Hardware architecture for real-time EEG-based functional brain connectivity parameter extraction

Hardware architecture for real-time EEG-based functional brain connectivity parameter extraction

Rafael Angel Gutierrez Nuno, Chi Hang Raphael Chung and Koushik Maharatna

Faculty of Engineering and Physical Sciences, Electronics and Computer Science, University of Southampton, SO17 1BJ, Southampton, United Kingdom

E-mail: {R.GutierrezNuno, chrc1u16, km3}@soton.ac.uk

Sept. 2020

Abstract.
Objective. Design a novel architecture for real-time quantitative characterization of functional brain connectivity networks derived from wearable Electroencephalogram (EEG). *Approach.* We performed an algorithm to architecture mapping for the calculation of Phase Lag Index (PLI) to form the functional connectivity networks and the extraction of a set of graph-theoretic parameters to quantitatively characterize these networks. This mapping was optimized using approximations in the mathematical definitions of the algorithms which reduce its computational complexity and produce a more hardware amenable implementation. *Main results.* The architecture was developed for a 19-channel EEG system. The system can calculate all the functional connectivity parameters in a total time of 131μs, utilizes 71% logic resources, and shows 51.84 mW dynamic power consumption at 22.16 MHz operation frequency when implemented in a Stratix IV EP4SGX230K FPGA. Our analysis also showed that the system occupies an area equivalent to approximately 937K 2-input NAND gates, with an estimated power consumption of 39.3 mW at 0.9 V supply using a 90 nm CMOS Application Specific Integrated Circuit (ASIC) technology. *Significance.* The proposed architecture can calculate the functional brain connectivity and extract the graph-theoretic parameters in real-time with low power consumption. This characteristic makes the architecture ideal for applications such as a wearable closed-loop neurofeedback systems, where constant monitoring of the brain activity and fast processing of EEG is necessary to control the appropriate feedback.

Keywords: Functional connectivity, Phase Lag Index, Graph Connectivity, FPGA, EEG, real-time processing. Submitted to: *J. Neural Eng.*

1. Introduction

Electroencephalogram (EEG) is a non-invasive modality for monitoring brain activities and has been widely applied in clinical neuroscience. Recently, using EEG in

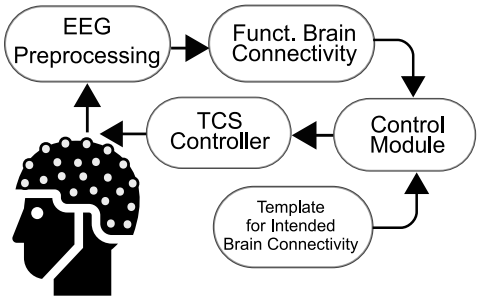

Figure 1. The overall concept for a real-time functional brain connectivity driven Transcranial Current Stimulation system.

conjunction with techniques such as functional brain connectivity (FC) and graph-theoretic characterization of these connectivity networks showed promising results in the detection/diagnosis of different cognitive disorders such as epilepsy [1, 2], Alzheimer's [3,4], Parkinson's [5,6], and autism [7,8]. The advent of wireless EEG with dry electrodes has widened its potential application in nomadic environments where brain activity monitoring and subsequent neurofeedback could be applied in a closed-loop system for treating neurological disorders "anytime, anywhere". The present work is part of a larger visionary project for developing real-time FC driven Transcranial Current Stimulation (TCS) based neurofeedback to treat cognitively impaired subjects in out-of-clinic settings. The overall concept of the system is shown in Figure 1. We envisage it as a wearable device with continuous capture of EEG signals which are used to formulate FC on-the-fly and compare it with templates of the desired "correct" functional brain connectivity. According to the outcomes of this comparison, a special controller module may control the appropriate multi-site TCS distribution with the aim of matching the desired FC. All of the associated operations will be performed in an iterative closed-loop control process that provides the patient with the appropriate stimulus at the right time instant. To make it a sustainable device, it needs to finish the overall process in the order of a second and will need low energy consumption for longer operational battery life owing to its potential continuous operation.

Traditional offline processing where the data is transmitted either to a mobile phone or a central computing facility is not applicable in this device owing to the real-time constraint to complete the entire control loop. The whole loop involves not only computationally demanding processes such as FC formulation and graph-theoretic parameter extraction but also an adjustment of stimulus current topography through the controller subsystem that would require a significant amount of time to complete using traditional offline processing. Another important point is the need for the system to have low energy consumption to sustain its operation. In [9], there is an extensive discussion about this issue in the general scenario of physiological parameter monitoring. It showed that instead of transmitting the data continuously to a central facility, better energy performance could be obtained by processing the signal locally at a sensor device provided the computationally expensive algorithms could be approximated as

a computationally light version without sacrificing the accuracy too much. Thus, a self-sustained and highly computation-demanding wearable system such as the one we envision is best realized using dedicated Application Specific Integrated Circuit (ASIC) design. This calls for an algorithm to architecture mapping optimization in terms of computational burden to make the algorithm "lightweight" without affecting the desired performance. To achieve this, in this work we propose a novel system hardware architecture for a 19-channel EEG system that has two main modules, namely, computation of PLI matrix resulting into the formulation of FC and, the extraction of graph-theoretic parameters from these FCs. This architecture is based on the proof-of-concept work presented in [10] and [11]. However, those two works only presented the two modules required for implementing the entire system, namely, the PLI computation and graph-theoretic parameters extraction respectively. In this work we present a complete and integrated solution to perform the whole calculation of the FC measurements from EEG signals in real-time. To our knowledge this is the very first attempt to implement such a system in real-time hardware. The main novelty lies in the proposed approximations in the mathematical definitions of the graph-theoretic parameters which make them "lightweight", i.e., reduce the numbers of computationally intensive arithmetic operations such as divisions, multiplications and square/cubic rooting that are fundamental in the conventional definitions of these parameters. We showed that such approximations have minimal impact on the final performance of the system by comparing the hardware outputs with the traditional software [12, 13] implementation of these parameters. This optimal algorithm-to-architecture mapping methodology made the overall design amenable for hardware implementation as the circuit components required for realising the above-mentioned arithmetic operations consume significantly large silicon areas and energy. Furthermore, this design can act as an emulation alternative in real-time to the current MATLAB simulation toolboxes (NBT [12], BCT [13] and HERMES [14]) which performs the calculation in an offline manner, enabling faster exploration of different cognitive phenomena. The proposed design is a stand-alone unit that takes appropriately preprocessed EEG signal (from a separate system) as its input to generate the quantitative graph-theoretic parameters for characterising the FC.

