[✓ ☒] **Public**

☐ **Confidential**

Project:	**ICESTARS**
Project Number:	FP7/2008/ICT/214911
Work Package:	WP2
Task:	T2.1
Deliverable:	D2.11 (version 1.1)

Title:	Public report on the developed scaling algorithms with benchmark results	
Author(s):	Taisto Tinttunen	(taisto@icestars.eu)
	Ville Karanko	(ville@icestars.eu)
Affiliation(s):	AWR-APLAC Corporation	
Date:	12 - Oct	

Table of Contents

Introduction 3

Harmonic Balance analysis 3

Scalability in multi-threaded analysis 4

Scalability with respect to number of independent time-scales (tones) and number of frequencies 7

Background 7

Implementation 8

Tensor methods for nonlinear equations 9

Hierarchical iteration of the Harmonic Balance Jacobian 11

Conclusions 13

References 14

Introduction

This document describes the development made in APLAC Simulators Harmonic Balance algorithm during ICESTARS project. The main emphasis has been in scalability to ensure analysis of large problems. By scalability we mean the increase in memory consumption and/or simulation time when the problem size, say number of frequencies taken into account or number of circuit elements, is increased.

In analysis of whole RF systems the circuit sizes are large. Instead of few circuit elements the analysis is performed on thousands of elements. The signals used are also more complicated including several independent frequencies. It is typical that it takes a lot of time to simulate this type of circuits. Therefore the analysis methods should also utilize the computational resources as optimally as possible.

Harmonic Balance analysis

Harmonic Balance (HB) is a frequency domain, nonlinear method developed to address the issue of fast and accurate simulation of circuits in the frequency domain. This means circuits whose normal operational condition is close to a periodic steady state (PSS) or almost periodic steady state in the case of multiple independent periodic inputs. It treats all signals as a sum of a predefined number of sinusoids, thus directly computes PSS of a circuit. Since the PSS is computed directly, an error-prone initial transient simulation is avoided resulting in considerable accuracy improvement. Furthermore modeling of many RF passive components is simplified in the frequency domain and a result from the HB analysis contains qualitative information on the frequency spectrum that cannot be obtained using a time-domain transient analysis. This is particularly important information in case of highly nonlinear electronic circuits where the input may be a 1-tone (periodic) or a multi-tone (almost periodic), but the output will contain distortion products up to n:th order.

HB has been successfully applied in the AWR-APLAC Corporation’s commercial circuit simulator APLAC and Infineon’s in-house circuit simulator TITAN for simulation of basic building blocks in an RF circuit, such as mixers, power amplifiers, frequency multipliers, modulators, and (with a built-in search of the oscillation frequency) also for autonomous circuits like oscillators.

HB analysis is a prerequisite for a number of further analyses which are built on top of it, such as HB noise analysis (HBNOISE), sweep of PSS (HBTRAN), small-signal PSS analyses (HBAC/HBTF/LSSS), analysis of multi-port networks in PSS (HBSPARAM), Volterra on HB (VoHB), envelope analysis etc. In addition, some time-domain analysis methods are closely related to HB analysis, such as DC or transient analysis (TRAN). Usually DC analysis result is used as an initial guess. However, for some difficult circuits, such as dividers, a transient analysis may be used to get the initial guess for node voltages. The HB analysis dependence on other analysis methods is shown in figure 1.

