CODESIGN IMPLEMENTATION OF A 3G BASE STATION RECEIVER

Sébastien Jomphe[1], Jean Belzile[1], Sofiène Affes[2] and Karim Cheikhrouhou[2]
[1]*École de technologie supérieure* [2]*INRS énergie, matériaux et télécommunications*
sebastien.jomphe.1@ens.etsmtl.ca

Abstract

In this paper we show how a software/hardware codesign approach can lead to the implementation of a Spatio-Temporal Array-Receiver (STAR) in a 3G WCDMA base station context. STAR has already proven substantial gains over the 2D RAKE [1], and current technology is now sufficiently integrated to allow its cost-efficient hardware implementation.

To this end, we demonstrate how the STAR algorithm can be translated to fit into a mixed FPGA/GPP (General Purpose Processor) architecture. We also define a data flow path including pipelining and sequencing which can be used to increase the number of channels that can be processed simultaneously within the system.

Keywords: *STAR, Codesign, 3G, WCDMA, Base Station.*

1. INTRODUCTION

As third generation cellular systems are being deployed, it has become apparent that new methods of channel estimation and equalization are required to cope with the anticipated higher data throughputs. 2D-RAKE structures have traditionally been employed for this purpose, not for their performance, but because other methods are often viewed as too burdensome.

However, the Spatio-Temporal Array-Receiver (STAR) developed earlier in [1] offers a significant performance advantage over RAKE-type receivers [2]. In this paper, we endeavor to show that STAR can be implemented into a software-defined radio context. Based on simplifications found in [2], we will show how pipelining and chaining of the structure-fitting subsystem can lower the inherent latency for STAR and enable the sharing of hardware ressources between between multiple users. We will also make use of a software/hardware codesign approach to offload high-latency, logical decisions onto an accompanying processor, such as a GPP (general purpose processor) or a DSP (digital signal processor).

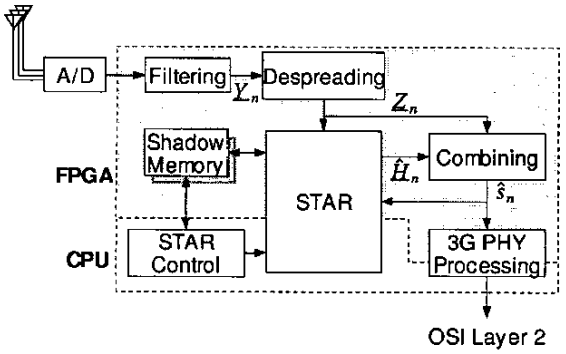

Figure 1: Global architecture of the PHY layer in a STAR-enabled 3G base station receiver.

2. GLOBAL ARCHITECTURE

2.1 STAR

The STAR algorithm exploits the time and space diversity offered by smart antenna arrays to estimate the parameters of the radio channel. With a quarter-wavelength of a few centimeters for 3G, spatial diversity could even be harnessed in handhelds, though the required processing power is more easily confined into a base station. Fig. 1 shows the PHY layer architecture of such a software-defined 3G base station.

What is novel about STAR is that it replaces the conventional finger-extraction approach of RAKE-type receivers by a more robust and accurate structure-fitting channel identification [2]. As shown on the top of Fig. 2, channel identification in STAR is achieved by two Decision Feedback Identification (DFI) modules. The constrained DFI version is different from the unconstrained one in that it applies structure-fitting (see blocks in grey color on Fig. 2) to its output $\tilde{H}_{n+1}$ (the update of $\hat{H}_n$) to extract a more accurate and robust channel estimate

CCECE 2004 - CCGEI 2004, Niagara Falls, May/mai 2004
0-7803-8253-6/04/$17.00 © 2004 IEEE