It is worth mentioning that from the overall system perspective, a set of preprocessing steps would be needed before inputting the signal into our architecture. This process will include the artifact removal and band-pass splitting so the analysis can be performed on the clean signal and at specific frequency bands (delta, theta, alpha, beta, gamma). The removal of artifacts in EEG signals has been extensively discussed in the literature and therefore we decide to leave it out of the scope of this work and focus only on the architecture design for the FC and graph-theoretic parameters.

Table 1. Summary of the graph-theoretic parameters and its relationship with brain functions [13].

Graph-theoretic parameter	Relationship to brain function
Degree	*Interconnection between one region and other regions of the brain.*
Degree (weighted)	*Similar to the degree but also represent the strength between these regions.*
Density	*The ratio between the brain regions connected and all the possible connections among all regions.*
Clust. coeff.	*Measures how the different regions of the brain tend to cluster between them*
Trans.	*Similar to the clustering coefficient but it measures the segregation in the whole brain instead of focusing on every region.*
Charact. path length	*Establish how efficient is the intercommunication among different brain regions*
Ecc., Rad. and Dia.	*Based on the characteristic path, these parameters establish the most/worst communication path form between the different brain regions.*

2. Background

2.1. Relationship between the functional connectivity parameters and brain functions

Functional brain connectivity has been shown as one of the main methods to understand the functions of the brain. Typically, FC creates a graph amongst different areas of the brain where each area of the brain represents a node of the graph whilst the arcs between them represent the connectivity strengths between different brain regions (nodes). The fundamental importance of such an approach is that it can characterize the information communications between different parts of the brain quantitatively in terms of graph network. Two global characteristics of the brain that are important for any neurodegenerative/neurodevelopmental disease (such as Alzheimer's, epilepsy, autism and Parkinson's [1–8]) are: 1) the ability of the brain to integrate information from a distant source and; 2) specialized local processing. The graph-theoretic parameters that could quantitatively describe such properties of brain are shown in the Table 1.

2.2. Phase Lag Index (PLI)

The PLI is a functional connectivity measurement commonly used to analyze EEG signals [4, 6, 15] introduced by [16] as a new measure to address the problem of volume conduction in functional connectivity. The PLI is calculated using (1), where ϕ is the instantaneous phase of the signal in the channels i and j, across a window with a total of L samples.

$$PLI(i,j) = \left| \frac{1}{L} \sum_{l=1}^{L} sign(\phi_i(l) - \phi_j(l)) \right| \tag{1}$$

The matrix obtained using (1) is an N-by-N matrix where the N corresponds to the number of nodes in the network (total number of electrodes in the EEG). Every "i, j" element in the matrix is known as an edge, and it represents the synchronization between these two i and j nodes.

2.3. Degree and Weighted Degree

The node degree is a measurement used to represent the interconnection of one node with the other nodes in the network. This provides information about how closely interconnected the nodes in the network are. The node degree can be estimated using (2), counting the number of edges (a_{ij}) greater than a certain threshold (i.e. zero) for a particular node i. Furthermore, a weighted degree (k_i^w) could also be calculated but in this case, the weights (w) of the edges are accumulated for a particular node. The weighted degree not only establishes that there is a connection between two nodes but also provides information about how strong this connection is.

$$k_i = \sum_{j=1}^{N} a_{ij} \; ; \; k_i^w = \sum_{j=1}^{N} w_{ij} \tag{2}$$

2.4. Density

The density can be obtained using (3), which counts the number of edges (w) greater than a threshold in the upper triangular elements of the matrix and divided by the total possible number of connections in the network. This ratio provides information on how dense the connection between the different brain regions is.

$$D = 2 \sum_{j,i \in N \; j>i} upper(w_{ij} > threshold)/(N^2 - N) \tag{3}$$

2.5. Clustering Coefficient

The clustering coefficient is used to measure the cliquishness between a group of nodes in the network, which in turn determines how clustered the network is [17]. The calculation

of the clustering coefficient is done as in (4), calculating the geometric mean (t_i) of the weights (w) for all the possible triangles around a specific node i.

$$t_i = \frac{1}{2} \sum_{j,h \in N} \sqrt[3]{w_{ij} w_{jh} w_{ih}} \; ; \; CC_i = \sum_{i \in N} \frac{2t_i}{k_i(k_i - 1)} \tag{4}$$

Also, it is possible to obtain the mean clustering coefficient as in (5) which gives a more general idea of how clustered the whole network is.

$$CC_{mean} = \frac{1}{N} \sum_{i \in N} CC_i \tag{5}$$

2.6. Transitivity

Transitivity is a variation of the mean clustering coefficient measurement used to understand the segregation of the whole network. The transitivity is calculated using (6), similar to the clustering coefficient but in this case, the values of the numerator and denominator of every node are accumulated before performing the division. This normalization helps the transitivity to be less susceptible to nodes with a lower degree [13] and therefore its advantage over the mean clustering coefficient.

$$T = \frac{\sum_{i \in N} 2t_i}{\sum_{i \in N} k_i(k_i - 1)} \tag{6}$$

2.7. Characteristic Path Length

The characteristic path length is a measurement of integration that describes the interconnection among the different nodes and how efficient the communication between them is. The characteristic path length is defined in (8). The shortest distance (d_{ij}) between two nodes can be obtained using (7), where $f(w)$ is a function to convert the edge values into distance values. A commonly used function is the inverse value of the weight.

$$d_{ij} = ShortestPath_{ij}(f(w)) \tag{7}$$

$$CPL = \sum_{i \in N} \frac{\sum_{j \in N \; j \neq i} d_{ij}}{N(N-1)} \tag{8}$$

Once the edge values are converted into distances, the shortest distance between the two nodes is found. The most common algorithm used to find the shortest distance between two nodes is Dijkstra's algorithm (DA) presented in [18]. The steps to perform the DA are the following:

(i) Create a set of unvisited nodes, which includes all nodes except node i.

(ii) Set the tentative distance d between the node i and every other node j to the edge length between them, i.e. $d_{i,j} = l_{i,j}$. Throughout the algorithm, $d_{i,j}$ will hold the shortest path length from i to j through visited nodes. Currently, only node i is visited, so the initial value of $d_{i,j}$ is $l_{i,j}$.

