Rev. 0, 12/2002

A Smart Antenna System for 3G Wireless Using the MSC8102 DSP Device

By Leon Turner and Dipesh Koirala

CONTENTS

This application note describes the resource requirements for implementing a high-level smart antenna or adaptive antenna (AA) uplink wideband Code-division multiple access (WCDMA) system using the StarCore®-based Motorola MSC8102 and ASIC(s). The baseline system supports 128 voice users, and the resources required to support adaptive antenna, chip rate, and symbol rate processing for a WCDMA uplink are all included in this implementation.

Two candidate adaptive algorithms are considered, which are based on the Minimum Mean Squared Error (MMSE), namely, Least Mean Square (LMS) and Recursive Least Squares (RLS). Both algorithms are computationally simplified, approximate methods of minimizing the mean squared error. The LMS algorithm was selected for this study, and the system requirements for the LMS adaptive algorithm are considered. The computations, memory, and input/out estimates for 128 voice or four 384 Kbps high data rate users for the Third Generation Partnership Project (3GPP) illustrate that the MSC8102 device can be used effectively and efficiently for smart antennas with LMS in a WCDMA system for 3G wireless infrastructure.

1 Adaptive Antenna Basics

Within a generic AA uplink system, the AA weight multiplication occurs between the control channel and data channel despreading (within the beam forming block shown in **Figure 1**) and conventional CDMA chip-rate control and data processing, which consists of automatic frequency control, channel estimation, time tracking, and maximal ratio combining. The AA processing also requires feedback from the conventional CDMA chip-rate processing. For details on a typical AA system, consult reference [3]. Various algorithms can be used to accomplish the adaptive weight control depicted in **Figure 1**. Several are based on the MMSE criterion [4]. The mean squared error is as follows:

$$mean\ |W^{H} \times r\ (n) - d|^{2}$$

where

- W is the weight vector (n_ants × 1 complex vector)
- $r\ (n)$ is the received de-rotated symbols (n_ants × 1 complex vector)
- d is the target output
- n is the time index

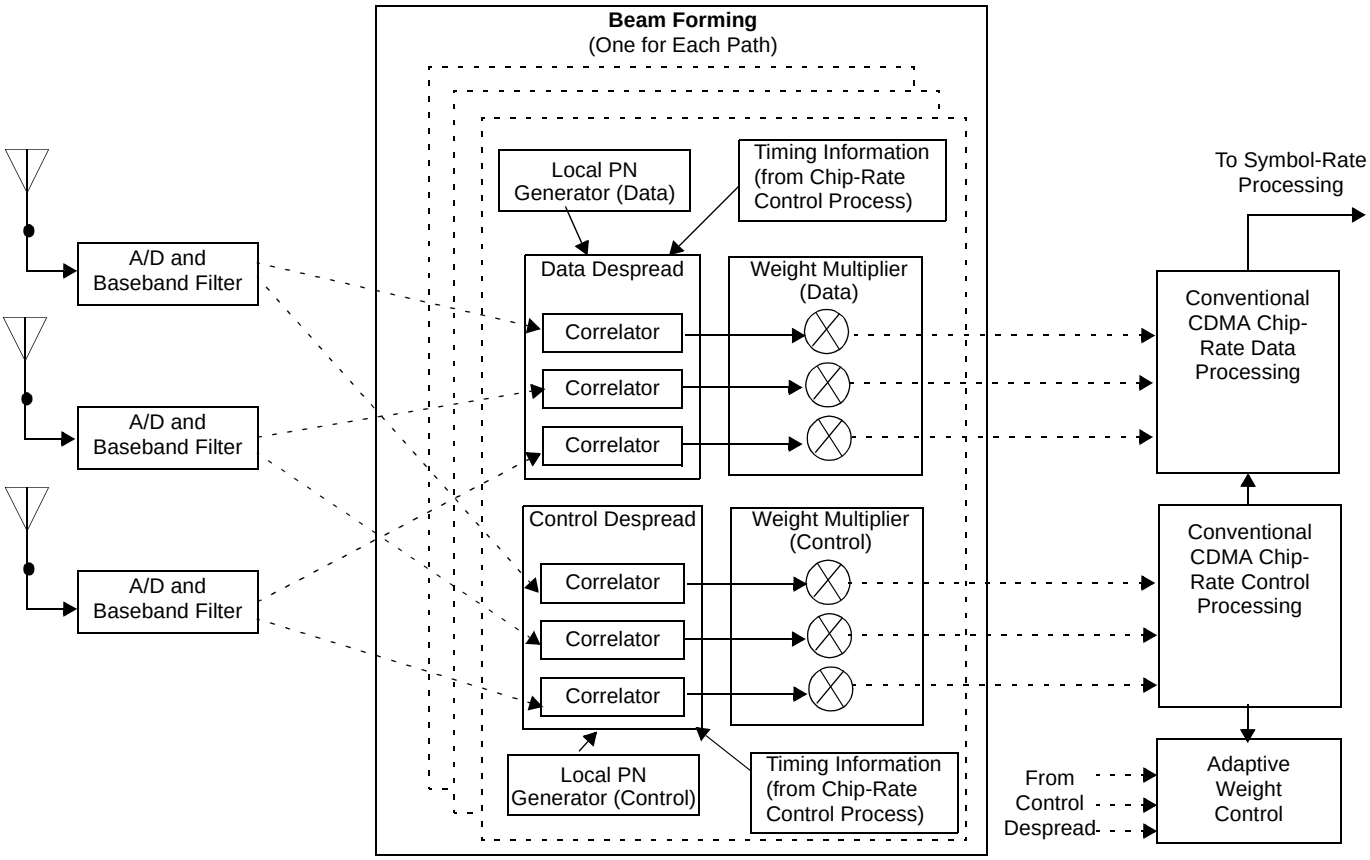

Figure 1. A Generic Adaptive Antenna (AA) Uplink WCDMA System

1.1 Least Mean Squares (LMS)

The system described in this application note uses ordinary LMS without normalization, with a trade-off between computational cost and performance. The performance of the normalized LMS algorithm is generally superior (especially if the signal power varies significantly due to signal propagation anomalies such as fading). However, normalization involves a division operation that is computationally very expensive. Flowcharts for the LMS and NLMS algorithms are shown in **Figure 2** and **Figure 3**.