TABLE I
Computational Burden for STAR

Block	Complexity n_{ID}=1 (Mops)	Complexity n_{ID}=10 (Mops)	Location
Constrained DFI	66	6.6	FPGA
Unconstrained DFI	66	6.6	FPGA
Space-time Separation	97	9.7	FPGA
Temporal Matrix Upd.	320	32.0	FPGA
FFT on $\tilde{D}$	246	24.6	FPGA
Time-Delay Update	42	4.2	FPGA
Impulse Generation	102	10.2	FPGA
FFT on $\hat{D}$	246	24.6	FPGA
Reconstruction	82	8.2	FPGA
Difference Extraction	195	19.5	CPU/stats.
Localization Spectrum	54	5.4	CPU/stats.
Appearing Path Decision[a]	20	0.2	CPU/ctrl
Vanishing Path Decision[a]	2	negl.	CPU/ctrl
Combining	65.3	65.3	FPGA
Total	**1603**	**219.1**	**mixed**
Total (FPGA)	**1332**	**192.0**	**FPGA**
Total (CPU)	**271**	**25.1**	**CPU**

Table 1: Computational burden for the proposed codesign architecture. Light gray denotes blocks running once every n_{ID}=10 symbol. Dark grey denotes $n_{IDCPU(CTRL)}$ =100. Results are shown for M=2 antennas, P=3 tracked paths and L=32. White blocks operate at symbol rate.

[a] Part of "STAR Control in Fig.1.

$\hat{H}_{n+1}$ [1]. The unconstrained channel estimate, $\breve{H}_{n+1}$, though coarse and less stable, is free to track new appearing paths. A comparison between the two is carried out every 100 symbols or so. A significant difference between the two suggests the detection of appearing paths and triggers an update of the number of tracked multipaths, P, which resets the constrained DFI output $\tilde{H}_{n+1}$ to $\breve{H}_{n+1}$ (see Fig. 2). The reader is encouraged to refer to [1],[2] for an in-depth explanation of STAR.

2.2 Codesign Approach

There are three parameters to consider when partitioning an algorithm between hardware and software, as in Fig. 1. The first is the computational load (in 10^6 ops./sec, or *Mops*) of each module. The second is the bandwidth required by closely coupled blocks to exchange data. The third is the branch complexity of decision blocks.

Modules with a high computational load are natural candidates for a purely hardware implementation. Those with a high branch complexity are more appropriately realized in a CPU. The separation must however insure that closely coupled modules remain within the same implementation domain, otherwise the computational savings could be outweighed by the bandwidth requirements.

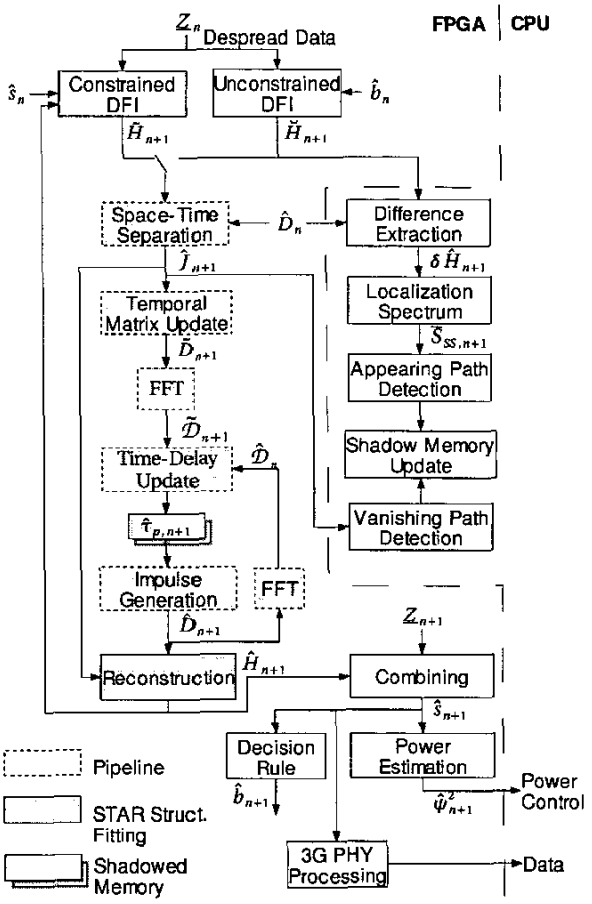

Figure 2: Pipeline Architecture of STAR

3. ARCHITECTURE

3.1 Computational Burden

Modifications to the original STAR algorithm have been carried out in [2], lowering the computational burden by 1 to 2 orders of magnitude.