(iii) In the set of unvisited nodes, find the node k with the shortest tentative distance.

(iv) For every node j, update the tentative distance as follows: $d_{i,j}(new) = minimum(d_{i,j}\ ,\ d_{i,k}+l_{k,j})$. The existing shortest path from i to j through visited nodes has a path length of $d_{i,j}$. If a path through node k (which has length $d_{i,k}+l_{k,j}$) is shorter, this would become the new shortest path between the nodes.

(v) Remove node k from the set of unvisited nodes. Repeat steps 3–5 until the set of unvisited nodes is empty.

(vi) After visiting all nodes, $d_{i,j}$ now holds the shortest path length from i to j through visited nodes.

2.8. Eccentricity, Radius and Diameter

The eccentricity is the maximum distance between two nodes in the network. In addition to this measurement, there is also the radius and diameter of a network, which corresponds to the minimum and maximum distance respectively for the whole graph. The distance between the nodes is calculated using (7) as in the characteristic path length.

3. Optimization of graph-theoretic metrics in terms of computation complexity

As described in the aforementioned section, each of the graph-theoretic parameters requires complex arithmetic computations such as computing cubic root, divisions and multiplications. Typically, such operations require significant arithmetic resources and therefore have a direct impact on the energy consumption of the architecture as energy consumption is directly proportional to the arithmetic complexity. Therefore, to make the system efficient, one may need to approximate the mathematical definitions of graph-theoretic parameters so that they can be amenably mapped to a hardware platform while reducing the overall computational complexity - ideally avoiding dividers and cubic root computations and using a minimal number of multiplications - without sacrificing accuracy. In this section, we describe the approximations in the definitions of the graph-theoretic parameters done in this work for an efficient algorithm to architectural mapping leading to optimization of arithmetic resources. From the equations (4) and (6) in section 2 it can be seen that transitivity and clustering coefficient all require computation of the geometric mean of triangles in the matrix and the Degree. Therefore, we have decided to consider the clustering coefficient computation as a core computational unit that apart from computing clustering coefficients also outputs the geometric mean of all the triangles t_i and the degree. These values are then shared with the transitivity and density to perform the respective calculations as shown in Figure 2. In this way, the values are reused and redundant hardware in the system is eliminated.

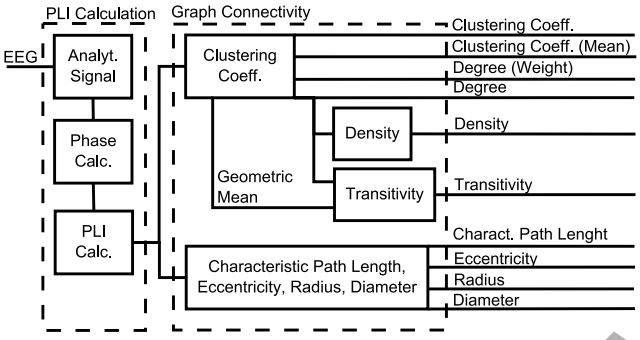

Figure 2. Block diagram of the general architecture for the PLI and the graph-theoretic measurements calculation.

3.1. Design considerations

Since this is the first attempt, to our knowledge, for implementing such a system in hardware, there is no existing design specification for such a system. However, given the visionary application described in Section 1, we considered the following conservative logical design constraints. The overall sensing, parameter extraction and the stimulation process (see Figure 1) need to be completed within 1 ms considering the neuronal firing rate is O(ms). This leaves only a fraction of a ms to complete the FC computation and corresponding graph-theoretic parameters extraction part, which is the main focus of this paper. Therefore, we conservatively estimated our real-time definition in this scenario as <0.5 ms. In terms of duration of operation over a day, we do not envisage that the system will need to operate continuously for more than 18 hours in the worst case and its battery could be recharged during the sleeping time of a subject. Considering the current battery capacity of a wearable system is 4200 mAh at 3.82 V [19], this means that the ceiling of the current consumption of the system <= 233 mA/hour. Considering this figure, the energy consumption of the system should be <= 890 mW.

3.2. Clustering Coefficient Optimization

The most computationally expensive operation needed in the clustering coefficient calculation is the computation of the geometric mean of all the triangles t_i. This is an iterative process that involves the multiplication of the weights for every possible triangle permutation as in (4) followed by cubic root evaluation. As can be seen in [20, 21] such computation needs significant arithmetic resources. To avoid that, we opt to take a different approach by completely avoiding evaluation of cubic root and simply using the multiplication between the weights as the numerator in (9) as in essence the cubic root will only contribute in terms of a scaling factor and not in terms of the actual nature of the t_i and hence for the value of clustering coefficients.

$$t_i = \frac{1}{2} \sum_{j,h \in N} w_{ij} w_{jh} w_{ih} \; ; \; CC_i = \sum_{i \in N} \frac{2t_i}{k_i(k_i - 1)} \tag{9}$$

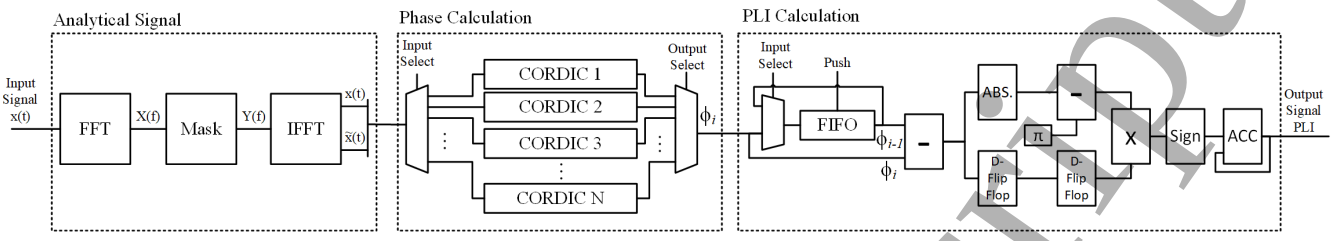

Figure 3. Block diagram of the hardware architecture proposed for the PLI calculation. Extracted from [10].

Table 2. Correlation between the clustering coefficient and the transitivity obtained with and without the cubic root.