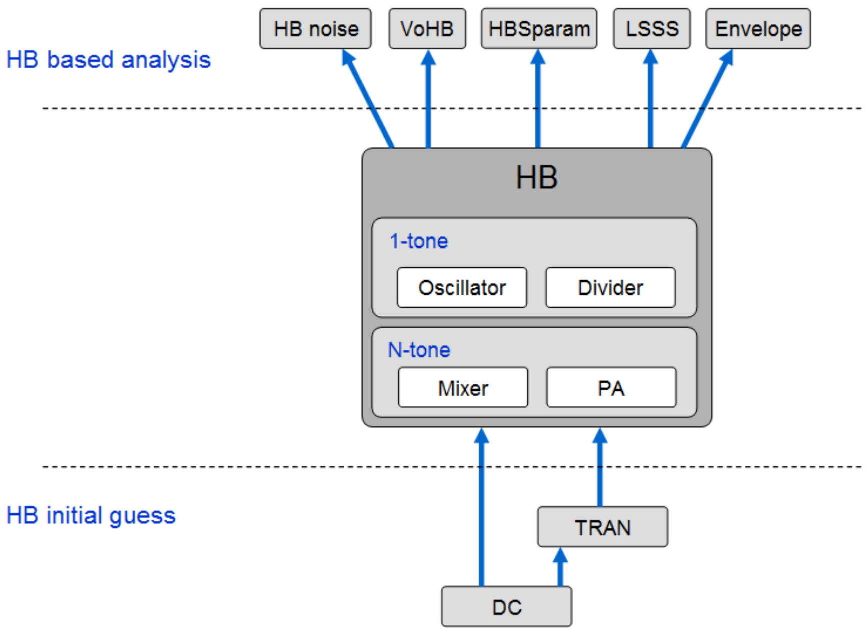

Figure 1: HB analysis and its links to other analysis methods

Typical applications of HB range from RFICs (with hundreds of nonlinear and very few linear components) to discrete board/PA modules (with few nonlinear devices but hundreds of linear components). The computational complexity comes from the fact that we need to solve the circuit simultaneously at every node and every frequency. To facilitate this, either direct sparse solvers or iterative solvers are used.

Scalability in multi-threaded analysis

The scalability for multiple processors has been profiled and the phenomenon behind it has been studied. The main motivation for this study is the fact the recent development in CPU's has brought multi-threading available for practically all the designers. Dual-core processors have become main stream in even home laptops and more "work-horse" oriented computers have dual-quad core processors. In the Intel Core i7 architecture these quad core CPU's have hyper-threading, which doubles the amount of cores seen by the Operating System. This means that average engineers work with computers that have 2-16 CPU's (or cores) available and they also expect that the tools, which they are using are able to take full advantage out of the available hardware resources.

In figure 2 it is shown how the algorithm scales for 2-8 CPU's or cores if the result for 2 threads gave a speed-up of 1.7 to 1.9. Here speed-up is defined as the ratio of the CPU times of single and multithreaded simulation. It is clearly shown that even though a result of 1.7 for 2 CPU's is basically a useful speed-up, the amount of single threaded parts results in poor scaling when the number of threads increases. Even though we have used 8 threads, a speed up of only 3.5 is achieved, in other words, we have used less than 50% of the computers resources.

