Design of Low Energy, High Performance Synchronous and Asynchronous 64-Point FFT

William Lee[1], Vikas S. Vij[1], Anthony R. Thatcher[2], Kenneth S. Stevens[1]
[1]University of Utah, [2]Intel Corporation

***Abstract*—A case study exploring multi-frequency design is presented for a low energy and high performance FFT circuit implementation. An FFT architecture with concurrent data stream computation is selected. An asynchronous and synchronous implementations for a 16-point and a 64-point FFT circuit were designed and compared for energy, performance and area. Both versions are structurally similar and are generated using similar ASIC CAD tools and flows. The asynchronous design shows a benefit of 2.4×, 2.4× and 3.2× in terms of area, energy and performance respectively over its synchronous counterpart. The circuit is further compared with other published designs and shows 0.4×, 4.8× and 32.4× benefit with respect to area, energy and performance.**

***Index Terms*—Asynchronous circuits, FFT, synthesis, timing analysis, low power digital, low energy digital, synchronous circuits, high performance**

I. Introduction

Scaling has enabled the transistor revolution leading to more than a billion transistors on a chip. This allows large, concurrent designs to be built. This growth in complexity has been supported by CAD algorithms and tools enabling the rapid development of complex circuits. However, this CAD has been targeted to the customary clocked design practice that employs a single frequency.

Given the large number of transistors now at our disposal, we feel that designs that operate at multiple frequencies afford an additional avenue for power and performance optimization. Asynchronous design is modular and easily integrates multiple design frequencies. Thus it is a good target for experimenting with multi-frequency designs. The power and performance difference between single and multiple frequency designs is highlighted in the asynchronous domain with two microprocessors. A clocked DLX microprocessor was translated into a single frequency asynchronous design using a "desynchronization" technique [1]. This resulted in the asynchronous design having a small penalty in terms of power, performance and area over its synchronous counterpart. On the other hand, an asynchronous version of the Pentium front end was redesigned to a multi-frequency asynchronous architecture operating at three frequencies: 720MHz for instruction decode, 3.6GHz for instruction selection, and 900MHz for instruction steering and issue [2]. The asynchronous design was fabricated in the same foundry as its commercial counterpart, and achieved three times the performance at half the energy per instruction.

This work applies a multi-frequency approach to the design of an FFT. A multirate FFT architecture was formulated which consists of distributed pipeline stages operating at different frequencies [3]. This FFT design does not use shared memory; rather it employs concurrency and parallel data streams using distributed pipeline elements. This results in high throughput, low energy, and multiple operating frequencies. An asynchronous design using this architecture was implemented using full custom design methods [4]. The low productivity limited that design to a 16-point realization. The work reported here

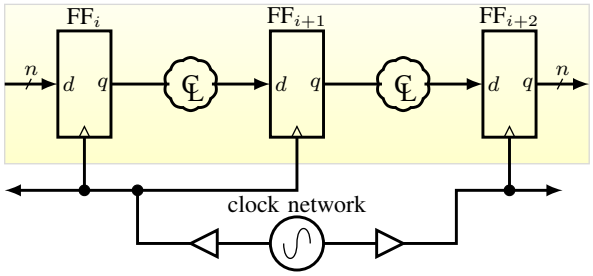

Fig. 1. Clocked design. Frequency and datapath delay of first pipeline stage is constrained by $FF_i/clk\uparrow_j \mapsto FF_{i+1}/d + margin \prec FF_{i+1}/clk\uparrow_{j+1}$

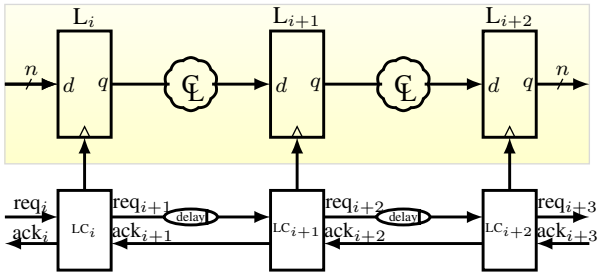

Fig. 2. Timed (bundled data) handshake design. Delay sized by RT constraint $req_i\uparrow \mapsto L_{i+1}/d + margin \prec L_{i+1}/clk\uparrow$. Each $req_i\uparrow$ handshake on LC_i indicates new data is presented to pin d of L_i.

implements both clocked and asynchronous 16 and 64 point versions of the multirate FFT architecture. The asynchronous designs employ a novel CAD flow that uses the same CAD tools and design flow as clocked design.

