Codesign Implementation of STAR in a 3G WCDMA Base Station Receiver

Sébastien Jomphe[1], Jean Belzile[1], Sofiène Affes[2], and Karim Cheikhrouhou[2]

[1] École de technologie supérieure
[2] Institut national de la recherche scientifique, énergie, matériaux et télécommunications
E-mails: sebastien.jomphe.1@ens.etsmtl.ca, jean@ele.etsmtl.ca,
{affes, cheikhro}@inrs-emt.uquebec.ca

***Abstract* – In this paper, we propose an architecture and a dataflow model for the implementation of the Spatio-Temporal Array-Receiver (STAR) in a 3G Base Station within a software-defined radio context. We outline the preferred codesign approach and show how the data can be exchanged between software and hardware efficiently. We then show post-synthesis figures of the FPGA ressources needed to implement STAR.**

Keywords - STAR, 3G, codesign, Base Station.

1. Introduction

With the promise of higher throughput as third generation cellular networks emerge, better methods of channel identification and synchronization will be required. 2D-RAKE-based receivers are still in widespread use today for fading channel identification, not for their performance, but because other methods are usually viewed as too complex.

A promising alternative, the Spatio-Temporal Array-Receiver (STAR) developed earlier in [1] shows a significant performance advantage over RAKE-type receivers. This paper shows how STAR can be cost-effectively implemented into a software-defined radio context using today's programmable components. We will review the simplifications of the original STAR algorithm proposed in [2] and the preferred architecture developed in [3] and show how they translate in terms of hardware complexity.

2. The STAR Algorithm

Fig. 1 shows STAR in a software-defined 3G base station receiver context. The STAR algorithm exploits the time and space diversity offered by smart antenna arrays to estimate the parameters of a radio channel.

STAR replaces the finger-extraction approach of RAKE-type receivers with a more robust and accurate structure-fitting (SF) channel identification [1]. This is achieved by two Decision Feedback Identification (DFI) modules, shown at the top of Fig. 2. The *constrained* DFI is different from the *unconstrained* one in that it applies structure fitting (grey blocks on Fig. 2) to its output $\underline{\tilde{H}}_{n+1}$ (the update of $\hat{H}_n$) to extract a more accurate and robust channel estimate $\hat{H}_{n+1}$ [1]. The unconstrained channel estimate $\underline{\breve{H}}_{n+1}$, though coarse and less stable, is free to track new appearing paths. A comparison is made between the two at approximately every 100 symbols: if we count n_A consecutive discrepancies due to a changing multipatrh profile, we reset $\underline{\tilde{H}}_{n+1}$ to $\underline{\breve{H}}_{n+1}$.

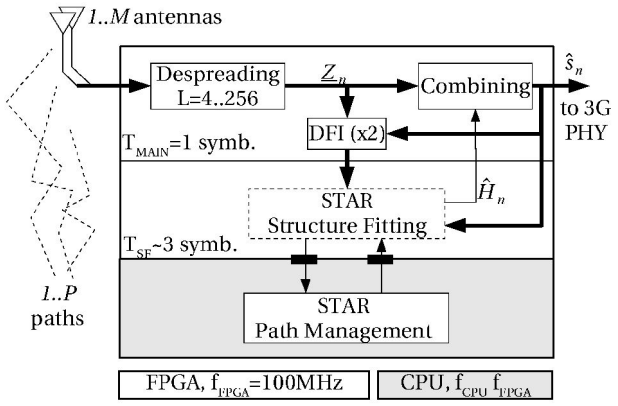

Figure 1. Global architecture of the PHY layer in a STAR-enabled 3G base station receiver. Black boxes represent distributed RAM.

3. Architecture

STAR comes with a global computational complexity of 10^9 complex ops/second, which translates roughly into 3×10^9 real ops/second [3]. Two classes of processing emerge from the analysis of the algorithm: computationally intensive but repetitive operations, and high branch complexity decision logic. The first class includes matrix products, linear filtering and correlation, and is well-suited for parallel hardware implementation. The second class consists of power control, multipath management and user management algorithms, all of which are better suited for a CPU implementation. Clearly STAR is a good candidate for a codesign architecture.

Work supported by NSERC, ISR Technologies inc. and a Canada Research Chair on High Speed Wireless Communications.