	Correlation	
Subject	Clustering Coefficient	Transitivity
1	0.898	0.928
2	0.920	0.948
3	0.918	0.936
4	0.905	0.923
5	0.942	0.959
6	0.937	0.960
7	0.914	0.932
8	0.901	0.915
9	0.943	0.948
10	0.953	0.957
Mean	0.923	0.941

To verify the impact of such modification in the definition of t_i, we carried out a correlation analysis of the clustering coefficient calculated using (4) and (9) withdrawing five-minute real EEG signals from 10 different subjects using data obtained from [22]. The data was recorded by 23-channels at 256 Hz with 16-bit resolution. The results are shown in Table 2. The average correlation between these signals is 0.923, which indicates that even though the scale of the signal is different, the morphology of the signals remains very close to the original and therefore could be used in the characterization of FC matrices.

To reduce the complexity even further, we utilize the symmetry properties of the PLI matrix. This leads to the computation involving only the elements of the upper triangle of the PLI matrix in the geometric mean calculation of t_i resulting in a reduction of computational complexity by half. This optimized geometric mean of t_i was then used to calculate clustering coefficients as well as the degree k_i (which was also used in the calculation of the clustering coefficient).

3.3. Transitivity

Since the transitivity calculation is similar to the clustering coefficient, we reuse the geometric mean and the degree for deriving transitivity. To validate that the modifications done to the clustering coefficient do not affect the calculations of the transitivity, the same correlation analysis has been done and its results are shown in Table 2. The average correlation is 0.9405, which demonstrates that not using the cubic root on the triangle calculation does not have a substantial impact on the transitivity either.

3.4. Density

Exploiting the symmetric properties of PLI, the summation of the binary degree k_i is equal to two times the summation of the non-zero values in the upper triangular elements of the matrix. Thus, it is possible to calculate the density as in (10) with the already calculated degree, reducing the arithmetic resources required.

$$D = \frac{\sum_{i \in N} k_i}{N(N-1)} \tag{10}$$

4. Hardware architecture

The system is composed of two main core modules: the PLI calculation and the graph connectivity module as shown in Figure 2. The first one calculates the PLI matrix from an EEG signal, while the graph connectivity module uses that matrix to extract the graph-theoretic parameters discussed in section 2.

4.1. PLI Architecture

The architecture used for the PLI calculation is the same as the one presented in [10] and it is shown in Figure 3. The main components of this architecture are the Analytical signal, Phase calculation and PLI calculation modules. The Analytical signal module performs a Fast Fourier Transform (FFT) to the EEG signal followed by a convolution with a mask function that doubles the power for frequencies higher than zero, keep the same power when the frequency is equal to zero, and makes the power equal to zero for the rest of the frequencies. After this, an inverse FFT (IFFT) is performed to obtain the analytical signal. The analytical signal is a complex signal in which the real component is the original signal ($x(t)$) and the imaginary component ($\tilde{x}(t)$) corresponds to the Hilbert transform. Then, the instantaneous phase is obtained by calculating the arctan function of the analytical signal. This is done using a COordinate Rotation DIgital Computer (CORDIC), which is a low complexity computing technique commonly used to perform arithmetic operations in hardware. Finally, the PLI calculation module uses these instantaneous phases to calculate the PLI value as in (1). This architecture was extended to 19-channels in this work instead of 16-channels as originally done in [10].

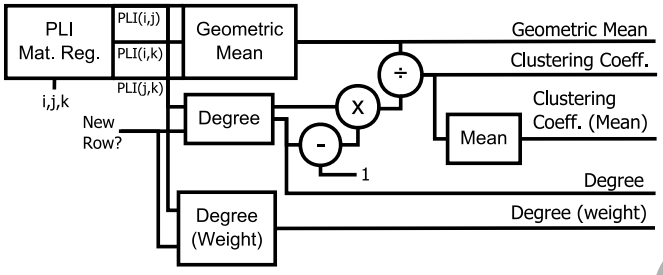

Figure 4. Block diagram of the hardware architecture proposed for the clustering coefficient.

4.2. Clustering Coefficient

As explained in section 3, the clustering coefficient module is considered as a core module that calculates not only the clustering coefficient but also the geometric mean of the triangles and the degree (binary and weighted). The block diagram of this module is presented in Figure 4, consisting of three basic processing blocks: geometric mean, degree and weighted degree. The geometric mean t_i was calculated first as in (9), multiplying and accumulating the weights of all the possible permutations for every node i in the matrix. At the same time, the degree (binary and weighted) are calculated in the respective blocks (explained in a later subsection). Once the geometric mean of all the permutations in node i have been done, the values are divided by $k_i(k_i-1)$. The same process is repeated for every node and the results are accumulated until the final clustering coefficient value is obtained.

4.3. Degree and Weighted Degree

The degree and the weighted degree are calculated as in (2), accumulating the number of edges greater than zero and the weights of the edges respectively for every node in the matrix. Both degrees are calculated in the clustering coefficient module. By doing all triangles permutations for every node, we guarantee that all the edges of this node will be used at least once, therefore it is possible to reuse these weight values in the calculation of the degree. By doing this it is not necessary to include an extra copy of the PLI values and the extra logic to perform the degree calculation in a separate process. The only disadvantage of this is that the degree calculation would take longer than it would have done using a separate process, but this does not have a critical impact on the performance since the degree result is still available before the other calculations, e.g. the clustering coefficient.

4.4. Density

As mentioned before, the density module reuses the previously computed values of the degree in the clustering coefficient module, and thus consists only of an accumulator and divider as shown in Figure 5. The density is calculated as in (10), accumulating the

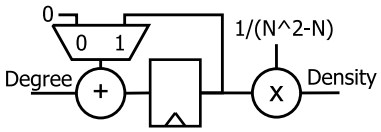

Figure 5. Block diagram of the hardware architecture proposed for the Density.

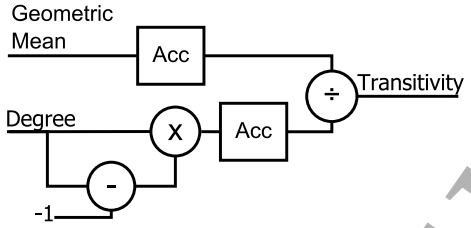

Figure 6. Block diagram of the hardware architecture proposed for the Transitivity.

degree values of every node followed by a division by the total number of elements outside the main diagonal. Since the total number of elements is constant, we precompute the inverse of it and multiply this inverted value with the output of the accumulator to perform the division operation. This approach reduces the total arithmetic complexity as opposed to using an arithmetically complex full divider.