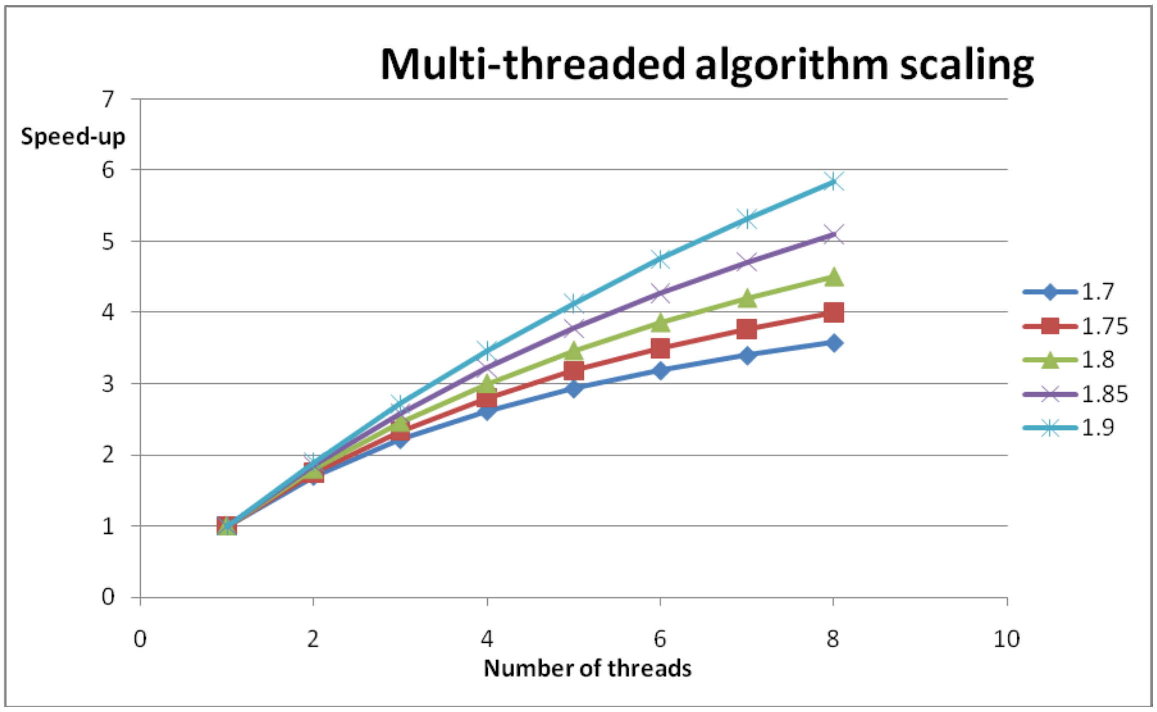

Figure 2: Scaling in the multi-threaded algorithm for 1-8 threads when the speed up of a dual core simulation (uses two threads) is 1.7, 1.75, 1.8, 1.85, and 1.9

We can also study the scaling as a function of the percentage of the algorithm that can be run in multiple threads. In the following figure 3 we show the speed-up when the multithreaded part of the algorithm is 70%, 75%, 80%, 85%, 90%, or 95% of the total amount.

It is clearly seen that a 15% single thread time means a speed up of 4x when run in 8 threads, i.e., the processor "efficiency" is 50%. For a 30% single thread time, speed up of only 2.5 is achieved as the single processor efficiency is less than 1/3. This actually highlights the multithreading problem – even though you have 70% of the code run in multiple threads – the scalability for 8 cores is not what most people (customers) would be expecting.

Source code profiling was used to find the bottlenecks slowing down multithreaded analysis. Naturally it was critical to minimize time spent in those parts of the algorithm that were run in a single thread. Figure 4 shows the achieved improvement for circuits where GMRES part of the analysis dominates.

