HSDPA-Adaptive Modulation and Coding: Simulation and Implementation

SASIKANTH MUNAGALA[1], RAJU MUCHANTHULA[1], YANNICK LE MOULLEC[1], PETER KOCH[1], LARS KRISTENSEN[2]
[1]KOM Department
Aalborg University
Niels Jernes Vej 12
DK-9220 Aalborg
[2]Rohde&Schwarz Technology Center A/S
Gasværkvej
DK-9000 Aalborg
DENMARK
http://kom.aau.dk/~ylm/

Abstract: - This paper deals with simulation and implementation issues of HSDPA-Adaptive Modulation and Coding (physical layer) of the 3GPP standard. Firstly, the system is simulated in Matlab. Then it is partitioned to hardware and software based on Matlab profiling. Furthermore, an algorithm has been developed to explore the design space based on optimization criteria for balancing area and time constraints. The algorithm helps during the design space exploration of different algorithms using Handel-C and DK-Suite tool for exploring the parallelism. Finally, back-end tools are used for applying low level optimizations for time and area constraints onto Xilinx Virtex II and Altera Stratix FPGA platforms. The results show that design trajectory at different abstraction levels help in progressing from system specification to an efficient final implementation within a reasonable design time.

Key-Words: - HSDPA, AMC, simulation, design-space exploration, parallelism, FPGA implementation

1 Introduction

One of the main drivers for the development of next generation wireless communication systems is increasing demand for high rate data services. First and second generation communication systems were mainly focused on voice services, while the third generation system is bringing a revolution in communications by providing high speed data transfer services. Increase in number of mobile users accessing internet and multimedia based applications demands more bandwidth in wireless networks. Today's Universal Mobile Telecommunication System (UMTS) offers data services of the acceptable quality, typically reaching data rates of 384Kbps. It can be foreseen that in the next couple of years, multimedia traffic will overrule voice traffic. By 2008, it is expected that multimedia communication will account for 1/4th of the mobile traffic [1]. Future applications will require even higher data rates, with the role of IP-based data services expanding. The ability to combine various services will also increase in importance. Applications such as intranet access and large file download result in an ever-increasing amount of data traffic in the mobile radio network. Mobile radio network operators are faced with increasing the capacity of their networks so that they can meet the demands of the increased data volume and provide quality service to subscribers of new services. For these reasons, High Speed Downlink Packet Access (HSDPA) [2-7] has been developed to enhance UMTS technology as specified by the 3rd Generation Partnership Project (3GPP) Release 5. The main focus of this work is the simulation, design space exploration and implementation of the physical layer of HSDPA-AMC onto FPGAs.

1.1 HSDPA-AMC

HSDPA utilizes other link adaptation techniques to substitute power control and variable spreading factor. HSDPA adapts modulation, the coding rate and the number of channelization codes to the instantaneous radio conditions. The combination of the first two mechanisms is called AMC. This ensures that each subscriber is provided optimum service. For example, a subscriber close to the base station needs only minimal protection against transmission errors because the transmission quality is so good. This subscriber can also receive a higher data rate than a subscriber at the edge of the cell.

Each mobile phone regularly sends messages to the base station about the channel quality.

The base station uses this information to decide how many resources can be made available to a subscriber and which modulation mode can be selected. This process improves the protection against transmission errors and optimizes the use of resources on the air interface. Besides QPSK, HSDPA incorporates the 16-QAM modulation to increase the peak data rates for users served under favorable radio conditions. Support for QPSK is mandatory for the mobile, though the support for 16-QAM is optional for the network and the UE.

The inclusion of the higher order modulation introduces some complexity challenges for the receiver terminal, which leads to estimate the relative amplitude of the received symbols, where as it only requires the detection of the signal phase in the QPSK case. The channel coding is based on the Release'99 1/3 turbo encoder and is always 1/3 rate (for every bit that goes into the coder, three bits come out). The effective code rate varies, however, depending on the parameters applied during the two-stage HARQ rate matching process.