The primary goal of this paper is to design and compare multi-frequency asynchronous and synchronous 16-point and 64-point FFT architectures. These designs are also compared against other outstanding single clock frequency FFT architectures for area, energy and performance. Overall the clocked multi-frequency 64-point FFT shows a 10.2× improvement in performance and 2.0× improvement in energy per operation over a comparable low energy single frequency design, but at a cost of 6.1× greater area. The multi-frequency asynchronous design improves upon the multi-frequency clocked design by a factor of 2.4× reduction in both area and energy per point, and a 3.2× improvement in performance.

II. Background

One of the major benefit of synchronous design is the presence of mature CAD tools and flows, which enables quick development of complex designs. To allow rapid development of asynchronous designs, a novel relative-timing (RT) based design flow is used. This flow allows asynchronous design elements to be specified and characterized for use with the synchronous CAD tools. These elements can be directly inserted in designs with supporting constraints to enable synthesis, place and route, timing driven sizing, optimization and validation to be performed on them. This flow also allows

978-3-9815370-0-0/DATE13/©2013 EDAA

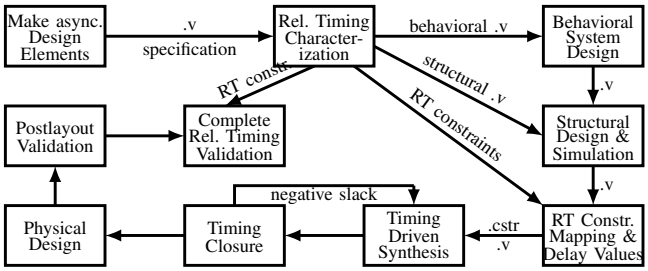

Fig. 3. Simplified Relative Timing Multi-Synch. Design Flow

a more accurate comparison between designs as the same algorithms and parameters for the tool flows, such as synthesis with Design Compiler, are used.

A. Relative Timing

The effect of time on a system is to order and sequence events. Relative timing (RT) is a method of modeling and controlling the results of circuit timing. A RT constraint consists of a common timing reference and a pair of events that are ordered in time for correct circuit operation. We call the common reference a point-of-divergence, or pod, and the ordered events the point-of-convergence, or poc. A constraint is represented as $\mathsf{pod} \mapsto \mathsf{poc}_0 + m \prec \mathsf{poc}_1$ where poc_0 must occur in time before poc_1 with margin m. Hence the maximum path delay from pod to poc_0 must be less than the minimum path delay from pod to poc_1. This can easily be represented by two related design constraint equations `set_max_delay` and `set_min_delay`, which perform timing driven synthesis that enforce the constraints on the logic paths. The RT constraints for a traditional clocked linear pipeline and a handshake linear pipeline are shown in Fig. 1 and 2.

B. Asynchronous CAD Tool Flow

The asynchronous CAD tool flow used is summarized in Fig. 3. Asynchronous controllers are designed and characterized [5]. Controller design begins with a specification, which is synthesized either by hand or by using asynchronous synthesis tools like petrify, 3D, minimalist. [6]. The result is a circuit definition which is then tech mapped to gates of the standard cell library. To ascertain the proper functioning of the circuit, a reset signal is added to the circuit to ensure all wires in it are defined and also to allow it to start in the correct state. A new specification is created with reset which is then verified against the implementation to generate RT constraints [7]. RT paths are used for performance and correctness reasons. RT constraints are represented as two separate delay paths mapped to `set_max_delay` and `set_min_delay` algorithms. Timing in sequential circuits must be represented as directed acyclic graphs (DAGs) to the synchronous CAD tools for performing timing driven sizing, optimization and static timing analysis (STA). Hence RT paths are used to constrain the timing arcs which can be cut by the tools to generate the acyclic timing graph. Structural asynchronous Verilog elements are hereby created and characterized with timing and cycle cut constraints.

The characterized Verilog design elements are used in a complete architecture. The RT constrained architecture can then be passed through the same synchronous CAD tools and flows as the clocked design it is compared against. This involves synthesis, place and route, simulation, timing validation and timing closure. Everything other than the asynchronous handshake control elements (the LC blocks in Fig. 2) is synthesized and place and routed similar to synchronous design. The exact same scripts were used for synthesis and place and route (other than the clock distribution and gating algorithms), providing a more accurate comparison between the clocked and asynchronous architectures.