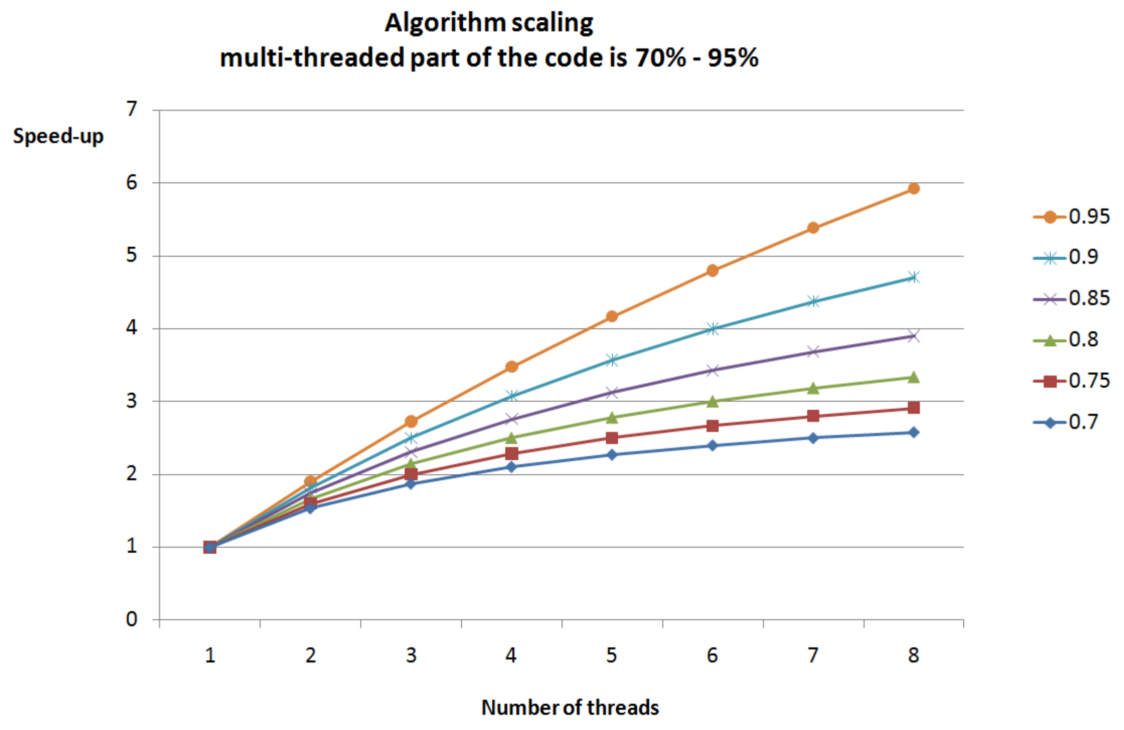

Figure 3: Scaling in the multi-threaded algorithm for 1-8 threads when the multi-threaded part of the algorithm is between 70% and 95%.