1.2 Design flow overview

The design flow, depicted in Fig.1 starts with a specification where the algorithms are developed in Matlab. After performing Matlab-based profiling and manual analysis of algorithm complexity, the most computationally expensive part of the AMC system is allocated onto the hardware (FPGA) and the rest of the system is allocated to software (Matlab). The partitioning examines design alternatives that meet the system constraints. It consists in mapping the functions of the algorithm onto interconnected hardware and software processors. The communication between hardware and software is achieved by a HW/SW interface. Here two different approaches are chosen to implement the hardware, i.e., DK methodology [8] and a Simulink approach. In the DK Methodology the hardware part is transformed into C code from Matlab. The C code is tested and refined until it has the same functionality as in Matlab specification. The C code, which is sequential in nature, is rewritten in Handel-C, introducing parallelism to the code. The Handel-C code is fed into DK-Suite. The Logic Estimator gives the estimated number of gates that would be used in the FPGA when the Handel-C code is synthesized and placed/routed. The goal of the Handel-C code is to maximize the use of parallelism while keeping the number of gates required below the number of gates available on the FPGA and reducing the time taken. In Simulink approach, two different methods, i.e., Xilinx System Generator and Altera DSP Builder are used for respective architectures. From these three different methods, back-end synthesis to gates and netlist generation are performed. Finally, the hardware and software integrated and the implementation is tested for its observance to the system specification.

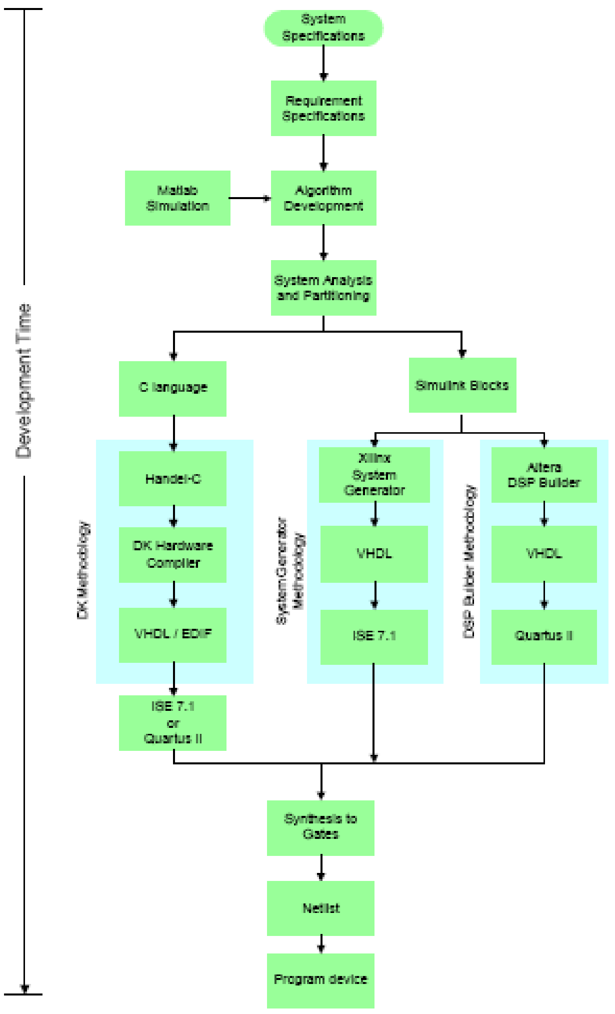


Fig.1 The overall design trajectory used to simulate and implement the AMC scheme of the HSDPA standard.

2. Simulations

2.1. Simulation setup

The main characteristics related to the simulation of the HSDPA-AMC (at the physical layer level) are described in Fig.1 are reported in Table 1. The choices of the values for each parameter are made according to the 3GPP specification and some of them are discussed below.

a) Coding Rate r: the turbo coder is composed of two RSCs and a systematic part, thus involving three outputs (the systematic output of each RSC is considered as a "sink"). Puncturing the outputs in order to increase the rate up to 1/2 is also possible.
b) Constraint Length K: in order to simplify the investigations, only K = 4 is used as constraint length for simulations.
c) Block length k: three different block sizes are simulated. The block lengths allows to investigate the effects of the AWGN channel used are 100, 320 and 1024. The interleaver size is equal to the block length.
d) Decoder: both Log-MAP and SOVA decoders have been investigated,
e) Modulation: QPSK and 16-QAM have been used,
f) Number of iterations = 1, 3 and 5: Increasing the number of iterations increases the decoding delay. Therefore it was decided to set the upper limit for the iterative process to 5 iterations.

