# OPTIMIZED FFT ARCHITECTURE FOR MIMO APPLICATIONS

*Ludwig Schwoerer and Ernst Zielinski*

Nokia Research Center
Meesmanstr. 103, 44807 Bochum, Germany
phone: + (49) 234 984 3483, fax: + (49) 234 984 3491, email: ludwig.schwoerer@nokia.com
web: www.nokia.com/research

## ABSTRACT

Upcoming wireless transmission systems, such as 4G, will most likely utilize MIMO in order to enhance data rate and link reliability. OFDM will be the enabler to combat the effects of a multipath environment.
In this paper we present results that we have obtained during the design of our FPGA-based MIMO concept demonstrator. Special attention will be given to FPGA suitable implementation of the key elements of our 4x4 MIMO OFDM system. In particular we demonstrate a fully pipelined FFT that is capable of processing up to 4 streams in parallel.

## 1. 4G SYSTEM SCENARIO

Right now, several proposals are under discussion on how to specify the next generation cellular wireless system. The 4G concept as shortly summarized here, is one of these. It shall provide a revolutionary approach in order to satisfy the high throughput and coverage targets that the increasing mobility demand is calling for.
The key parameters of the particular 4G proposal we are aiming at are:

- 100 MHz Bandwidth
- Multi-Carrier Transmission (2k-OFDM)
- High Spectral Efficiency
    - 1 bps/Hz (mobile wide area)
    - 10 bps/Hz (pedestrian 'hot-spot')
- MIMO processing
    - Up to 4 x 4 configuration

In order to give implementation feedback already in the early phases of system and algorithm design, we are building up an FPGA based demonstrator.
This gives the basis for architectural exploration and to do some trade-offs between algorithm performance and implementation complexity. The coarse block structure of our demonstrator is outlined in the next section.

## 2. DEMONSTRATOR STRUCTURE

The high computational complexity of the underlying algorithms together with the high throughput requirements of our system are out of range with any single-FPGA solution. Therefore, the target is to have a 6 FPGA system, where the algorithmic building blocks are mapped to the individual FPGAs as depicted in Figure 1.

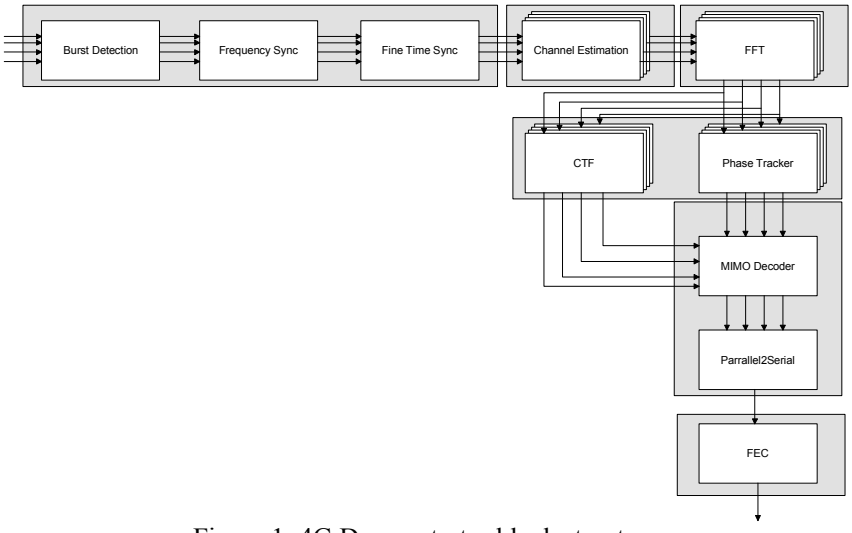


Figure 1: 4G Demonstrator block structure

One of the central algorithms in this system is the FFT, which must be capable of transforming 4 streams at 100 MSamples/s each. Therefore an implementation is needed that can be efficiently pipelined in order to achieve this high throughput, and at the same time minimizes computational complexity, memory size, and control overhead. Also a straightforward extension approach towards an ASIC integration with much higher clock rate compared to FPGA is favourable, so that the architecture can be efficiently reused in final implementations.

## 3. TRADITIONAL MIMO-FFT

In this section, we want to outline two common solutions, how every kind of FFT architecture can be expanded to support 4 input streams. For both of them the required memory size is indicated.