4.5. Transitivity

The transitivity calculation is almost identical to the clustering coefficient with the exception that the numerator and denominator are accumulated before performing the division. Thus, the triangle and the degree values from the clustering coefficient calculation are reused here and the only extra hardware needed is an accumulator for the numerator, a multiplier and accumulator for the denominator and the final divider to perform the calculation. The architecture for the transitivity is shown in Figure 6.

4.6. Characteristic Path Length (CPL)

To perform the CPL calculation it is necessary to map the weights of every edge in the matrix to an equivalent distance. Here, we decided to perform the mapping as in (11). With this mapping, the weight values closer to one which represents a strong connection on the brain connectivity matrix will have a small distance value. On the other hand, values close to zero will have a greater distance value representing a weak connection between the nodes.

$$f_{ij}(w_{ij}) = 1 - w_{ij} \tag{11}$$

Once the mapping is done, the shortest path is found using Dijkstra's algorithm (DA) discussed in Section 2. The Shortest Path Finding Unit (SPFU) in Figure 7 (a), performs the distance mapping shown in (11) and compares the actual minimum distance from the node against the particular node distance plus the current node distance, as

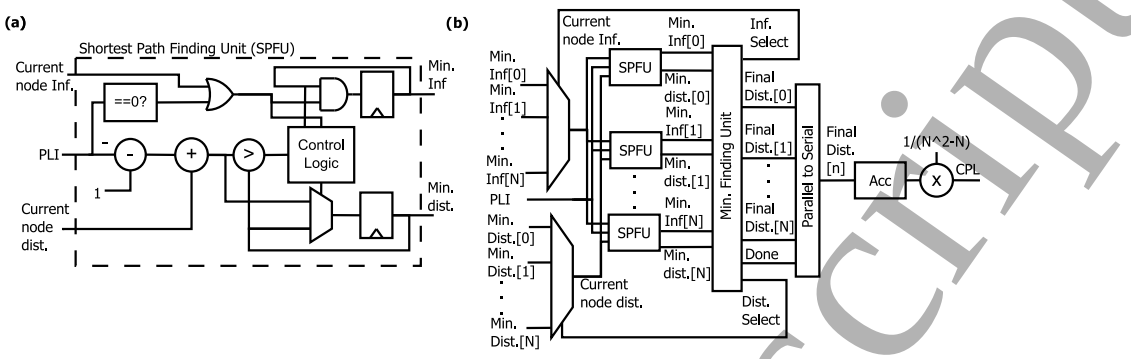

Figure 7. (a) Block diagram of the hardware architecture proposed for the Shortest Path Finding Unit (SPFU). (b) Block diagram of the hardware architecture proposed for the characteristic path length.

it is done in step 4 in Section 2. The Minimum finding unit finds the node with the minimum distance that will be used as the new node (step 5) and the current node distance is updated with this node distance. The process is repeated until the distance from the initially selected node and the rest of the nodes is found. Then, the whole process is repeated with a different initial node and the minimum distance from that node with respect of the other is found. When all the minimum distances from all the nodes with respect to the other nodes is completed, these values are accumulated and later divided by '$N^2 - N$' as in (8), obtaining the final CPL value.

4.7. Eccentricity, Radius and Diameter

To calculate these three measurements, it is necessary to calculate first the shortest distance between nodes. This is already done during the calculation of the CPL, then these values are stored and reused here. Then the calculation is just as simple as finding the maximum values for every node (eccentricity) and finding the maximum and minimum of the whole network (diameter and radius respectively). This is done by checking node distance and using a comparator to determine if the new value becomes a maximum/minimum against the previous value examined.

5. Validation and Results

To verify the proper behavior of the complete system, all the modules were connected together, and a functional test was performed using simulations.

5.1. Validation data

The validation was done using a five-minute, 19-channel EEG signal recorded at 256 Hz, from 10 different subjects. These records were obtained from the database of [22]. These EEG signals are the same used in the parameter optimization for section 3.

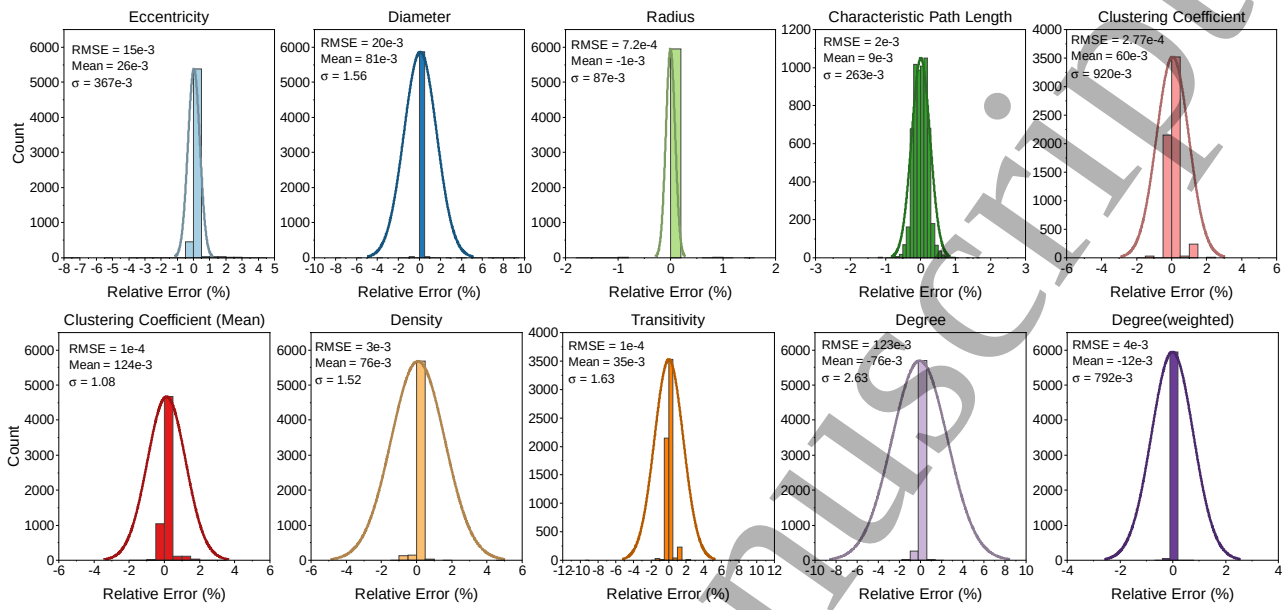

Figure 8. Histogram with the relative error results from the functional validation for every graph-theoretic parameter calculated for the 600 windows and the 10 subjects. The straight line shows the histogram fit of the relative error in order to appreciate better the distribution of the error. Inside every plot was added the mean RMSE, mean and standard deviation value for all parameters.