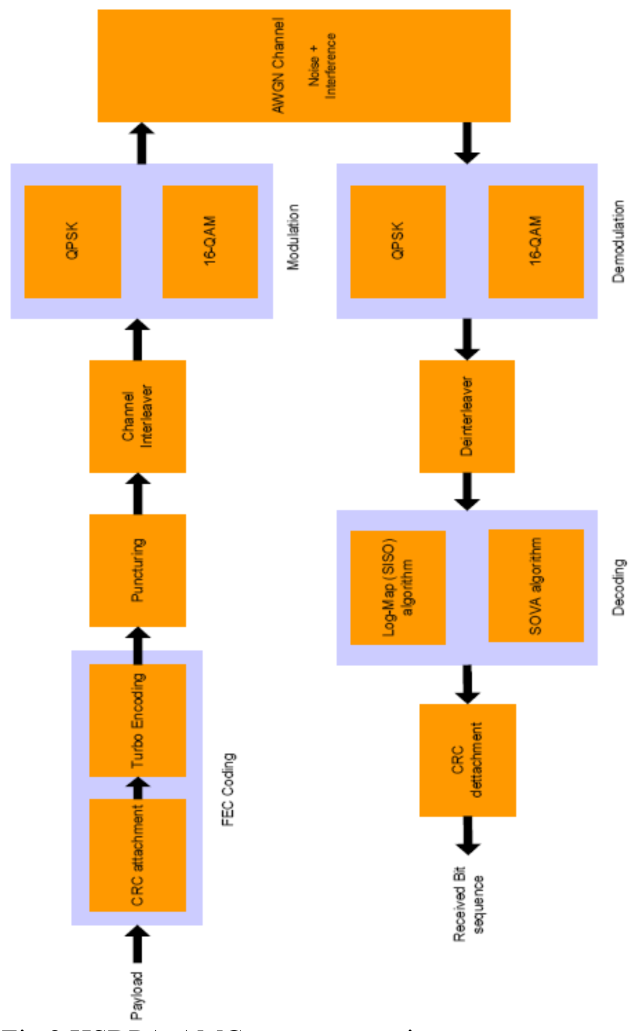


Fig.2 HSDPA-AMC system overview

Parameter	Type /Size
Channel	AWGN
CRC length	24
Coding Rate r	Rate 1/3 Turbo coder
Constraint length K	4 (1011, 1101)
Block length k	100, 320 and 1024
Turbo interleaving	Pseudo-random interleaving
Decoder	Log-MAP and SOVA
Modulation	QPSK and 16-QAM

Table 1. HSDPA-AMC simulation parameters

2.2 Simulation results

Several inter-leaver lengths are simulated and investigated. The effects of the number of decoding iterations are also analyzed. The SER and BER performances of the QPSK and 16-QAM modulation schemes are shown in Fig.3. The BER and SER are higher for 16-QAM comparing with QPSK on noisy channel conditions. These modulation schemes are performed utilizing non-spread data burst. There is an improvement of 7dB in Eb/N0 using 16-QAM for SER at the order of 10^{-3} and similarly an improvement of 1dB in Eb/N0 using 16-QAM for BER at the order of 10^{-3} compared to QPSK.

Thus, it is observed that as the noise level increases in the channel, QPSK gives a better modulation compared to 16-QAM. The performance of turbo code in terms of BER and FER improves with the increase in iterations, block length and the inter-leaver size. This implies that one would select the largest possible block length. The inter-leaver design plays significant role in performance of turbo code particularly with high SNRs. A random inter-leaver design in this case gives good performance. As inter-leaver size increases, it requires more iteration to give good performance. Thus, there is a trade-off between the block length and the inter-leaver size as more delay is produced with the increase in the inter-leaver size.