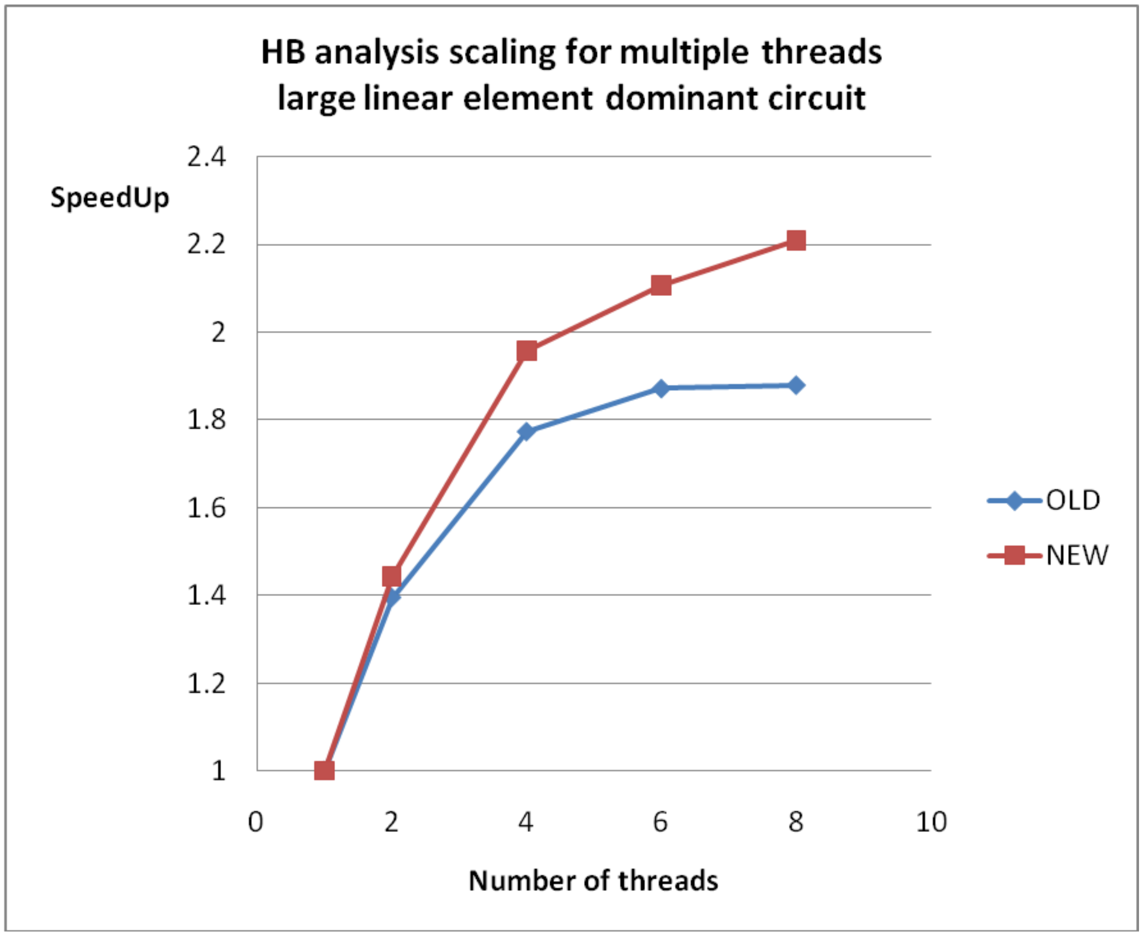

Figure 4. Scaling for a large linear element dominant circuit. With 8 threads the efficiency has not saturated in the new implementation. Practically no speed-up was achieved after 6 threads in the old implementation.

Scalability with respect to number of independent time-scales (tones) and number of frequencies

Background

Fourier transformation techniques are used in HB to compute Fourier components of currents of nonlinear devices (nonlinearity being defined in the time domain). Two things seem to be important here: being able to exactly represent a trigonometric polynomial of sufficiently high a degree (trigonometric degree) and doing the inverse and forward transformations with a fast algorithm.

Usually a technique known as “mapping” or “artificial frequencies” is used in multitone simulations. There, given a combination of harmonics, say, $k_1*f_1 + k_2*f_2$ is mapped to a single index $k:= k_1*m_1 + k_2*m_2$ using the integers m_i. That index defines the “slot” where the voltage coefficient at that inter-modulation frequency is located in a one-dimensional FFT (modulo the length of the FFT). If no two interesting coefficients are mapped on the same index the method will yield a meaningful result. Having done sampling of current in the artificial time-domain, the output current coefficient is read from the same location.

In dimensions higher than 2 the mappings are not dense so that some padding or unused locations occur (at arbitrary locations). These diminish the efficiency of the FFT. The problem is more pronounced as the dimensionality increases.

An alternative to the above technique is the sparse grids approach that can be developed also for trigonometric functions. It readily extends to arbitrary dimensions. It has the downside of not allowing for inter-modulation terms of high trigonometric degree except at the expense of considerably increasing harmonic order. It is not known whether such an approximation is good for RF-circuit problems.

Implementation

In search for tone-5 FFT mapping techniques for HB, a good mapping was not found using previously described techniques. The development for enabling more than 4-tones in a reasonable mapping was redirector to utilizing the existing optimal mappings for 1-4 tone problems to generate 5-8 tone mappings.

Previously, if the number of tones in APLAC simulations has exceeded 4, only BOX sampling has been available which has resulted in enormous number of sampling points. The results for the new one-dimensional mapping algorithm are presented in Table 1.

Number of Tones	Generated from	Resulting sampling points for DIAMOND 5	Sampling points in the old implementation BOX 5	Improvement
4	4	1.35e4	1.97e5	14.6
5	4+1	2.70e5	4.10e6	15.2
6	4+2	2.99e6	8.61e7	28.8

7	4+3	3.11e7	1.81e9	58.2
8	4+4	1.82e8	3.78e10	208

Table 1. Improved one-dimensional frequency mappings result in fewer sampling points. The 4-tone results are included as well for reference.