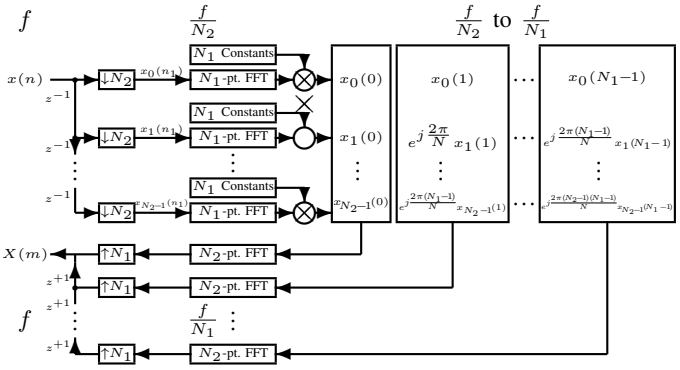

Fig. 4. Multirate FFT Architecture [3]

C. FFT Architecture

The FFT is an algorithm that requires global dependencies, but it can be derived in a multirate form that allows a hierarchical representation as shown in Eqn. 1 [3]. This multirate architecture exploits performance from concurrency by allowing parallel computations to occur at reduced frequencies. The equation represents N_2 FFTs using N_1 values as the inner summation, which are scaled and then used to produce N_1 FFTs of N_2 values. This representation has the advantage that it takes a high frequency stream and decimates it so that each of the internal FFTs operate at a lower decimated data stream frequency. This allows the architecture to simultaneously have lower energy and higher performance.

$$X_{m_1}(m_2) = \sum_{n_2=0}^{N_2-1} \left[W_N^{m_1 n_2} \sum_{n_1=0}^{N_1-1} x_{n_2}(n_1) W_{N_1}^{m_1 n_1} \right] W_{N_2}^{m_2 n_2} \tag{1}$$

The general architecture derived from Eqn. 1 is shown in Fig. 4. There are three architectural control structures: a decimator, expander, and crossbar block. Each of the N_i blocks can be another hierarchical instance of the design where i is the size of the FFT performed in that block. The values of $N_1 \times N_2$ equals N_1 or N_2 at the higher level in the hierarchy.

The decimator block down-samples the input stream [8]. For a sampled signal $x(n)$, the output of the M-fold decimator is given by $y(Mn)$. The sampling of the N_2 decimator is arranged in a regular repeating fashion where the first sample is steered to the first output stream, the second to the second stream and so on. The M^{th} item is steered back to the first stream. This effectively produces M parallel streams operating at $1/M$ the frequency of the input.

The expander block is the dual of the decimator block. They take M low-frequency streams and up-sample by combining them into a stream that has an M-fold higher frequency. In the FFT architecture, the expander operates on a stream of data $x_0(m_2), \ldots, x_{N-1}(m_2)$ reproducing a stream at the original frequency and in the correct functional order for the algorithm.

Product blocks multiply a stream of results coming from the N_1 point FFT units by a set of constant values. Both constants and results are complex numbers, requiring four multiplications and two additions per sample. The constants

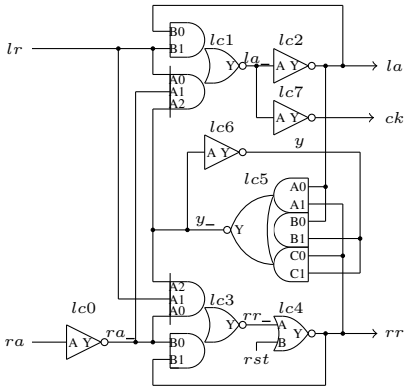

Fig. 5. LC circuit implementation [5]

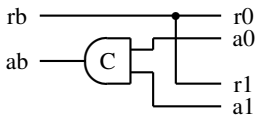

Fig. 6. Fork/Join Template

are calculated by $W_N^{m_1 n_2}$, where $m_1 = 0, \ldots, N_1 - 1$ and $n_2 = 0, \ldots, N_2 - 1$.

The crossbar switch maps results from the product block to the N_2 FFT units. The N_2 FFT units take a transform of time displaced Fourier transform samples. Each N_1-point FFT provides one data sample to each of the N_2-point FFT units. The first row of the N_2 FFT units takes the first sample from each of the N_1 rows, the second row the second sample, and so on. This is implemented by performing an N_2 up-sampling followed by a N_1 down-sampling. Another solution is to steer the data to N_2 N_1-way decimators, followed by N_1 N_2-way expanders. Decimator sequencing here is different than that of the top level block because it steers the first N_2 samples to each row before moving onto the next row.

III. FFT Design

Multi-frequency asynchronous and clocked 64-point FFT designs are implemented from the architecture block diagram shown in Fig. 4. Both designs are hierarchically decomposed at the top level such that $N_1 = 16$ and $N_2 = 4$. The 16-point FFT implementations are also hierarchically decomposed with $N_1 = N_2 = 4$. The terminal hierarchical nodes in the designs is the 4-point FFT blocks since it can be implemented with simple add and subtract operations due to the value of the constant data values. There are four frequency domains in this design. The frequency of the incoming data is f, which gets decimated to derive f/4, f/16 and f/64 frequencies.