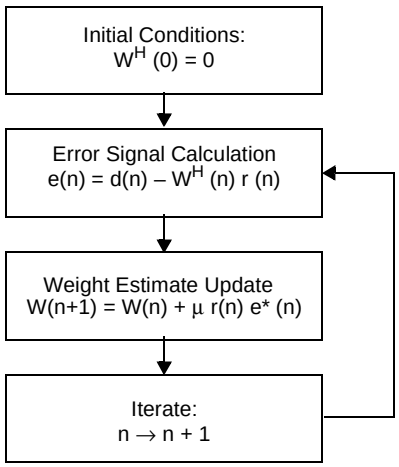

Figure 2. LMS Algorithm

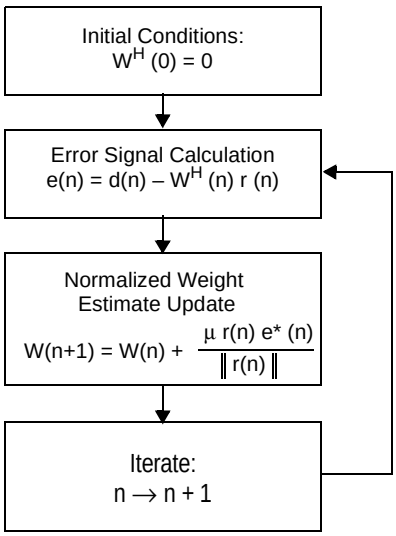

Figure 3. Normalized LMS Algorithm

The notation in **Figure 2** and **Figure 3** is as follows:

- *e(n)* and *d(n)* are the complex error and real target outputs, respectively
- W(n) is the N × 1 complex array weight vector
- *r(n)* represents the N × 1 complex array input vector.
- *N* is the number of antennas in the smart antenna system
- *n* is the time index
- μ is the convergence parameter normally called step size in gradient descent-based algorithms.

The LMS algorithm is implemented as specified in [1], according to the following Matlab code:

```
% Definitions:
% R - Input data vector(complex, n_ants X 1)
% W - Weight vector output matrix (complex, n_ants X n_inputs)
% d_fix - target output (real)
% e - error (complex)
% mu - convergence parameter (supplied by user)
% mu = 0.5 in current simulation

% Initial conditions
W(:,1) = 0;

% Update step
e = d_fix - W(:,iter)'*R;
W(:,iter+1) = W(:,iter) + mu*(e'*R);
iter = iter+1;
```

Figure 4 shows the directional power response for an 8-element linear antenna array that is adapted using the Matlab algorithm. Powers values are normalized by the maximum direction power response. Circular marker size in **Figure 4** indicates the relative strength of interfering sources. As the figure shows, the response is successfully steered towards the desired signal source and away from the strongest interfering sources.

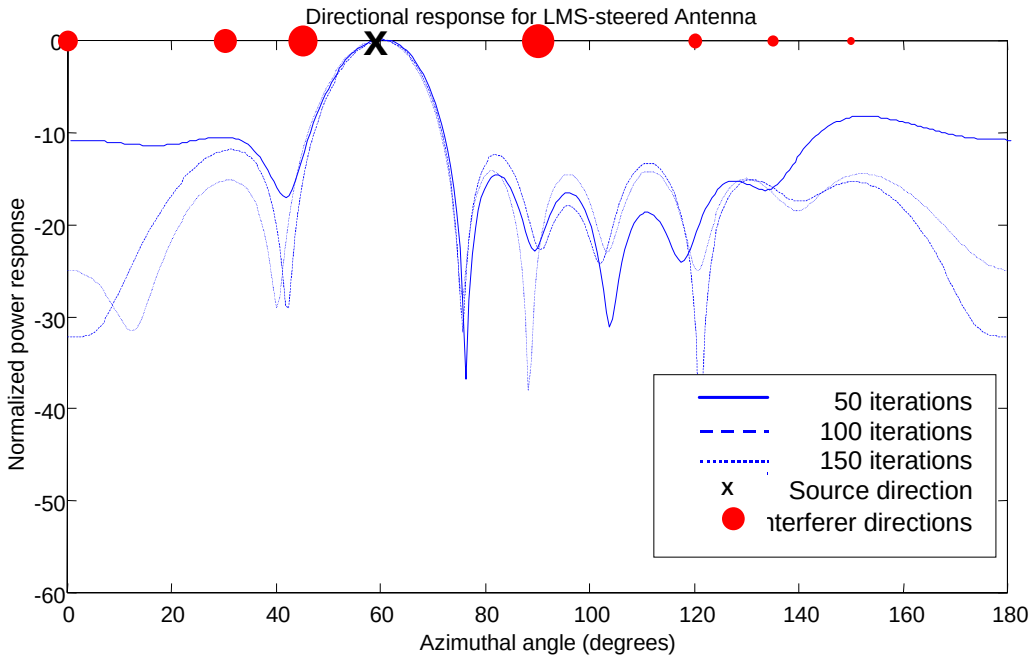

Figure 4. Power Coverage for LMS-Steered Antenna Array

1.2 Recursive Least Squares (RLS)

Based on the method of least squares, the recursive implementation of the algorithm starts from known initial conditions and uses the information contained in new data samples to update the old estimates. An exponential weighting factor or forgetting factor is defined as follows:

$$ß\,(n,i) = \lambda^{n-i}, \qquad i = 1, 2, \ldots, n$$

where λ is a positive constant close to but less than 1. Since this factor is used in the cost function, the algorithm is also called exponentially weighted least squares. When λ equals 1, there is no weighting on the square errors, so we have the ordinary method of least squares. The inverse of 1 – λ is a measure of the memory of the algorithm. The special case of λ = 1 corresponds to infinite memory and is the case considered in this implementation. **Figure 5** shows a flowchart for the RLS algorithm.