The new algorithm simply splits the total dimension as indicated in the column "Generated from" in Table 1 and maps the corresponding dimensions (co-ordinate projections) to a one-dimensional space using existing dense mappings. The result is a lower dimensional index-set (in the listed examples two-dimensional) that can be further mapped to a one dimensional set using a trivial bounding box mapping (BOX). This last mapping is no longer particularly dense but the end result is still significantly more economic than the trivial BOX mapping.

For a single diode, the current function evaluations (in the time domain) takes around 1E-6 seconds. For even the simplest circuit, this function would have to be calculated in frequency domain over ten times. As this is done by FFT, each of these circa 10 times the function would be called N times, where N is the required number of sampling points.

In practice, a 4-tone simulation can be run for a voltage source, one diode and one resistor circuit in 0.3 seconds. A 5-tone simulation samples over 300 times more (see Table 1) when the mapping had to be changed and that can be seen in simulation time. The simple circuit takes 1min 36s, which is exactly the ratio in nonlinear element sampling. The same circuit would take over 30 minutes in 6-tone analysis. With the new frequency mapping, the 6-tone simulation can be run in one minute. The large gap in the simulation times has been diminished to the same qualitative level seen earlier in the lower dimensional cases.

Tensor methods for nonlinear equations

Tensor-Krylov methods have been implemented in the open source large-scale numerical solver package Trilinos of Sandia National Laboratories (USA) [1]. Additionally, other numerical solvers for nonlinear systems are available in that package. Therefore, for trying out whether tensor methods offer an edge over inexact-Newton and for benchmarking purposes the coupling of the Trilinos nonlinear solver package NOX to APLAC was done.

The NOX package is written in C++ and specifies an abstraction (an interface class) of a nonlinear equation that needs to be concretely implemented before the solvers are called. The interface includes certain basic operations such as

- setting the point (vector) where the function value is computed
- computing the function value
- computing the Jacobian
- applying the Jacobian to a vector
- applying the Jacobian transpose to a vector
- copying the object

We implemented the equation class for the APLAC Harmonic balance equations. The implementation benefits from the existing efficient components of APLAC HB. These include

- matrix-implicit Jacobian computations
- various preconditioners
- multithreaded operations

Since in practice only one copy of a Jacobian representation can be retained (due to memory consumption) we implemented a strategy where setting a new point of computation invalidates the previous Jacobian. Then the function and Jacobian need to be possibly recomputed. In practice, it turns out that the NOX solvers use the possibility to copy the equations object only for storing the previous point of computation so that unnecessary calls to the interface are minimized.
Calling NOX from APLAC was enabled through a C function call interface and loads as a DLL on Windows. Trilinos is LGPL licensed (at the time of writing the latest version 10.4 is still LGPL 2.1 or later) so it needs to be dynamically linked if ever distributed to commercial clients.
As an example of this methods behavior a GSM-related design was simulated. The analysis performed is a 2-tone HB with a circuit of 2200 nodes (s3_onedim.i, confidential). For this circuit the NOX solver showed a significant improvement, see Table 2.

s3_onedim			
Intel Core2 dual-core T7200 2GHz 4MB L2 cache			
Single execution thread			
Solver	Function calls (M)	Matrix-vector products	time (s)
APLAC HB mode 1	4.4	933	49
NOX Inexact Newton	5.1	462	40
NOX Tensor-Krylov (Tensor 2)	2.2	288	24

Table 2. 2nd test circuit simulation time, function calls and matrix-vector products for APLAC Inexact Newton, NOX Inexact Newton and NOX Tensor solvers.

Hierarchical iteration of the Harmonic Balance Jacobian

The Harmonic Balance (HB) problem for RFIC circuits is in essence a problem of solving moderate to large nonlinear system of algebraic equations. The only generally efficient methods for solving such systems require iteration based on local linearization. An example of this is inexact-Newton.