The datapath for all the designs are specified behaviorally with the control being the only differentiating point. The asynchronous design is implemented as a bundled data pipeline (Fig. 2). The LC block that controls timing and sequencing is a 4-phase handshake protocol similar to that in Fig. 5. This cell generates a local clock signal to control the pipeline stage based on the handshake with the adjacent handshake controllers.

These designs operate on fixed-point data. The input and output are 32 bits wide, with the upper 16 bits representing the real value and the lower 16 representing the imaginary value. The fields use twos complement representation of signed numbers that are decimal values less than or equal to plus or minus one. The first four bits are used for the whole part of the number and the rest 12 bits for the fractional part.

A. Asynchronous Design

The first step in an RT asynchronous design is to create and characterize the handshake elements. This design uses four circuit elements: a linear pipeline controller (LC) (Fig. 5), a 2-input Fork/Join element, the decimator and expander. The LC circuit interfaces two pipeline stages by controlling the protocol between the stages and storing one data word (Fig. 2). The fork (Fig. 6) broadcasts a request from a sender to two receivers. The ack from the two receivers is synchronized with a C-element before being passed on to the sender [9]. The join element contains the same logic and is the dual of the fork. Requests from two senders are first synchronized before being sent to a receiver, while the ack signal from the receiver is broadcasted to both the senders.

Fig. 8 shows the design of the 4-way decimator. It consists of a ring connected shift register with one bit asserted to steer the requests to four different pipelines based on the value in the shift register. The req and the ack signals are active high. Since only one acknowledgment is active at a time, the four ack signals are passed through an OR gate. Values in the shift register update when the input request goes low. The circuit is characterized for its timing constraints. As long as the shift register can change in one half cycle time (before the next req occurs), this logic will operate correctly. Note that this block adds a 2-input AND gate delay on the request path and a 4-input OR gate delay on the acknowledge path. This is the only overhead of the decimator, and adds approximately 8 gate delays to the cycle time of the architecture, allowing it to operate at approximately a 16 gate delay cycle time. This resulted in a frequency that was close to 1.3 GHz, which we deemed as a sufficiently fast performance target.

The design of the asynchronous expander in Fig. 10 is similar to the decimator. It includes an N_i-bit ring connected shift register and some combinational gates to select the data and control signals to be driven to the output channel.

Once these blocks were designed the top level asynchronous architecture was built by simply composing the pipeline control and datapaths together. We employed a hierarchical structural design style which was almost identical to drawing and connecting block level schematics for the design. In this method a functionally correct design was hierarchically designed and validated for performance and correctness. First a simple 4-point FFT was built, which was used to build a 16-point FFT, and then these components were integrated into the 64-point FFT. The dataflow graph of a 4-point FFT is shown in Fig. 11. The pipelined asynchronous control logic for that design is shown in Fig. 12. The design of the pipeline was almost as simple as drawing the figure for the paper, where the butterfly network and first set of adders are between stages LC1 and LC2, the second butterfly network and adders are between stages LC2 and LC3, and the last network convolution is between stages LC3 and LC4. Following is a code snippet from the design to give you a flavor of the RTL. Some liberty is taken in the syntax to compress the example. This shows a pipeline stage at the input of the design that feeds into the next stage of 16-point FFTs. Each pipeline stage and structural block is similarly designed.

```
module FFT_64 (ri, ai, DI, ro, ao, DO, rst);
   input  [`WORD_SIZE-1:0] DI; ...
   // input pipeline
   linear_control  LC0 (.lr(ri), .la(ai), .rr(p0r),
           .ra(p0a), .ck(ck0), .rst(rst));
   latch   P0   (.d(DI), .clk(ck0), .q(P0D0));
```