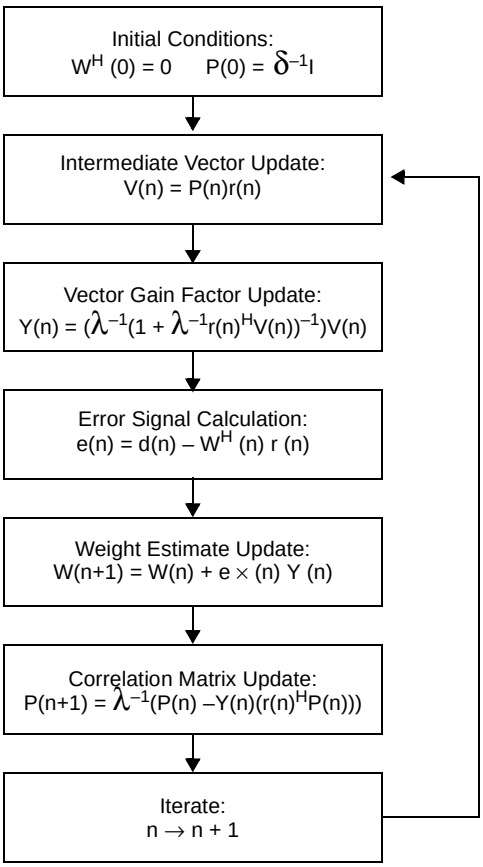

Figure 5. RLS Algorithm

The notation in **Figure 5** is as follows:

- *P(n)* is the inverse of the (exponentially weighted) correlation matrix for the input vectors *{r (1) . . . r(n)}*.
- *V(n)* is an N × 1 intermediate complex vector.
- *Y(n)* is an N × 1 complex vector gain factor.
- *W(n)* is the N × 1 complex weight vector.
- *d(n)* is the real target signal (in some applications, *d(n)* may be complex).
- *e(n)* is the complex error signal.

The initial conditions shown in **Figure 5** are $W(0) = 0$ and $P(0) = \delta^{-1}I$, whereδ is a positive number much less than $N–1(E[|r^H r]$ ($\delta = 0.01$ in our case). This initialization procedure is normally referred to as a soft-constrained initialization [1]. The RLS algorithm is implemented as specified in [1], according to the Matlab code, as follows:

```
% Definitions:
        % PP - Inverse correlation matrix (complex n_ants X n_ants)
        % R - Input data vector(complex, n_ants X 1)
        % V - Intermediate vector (complex, n_ants X 1)
        % W - Weight vector output matrix (complex, n_ants X n_inputs)
        % Y - Intermediate vector (complex, n_ants X 1)
```

```
% d_fix - target output (real)
% e - error (complex)
% idelta - initial stiffness parameter (supplied by user)
% idelta = 20 in current simulation
% ilambda - forgetting factor (ilambda=1 means no forgetting) (user suppl.)
% ilambda = 1 in current simulation

% scale = 2^k, where k is chosen large enough to keep parameters in range
% scale = 64 in current implementation

% Initial conditions
PP = idelta*eye(n_ants,n_ants);
W(:,1) = 0;

% Rescaling for proper range in fixed point
PP = PP/scale;

% Update step
V = PP*R;
Y = ilambda / (1/scale + ilambda*R'*V) * V;
e = d_fix - W(:,iter)'*R;
W(:,iter+1) = W(:,iter) + e'*Y;
PP = ilambda * (PP - Y*(R'*PP));
iter = iter + 1;
```

The parameter scale ensures that all quantities remain fractional so that overflow does not occur during the fixed-point computations. **Figure 6** shows the directional power response for an 8-element linear antenna array for a system with specifications given in [2] that was adapted using the preceding Matlab algorithm. As **Figure 6** shows, the response is successfully steered towards the desired signal source and away from interfering sources. The antenna coverage pattern is fairly stable after 50 iterations. Power values are normalized by the maximum directional power response. Circular marker size indicates the relative strength of interfering sources.

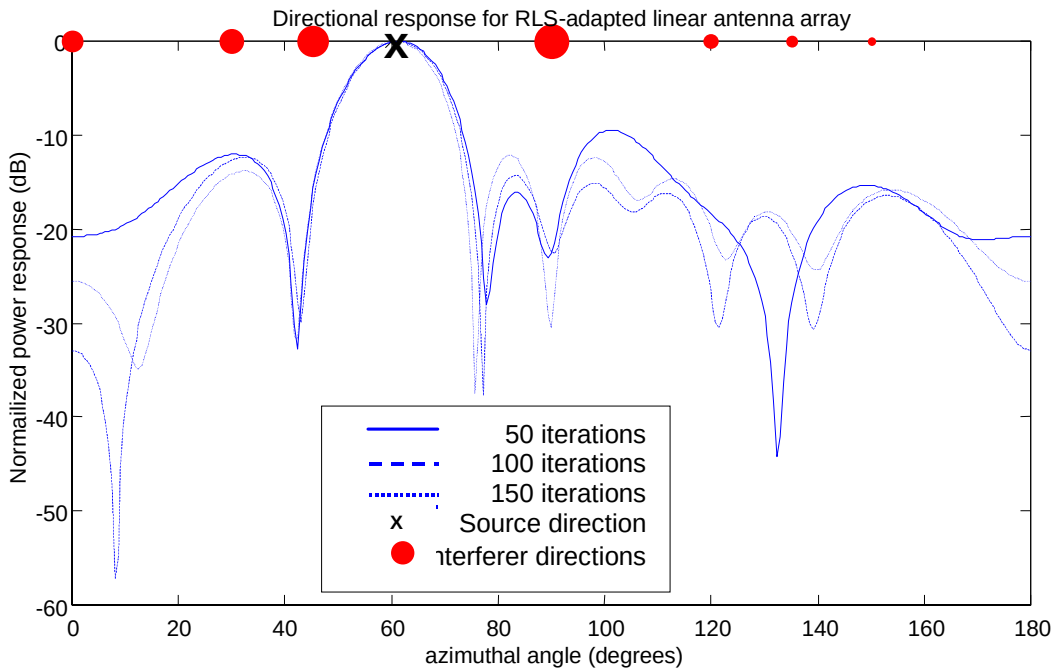

Figure 6. Power Coverage for RLS-Steered Antenna Array

2 AA System

Figure 7 shows a top-level block diagram of an MMSE-based AA system. Options for partitioning the system between hardware and software are indicated in the figure. Such partitioning is useful because putting as many functions as possible in the DSP allows you to take advantage of the flexibility inherent in software. However, some blocks are placed in hardware because they exceed the I/O or MIPS capacity of the DSP. The beam combiner block is a candidate for either hardware or software. If the beam combiner is placed in software, the hardware/software interface is more straightforward because no feedback is required from the beam estimator to the hardware. However, a software beam combiner requires much greater input bandwidth per user, which limits the number of users the system can accommodate. Also, a hardware beam combiner adds considerable complexity to the hardware, since multipliers and control capabilities must be added for the beam combining. The hardware buffer requirements are also greater, since data symbols must be stored until the channel estimates required for beam combining are returned from the software.