5.2. Functional Validation

The proposed system was simulated using the ModelSim software and the output of this system was compared with the output obtained using the toolbox from [13]. The window size used in the PLI calculation was 128 samples. Since the EEG used was sampled at 256 Hz, it is possible to calculate two windows for every second of the EEG. Therefore, a five-minute EEG signal will generate a total of 600 PLI matrices for every subject. From every PLI matrix, the graph-theoretic parameters mentioned in section 2 are calculated, followed by calculation of the relative error between the hardware architecture and the MATLAB toolbox. The same process was repeated for every one of the 10 subjects.

5.2.1. Error Calculation: The results of the relative error are shown in Figure 8 as a histogram for every parameter calculated across all the windows and all the subjects. Based on the distribution fit of the histogram fit in this figure, it can be observed that the center of the curve is located close to zero, which indicates that for most of the values obtained the relative error is concentrated in this area. The degree is the one with the maximum relative error range that goes from -7.97% to 7.82% (mean= -0.076%, 3σ= 7.9%). Despite this wide range, most of the errors obtained were concentrated in a range of about -2% to 2%. In addition, the RMSE value was calculated for all the parameters. The maximum RMSE found among all the parameters is 0.123 for the degree, a small value that also proves the correct behavior of the system.

Table 3. Number of clock cycles required for the calculation of the different parameters.

	Clock Cycles
Degree	$N\binom{N-1}{2}+1$
Degree (weighted)	$N\binom{N-1}{2}+1$
Clustering Coefficient	$N\binom{N-1}{2}+7$
Transitivity	$N\binom{N-1}{2}+6$
Density	$N\binom{N-1}{2}+2$
Characteristic Path Length	N^2+N+1
Eccentricity, Radius, Diameter	N^2+N+1

5.2.2. Time Calculation: The number of clock cycles required to perform the calculation of every connectivity measurement can be found in Table 3, where N corresponds to the number of Channels and the $\binom{N-1}{2}$ is a function to obtain all the possible combinations for $N-1$ channels for 2 positions.

In Table 3, the measurement with the longer calculation time was the clustering coefficient. The reason for this is the extensive iterative processing done to calculate the geometric mean of all the possible triangles for a specific node. This triangle calculation could be performed in parallel if desired, instantiating as many modules as the number of nodes in the system. Nevertheless, without parallelization the design is still fast enough for real-time implementation, thus an increase in the speed is not necessary and will only increase the resources needed and the power consumption of the system. The total calculation time on MATLAB is 3.74ms, which is about 28 times higher than ours (131us). It is worth mentioning is that the time required to perform the degree is almost as long as the Clustering coefficient because the degree is calculated during the same process. Indeed, the degree is a simple measurement that could be obtained faster than the actual time, but that implies creating another instance of the PLI matrix and extra logic to perform the calculation independently from the clustering coefficient. Since there is no benefit in calculating the degree faster, in this work it was opted to keep the degree calculation in conjunction with the clustering coefficient.

5.3. Hardware savings

In this subsection, we compare the hardware savings achieved with our approximations compared to a one-to-one implementation of the algorithm. This is shown in Table 4 in terms of 2-input NAND gates. In this table we converted all the adders, multipliers, dividers, and comparators complexities in terms of 2-input NAND gates, providing an easier way to compare these elements since each of them may require a different number of logic gates for its implementation. It is important to mention that on this table, the optimized degree is considered zero since it is implemented inside the Clustering

Table 4. Comparison between the one-to-one implementation against the optimized proposed architecture in terms of 2-input NAND gates.

	NAND gates		
Module	Not Opt.	Opt.	Savings (%)
Degree	150	0	100
Degree (weigh.)	135	0	100
Density	2,002	1,296	35.26
Clust. coeff.	68,863	59,472	13.63
Trans.	69,013	43,081	37.57
Charact. path len.	15,548	11,676	24.90
Ecc., Rad. and Dia.	315	315	0
Total	156,026	115,840	25.75

Table 5. Synthesis results for the different modules of the system and the final design.

	Comb. ALUT	Reg.	Mem. Blocks (bits)	DSP Blocks	Freq. (MHz)
PLI					
PLI calculation system	22,398	95,406	40,408	192	215.05
Graph-theoretic Parameters					
Clust. coeff.	3,892	2,105	0	9	23.74
Trans.	1,873	86	0	0	564.33
Density	65	15	0	0	800
Charact. path length	4,517	2,002	0	4	61.39
Ecc., Rad. and Dia.	200	546	0	0	200.36
Graph-theoretic parameters system	11,835	6,060	0	17	22.16
PLI + Graph-theoretic parameters					
Final design	33,679	101,205	40,408	209	22.16

coefficient module. After all the optimizations the total saving comes to around 25% which translates in an equal reduction of area and power consumption of the final system compared to a one-to-one implementation.

5.4. Synthesis Results

After the functional test was performed and the system was performing as expected, the next step is to perform the synthesis of the system on an FPGA. The FPGA used for this system is the Stratix IV GX EP4SGX230K. The synthesis results for every module are shown in Table 5.

Table 6. Architectural complexity estimation using NAND gates.