Ideally, as in [1], channel estimation should take place every symbol. That is, the structure-fitting subsystem (grey blocks in Fig. 2 and Table 1) should be excited by every incoming symbol, requiring a total processing power of nearly 1.6 Gops. However, [2] makes the key assumption that the channel parameters are nearly time-invariant for the duration of a few symbols. Updating the

channel parameters every n_{ID}=10 symbols still preserves the channel tracking performance but dramatically reduces the required processing power. Table 1 shows that activating the structure-fitting pipeline every n_{ID}=10 symbols and re-examining the number of tracked paths every $n_{IDCPU(CTRL)}$=100 symbols requires a reasonable complexity of 219 Mops per radio channel.

Another key element of [2] is the introduction of a reduced processing window of length L_Δ. That is, the despreading operation is carried out with the entire spreading code over $2L$-1 chips, but only the first L_Δ elements of the cross-correlation vector are considered. When tracked properly, the main power components should not escape this reduced window. Experimental results in [2] suggest that a value of L_Δ=32 is sufficient. All H and D operands are dimensioned along L_Δ, so a lower value has a direct impact on matrix multiplications and FFT operations. With spreading factors smaller than 32, L_Δ takes the value of L.

3.2 Partitioning the algorithm

Implementing STAR as a software/hardware codesign architecture implies separating the highly regular computations from the control algorithms.

3.2.1. Core Computations. From the description of STAR [2] and our discussion about n_{ID}, we establish a frontier in the STAR block of Fig. 2 between the structure-fitting and the path management modules. Used every n_{ID} symbols, the structure-fitting pipeline consists mainly of matrix multiplications and sums. Though the dimensions of the operands can vary (eg. with changing P), an upper limit of M=2, L_Δ=32 and P_{MAX}=5 has been impose for implementation within an FPGA.

3.2.2. Decision Blocks. We can further partition STAR by recognizing that radio paths do not appear and vanish instantaneously. Therefore, the decision to add or remove them from the tracking process need not be made on a symbol-by-symbol basis. In fact, STAR only triggers such a change after $n_A = n_V \approx 100 = n_{IDCPU(CTRL)}$ iterations of continuous detection of an appearing or vanishing path, respectively. This translates to an update rate of only a few hundred Hz, hence moving these blocks to the CPU is natural. Moreover, since a given decision need not be taken at a specific moment, the underlying statistics can suffer some lag and thus, can be moved into the CPU where they will be updated every $n_{IDCPU(STAT)} \approx 10$ iterations.

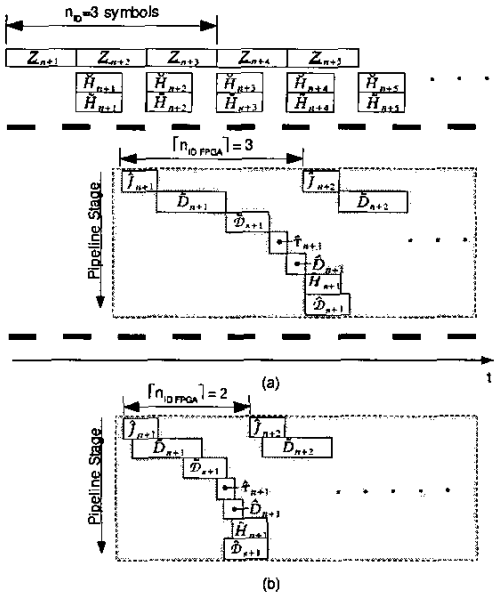

Figure 3: Fastest possible n_{ID} for STAR at L=32 for (a) the unchained pipeline and (b) the chained pipeline.

3.2.3. Inter-process Communications. The preferred mechanism to exchange data between the FPGA and the CPU is through shadow memory. In the current implementation, when the CPU detects a change in the number of paths, it rebuilds a coherent set of matrices in shadow memory and the FPGA is instructed to swap active and shadow memories at its next structure-fitting iteration. The CPU then reclaims the formerly active memory as shadow memory for a subsequent update. Since both CPU and FPGA can access the same physical memory resource on their own clock domain, the exchange from CPU to FPGA is a single data word: a new memory base address. This is known as the mailbox principle.