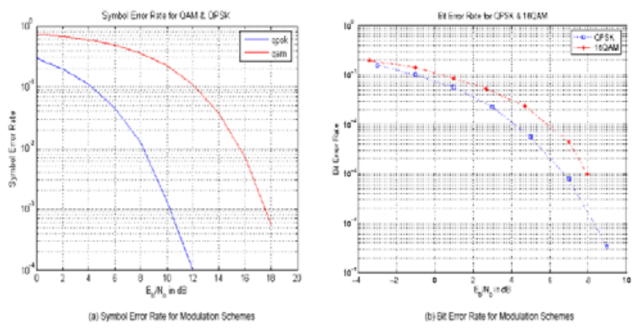


Fig.3 Simulation results for the SER and BER performances of the QPSK and 16-QAM modulation schemes.

The simulated performance of a turbo code with a code rate of 1/3 with block lengths of 100, 320 and 1024 bits using log-MAP and SOVA iterative decoders in terms of BER and FER are shown in figures 4 and 5 respectively. After 5 iterations, a BER of 10^{-5} is reached for a block length of 1024 for both decoders. It is observed from figure 5 that the BER for log-MAP is much lower than that of SOVA for different block lengths. As the iterations increase, the improvement decreases for both the decoders with respect to iterations. The experiment shows that log-map decoder gives a better performance with low BER (order of 10^{-7}) and FER (order of 10^{-4}) compared to SOVA decoder (order of BER is 10^{-4} and FER is 10^{-3}).

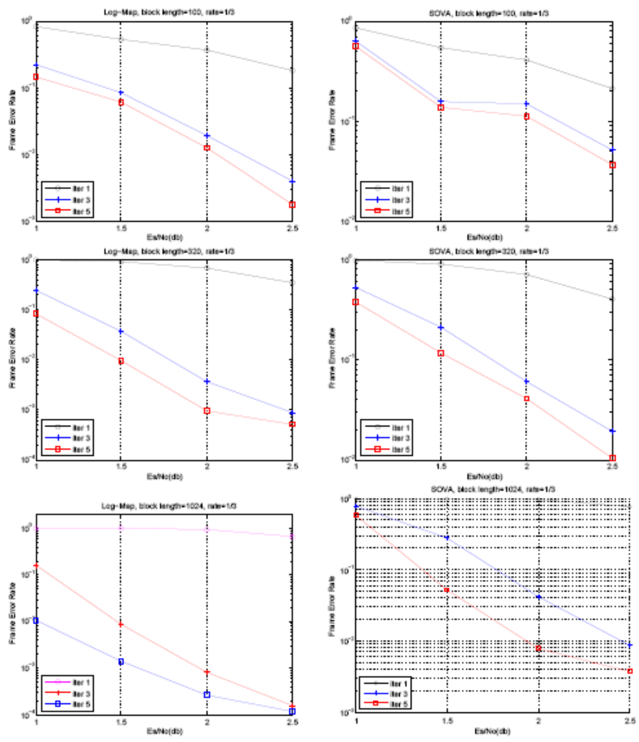


Fig.4 Comparison of Frame Error Rates using Log-Map and SOVA algorithms for Block Lengths of 100, 320 and 1024

3. Implementation

3.1. Design space exploration

The optimization criteria for mapping an algorithm to FPGA-based systems are as follows:
1. the design must not exceed the capacity constraints of the system,
2. the execution time should be minimized,
3. for a given level of performance, FPGA space usage should be minimized.