### 3.1 Parallel transformation

The FFT transformation is a central process in conventional OFDM (SISO-OFDM: single-input-single-output OFDM) systems. The transition to MIMO technique results in an OFDM system with $M_R$ (= number of receiver antennas) FFT transformation processes in parallel. Typically, $M_R$ FFT blocks are implemented, i.e. one for each stream (see Figure 2 for $M_R$=4). It can be seen that the complexity of such a system grows linearly with the number of antennas (i.e. $M_R$ times the FFT complexity).

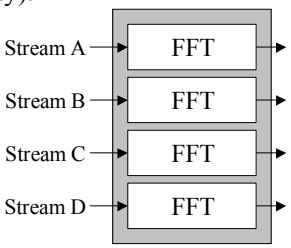


Figure 2: Parallel FFT transformation

The required memory size is given by:

$$FFTMem \quad M_R \qquad (1)$$

with *FFTMem* the memory size of one FFT transformation block.

### 3.2 Serial blockwise transformation

For complexity reduction of the system, the transformation process can be done serially by a smaller number ($M_{FFT}$) of FFT blocks (straightforward serial FFT solution). In order to transform $M_R$ parallel streams serially, the FFT has to work at higher rate $M_{Rate}$. Figure 3 illustrates the block diagram for $M_R$=4 and $M_{FFT}$=1.

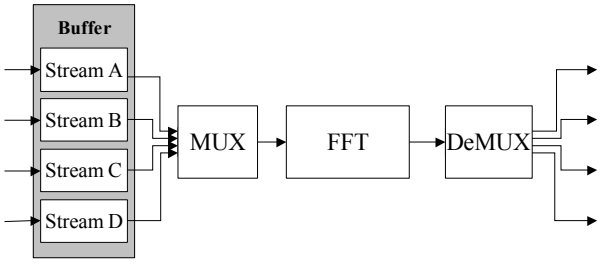


Figure 3: Serial blockwise transformation

Due to this processing, the input streams are multiplexed in a blockwise fashion in front of the FFT and demultiplexed after. This strategy results in a reduction of computational complexity, depending on the sharing ratio ($M_R/M_{FFT}$).

Unfortunately, each stream requires an additional input buffer that collects one OFDM symbol before sending it to the FFT. In Figure 4 this buffering is indicated as Buffer area I.

Furthermore, additional buffer space is needed, because read accesses are scheduled in a way that reading overlaps with writing of the subsequent OFDM symbols. This additional FIFO operation is shown as Buffer area II in Figure 4.

Of course, both memory areas can be merged to form one circular buffer per stream.

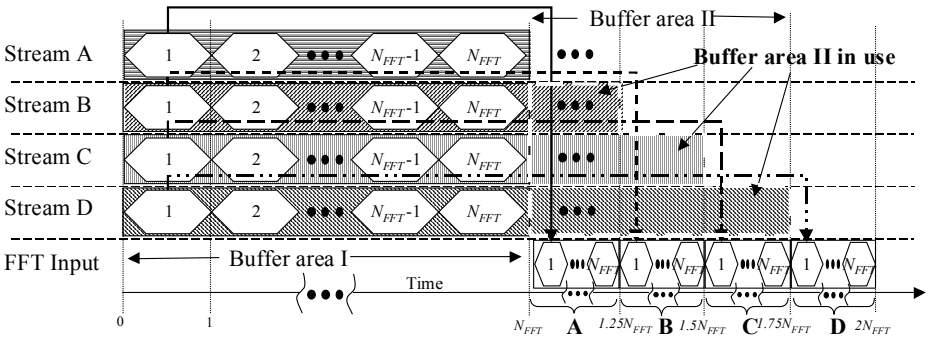


Figure 4: Timing diagram for the serial blockwise FFT transformation

In a first step, $N_{FFT}$ (the size of the FFT) samples of each stream are written to the corresponding stream buffer. Finally, after the buffering period, each buffer successively shifts its content into the FFT block, which works at a higher rate. Since the buffer content of the streams is used sequentially and new data symbols are continuously fed to the FFT at the same time, additional buffer space is needed. The additional memory requirement for this multiplexing scheme is given by:

$$\underbrace{M_R N_{FFT}}_{BufferI} + \underbrace{\left( \frac{M_R^2}{M_{FFT}} \quad M_R \right) \frac{N_{FFT}}{2M_{Rate}}}_{BufferII}. \qquad (2)$$