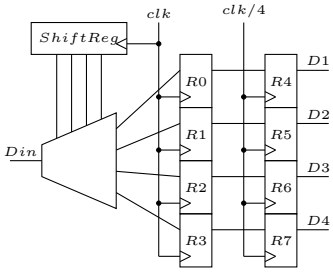

Fig. 7. Synchronous Decimator

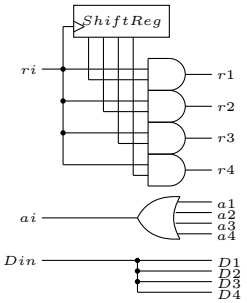

Fig. 8. Asynchronous Decimator

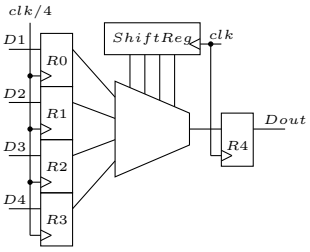

Fig. 9. Synchronous Expander

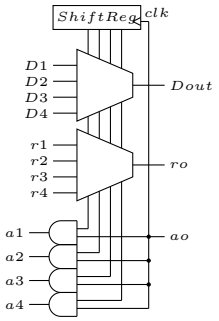

Fig. 10. Asynchronous Expander

```
decimator_4 D4_0 (.DI(P0D0), .D1(P0DT1), .D2(P0DT2),
   .D3(P0DT3), .D4(P0DT4),
   .ri(p0r), .ai(p0a), .rst(rst),
   .r1(p0rt1), .r2(p0rt2), .r3(p0rt3), .r4(p0rt4),
   .a1(p0at1), .a2(p0at2), .a3(p0at3), .a4(p0at4));

// The FFT_16 modules.
FFT_16 F16_0 (.ri(p0rt1), .ai(p0at1), .ro(p1rt1),
  .ao(p1at1), .DI(P0DT1), .DO(P1DT1), .rst(rst));
```

Performance and functionality optimizations in an asynchronous design are somewhat independent operations. A design that is functionally correct can be created relatively quickly. However, particularly for multi-frequency designs, some effort is needed to balance the cycle times and pipelining to optimize performance. This is very different from clocked design where performance and pipelining are essential for correct functionality, and part of the initial specification.

A primary aspect of optimizing performance of an asynchronous architecture is to calculate the critical paths and focus on those. Experimenting with the power-performance tradeoffs allowed us to quickly identify the critical paths in the asynchronous design. Due to the multirate architecture, it was not the complex multipliers or adders that operate at 1/4, 1/16, or 1/64 the input frequency. Rather, the top level decimators and expanders limit the operating frequency of the design. We therefore focused on designing high throughput decimators and expanders.

The performance optimizations will be illustrated with the 4-point FFT pipeline shown in Fig. 11 and 12. From a correctness perspective, the data through the expander could pass straight through the expander through the butterfly network to the adders. However, this would create too long a cycle time at the decimators. Increased performance is obtained by adding pipeline stages before and after the decimators and expanders since they are the critical paths in the design.

The next power-performance optimization of the asynchronous 4-point FFT design was to determine the frequency target for the smallest area and lowest power adder in the given technology. A 16-bit ripple carry adder needed about 860ps in this technology, so that became the performance target of the 4-point FFT design. This was less than the time available for the computation (3ns in the top level 64-point block and 12ns in the 16-point blocks at 1.3 GHz operating frequency). However slowing the operation down beyond 860ps simply adds more area, energy, and latency to the control path.

An additional performance critical aspect of a design is due to pipeline synchronizations. Adding or removing pipeline stages in an asynchronous design can be employed to remove forward and backward stalls in an architecture. This has been referred to as "slack matching" in the asynchronous

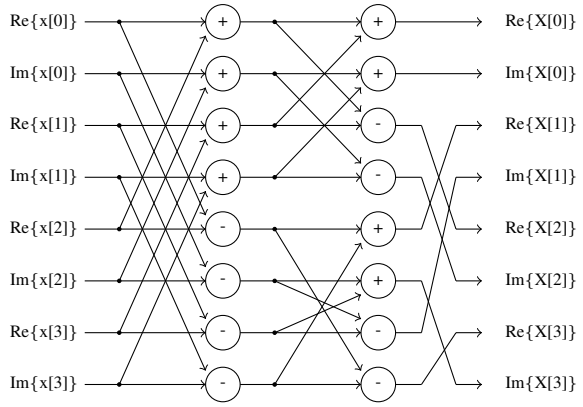

Fig. 11. Data Flow Graph of 4-point FFT Calculation

literature [10]. Therefore a version of the performance critical 4-way decimator was designed as a 2×2 pipelined decimator to increase throughput and reduce sensitivity to backward stalls. Likewise the crossbar and 16-way expander in the 64-point design of Fig. 13 have been pipelined.

The asynchronous design was built using "natural" pipelining for each block with pipeline performance targets based on the top level architecture. For example, pipeline stages exist between the adders of the design whereas they can be removed from a performance perspective. A few modifications to the original pipeline structure have been made to improve area in the "async-opt" design.

B. Synchronous Design

The synchronous FFT is designed using the same architecture in Fig. 4. It consists of clocked decimators and expanders (Fig 7 and 9). The 4-point synchronous FFT design is a six deep pipeline.

Fig. 7 shows the clocked 4-way decimator. The design consists of a high frequency register bank and a low frequency register bank, a clock divider, and a shift register to track the relationship between the two clocks. The shift register must be properly initialized in relation to the global state of the circuit based on data arrival to ensure proper data steering. The data is incrementally latched into the high frequency register bank. At the low frequency clock the data is then shifted into the low frequency register bank, where it is sampled at a $1/N_2$ frequency. The expander in Fig. 9 is the dual of the decimator. The parallel data stored into a low frequency register is streamed and stored in the output register based on the higher frequency clock. The channel selection is dependent on the shift register and requires properly initialization similar to the decimator.