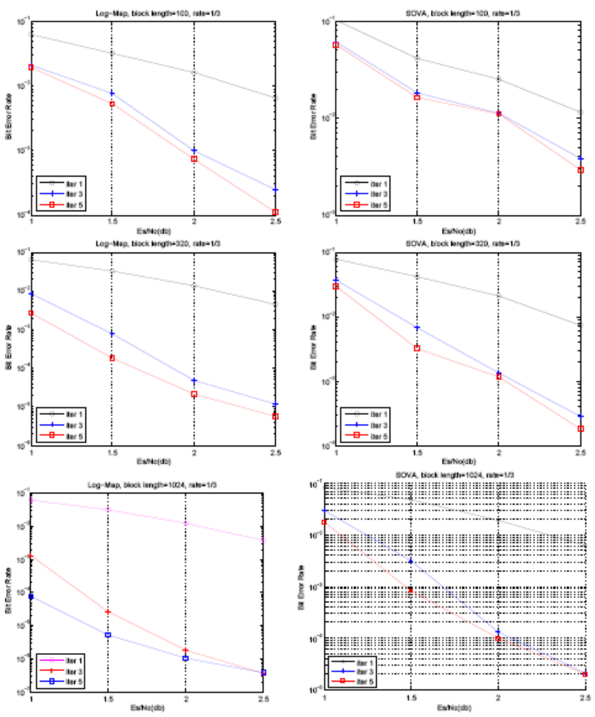


Fig.5 Comparison of Bit Error Rates using Log-Map and SOVA algorithms for Block Lengths of 100, 320, 1024

The motivation for the first two criteria should be apparent, but the third criterion is needed for several reasons such as, if two designs have equivalent performance, the smaller design is more desirable. A smaller design usually has less routing complexity, and as a result could achieve a faster target clock rate. The algorithm uses two metrics to guide the selection process. Firstly, the result of estimation provides space usage of the design, related to criterion 1 above. Another important metric used to guide the selection of a design, related to criteria 2 and 3, is balance (B). When a design is not balanced, this metric suggests whether more resources should be devoted for improving computation time and memory usage. Algorithm 1 starts with the code (e.g., Handel-C code). The estimation of the different constraints is done during synthesis. The Balance B, which depends on the estimates, code and performance, is calculated. Based on the value of B (whether it is greater or less than 1), the design is considered to be memory bounded or computation bounded. Starting the design from the saturation point, considering the design to be space bound, if the estimate of the space is less than the capacity of the target platform then jump to the next step of time. Else optimization of the design is done. During the next process, if the estimated time is less than the specification of the application, then the architecture of the design is programmed thus obtaining an optimal solution from the DSE. Else parallelism is applied to the design to meet the timing conditions and optimization is performed. This process continues until the system constraints are met. The output of the algorithm is the modified code.

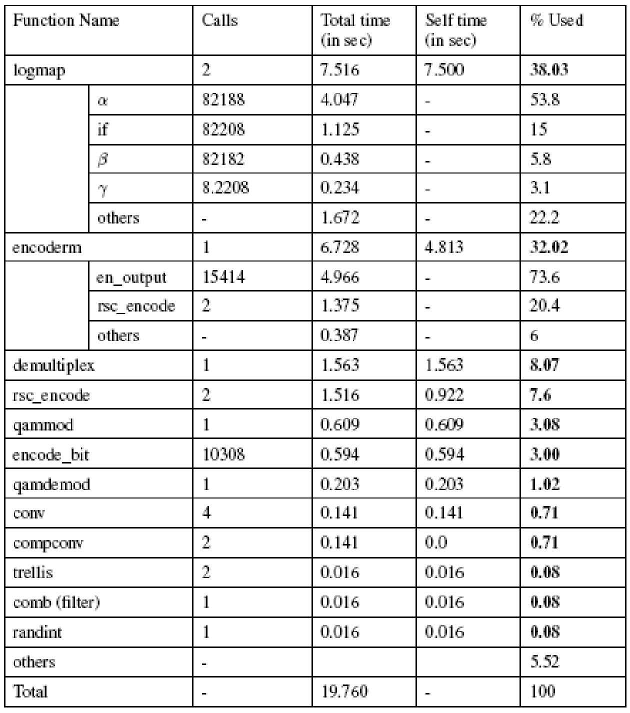
Balance = F/C

```
***Search Algorithm for Area and time Constraints:***
*Input : Code /* Handel-C Code */*
*Output : Adapted Code /*Handel-C Code*/*
*Begin*
*Estimate = synthesise (Code)*
*B=Balance (Code.Estimate.Performance)*
***Space** /* First deal with space-constrained design*/*
*if (Estimate.Space < Capacity) then*
*goto Time*
*else*
*Optimization (Globally and locally) /*Search for possible solution*/*
*end*
***Time** /* Deal with Time constrained design */*
*if (Estimate.Time < Applicable)*
*program FPGA*
*else if*
*search for parallelism*
*goto Space*
*else*
*Optimization at different abstraction levels (Globally and Locally)*
*goto Space*
*end*
*end*
```