Numerical solvers of the Newton type require a linear system solve for each iteration. For large systems that is usually done iteratively with, e.g., GMRES. The iterative linear solver, however, needs good search directions and they are computed by solving another linear system called a preconditioner. The preconditioners are required to be approximations that are much less expensive to solve than the original linear system. The most common preconditioner used with HB is a block-diagonal (block-Jacobi) system that is exact for linear time-invariant systems.

Block-Jacobi preconditioning becomes less effective and eventually useless as the nonlinear effects more and more determine the state changes in the circuit. However, it turns out that the conversion matrices have a property that can be exploited. Looking at a single conversion matrix or a one node circuit the matrix G has the hermitian Toeplitz structure (in exp-basis)

$$\begin{bmatrix} c_0 & c_1 & c_2 & \cdots & c_n \\ c_{-1} & c_0 & c_1 & \cdots & c_{n-1} \\ c_{-2} & c_{-1} & c_0 & \cdots & c_{n-2} \\ \vdots & & & \ddots & \vdots \\ c_{-n} & c_{-n+1} & c_{-n+2} & \cdots & c_0 \end{bmatrix}$$

It follows that a diagonal block of this matrix is just another conversion matrix with a smaller number of harmonics of the conductance (or capacitance) waveform. The hierarchical GMRES splits the frequencies of unknowns in two approximately equal parts and uses the iterative solution of the smaller problems as a preconditioner to the whole problem. If this idea is used for the sub-problems we eventually end up with blocks of one frequency only – the block-Jacobi preconditioner – that can usually be solved directly with a sparse solver.

The sparsity pattern of the HB Jacobian from QPSK receiver 1 tone simulation with 7 harmonics in complex form (15*119 rows and columns) is shown in Figure 5. It is shown in frequency major ordering (i.e. circuit ordering within each frequency index) and complex equation formulation. The upper left corner of the matrix corresponds to the largest negative frequency (index -7) and the lower right corner to the largest positive frequency (index 7). DC is at the center. Multilevel splitting is depicted by the red boundaries of proper low frequency blocks (with DC) and green boundaries that represent high-high conversion. Not all subdivisions are included for clarity.

The algorithm that is used to solve this problem is based on splitting the matrix according to the frequency-blocks and applying GCR-iteration for each block until some stopping criterion has been reached. No restarting was done and subspaces were reused.

Table 3 shows number of top level iterations and the estimated amount of (relative) floating point operations required by the same "basic" algorithm using single level or multilevel formulation.

	niter (upper level)	relative work estimate
Single level	43	1.0
Multilevel	15	0.39