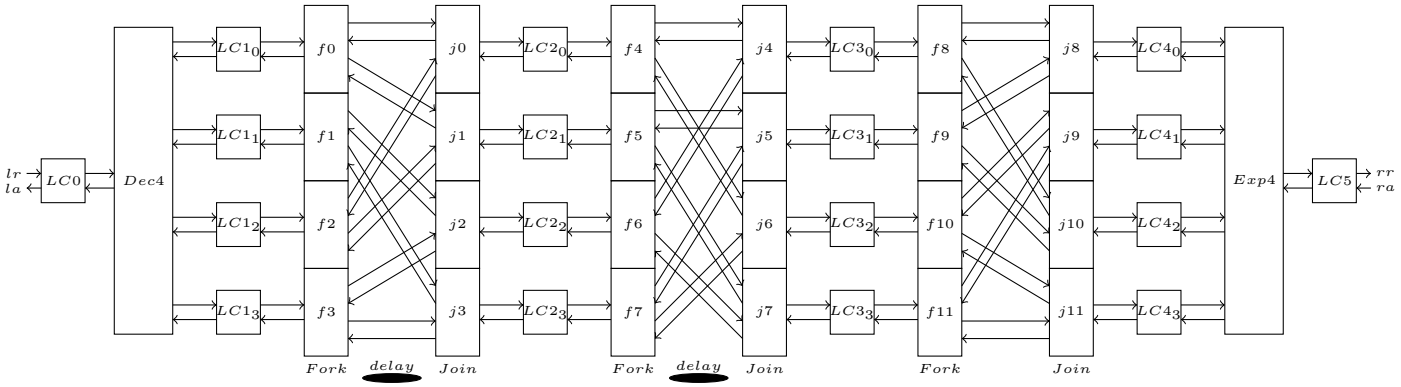

Fig. 12. 4-point FFT Design

IV. Results

These circuits use the Artisan academic library in IBM's 65nm 10sf process. The circuits were designed in behavioral Verilog, synthesized using Design Compiler, and place and routed using SoC Encounter. Circuits were simulated for timing and functional correctness using Modelsim with postlayout parasitics back-annotated. Testing was performed using pre-defined input vectors which included 1024 random numbers. Both 64-point FFT circuits have less than $\pm 0.3\%$ variation as compared to MATLAB FFT computation. Various performance parameters including forward latency, cycle time, and throughput were also generated from the simulation along with VCD (Value Change Dump) file. The simulation VCD file along with the parasitics of the place and routed design was used to calculate the power numbers for each design by PrimeTime PX.

Tab. I and II summarize these multirate designs against several other designs. These 16-point implementations are compared against a design that is similar in architecture [11]. The 64-point benchmark is a low power Texas Instruments design [12]. Performance is measured as the time to completely process 1024 samples.

The simplicity of making architectural and performance modifications to the asynchronous design allowed us to quickly explore a simple area improvement to our asynchronous architecture. The 64-point architecture contains four 16-point FFT's. Each of these contain three complex multipliers operating at 1/16 the top level frequency. At this frequency the multipliers could be shared, removing 8 complex multipliers from the design (Async-opt) resulting in an overall 18% area reduction. This modification had a minor positive affect on performance and negative affect on latency and energy per point. Other modifications that reduce area at little or no energy and performance cost can also be explored, as well as other optimizations based on target versus required frequencies. Asynchronous designs are particularly amenable to such architectural explorations.

For comparison, results in these tables are optimistically scaled to an equivalent for 65nm technology node by using theoretical constant-field scaling assuming the scaling factor $\kappa = 1.43$ per node (let $s = 1/\kappa = 0.7$) [13]. This results in delays in the tables multiplied by s, s^2, and s^6 for the 90nm, 130nm, and 600nm nodes. Energy values are scaled by s^3, s^6, and s^{18}. Area reduces by s^2 per generation.