Algorithm 1. Algorithm for Design Space Exploration where F refers to the total data bits the memory can provide per cycle, and C refers to the data consumption rate (total data bits the computation can consume during the computational delay). If F < C, the design is memory bounded, else computation bounded.

3.2 Matlab profiling

The timing properties of the algorithms of the system can be seen in table 2. The functions logmap refers to the log-MAP decoder, encoderm refers to the Turbo encoder, rsc_encode refers to RSC encoder, demultiplex, qammod, conv, compconv, trellis refers to demultiplexer, 16-QAM modulation, convolution, a function showing convolution between signal & filter, function performing the trellis state transition



Function Name		Calls	Total time (in sec)	Self time (in sec)	% Used
logmap		2	7.516	7.500	**38.03**
	α	82188	4.047	-	53.8
	if	82208	1.125	-	15
	β	82182	0.438	-	5.8
	γ	8.2208	0.234	-	3.1
	others	-	1.672	-	22.2
encoderm		1	6.728	4.813	**32.02**
	en_output	15414	4.966	-	73.6
	rsc_encode	2	1.375	-	20.4
	others	-	0.387	-	6
demultiplex		1	1.563	1.563	**8.07**
rsc_encode		2	1.516	0.922	**7.6**
qammod		1	0.609	0.609	**3.08**
encode_bit		10308	0.594	0.594	**3.00**
qamdemod		1	0.203	0.203	**1.02**
conv		4	0.141	0.141	**0.71**
compconv		2	0.141	0.0	**0.71**
trellis		2	0.016	0.016	**0.08**
comb (filter)		1	0.016	0.016	**0.08**
randint		1	0.016	0.016	**0.08**
others		-			5.52
Total		-	19.760	-	100

Table 7.1: Algorithm analysis using Matlab profiling with block length= 100

respectively. Different Matlab functions along with the number of times the functions are called can be observed. The difference between total time and self time is that total time is the overall time taken by the function where as self time is the time taken by the function excluding sub-functions called by the particular function. The percentage column shows the time taken by the function to the overall system. Maximum time taken is the Log-Map algorithm with 38% and this function is called twice by the main function because 2 logmap functions are used by the decoder. This function has sub-functions such as α and γ which takes 53.8% and 5.1% respectively. Here it is to be noted, that these functions are called about 8000 times within the function logmap because of its dependency on the input block length. As the input block length increases the number of calling functions (like α) also increases. Thus as the block length increases the complexity increases. Here it is to be noted that 15% of the logmap algorithm is taken by “if” statements. This shows that there is a high number of conditional operations performed in this functionality. The encoder function takes the second most time with 32%. Here two rsc_encoder functions take about 20% of the total encoder timing. Later the functions modulation and demodulation takes about 3% and 1% respectively. It is worth noting that the algorithm developed in Matlab which is a purely mathematical point of view, it is a procedural language suited for un-timed algorithms with mathematically friendly data types. At the implementation level, where it takes the algorithm and implement it in either software or hardware. Most of the FPGAs support fixed point data types; whereas Matlab focuses on floating point; when the design flow from Matlab to FPGA, it creates trouble to implement on a FPGA directly. It is also to be considered that Matlab works more as an interpreter rather than a compiler. So, the basic idea for using the Matlab profiling is to obtain the rough estimates of the time taken by each function. Considering these timing properties from Matlab profiler, partitioning of hardware and software is performed. The Log-map algorithm is considered on a software platform and the rest of the system is chosen to do on the hardware platform which includes the channel coding and the modulation schemes. The reason for this choice was that the complexity of the Log-Map decoder increases with the increase in the input block length. Here the parameters α, β and γ are interdependent on each other for each Log-map block. Because of more control dependencies in Log-map, possibility of parallelism is low. For this reason, it was decided not to proceed with the implementation of Log-Map onto the hardware whereas a better optimization can be obtained on the software platform.