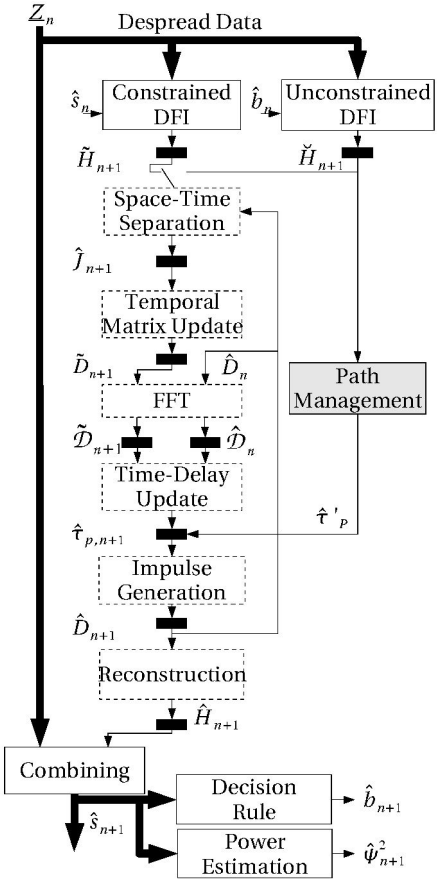

Figure 2. Pipeline structure of STAR. In FPGA: white solid-line blocks form the main data path (T_{MAIN}=1 symbol), white dotted-line blocks make up the structure-fitting pipeline (T_{SF}~3 symbols). In CPU: the gray block performs adaptive path management (T_{CPU}~100 symbols). Black boxes are dual-port RAM elements.

When allocating these functions between FPGA and CPU, we must ensure that closely coupled modules remain within the same implementation domain. Ignoring this would create a high bandwidth requirement through the FPGA/CPU boundary which would in turn outweigh the benefits of the codesign scheme.

Further splitting the structure-fitting portion of STAR into basic operations results in the pipelined model of Fig. 2. Given the serial nature of the algorithm, most of the pipe remains idle during a structure-fitting pass (see [3]), which enables the sharing of one SF between multiple users.

The building blocks required are matrix multipliers (MM), a dynamically-adjustable length FFT processor, a custom regression module capable of tracking sub-sample delays and a fractional-delay filter (FDFIR) for accurate temporal impulse generation.

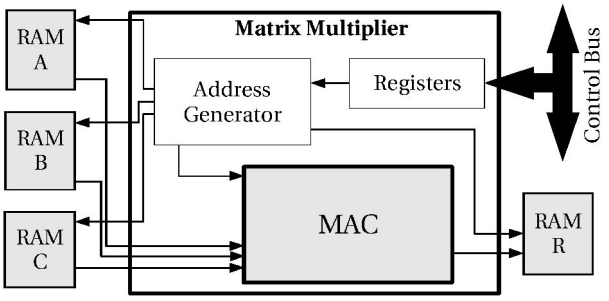

Figure 3. Matrix Multiplier architecture.

Interconnection between the modules is accomplished through dual-port distributed RAM elements (black boxes in Figs. 1 and 2). This allows in some instances a module to process a data set even before the preceding stage has completed its task. In other situations, it lets the CPU update shadow memory areas while processing continues uninterrupted in the FPGA.

Given the worst-case dimensions of the operands, only a quarter of each RAM is required to hold the active data set of one user. With two data sets per user (shadow and current), we can fit two users within the available RAM ressources.

4. Implementation

In order to promote design reuse, we keep the number of different module types to a minimum. As such, we need to generalize the modules so they can accomplish different tasks, being careful not to over-design.

The FPGA part of the algorithm shown in Fig. 2 is handled by five different types of modules. While each type has a different core computation, they all exhibit the same interface. For example, the matrix multiplier (Fig. 3) utilizes an embedded specialized multiply-accumulate (MAC) unit to carry out the sequential multiplication, and has the necessary signals to interface to 4 RAMs and a STAR-wide unidirectional setup bus. The MM by itself simply feeds the right sequence of data from RAMs A, B and C to the MAC and stores the results in RAM R.

All operands depicted on Fig. 2 are matrices using fixed-width binary representation for every element. This width must be specified prior to synthesis for every module. However, each module contains a set of registers that keep track of the dimensions *L*, *M* and *P* (Fig. 1) of the matrices. These values must be dynamically updated by the CPU to reflect the current state of channel identification. With a few more dynamic settings, the MM can perform vectorial product, dot product or norm on its three operands at no further hardware cost.