The biggest advantage of this multi-frequency architecture against the others comes in the form of throughput. These designs can sustain a rate of one data point per clock cycle,

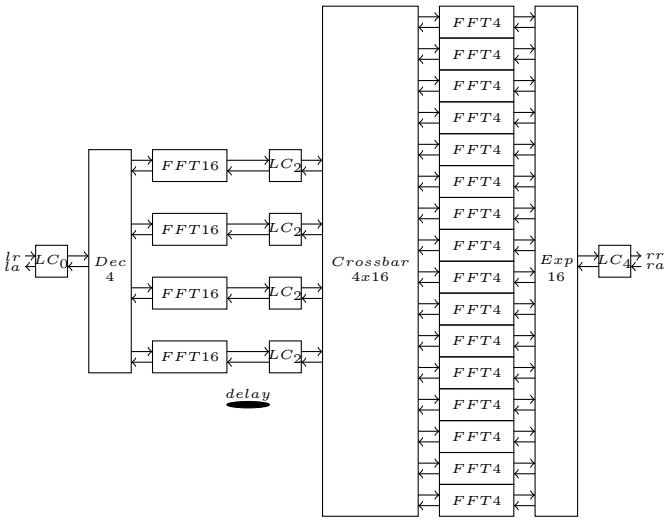

Fig. 13. 64-point FFT Design

at a relatively constant frequency regardless of the point size. The asynchronous design also provides a substantial reduction in latency. From an idle start, the asynchronous 16 and 64-point designs can complete processing 1024 samples over 8 and 32 times faster respectively than the benchmark designs. Multi-frequency design also shines in energy per sample. The asynchronous designs consume approximately one-fourth the energy per sample of the competitors. This 16-point pipelined design is less than half the size of this comparable clocked hierarchical pipelined design. When comparing this design against the low power 64-point design from Texas Instruments, the clocked design and area optimized asynchronous designs consume six and two times the area. This points out the very different design targets and architecture styles. Their architecture shares the computation units for area efficiency at a cost of higher energy and much lower performance.

The Async-opt design is significantly better than the clocked design of the same architecture. The 64-point design shows an improvement of 2.28× the energy per data point and 3.16× the performance while costing only one-third the area.

Accurately comparing FFT designs with different point sizes, technology nodes, and architectures is challenging. Tab. III therefore provides a design comparison based on three metrics - *Benefit Product* [15] and $e\tau^2$ using Baireddy as the reference, and *Normalized FFTs per Energy* [14]. Benefits Product is the product of the area, energy, and execution time. Baas and Chong employ voltage scaling to quadratically reduce energy. Energy times square of the execution time ($e\tau^2$)

TABLE I
THE 16-POINT FFT COMPARISON RESULT (* CONSTANT FIELD SCALED TO 65 NM TECHNOLOGY)

Design	Tech. nm	Points – Samples	Word $bits$	Clock MHz	1K-point Exec. Time μs	Power mW	Energy/point pJ	Area Kgates	Exec.Time Benefit	Energy Benefit	Area Benefit
This Design (Async)	65	16-1024	16	1,274	0.83	30.9	25.05	54	8.32	3.93	2.73
This Design (clock)	65	16-1024	16	588	1.73	24.7	41.83	71	3.98	2.35	2.07
Guan [11]	130	16-1024	16	653*	6.91*	14.6*	98.33*	147	1.00	1.00	1.00

TABLE II
THE 64-POINT FFT COMPARISON RESULT (* CONSTANT FIELD SCALED TO 65 NM TECHNOLOGY, + NOMINAL PROCESS VOLTAGE)

Design	Tech. nm	Points – Samples	Word $bits$	Clock MHz	1K-point Exec. Time μs	Power mW	Energy/point pJ	Area μm^2	Exec.Time Benefit	Energy Benefit	Area Benefit
This Design (Async-opt)	65	64-1024	16	1,357	0.87	69.4	59.23	395	32.24	4.51	0.47
This Design (Async)	65	64-1024	16	1,316	0.87	65.5	55.65	479	32.39	4.80	0.39
This Design (Clock)	65	64-1024	16	667	2.76	50.2	135.30	1,160	10.21	1.97	0.16
Baireddy [12]	90	64-4096	–	514*	28.18*	9.7*	266.95*	186*	1.00	1.00	1.00
Chong (1.1V) (Async) [14]	350	128-128	16	–	1,633.64*	–	4.45*	45*	0.02	59.98	4.12
Chong (3.5V) (Async)+[14]	350	128-128	16	–	513.43*	–	45.06*	45*	0.05	5.92	4.12
Baas (3.3V) [15]	600	1024-1024	20	1,470*	3.53*	11.7*	40.31*	679*	7.98	6.62	0.27
Baas (5V)+[15]	600	1024-1024	20	2,228*	2.33*	40.7*	92.55*	679*	12.10	2.88	0.27

TABLE III
DESIGN COMPARISONS (+ THE NOMINAL PROCESS VOLTAGE)

Design	$e\tau^2$ Advantage	Normalized FFTs per Energy [14]	Benefit Prod. [15]
This Design (Async-opt)	4,683.38	17.35	68.54
This Design (Async)	5,031.00	18.47	60.37
This Design (Clock)	205.60	7.60	3.23
Baireddy [12]	1.00	–	1.00
Chong (Async-1.1V)	0.02	8.33	4.26
Chong (Async-3.5V)+	0.02	17.01	1.34
Baas (3.3V) [15]	421.98	8.44	14.48
Baas (5V)+[15]	421.98	3.31	9.56

provides a reference that is independent of voltage scaling. The Normalized FFTs per Energy metric largely disregards performance. Those results are normalized to the 350nm node to produce the same values as reported in [14].