3.3 Estimation of Time and Area using DK-Suite

In hardware/software co-design, Handel-C is good option as it describes the behavioral level of a system in terms of C syntax and supports parallel execution in digital design. Handel-C can be used to model in data path and control logic of a design similar to software language like C. In Handel-C each assignment takes one clock cycle and everything else is free. This feature gives control over clock timing and it could be possible to change the design starting from sequential execution of statements to add parallelism meeting certain timing constraints. Different experiments are conducted for the algorithm using Handel-C for parallelism. Figure 6 shows the trade-off curve between time and area for 16-QAM of a generic architecture. The curve shows the experiments conducted from completely sequential code through different stages of parallelism added to the code using “par” statement to the complete parallel code and calculating the time taken (in clock cycles) for each stage. It is observed that for this particular algorithm (16-QAM), the sequential program takes 1250 clock cycles with an area estimate of 1608038 gates where as the complete parallel code takes 400 clock cycles for 1613780 gates. This shows there is an increase of 0.38% in number of gate counts from complete sequential to complete parallel code. But the time has decreased to 3 times from 1250 to 400 clock cycles. Considering the different scenarios, for different parallelism options obtained, the experiments are continued to different target FPGA platforms (Xilinx Virtex II and Altera Stratix EP1S10F780C6).

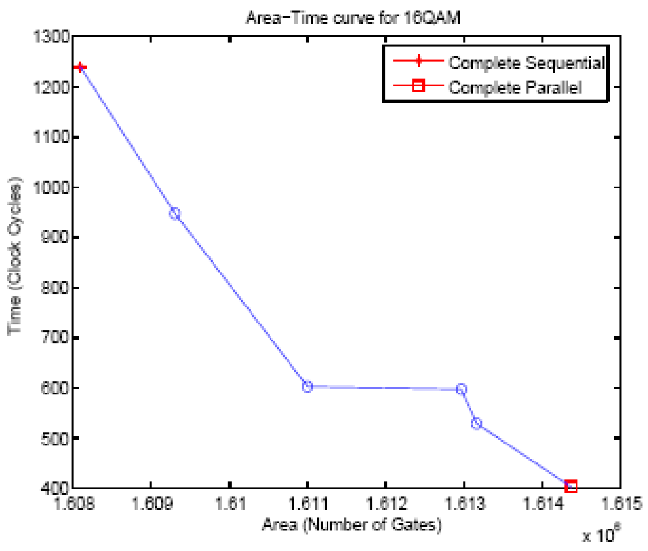


Fig.6 Trade-off curve (Area vs. time plot) for 16-QAM algorithm

3.4 FPGA implementations

Table 3 shows the results obtained from DK-Suite for these two target platforms while generating the EDIF output. The left most column is the type of operation performed and the respective gate and LUT count. The left columns are for Xilinx FPGA whereas the right most columns refer to Altera FPGA. The time taken for these respective platforms is the same, i.e., 1250 clock cycles and 400 clock cycles for sequential and parallel codes respectively. The timing properties can only be obtained when the tool is in “Debug” mode. In table 6, the optimizations are performed at three different levels of abstraction: the highest level after the compilation of the code (software optimization, no hardware slices involved), later after the expansion and finally at the gate level (specific to the FPGA) after the map process. The gate count, flip flops and the LUTs vary for the two platforms while mapping. This is because of the optimization performed by DK-Suite for the specific architecture and thus the partitioning of the system is done. It is observed that both target platforms are suitable for the implementation of the 16-QAM algorithm as per the estimation from DK-Suite.