Module	No. of NAND
Analytical sig.	409,374
Phase Calc.	92,160
PLI Calc.	318,960
PLI Unit (19 Ch. system)	820,494
Degree	0
Degree (weighted)	0
Density	1,296
Clust. coeff.	59,472
Trans.	43,081
Charact. path length	11,676
Ecc., Rad. and Dia.	315
Graph-theoretic parameters system	115,840
PLI + Graph-theoretic parameters	936,334

The module with the highest resource usage was the PLI calculation, using 61% of the available resources on the board. In comparison to this, the proposed architecture for the calculation of the graph-theoretic parameters requires 9% of the total resources available at the board. Nonetheless, the calculation of the graph-theoretic parameters is slower than the PLI calculation, achieving a maximum frequency of 22.16 MHz. However, it is a well-known fact that although FPGA implementations provide design flexibility, they are inherently not optimized for area, speed and power compared to specialized ASIC implementation. To get a better understanding of hardware complexity, we performed an estimation of the logic gates required for the design by converting all the adders, multipliers, dividers, and comparators complexities in terms of 2-input NAND gates. We did not implement the FFT module here but implemented the full PLI and graph-theoretic parameter extraction circuits. The reason for not implementing FFT is that, there are several optimized implementations of it already proposed over the decades and any could be adopted for the current purpose depending upon the area and power budget. Therefore, for this estimation, we selected the implementation presented in [23] due to its low area and power consumption. The result is shown in Table 6 in terms of 2-input NAND gate equivalent.

5.5. Power Consumption

One important point for this design was to create a low power system. To demonstrate the low power capabilities of the devices a power estimation was obtained using the software provided by the manufacturer of the FPGA. This software guarantees a high

level of accuracy when the software is provided with enough switching activity from the simulations. The switching activity was generated running the simulations previously mentioned in section 5.2. The EEG data is provided at the same sample rate at which it was acquired, and the frequency used in the system is 22.16 MHz which is the maximum frequency achieved. The dynamic power consumption of the system in this particular FPGA at a VCC voltage of 0.9V is 51.84mW. This total power consumption is important if this design is meant to be implemented as a portable continuous monitoring system for instance. Based on this power consumption and the current battery capacities in portable devices like mobile phones, with a capacity up to 4200 mAh [19], the proposed system could be energized continuously for days. It could be integrated with other systems for a further, more advanced processing of the signal as a standalone system. Despite this promising result, however, as mentioned earlier, it is a well-known fact that FPGAs are inherently not as efficient in terms of power consumption as an ASIC implementation where a design can be optimized from the point of view of power consumption. Therefore, we expect the system to show even better power performance when implemented in ASIC. To establish that, we performed an estimation of the power consumption in a 90nm TSMC CMOS technology using Synopsis Design Compiler. In this estimation, we used the power reported in [23], which is an FFT module implemented in the same CMOS technology. Our results are presented in Table 7. Based on the estimations, the power consumption of the complete system will be around 39.37 mW, which represents a power saving of around 24% from the 51.84 mW estimated for the FPGA implementation. Furthermore, it is important to note that FPGA uses a CMOS technology of 40nm, less than half the size of the library used for the estimations. Thus, if a similar technology to the FPGA were to be used for the ASIC implementation, further power reduction can be expected.

5.6. Comparison

In literature, only the authors from [11] have presented an architecture that performs a calculation of the graph connectivity measurements. The comparison between this and the proposed system is presented in Table 8.

There are two major differences between these two systems. The first one is the inclusion of the module to calculate the PLI calculation in the proposed system. In [11], the authors did not perform the calculation of the instantaneous phase used for the calculation of the PLI because it was assumed that these phases were given as an input. In the proposed system, the input will be an EEG signal instead of phase values, and therefore includes the PLI calculation that was avoided in [11]. The second major difference is the way the graph-theoretic parameters are calculated. The whole architecture was redesigned with the main goal to reduce the number of arithmetic operations in the calculation. This not only allows to reduce the size of the final system but also is greatly reflected in the power consumption of the system. By reducing the number of arithmetic operations used, the total energy of the system is decreased as

Table 7. Power consumption estimation for every module in the system.

Module	Power (mW)
Analytical sig.	6.932
Phase Calc.	2.522
PLI Calc.	28.800
PLI Unit (19 Ch. system)	38.256
Degree	0
Degree (weighted)	0
Density	0.009
Clust. coeff.	0.787
Trans.	0.148
Charact. path length	0.021
Ecc., Rad. and Dia.	0.149
Graph-theoretic parameters system	1.116
PLI + Graph-theoretic parameters	39.372

Table 8. Comparison between the design presented in [11] and the design presented here.

	Error		Adders		Mult. & Div.		Clock Cycles	
	[11]	This work	[11]	This work	[11]	This work	[11]	This work
Degree	0	$O(2^{-7})$	8	0	0	0	88	169
Degree (weighted)	0	O(2−9)	64	0	0	0	88	169
Density	$O(2^{-8})$	$O(2^{-9})$	102	1	0	1	142	170
Clustering Coefficient	0	$O(2^{-14})$	896	5	1,024	5	87	175
Transitivity	$O(2^{-10})$	$O(2^{-15})$	903	3	1,024	2	88	174
Characteristic path length	$O(2^{-8})$	$O(2^{-9})$	21	3	0	1	157	73
Eccentricity, Radius, Diameter	0	$O(2^{-9})$	37	0	0	0	168	73

well. The reduction in resources used can be estimated by comparing the number of logic 2-input NAND gates that both architectures use. The number of gates reported in [11] is equal to 1,116K while in this work the estimate for the complete system came to around 937K (Table 6), about 16% less. Nevertheless, the architecture of [11] does not include the 19 channel PLI calculation and it is only doing the graph-theoretic parameter calculation. Thus, if we compare only the graph-theoretic parameter calculation of this work against [11], the 2-input NAND gates required for this architecture will be around 116K, representing a reduction of about 89% in the total 2-input NAND gates used. Additionally, even though is not possible to compare the power between both architectures since the CMOS technology is different(130 nm) from ours(90 nm), it is

expected that the reduction on logic elements previously discussed will have a similar impact in the total power consumption of the system. Even though there was a reduction in the number of arithmetic elements used for the operations, the proposed system is still able to perform the calculation in a similar time to that of the system presented in [11]. The difference in the calculation time between these two is seven clock cycles only. At the maximum frequency achieved, the total time required to calculate the graph-theoretic measurements in this design is equal to 7.89us compared to the 6.7us reported in [11]. Although the calculation time is longer in this design, this is a minor tradeoff that is compensated with fewer logic components and energy consumption.

6. Discussion

The connectivity method proposed here is at scalp level, however a source space analysis is also possible. By analyzing the signal at the source space, it is possible to overcome the field spread or volume conduction problem that affects many connectivity measurements [24]. Nevertheless, such analysis would require very high computational burden to perform the source localization step from EEG as this is an inverse problem. We are currently at a stage of exploring the possibilities of developing a low-computational method for implementing this stage which will be presented in our future work.