Table 3. QPSK matrix solution results

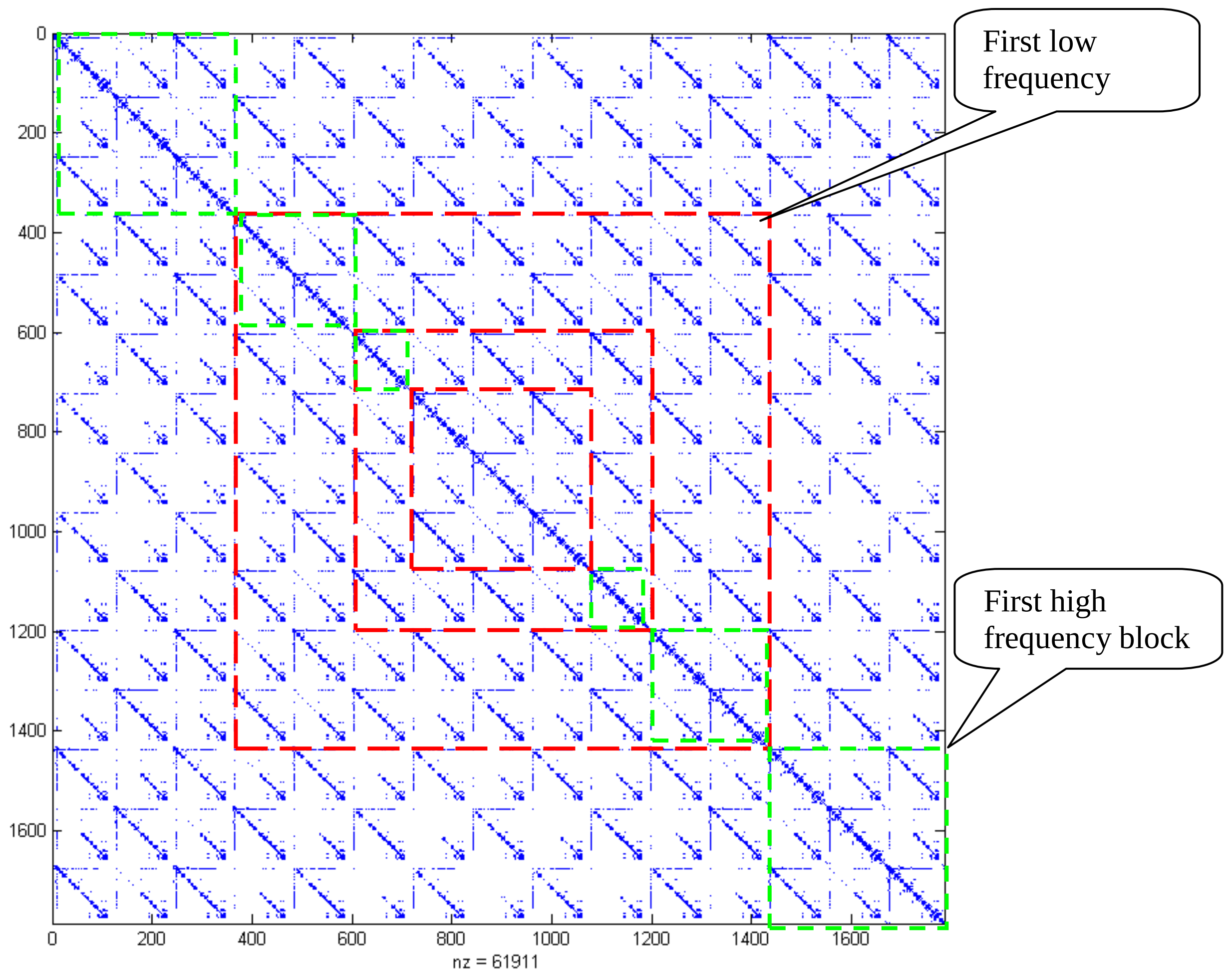

Figure 5. QPSK circuit sparsity pattern

Conclusions

Several different parts of Harmonic Balance algorithm were improved. The authors worked and further developed the Harmonic Balance algorithm in the APLAC Simulator of AWR-APLAC, which has been an industry leading high quality simulator for over a decade.

The focus was in scalability in terms of optimal utilization of computer resources as well as to methods that are suitable for very large circuits or large number of analysis frequencies. Such methods are iterative and matrix implicit. For utilizing computer resources, some bottlenecks in single threaded parts of HB implementation were removed which enabled better utilization of CPU's or cores.

For alternative solver technology, the Trilinos Tensor-Krylov solver from Sandia National Laboratory was integrated in the APLAC environment. It offers a high quality solver as an alternative in cases where the classic HB work horse, Inexact-Newton-GMRES solver looses orthogonality.

As a new method for preconditioning of the iterative problem, a hierarchical version of GMRES was implemented as well. Initial tests indicate that it promises smaller iteration counts so that the problematic restarting of the iteration might be avoided.

For better scalability for several independent excitations, a new method for defining a one-dimensional mapping based on existing optimal 1-4 tone mappings was developed. The new method reduces the number of required sample points dramatically enabling e.g. 5-tone simulations over 10 times faster.

References

[1] The Trilinos project webpage: http://trilinos.sandia.gov/