	Xilinx Virtex II				Altera-Stratix EP1S10F780C6			
Complete Sequential	Nand Gates	Flip flops	Memory Bits	LUTs	Nand Gates	Flip flops	Memory Bits	LUTs
After Compilation	1611530	6051	172	-	1611530	6051	172	-
After optimization	46051	3838	172	-	56076	5492	172	-
After Expansion	65320	3827	496	-	75323	5491	172	-
After optimization	1891	230	0	-	3259	430	0	-
After Mapping	-	230	0	60	-	430	0	76
After Post-optimization	-	23	0	74	-	430	0	76
Partially Parallel	Nand Gates	Flip flops	Memory Bits	LUTs	Nand Gates	Flip flops	Memory Bits	LUTs
After Compilation	1611530	6051	172	-	1611530	6051	172	-
After optimization	56076	5492	172	-	46051	3838	172	-
After Expansion	76113	5491	496	-	64530	3827	172	-
After optimization	3091	430	0	-	2059	230	0	-
After Mapping	-	430	0	60	-	230	0	77
After Post-optimization	-	24	0	86	-	230	0	77
Complete Parallel	Nand Gates	Flip flops	Memory Bits	LUTs	Nand Gates	Flip flops	Memory Bits	LUTs
After Compilation	1611530	6051	172	-	1611530	6051	172	-
After optimization	44239	3526	172	-	44239	3526	172	-
After Expansion	63509	3525	496	-	62718	3525	172	-
After optimization	1069	93	0	-	1891	93	0	-
After Mapping	-	93	0	60	-	93	0	76
After Post-optimization	-	23	0	66	-	93	0	76

Table 3 Area Estimation of the 16-QAM system for different parallelism for two architectures

5. Conclusion and future work

This paper presented and discussed simulation and implementation issues of HSDPA-Adaptive Modulation and Coding (physical layer) of the 3GPP standard.

Firstly, the system has been simulated, showing that 16-QAM is more sensitive to channel errors than QPSK. Log-MAP is 2.5 times more complex than Viterbi algorithm with an increase on 0.5dB gain.

Furthermore, in order to implement the system, an algorithm has been developed to explore the design space based on optimization criteria for balancing area and time constraints. The algorithm helped during the design space exploration of different algorithms using Handel-C and DK-Suite tool for exploring the parallelism.

Finally, back-end tools have been used for applying low level optimizations for time and area constraints onto Xilinx Virtex II and Altera Stratix FPGA platforms. We have observed that both target platforms are suitable for the implementation of the 16-QAM algorithm as per the estimation from DK-Suite. Finally, the results show that design trajectory at different abstraction levels help in progressing from system specification to an efficient final implementation within a reasonable design time.

References:

[1] Hans-Peter Schwefel, Wireless Networks-III., Lecture slides, Aalborg University, 2005

[2] 3GPP TS 25.211 V6.5.0 (2005-06),Technical Specification Group Radio Access Network; Physical channels and mapping of transport channels onto physical channels (FDD) (Release 6)

[3] 3GPP TS 25.212 V6.5.0 (2005-06), Technical Specification Group Radio Access Network; Multiplexing and channel coding (FDD)(Release 6)

[4] 3GPP TS 25.213 V6.4.0 (2005-09), Technical Specification Group Radio Access Network; Spreading and modulation (FDD)(Release 6)

[5] 3GPP TS 25.308 V6.3.0 (2004-12), Technical Specification Group Radio Access Network; High Speed Downlink Packet Access (HSDPA); Overall description; (Release 6)

[6] 3GPP TR 25.950 V4.0.1 (2005-07), Technical Specification Group Radio Access Network; UTRA High Speed Downlink Packet Access (Release 4)

[7] 3GPP TS 25.302 V6.5.0 (2005-09), Technical Specification Group Radio Access Network; Services provided by the physical layer (Release 6)

[8] Celoxica DK-Design Suite, www.celoxica.com