3.3 Pipeline Architecture

We have identified the structure-fitting pipeline in Fig. 2 as the critical path in the system. Our efforts are directed at reducing the length of its longest feedback.

A number of feedbacks are required from iteration n to n+1. This precludes the instantiation of a true pipeline structure. Specifically, the longest feedback begins with the temporal propagation matrix $\hat{D}_n$ and branches to the next iteration into the spatial propagation matrix $\hat{J}_{n+1}$. This dependency has not been resolved and we are forced to accept that the pipeline will be mostly empty. This situation can however be harnessed in two ways.

A first interesting concept is to feed many independent sets of data to the structure-fitting pipeline in an interleaved fashion. With each set representing the radio channel of a given end-user, we can effectively share a single structure-fitting pipeline between many users. A second way is to chain the pipeline, as described in the next section.

3.4 Chaining the Pipeline

Chaining allows the processor at stage k+1 access the output of stage k before actual completion. Since data exchange between neighboring modules is achieved through distributed shared memory, we must grant two separate processes access to the same memory resource at the same time. The preferred method is to use distributed dual-port memory resources. In doing so, we must however make sure that the input rate of stage $k+1$ does not exceed the output rate of stage k or a WAR ("Write-After-Read") data hazard could occur. The FFT core [3] is a good candidate for chaining because its I/Os are both serial and of the same rate.

Considerable care is required when designing a chained pipeline since different processors may require their input data in row-major and others may not (eg. for transpose matrices). This affects the actual chaining indent that each processor can gain.

3.5 Modified MAC

The bulk of the operations contained in the structure-fitting chain consists of matrix-MUL-matrix-ADD-matrix-type operations. Since all operands are dense matrices, no simplification can take place, so a specialized multiply-accumulate (MAC) architecture is introduced.

As shown in Fig. 4, a third input has been added to a traditional MAC structure in order to facilitate the matrix-times-vector-plus-vector operation which is frequent in STAR. Two stages (one per clock cycle) have been defined in the MAC, which considerably reduces the overall pipeline delay.

In comparison, a sequential processing unit requires 4 multiplications and 6 additions (2 real and 2 complex) to complete a partial result, for a total of 10 operations. While our MAC requires two cycles to produce a result, its pipelined architecture permits the chaining of operations. As the number of elements in the input vectors increases, the speedup over a purely sequential processor approaches 10.

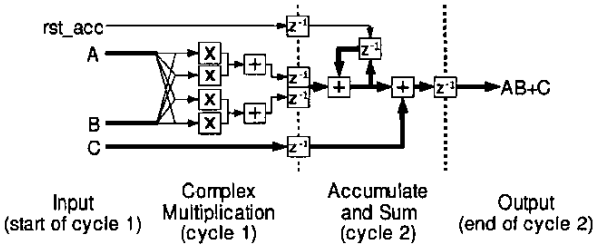

Figure 4: Two-cycle, complex operand MAC structure for the STAR pipeline. Bold lines denote complex numbers. Thin lines are either real or imaginary parts.

4. CONCLUSION

Through the use of a software/hardware codesign approach, we have shown how the Spatio-Temporal Array-Receiver (STAR) can be implemented using available technology. This receiver outperforms the 2D-RAKE in a WCDMA base station context.

By sorting the highly repetitive calculations from the branching operations, we have worked out a codesign architecture. Through algorithmic modifications that further emphasize the difference between software and hardware, we have reduced the latency of the structure-fitting pipeline thereby increasing the idle time of the pipeline.

We have shown how the structure-fitting pipeline can be further optimized to allow interleaved processing of multiple users or, through chaining, allow faster and more accurate processing of a single user.

Finally, a modified MAC allowed us to gain a speedup factor of nearly 10 relative to a sequential architecture.

Acknowledgments

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

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] M.-E. Grandmaison, J. Belzile, C. Thibeault and F. Gagnon, "Reconfigurable and Efficient FFT/IFFT Architecture," accepted for publication at CCECE, Niagara Falls, May 2004