elimination in HOL," in *Supplemental Proc. Int. Conf. Theorem Proving Higher Order Logics*, Edinburgh, U.K., Sep. 2001, pp. 159–174.
[22] J. R. Harrison and L. Théry, "A skeptic's approach to combining HOL and Maple," *J. Autom. Reason.*, vol. 21, no. 3, pp. 279–294, Dec. 1998.
[23] M. Huhn, K. Schneider, T. Kropf, and G. Logothetis, "Verifying imprecisely working arithmetic circuits," in *Proc. Design Automation and Test Eur. Conf.*, Munich, Germany, Mar. 1999, pp. 65–69.
[24] *IEEE Standard for Binary Floating-Point Arithmetic*, 1985. ANSI/IEEE Standard 754-1985.
[25] T. Kaneko and B. Liu, "Accumulation of round-off error in fast Fourier transforms," *J. Assoc. Comput. Mach.*, vol. 17, no. 4, pp. 637–654, Oct. 1970.
[26] H. Keding, M. Willems, M. Coors, and H. Meyr, "FRIDGE: A fixed-point design and simulation environment," in *Proc. Design Automation and Test Eur. Conf.*, Paris, France, Feb. 1998, pp. 429–435.
[27] B. Liu and T. Kaneko, "Roundoff error in fast Fourier transforms (decimation in time)," *Proc. IEEE*, vol. 63, no. 6, pp. 991–992, Jun. 1975.
[28] Mathworks, Inc., *Simulink Reference Manual*, 1996.
[29] T. Melham, "Higher order logic and hardware verification," in *Cambridge Tracts in Theoretical Computer Science*, vol. 31. Cambridge, U.K.: Cambridge Univ. Press, 1993.
[30] J. Misra, "Powerlists: A structure for parallel recursion," in *ACM Trans. Program. Lang. Syst.*, vol. 16, no. 6, Nov. 1994, pp. 1737–1767.
[31] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1989.
[32] A. V. Oppenheim and C. J. Weinstein, "Effects of finite register length in digital filtering and the fast Fourier transform," *Proc. IEEE*, vol. 60, no. 8, pp. 957–976, Aug. 1972.
[33] T. Thong and B. Liu, "Fixed-point fast Fourier transform error analysis," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-24, no. 6, pp. 563–573, Dec. 1976.
[34] C. J. Weinstein, "Roundoff noise in floating point fast Fourier transform computation," *IEEE Trans. Audio Electroacoust.*, vol. AU-17, no. 3, pp. 209–215, Sep. 1969.
[35] P. D. Welch, "A fixed-point fast Fourier transform error analysis," *IEEE Trans. Audio Electroacoust.*, vol. AU-17, no. 2, pp. 151–157, Jun. 1969.
[36] J. H. Wilkinson, *Rounding Errors in Algebraic Processes*. Englewood Cliffs, NJ: Prentice-Hall, 1963.
[37] Xilinx, Inc. (2000, Aug.). *High-Performance 64-Point Complex FFT/IFFT V2.0, Product Specification*. [Online]. Available: http://xilinx.com/ipcenter

Behzad Akbarpour received the B.Sc. degree in electrical engineering from Shiraz University, Shiraz, Iran, in 1993, the M.Sc. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1997, and the Ph.D. degree in electrical engineering from Concordia University, Montreal, QC, Canada, in 2005.

He was a Design Engineer at Emad Semicon. Co. Ltd., Tehran, Iran from 1997 to 2000, and Research Assistant at the Hardware Verification Group, Concordia University from 2000 to 2005. Currently, he is Research Associate at the Automated Reasoning Group of the University of Cambridge Computer Laboratory, Cambridge, U.K. His research interests include formal hardware verification, theorem proving, digital signal processing, and very large scale integration (VLSI) design automation.

Sofiène Tahar (M'96) received the Diploma degree in computer engineering from the University of Darmstadt, Darmstadt, Germany, in 1990 and the Ph.D. degree in computer science from the University of Karlsruhe, Karlsruhe, Germany, in 1994.

From 1995 to 1996, he was a Postdoctoral Fellow at Universitè de Montrèal, Montreal, QC, Canada. He is currently an Associate Professor in the Department of Electrical and Computer Engineering, Concordia University, Montreal. He has made contributions and published papers in the areas of formal hardware verification, microprocessor verification, modeling and verification of communications architectures and protocols, and mobile code division multiple access (CDMA) communications. He has been involved in various international conference program committees as well as national research grant selection committees.

Dr. Tahar holds a Concordia University Research Chair in Formal Verification of Microelectronics Systems since 2001. In 1998, he received a Canada Foundation for Innovation (CFI) Researcher Award. He is a Professional Engineer in the Province of Quebec.