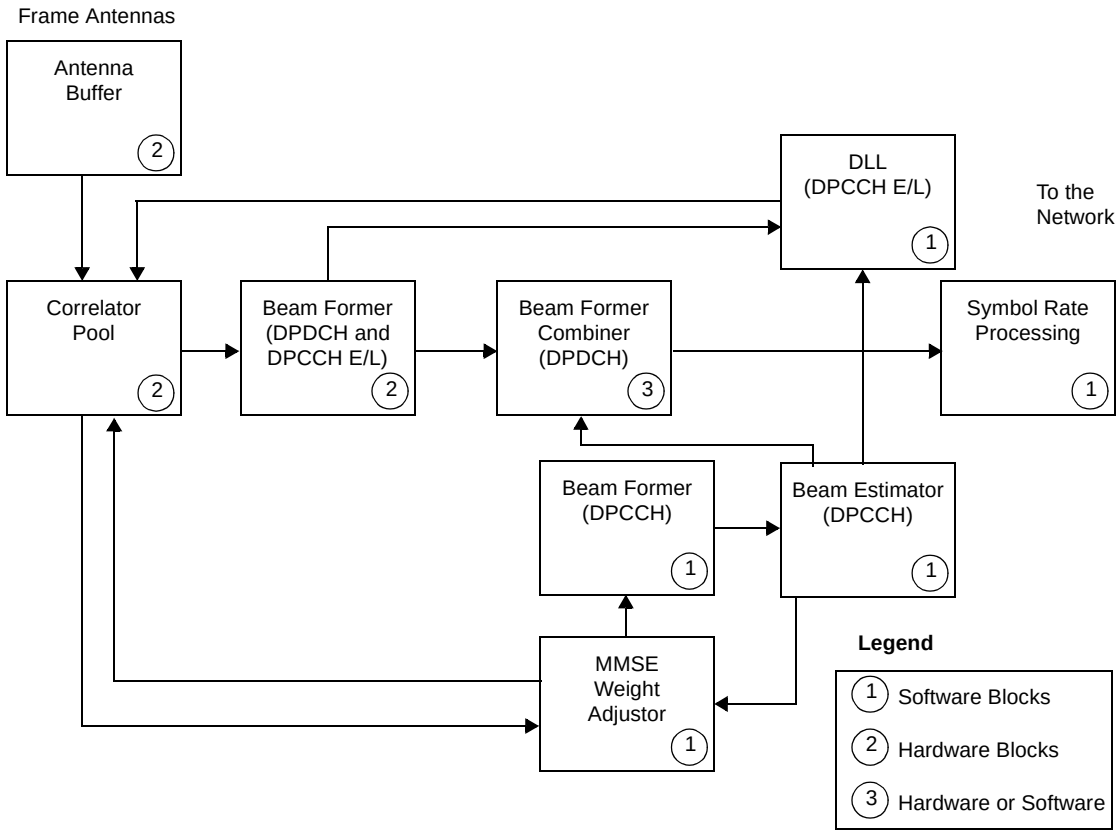

Figure 7. MMSE-based AA System with Partitioning

The system blocks shown in **Figure 7** are as follows:

- *Antenna buffer*. This buffer consists of two banks, one to read in antenna samples and the other to write the samples out to the correlator. This two-bank structure is called a *ping pong* buffer. The antenna buffer structure consists of one ping-pong buffer for each antenna element in the system.
- *Correlator pool*. Correlators are shared among all users, beams, and antenna elements. For each specific user/beam/antenna element, four correlations are required: one for DPDCH and three for DPCCH (early, on-time, and late). Beams are sector-specific. That is, if two sectors lock onto the same multipath, each sector forms its own beam.

- *Beam former*. Beams are formed via the weighted combination of despread antenna data. The MMSE weight adjustment algorithm (RLS or LMS) determines the weights. Beams must be formed for DPDCH (for data demodulation), DPCCH early/late (for time tracking in the DLL), and for on-time DPCCH (for channel estimation)
- *DLL (Early/Late DPCCH)*. This function is the same as in conventional (non-AA) systems. The DLL estimates the timing drift of the beam, with the aid of data obtained from the early/late DPCCH beams. The DLL requires AFC correction and channel estimation, which is supplied by the beam estimation block.
- *Beam combiner (DPDCH)*. For the DPDCH, this function is analogous to path combining in non-AA systems. Each user's data signal is computed via a weighted combination of the user's AFC-corrected beams. The weights come from the output of the MMSE weight adjuster block and are updated periodically (in our computations we assume an update rate of 1 weight update per slot).
- *Beam estimation (on-time DPCCH)*. This is analogous to channel estimation in non-AA systems. Automatic frequency control (AFC) and channel estimation functions are performed on each on-time DPCCH beam. AFC estimates and corrects the overall frequency offsets of the beams on a per-user basis, and the channel estimation estimates the residual beam variation (subsequent to AFC correction) in the baseband (IQ) plane.
- *MMSE weight adjuster*. Uses the overall beam estimation (AFC plus channel estimation) plus the despread antenna data for the DPCCH to update the weights for antenna combination.

3 System Throughput Estimates

This section presents estimates of throughputs between system blocks shown in **Figure 7**. The throughputs of greatest concern are those from hardware blocks to software blocks, since the hardware-to-DSP I/O throughput is the bottleneck that limits the number of users the system can handle. Hardware block-to-hardware block throughputs are less of an issue because they can be decreased by pipelining. Throughputs from software block-to-software block are not computed because these do not correspond to physical data throughput (since all software blocks perform on the same device). **Table 1** lists the parameters used in the throughput analysis.