In addition, the FFT uses a memory in the size of *FFTMem*. Thus, the overall memory size (complex symbols) is given by:

$$\underbrace{M_R N_{FFT}}_{BufferI} + \underbrace{\left( \frac{M_R^2}{M_{FFT}} \quad M_R \right) \frac{N_{FFT}}{2M_{Rate}}}_{BufferII} + \underbrace{(FFTMem) M_{FFT}}_{FFT} \qquad (3)$$

For our demo system with $(M_R/M_{FFT}) = M_{Rate} = 4$ and $M_{FFT} = 1$ equation (3) simplifies to:

$$\underbrace{M_R N_{FFT}}_{BufferI} + \underbrace{\frac{3M_R N_{FFT}}{8}}_{BufferII} + FFTMem \qquad (4)$$

Assuming the FFT core utilizes only one buffer of size $N_{FFT}$ (the minimum number for an FFT to run), *FFTMem* equals $N_{FFT}$ and thus equation (4) becomes:

$$\frac{11M_R + 8}{8} N_{FFT} = M_R N_{FFT} + \frac{3M_R + 8}{8} N_{FFT} \qquad (5)$$

So as a conclusion, the serial blockwise transformation, while reducing the computational complexity by $(M_R/M_{FFT})$, requires $\frac{3M_R + 8}{8} N_{FFT}$ more memory compared to the parallel transformation.

This approach can be applied regardless of the internal structure of the FFT. But anyhow, as the FFT is targeted to run at 4 times the sample frequency, a fast implementation of the FFT is needed.

While looking at possible architectures for such a high-speed FFT, we have identified the Radix 2² single-path delay feedback (SDF) architecture [1] to be very suitable for our application.

## 4. OPTIMIZED MULTISTREAM-FFT

The Radix 2² FFT belongs to the class of pipeline FFT architectures. For a hardware-oriented implementation, this approach combines the advantages of the radix 4 and radix 2 approaches. The radix 4 requires minimum of non-trivial multipliers, whereas the radix 2 uses a simple butterfly structure. Its hardware complexity is thus shown to be minimal on both dominant components: $\log_4 N_{FFT} - 1$ complex multipliers or CORDIC [2] elements and $N_{FFT} - 1$ memory size [1]. This makes it very attractive for our MIMO OFDM system, where FFT has to be done on $M_R$ streams. Its internal structure is depicted in Figure 5 for $N_{FFT} = 2$k.

In N-by-$M_R$ MIMO systems, there are $M_R$ data input streams in parallel. For this reason, an FFT architecture is also implemented which is able to process several data streams simultaneously at a rate $M_R$ the sample rate.

Due to the internal structure of the Radix 2² FFT it is possible to change the multiplexing scheme from blockwise to samplewise.

Figure 6 illustrates an FFT architecture for 4 parallel data streams.

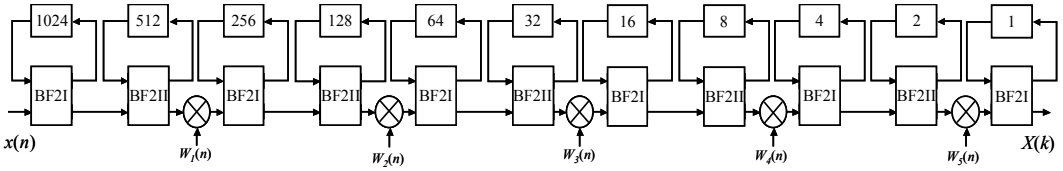


Figure 5: Block diagram of the Radix 2² FFT, $N_{FFT} = 2$k [1]

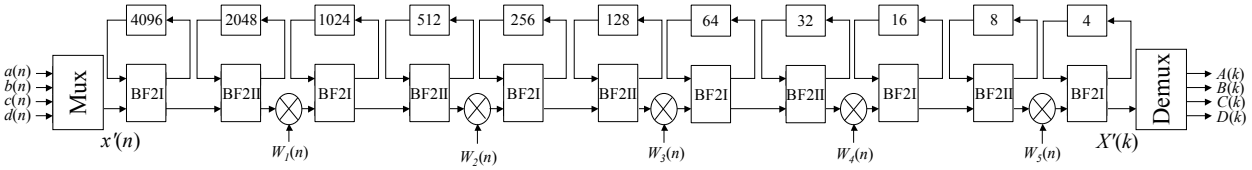


Figure 6: Multi-stream FFT architecture for a 4 antenna MIMO receiver.