The correct value for these registers is gathered from the CPU by the global STAR Control Unit (SCU) through a set of mailbox registers. This method facilitates the

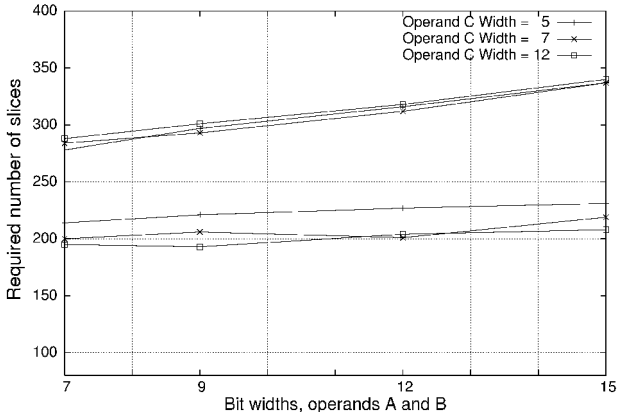

Figure 4: Complexity of modules in number of slices for the complete Matrix Multiplier (upper set) and the Matrix Multiplier without its embedded MAC.

crossing of the FPGA/CPU boundary. Following the precise order of pipeline operations, the SCU reprograms the modules in sequence through a simple unidirectional bus and can effectively bootstrap a whole new identification scheme if needed.

5. Synthesis Results

Design reuse gives us the ability to extrapolate the results of synthesis and place-and-route of a few modules to predict the global size of the STAR receiver.

Taking the MM as an example, we demonstrate that the selection of pre-synthesis generics has a linear impact on the core computation submodule (the MAC in this case), but nearly no impact on the wrapper itself. The upper set of curves in Fig. 4 shows the place-and-route results of several combinations of bit widths for the two multiplicative inputs (operands A and B), and the additive input (each curve in the set) in terms of number of slices. For bit widths varying from 7 to 15, the slice usage varies from ~275 to ~340. We should point out that the width of the result has minor impact on the slice utilization since it only discards unwanted bits which are computed anyway.

The lower set of Fig. 4 shows the size of the MM under the same conditions, omitting the MAC submodule. The nearly-constant number of slices throughout the different bit widths shows the complexity of the MM wrapper at around 225 slices, depending on the size of operand C. We should note however that these figures take into account the presence of operand C in the MAC, whereas it is seldom encountered in STAR, and hence, discarded by the synthesis process, leading to smaller synthesized modules.

The frequency of operation was targeted at 100MHz for a -5 grade XC2VP40 FPGA and was attained without much effort.

TABLE I
Number Of Slices Per Module

Block	Complexity (slices)	Number required	Total
DFI	203	2	406
Matrix Multiplier	309	5	1545
FFT Processor	800	1	800
Time-Delay Regr.	198	1	198
Fract. Delay Filter	213	1	213
Despreader	498	1	498
STAR Control Unit	595	1	595
Total	**---**	**12**	**4255**

Table 1 gives a by-module breakdown of the complexity of STAR for plausible bit widths. Without benefiting from the above-mentioned simplifications, the total number of slices required still shows that STAR can be instantiated within our target platform.

6. Conclusion

In this article we have shown that the STAR algorithm can be implemented in a 3G Base Station within a software-defined radio context.

Through algorithm partitioning we have established a natural frontier between hardware- and software-specific portions of STAR, leading to a codesign implementation.

Furthermore, we have explained the dataflow between modules and over the FPGA/CPU boundary as well as the proposed architecture for constituting modules. Design reuse has been promoted in that many generalized modules are used for many functions.

Finally, we have given figures on the hardware usage both by module and globally, proving that STAR can indeed be implemented cost-effectively in off-the-shelf FPGAs.

REFERENCES

[1] S. Affes and P. Mermelstein, “A New Receiver Structure for Asynchronous CDMA: STAR—The Spatio-Temporal Array-Receiver,” *IEEE J. Select. Areas Commun.*, vol. 16, pp. 1411-1422, Oct. 1998

[2] K. Cheikhrouhou, S. Affes and P. Mermelstein, “Impact of Synchronization on Performance of Enhanced Array-Receivers in Wideband CDMA Networks,”, *IEEE J. Select Areas* Commun., vol. 19, pp. 2462-2476, Dec. 2001

[3] S. Jomphe, J. Belzile, S. Affes and K Cheikhrouhou, “Codesign Implementation of a 3G WCDMA Base Station Receiver”, *IEEE Canadian Conference on Electrical and Computer Engineering, Niagara Falls, to appear, May 2-5 2004.*