Table 1. Parameters Used in System Analysis

Parameter	Description
N_ants	Number of antenna elements per smart antenna per sector. Typically, N_ants ranges from 4 to 8. In the current analysis, N_ants has a value of 6.
N_sectors	Number of sectors. Typically, there are 3 or 6 sectors. In the current analysis N_sectors has a value of 3.
N_beams	Maximum number of beams per user.
N_users	Number of users supported by the system (including all sectors). In the current analysis, N_users has a value of 128.
N_multicodes	Maximum number of DPDCH code channels per physical channel (currently 6 according to 3GPP)
Chip_rate	System chip rate (3.84 Mcps for 3GPP).
DPCCH_sym_rate	Symbol rate for control symbols (15 Ksps for 3GPP)
mean_max_DPDCH_sym_rate	Mean of all users' maximum data symbol rates
Slot_rate	Equal to 1.5 ksps for 3GPP
B_1	DPCCH and DPDCH correlator output byte width (B_1 = 1 is assumed)
B_2	DPCCH and DPDCH beam former output byte width (B_2 = 1 is assumed)

Table 1. Parameters Used in System Analysis (Continued)

Parameter	Description
B_3	Beam estimator output byte width (B_3 = 1 is assumed)
B_4	DPCCH and DPDCH Beam combiner output byte width (B_4 = 1 is assumed)
B_5	DLL output byte width (B_5 = 4 is assumed, as in current CRRS design)
B_6	Weight adjuster output byte width (B_6 = 1 is assumed)

The mean_max_DPDCH_sym_rate parameter requires additional explanation. The system is configured so that a minimum spreading factor is determined for each user. For instance, users designated as voice users are allowed a minimum spreading factor of 64, corresponding to a maximum DPDCH symbol rate of Chip_rate/64. These voice users have the option of using a larger spreading factor (for example, lower data rate), but not a smaller one. In addition, different data users could be assigned different minimum spreading factors from the set $\{2^k, k=2,3,4,5\}$, corresponding to a maximum DPDCH symbol rate of Chip_rate/2^k. The mean_max_DPDCH_sym_rate parameter is defined as the average over all users of these maximum symbol rates. The maximum system data symbol throughput is given by N_users × mean_max_DPDCH_sym_rate.

Table 2 shows the throughputs (in MB/sec) between the individual blocks of the smart antenna system shown in **Figure 7**. The system scenario is for 128 voice users at minimum spreading factor 64, corresponding to mean_max_DPDCH_sym_rate = Chip_rate/64 and a total maximum DPDCH symbol throughput of 2 × Chip_rate.

Table 2. System Throughput Estimates

System Parameters	
Number of antennas/beam	6
Number of beams/user	6
Number of users	128
Chip Rate	3.84E+06
DPCCH symbol rate	15000
Mean Max DPDCH symbol rate	60000
Slot Rate	1500
Number of real DPDCH for multicode	6

Output Byte Widths	
Correlator output	1
Beam former output	1
Beam estimator output	1
Beam combiner output	1
DLL output	4
Weight adjustor output	1

Throughput Estimates

Output From	Input To	Maximum Throughput (MB/Second)			
		DPCCH On-Time	DPCCH Early/Late	DPDCH	Control
Correlator pool	Beam former		276.48	276.48	
Correlator pool	Weight adjustor	138.24			
Beam former	DLL		46.08		
Beam former	Beam combiner			46.08	
Beam estimator	Beam combiner				2.304
Beam combiner	Symbol rate process			7.68	
DLL	Antenna buffer and correlator bank				4.608
Weight adjustor	Beam former				13.824

4 DSP Memory

Memory can be conveniently divided into DPDCH memory and DPCCH memory, which is described in this section. These buffers all reside within the DSP.

- DPDCH memory requirements:
  - — *DPDCH interface buffer*. The buffer to which the hardware writes. This buffer stores one slot worth of data symbols.
  - — *DPDCH intermediate buffer*. Holds the data symbols prior to spread factor combining and frequency correction.
- DPCCH memory requirements:
  - — *DPCCH interface buffer*. The buffer to which the hardware writes. This buffer stores one slot worth of control symbols.
  - — *DPCCH intermediate buffer*. The working buffer for the DSP. The types of buffers required are on-time DPCCH before beam forming, on-time DPCCH after beam forming, and early and late DPCCH.
- Control buffer requirements. Provides control information to the hardware, along with antenna coefficients and channel estimates:
  - — *Channel estimate coefficient*. Buffers to store channel estimate coefficients.
  - — *Antenna coefficients*. Buffers to store antenna coefficients.
  - — *Hardware control fields*. Buffers to store hardware control information.

5 MCPS Estimates for Software Blocks

Estimations of the computational requirements for the proposed AA system must include both AA and chip rate processing. The computational load is measured in million cycles per second (MCPS). The MCPS requirements are computed only for software blocks, as indicated in **Figure 7**. AA is an enhancement to the fundamental chip rate system. Therefore, to evaluate resource requirements for an AA-enhanced system, we require a reference chip rate receiver architecture, to which we add AA features. **Figure 8** shows the reference chip-rate architecture, which we refer to as the Chip Rate Reference System (CRRS). Of course, if the basic underlying chip rate architecture is changed, the resource requirements may also change. Although CRRS is the reference system, some changes in CRRS may be required when it is incorporated into an AA system. As indicated in **Figure 7**, the beam combiner function (corresponding to the path combiner in CRRS) can be moved from software to hardware to meet resource requirements.

The AA system is differentiated from a non-AA system by the addition of the beam former and MMSE weight adjuster blocks (see **Figure 7**). Also, the AA system has an increased correlator load because separate correlators are required for each antenna in the array for each beam. However, for the systems blocks assigned to the DSP according to **Figure 7**, the MCPS load is equal for AA and non-AA systems. Therefore, estimates of MCPS requirements need only compute the MCPS requirements for the two AA-specific system blocks. All other MCPS estimates can be carried over from previous estimates computed for the non-AA chip-rate system.