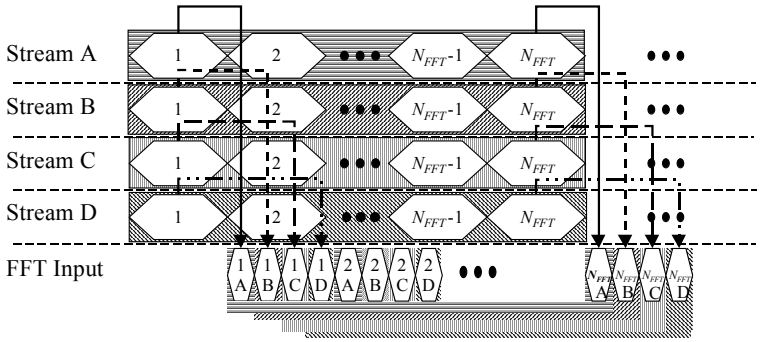


Figure 7: Timing of the proposed FFT architecture

Figure 7 shows the read/write timing of the samplewise multiplexing and thus also the FFT internal processing.

In the first step of the process, the data streams $a(n)$, $b(n)$, $c(n)$ and $d(n)$ are multiplexed to a single stream $x'(n)$ that is directly fed to the FFT pipeline. For this reason, there is no need to introduce input buffers, which would have at least the size given in (2). For the transformation of the input $x'(n)$, the architecture is slightly modified. Due to the four-fold amount of data at each stage, the FIFO memory size is extended by a factor of four. In addition, since the same twiddle factors are used for each of the four streams, the twiddle factors change four times slower compared to the single stream FFT. Finally, the transformed data streams are demultiplexed corresponding to the multiplexing at the beginning of the FFT.

The overall memory size is $M_R$ ($N_{FFT}$ -1), which is exactly the size given in (1), if the Radix 2² FFT is used with $FFTMem = N_{FFT} - 1$, but with the computational complexity of just one FFT. Thus it is of the same complexity as the architecture described in chapter 3.2, but with a smaller memory size. Because of the interleaved data processing within the FFT, there is no need for buffering of the FFT inputs. Table 1 summarizes complexity figures (number of complex multipliers) and memory consumption of all three investigated approaches, assuming Radix 2² FFT as basic core.

| | Complexity | Memory Size |
|---|---|---|
| Parallel FFT | $M_R (\log_4 N_{FFT} - 1)$ | $M_R (N_{FFT} - 1)$ |
| Blockwise Multiplexing | $\log_4 N_{FFT} - 1$ | $M_R N_{FFT} + \frac{3M_R + 8}{8} N_{FFT}$ |
| Samplewise Multiplexing | $\log_4 N_{FFT} - 1$ | $M_R (N_{FFT} - 1)$ |

Table 1: Complexity and Memory requirements

## 5. CONCLUSIONS

An optimized FFT architecture has been presented that achieves both lowest computational complexity and smallest memory size for MIMO OFDM applications.

If timing constraints on the FPGA permit, the proposed structure can directly be utilized, otherwise the parallel implementation is to be preferred. Also then, a clear migration path towards later ASIC implementation is given by changing the structure from parallel to samplewise multiplexing. This change is easily accomplished by just quadruplicating the FIFO lengths in the Radix 2² FFT core.

## 6. REFERENCES

[1] S. He and M. Torkelson, "A New Approach to Pipeline FFT Processor." *Parallel Processing Symposium*, 1996., *Proceedings of IPPS* '96, *The 10th International* , 15-19 *April* 1996 Pages:766 – 770

[2] J.E. Volder, "The CORDIC Trigonometric Computing Technique", IRE Trans. Electronic Computers, Vol EC-8, page 330-334, 1959