V. CONCLUSIONS

Multirate asynchronous and a synchronous 16 and 64-point FFT circuits were implemented and compared against published FFT designs. A novel relative-timing based flow which enables the use of pre-characterized sequential templates with synchronous CAD tools and flows to develop asynchronous circuits is used. This work demonstrated that the flow can be efficiently applied to a large asynchronous design. The relative cost of development of an asynchronous circuits with the new flow is similar to its synchronous counterpart for development of these multirate designs. The FFT circuit operates at 1.4GHz and consumes 59.2pJ of energy per data point. A 2.4×, 2.4× and 3.2× benefit in terms of area, energy and throughput respectively over its synchronous counterpart is achieved. Also a 0.48×, 4.5× and 32.20× benefit over a low power 64-point FFT design by Texas Instruments [12] as well as a 2.77×, 8.01× and 8.32× benefit over a similar 16-point FFT architecture [11] are reported respectively for area, energy and throughput.

VI. ACKNOWLEDGMENTS

This material is based upon work supported by Semiconductor Research Corporation Tasks 1817.001 and 2235.001, and the National Science Foundation under Grant No. 1218012.

REFERENCES

[1] J. Cortadella, A. Kondratyev, L. Lavagno, and C. P. Sotiriou, "Desynchronization: Synthesis of Asynchronous Circuits From Synchronous Specifications," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 1904–1921, Oct 2006.

[2] K. Stevens, S. Rotem, R. Ginosar, P. Beerel, C. Myers, K. Yun, R. Kol, C. Dike, and M. Roncken, "An Asynchronous Instruction Length Decoder," *IEEE Journal of Solid State Circuits*, vol. 36, no. 2, pp. 217–228, Feb. 2001.

[3] B. Suter and K. Stevens, "Low Power, High Performance FFT Design," in *Proceedings of IMACS World Congress on Scientific Computation, Modeling, and Applied Mathematics*, no. 1, 1997, pp. 99–104.

[4] B. W. Hunt, K. S. Stevens, B. W. Suter, and D. S. Gelosh, "A Single Chip Low Power Asynchronous Implementation of an FFT Algorithm for Space Applications," in *International Symposium on Advanced Research in Asynchronous Circuits and Systems*. IEEE, April 1998, pp. 216–223.

[5] K. S. Stevens, Y. Xu, and V. Vij, "Characterization of Asynchronous Templates for Integration into Clocked CAD Flows," in *15th International Symposium on Asynchronous Circuits and Systems*. IEEE, May 2009, pp. 151–161.

[6] R. M. Fuhrer and S. M. Nowick, *Sequential Optimization of Asynchronous and Synchronous Fininte State Machines: Algorithms and Tools*. Kluwer Academic, 2001.

[7] Y. Xu and K. S. Stevens, "Automatic Synthesis of Computation Interference Constraints for Relative Timing," in *26th International Conference on Computer Design*. IEEE, Oct. 2009, pp. 16–22.

[8] B. W. Suter, *Multirate and Wavelet Signal Processing*. Academic Press, 1997.

[9] R. E. Miller, *Switching Theory*. New York, New York: Wiley, 1965, vol. 2, chapter 10 reviews Muller's work on speed independent circuits.

[10] A. M. Lines, "Pipelined Asynchronous Circuits," Master's thesis, California Institute of Technology, Pasadena, CA, 1998.

[11] X. Guan, Y. Fei, and H. Lin, "Hierarchical Design of an Application-Specific Instruction Set Processor for High-Throughput and Scalable FFT Processing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 3, pp. 551–563, March 2012.

[12] V. Baireddy, H. Khasnis, and R. Mundhada, "A 64-4096 point FFT/IFFT/Windowing Processor for Multi Standard ADSL/VDSL Applications," in *International Symposium on Signals, Systems and Electronics*. IEEE, 2007, pp. 403–405.

[13] A. Chandrakasan, W. Bowhill, and F. Fox, *Design of High-Performance Microprocessor Circuits*. Wiley-IEEE Press, 2000.

[14] K. Chong, B. Gwee, and J. Chang, "Energy-efficient synchronous-logic and asynchronous-logic FFT/IFFT processors," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 9, pp. 2034–2045, 2007.

[15] B. Baas, "A low-power, high-performance, 1024-point fft processor," *Solid-State Circuits, IEEE Journal of*, vol. 34, no. 3, pp. 380–387, 1999.