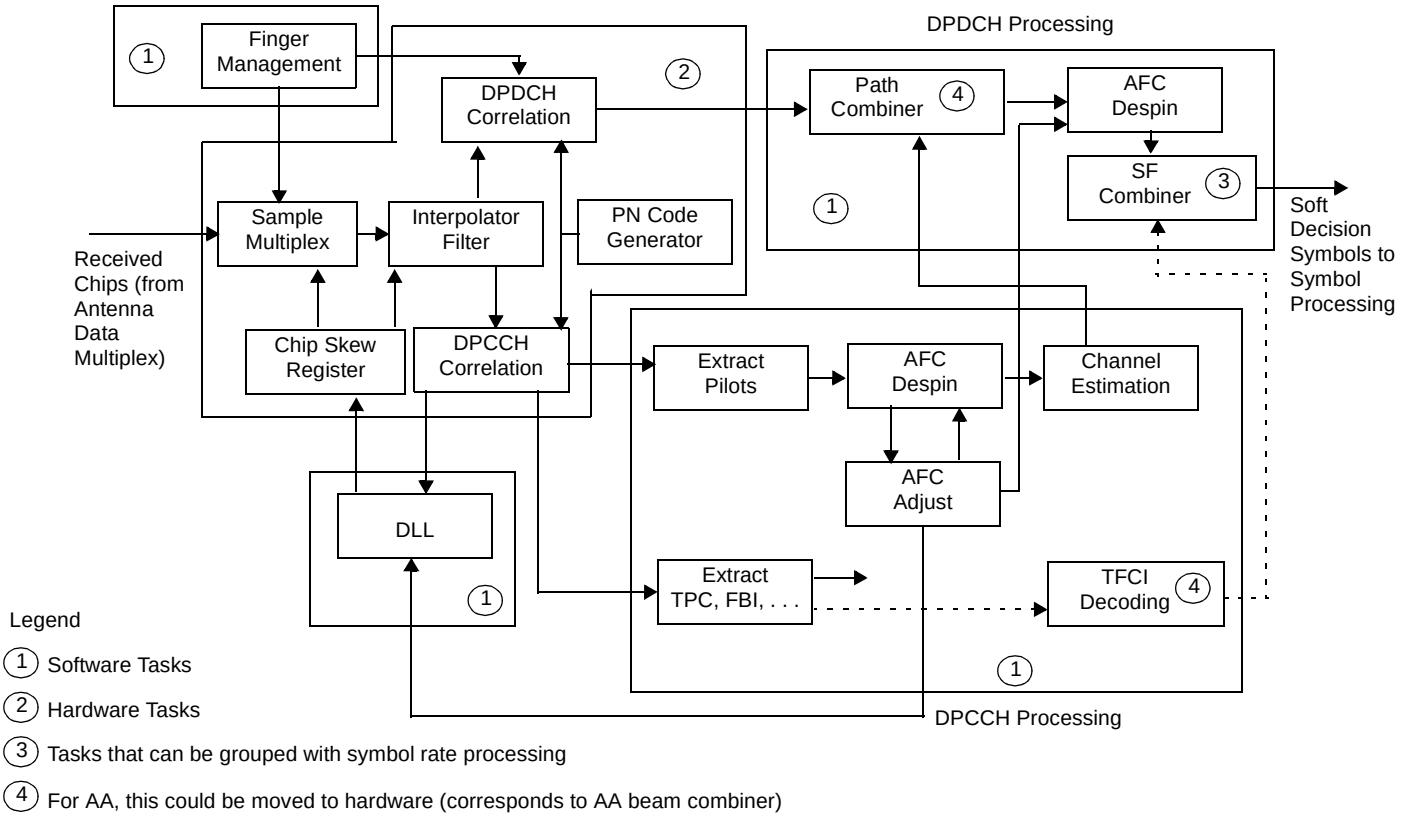

Figure 8. Chip Rate Reference System (CRRS) Architecture

Following are MCPS formulas for the system blocks specific to the AA. **Table 3** lists the number of MCPS for all system blocks (including those not specific to the AA). The following estimates are based on four MAC operations per complex multiply:

- *MCPS cost for LMS algorithm*. The single beam estimate of 0.44 MCPS for 8 antennas is obtained from reference [5] and is based on measurements taken from an actual implementation of LMS on an MSC8101 device.
- *MCPS cost for RLS algorithm*. The single beam estimate of 5.93 MCPS for 8 antennas is obtained from reference [5] and is based on measurements taken from an actual implementation of RLS on an MSC8101 device.

In **Table 3**, the MCPS is for 128 voice users with 6 fingers (beams) per user. The columns labeled *CRRS+AA(1)* refer to AA systems with the beam combiner in hardware.

Table 3. Resource Requirement Computations for AA System

MCPS Comparisons		
	6 Antenna Elements	
	CRRS	CRRS + AA(1)
Buffer input management	460.800	691.20
DPCCH beam combining	56.256	43.96
DPDCH frequency correction	0	0
SF combining	184.520	21.12
Extract pilots	150.530	112.89

Table 3. Resource Requirement Computations for AA System (Continued)

MCPS Comparisons cont.		
	6 Antenna Elements	
	CRRS	CRRS + AA(1)
DPCCH frequency correction	176.640	132.480
Channel estimation	43.000	32.200
Frequency estimation	54.530	43.800
TFCI decoding	27.648	27.648
DLL + energy computation	119.350	89.840
Finger management + control	459.200	334.400
Target generation	0	81.790
Beam former (DPCCH on-time)	0	172.800
MMSE weight adjuster (LMS)	0	350.000
Total	1732.474	2134.338
I/O Comparisons (MByps)		
DPDCH	92.160	15.360
DPCCH early/late	46.080	46.080
DPCCH on-time	23.040	138.240
Control feedback	2.304	20.736
Total	163.584	220.416
Memory Comparisons (KB)		
	CRRS	CRRS + AA(1)
DPDCH	148.224	96.328
DPCCH	72.960	285.696
Control feedback	6.144	13.824
Total	227.328	398.848