Having said that, in this particular work we used PLI as the main method for the connectivity analysis since it is a measurement that was created to overcome the confounding problem of volume conduction found in other connectivity measurements such as Phase Locking Value (PLV). Therefore, the architecture presented here is expected to be less affected by this volume conduction problem and should represent true interactions between the different regions of the brain.

Moreover, as it was previously mentioned, this process cannot be started before the necessary preprocessing of the signal. The connectivity analysis can only be applied after the EEG is clean of artifacts and separated into different frequency bands (e.g. delta, theta, alpha, beta, gamma) – as we are interested in band-specific functional connectivity characterisation. While the signal splitting into different bands can be done with standard filtering approach, an appropriate way to separate the artifacts needs to be investigated from the hardware implementation point of view, under the constraints of real-time computation and low energy consumption requirements, as we did in this work.

7. Conclusions

In this work, a new architecture was presented that is capable to extract the PLI and the graph connectivity measurements from an EEG signal. For validation, a 19-channel version of the architecture was synthesized. The synthesized design requires a total of 71% of the total resources of the FPGA and can operate at a maximum frequency of 22.16 MHz. The unused resources could be used then to extend the number of channels

or include further processing of the graph connectivity measurements, according to the application needs. The power consumption of the system was equal to 51.84mW. The validation results show that the relative error percentage is low for all the measurements, with the greatest error being the degree within the range of +-7.9%. Additionally, the total time required for the calculation of all the graph connectivity measurements was equal to 131us in our design, accelerating the calculation by at least one order of magnitude compared with the time required to perform the same calculation on MATLAB (3.74ms).

The major application envisioned for this design is in neurofeedback systems. These systems require continuous monitoring of the brain and fast processing of the EEG signal to provide the correct stimulus at the appropriate time, and therefore a system like the one proposed in this works suits it perfectly. Additionally, since the whole system can be implemented on an FPGA it is not necessary to have an external processing unit, which will remove the need to have complicated setups. This could be beneficial for neurofeedback systems used in rehabilitation, allowing the patient to perform the rehabilitation sessions at home, saving time in transportation to the rehabilitation center and increasing the number of sessions that the patient can have, reducing the time required for rehabilitation. Moreover, this system can be combined with portable EEG acquisition systems and allow continuous monitoring of the brain activity on the go, bringing the possibility for new applications.

Future work for this design will be its implementation on an ASIC. This will reduce the total size of the final system and will improve the operation frequency and power consumption. This translates into a more efficient and smaller system that can be easily set up with longer battery life.

Acknowledgments

This work was partly funded by the scholarship program of the Mexican National Council for Science and Technology (CONACYT)

References

[1] Douw L, De Groot M, Van Dellen E, Heimans J J, Ronner H E, Stam C J and Reijneveld J C 2010 *PLoS One* **5**

[2] Sargolzaei S, Cabrerizo M, Goryawala M, Eddin A S and Adjouadi M 2015 *Computers in biology and medicine* **56** 158–166

[3] Stam C, Van Der Made Y, Pijnenburg Y and Scheltens P 2003 *Acta Neurologica Scandinavica* **108** 90–96

[4] Yu M, Gouw A A, Hillebrand A, Tijms B M, Stam C J, van Straaten E C and Pijnenburg Y A 2016 *Neurobiology of aging* **42** 150–162

[5] Stoffers D, Bosboom J, Deijen J, Wolters E C, Stam C J and Berendse H W 2008 *Neuroimage* **41** 212–222

[6] Utianski R L, Caviness J N, van Straaten E C, Beach T G, Dugger B N, Shill H A, Driver-Dunckley E D, Sabbagh M N, Mehta S, Adler C H *et al.* 2016 *Clinical Neurophysiology* **127** 2228–2236

[7] Jamal W, Das S, Oprescu I A, Maharatna K, Apicella F and Sicca F 2014 *Journal of neural engineering* **11** 046019

[8] Ahmadlou M and Adeli H 2017 *Neuroscience letters* **650** 103–108

[9] Maharatna K, Mazomenos E B, Morgan J and Bonfiglio S 2012 Towards the development of next-generation remote healthcare system: Some practical considerations *2012 IEEE International Symposium on Circuits and Systems* (IEEE) pp 1–4

[10] Nuno R A G and Maharatna K 2019 A phase lag index hardware calculation for real-time electroencephalography studies *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)* (IEEE) pp 644–647

[11] Pal C, Biswas D, Maharatna K and Chakrabarti A 2017 Architecture for complex network measures of brain connectivity *2017 IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE) pp 1–4

[12] Hardstone R, Poil S S, Schiavone G, Jansen R, Nikulin V V, Mansvelder H D and Linkenkaer-Hansen K 2012 *Frontiers in physiology* **3** 450

[13] Rubinov M and Sporns O 2010 *Neuroimage* **52** 1059–1069

[14] Niso G, Bruña R, Pereda E, Gutiérrez R, Bajo R, Maestú F and del Pozo F 2013 *Neuroinformatics* **11** 405–434

[15] Engels M M, Stam C J, van der Flier W M, Scheltens P, de Waal H and van Straaten E C 2015 *BMC neurology* **15** 145

[16] Stam C J, Nolte G and Daffertshofer A 2007 *Human brain mapping* **28** 1178–1193

[17] Watts D J and Strogatz S H 1998 *nature* **393** 440

[18] Dijkstra E W *et al.* 1959 *Numerische mathematik* **1** 269–271

[19] Liang Y, Zhao C Z, Yuan H, Chen Y, Zhang W, Huang J Q, Yu D, Liu Y, Titirici M M, Chueh Y L *et al.* 2019 *InfoMat* **1** 6–32

[20] Pineiro A, Bruguera J D, Lamberti F and Montuschi P 2008 *IEEE Transactions on Computers* **57** 562–566

[21] Bruguera J D *et al.* 2011 Composite iterative algorithm and architecture for q-th root calculation *2011 IEEE 20th Symposium on Computer Arithmetic* (IEEE) pp 52–61

[22] Shoeb A H 2009 *Application of machine learning to epileptic seizure onset detection and treatment* Ph.D. thesis Massachusetts Institute of Technology

[23] Yu C and Yen M H 2014 *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **23** 1793–1800

[24] Schoffelen J M and Gross J 2009 *Human brain mapping* **30** 1857–1865