6 MSC8102 Implementation

The current CRRS chip rate architecture can be implemented with a single MSC8101 device [6]. However, since a practical AA system implementation has more stringent resource requirements, we migrated the implementation to a multi-core processor, that is, a CRRS-based AA system on the MSC8102, which is a 4-core device. Our study shows that for our benchmark system (128 voice users with 6 beams/user and 6 antennas/beam), two MSC8102 devices can perform all required AA and chip rate processing. For a complete system with chip and symbol rate processing, a third MSC8102 is required. The MSC8102 implementation incurs minimal software changes in the chip rate portion of the system because both the MSC8101 and the MSC8102 processors use the same instruction set (although changes are necessary because of the transition from a single core to a multi-core architecture). Additions to the software to incorporate AA functions include the following:

- a module for performing MMSE (to recalculate antenna coefficients dynamically)
- an on-time beam-forming algorithm, which is identical to the beam combiner, except for changes in the input and weight coefficients

The transition from the current CRRS architecture to the suggested AA system also requires some changes in the hardware, as follows:

- An increased number of correlators to perform correlations for each antenna element
- Complex multipliers to perform beam forming and subsequent beam combining.

Figure 9 depicts a complete AA system (including all AA, chip rate, and symbol rate processing) for 128 voice users, using three MSC8102 devices. The hardware/software partitioning is as shown in **Figure 7**, with the beam combiner in hardware. The system can accommodate the indicated number of voice users performing both symbol rate and chip rate processing. The chip rate+AA(1) processing load of 2135 MCPS is based on scaled results from a 32-voice users chip rate system implemented on an MSC8101ADS board [6] and on measurements taken from an actual implementation of LMS on an MSC8101 device[5]. The symbol rate estimate of 611 MCPS is based on scaled results from a 32-voice users symbol rate system implemented on an MSC8101ADS board. Three buffers reside in hardware (antenna buffer, correlator buffer, and control buffer). The buffer size estimates derive from our analysis of the CRRS system [6]. The buffers described in **Section 4**, *DSP Memory* are all contained within the MSC8102 devices, except for the DPDCH symbol rate buffer, which is shown in **Figure 9**. The resource requirements for the system are based on the number of voice users. Each MSC8102 device is capable of 1.2 Giga instructions per second at 300 MHz, and it has a bus bandwidth I/O of 560 MB per second on the 60x-compatible system bus/280 MB per second on the direct slave interface (DSI) bus (or 280 MB per second on the system bus/560 Mbyps on the DSI bus), assuming 70 MHz, 224 × 4 KB of L1 memory, and 476 KB of L2 shared memory.

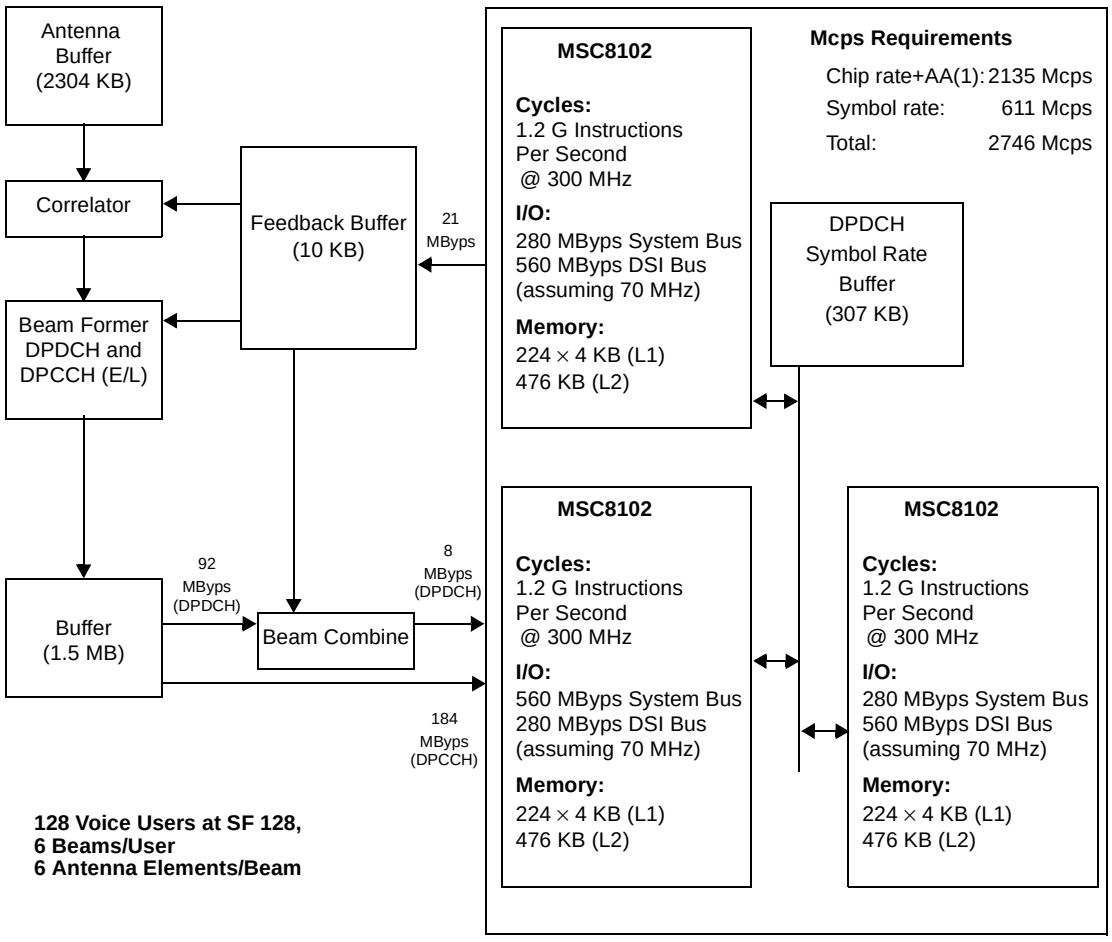

Figure 9. Complete Smart Antenna System

The smart antenna system processing load of 2718 Mcps is well within the processing capabilities of the three MSC8102 devices. The system data flow is described in **Figure 10**. The total system bus usage (221 Mbyps) includes all data transfers between the MSC8102 devices and the ASIC. The ASIC performs the correlations, beam forming, and beam combining functions. The total DSI bus usage of 50 Mbyps includes all data transfers between the three MSC8102 devices and the external memory (also referred to as the symbol rate buffer in **Figure 9**). If the spread factor for certain users decreases to accommodate higher data rates, the system resource requirements increases. The bottleneck in how many high data rate users this system can handle is the symbol rate MCPS requirement. A reasonable trade-off between high data rate users and voice users is that one data user at 384 Kbps (spread factor 4) consumes the same amount of symbol rate processing as 32 voice users. Four data users at 64 Kbps (spread factor 16) require the same amount of symbol rate processing as 32 voice users.

6.1 CRRS Software

The current CRRS is designed on an FPGA, along with an MSC8101 device. The transition from a single core to a multi-core device necessitates some changes to the software. Otherwise, the hardware to DSP interface is designed so that the core software can be reused even if the hardware design changes. The only software module that must be rewritten is the buffer input manager interface module that orders the partial symbols from the hardware. The interface for each stream of data generated by the multiple antenna elements can be managed by replicating the same interface currently in use in CRRS. For a system with beam combining in hardware, there is little impact on the current CRRS software structure since path combining is the first module the DPDCH data stream encounters. The relocation of beam combining to hardware requires a change only in the interface routines; the interface must be changed so that the beam former coefficients generated by the AA algorithms are provided to the hardware.

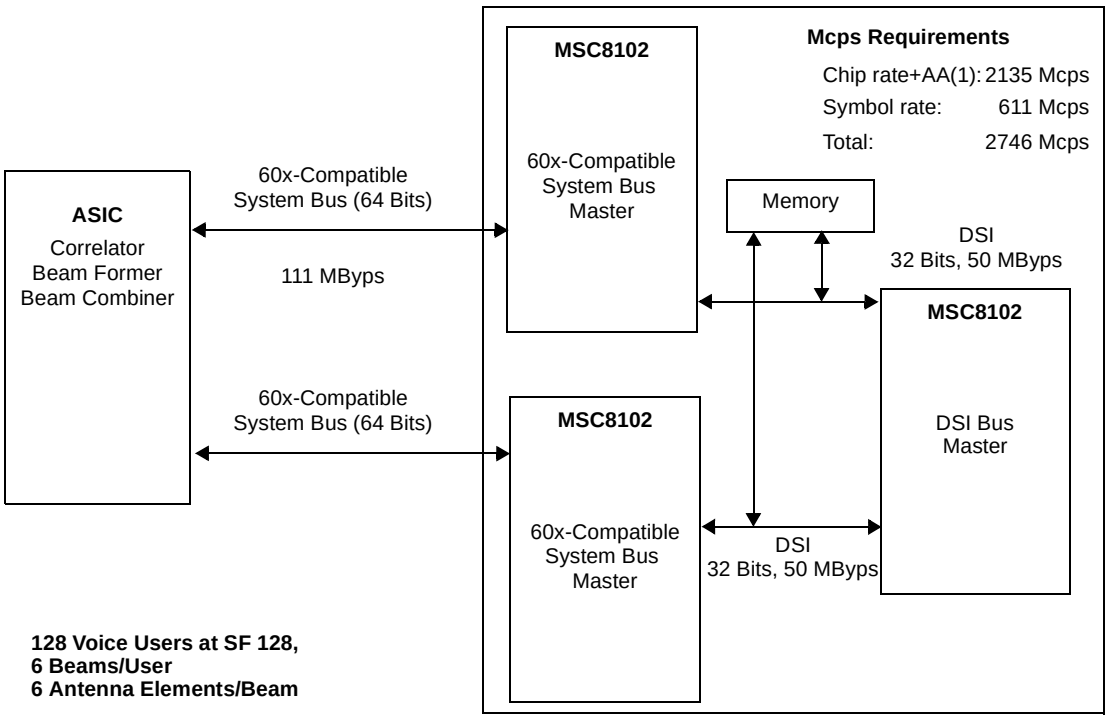

Figure 10. Connectivity Diagram For the Complete Smart Antenna System

6.2 CRRS Hardware

The hardware must be scaled up from the current CRRS structure for an AA system. The current system is for single-element antennas, but the AA system employs multiple-element antennas. Therefore, the AA system requires up to *N*-fold duplication of the CRRS system (where *N* is the number of elements per antenna) and related changes in control. More correlators are required to accommodate data from multiple antenna elements, but each correlator is identical to those in the current structure. The only added feature in the system is the beam former, which is a complex MAC unit. If the DPDCH beam combiner is added to the hardware, an additional complex MAC unit must be added to the hardware.The channel estimates must also be sent back from software to hardware. In summary, migration from the current CRRS system to an AA system requires some enlargements and additions to the hardware, with attendant changes to the control software.

7 References

[1] Simon Haykin, *Adaptive Filter Theory*, 3rd Edition, Chapter 13, Prentice-Hall, NJ, 1996.

[2] Koga, Hisao and Taromaru, Makoto, “A Simple and Fast Converging Algorithm for MMSE Adaptive Array Antenna”, *IEICE Transactions Commun.*, Vol. E83-B, No. 8, August 2000, pp. 1671–1677.

[3] S. Tanaka, M. Sawahashi, F. Adachi, “Pilot Symbol-Assisted Decision-Directed Coherent Adaptive Array Diversity for DS-CDMA Mobile Radio Reverse Link Source,” *IECE Transactions Fund.*, Vol E80-A, No. 12, December 1997, pp. 2445–2454.

[4] J. C. Liberti and T. S. Rappaport, *Smart Antennas for Wireless Communications: IS-95 and Third Generation CDMA Applications.* (Prentice-Hall, NJ, 1999.

[5] *Resource Requirements and Partitioning for MMSE-based Adaptive Antenna Uplink WCDMA System Using the Motorola MSC8102 Processor*, Motorola application note release is pending.

[6] *MSC8102 and MSC8102-Based 3G Channel Card*, Motorola application note release is pending.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale™
semiconductor