# Design and soc implementation of a low cost smart home energy management system

Trung Kien Nguyen

# UNIVERSITÉ NICE SOPHIA ANTIPOLIS
# POLYTECH'NICE-SOPHIA

## École Doctorale des Sciences et Technologies de l'Information et de la Communication

*Electronique pour Objets Connectés*

# THESE

Pour obtenir le titre de
**Docteur en Sciences spécialité Electronique**
de l'Université Nice Sophia Antipolis

présentée et soutenue par

# Kien Trung NGUYEN

## Conception et réalisation d'un système de gestion intelligente de la consommation électrique domestique

Thèse dirigée par Gilles JACQUEMOD
Soutenance prévue le 11 Décembre 2015

**Jury :**

| | | |
|---|---|---|
| **P. GARDA** | Rapporteur | Professeur, Université Pierre et Marie Curie Paris |
| **S. WEBER** | Rapporteur | Professeur, Université de Lorraine Nancy |
| **G. JACQUEMOD** | Directeur | Professeur, UNS Sophia Antipolis |
| **E. DEKNEUVEL** | Co-Encadrant | Maître de Conférences, UNS Sophia Antipolis |
| **L. HEBRARD** | Examinateur | Professeur, Université de Strasbourg |
| **B. NICOLLE** | Examinateur | Ingénieur, Qualisteo Nice |
| **O. PARSON** | Examinateur | Associate Professor, University of Southampton |

# Table of Contents

# Acronyms, abbreviations and definitions

| Term | Description |
|---|---|
| ADC | Analog to Digital Converter |
| AMS | Analog Mixed Signal |
| API | Application Program Interface |
| ASIC | Application Specific Integrated Circuit |
| CLIP | Component-Level IP |
| CPU | Control Process Unit |
| CUSUM | CUmulative SUM |
| DFD | Data-Flow Diagram |
| DMA | Direct Memory Access |
| DSP | Digital Signal Processing |
| EPC | Energy Performance Certificate |
| FIR | Finite Impulse Response |
| FFT | Fast Fourier Transform |
| FIFO | First In, First Out |
| FPGA | Field Programming Gate Array |
| FSM | Finite State Machine |
| GA | Genetic Algorithm |
| GUI | Graphical User Interface |
| HDL | Hardware Description Language |
| HLS | High Level Synthesis |
| HMM | Hidden Markov Model |
| HW SW | HardWare SoftWare |
| HVAC | Heating, Ventilation and Air Conditioning |
| HID | High Intensity Discharge |
| IIR | Infinite Impulse Response |
| IP | Intellectual Properties |
| KPN | Kahn Process Network |
| LUT | Look Up Table |
| MCU | Micro Controller Unit |
| MoC | Model of Computation |
| NIALM | Non-Intrusive Appliance Load Monitoring |
| NoC | Network on Chip |
| NRE | Non-Recurring Engineering |

| | |
|---|---|
| OCED | Organization for Economic Cooperation and Development |
| OVP | Open Virtual Platform |
| PRA | Pyramid Recursive Algorithm |
| REDD | Reference Energy Disaggregation Data set |
| RMS | Root Mean Square |
| RPN | Reactive Process Network |
| RTL | Register-Transfer Level |
| SDF | Synchronous Dataflow |
| SoC | System on Chip |
| THD | Total Harmonic Distortion |
| TLM | Transaction-Level Model |
| UML | Unified Modeling Language |
| USOM | USer Operating Mode |
| VAR | Volt-Ampere Reactive |
| VFD | Variable Frequency Driver |
| VLSI | Very Large Scale Integration circuits |
| XSG | Xilinx System Generator |

# CHAPTER 1. INTRODUCTION

**Contents**




**Abstract:**

*This chapter is the useful background information of the Non-Intrusive Appliance Load Monitoring (NIALM) technology and the motivation of this research. Section 1.1 presents a big image about the relation between energy usage of human and environmental pollution. This section also talks about smart meters, which can monitor and give people more information about their energy usage to engage them in saving energy. Section 1.2 gives a brief introduction about the NIALM technology, a state of the art, and its daily area in life. Section 1.3 introduces some electrical signatures, which a NIALM system must extract, analyze to recognize and monitor power usage of appliances in an electrical network. Section 1.4 discusses about the trend of NIALM technology to be able to solve its remain challenges. Thus, section 1.5 presents our proposal for an innovative real-time NIALM system based on System on Chip (SoC) and the contributions of this thesis. Finally, section 1.6 will present the organization of this thesis.*

# Chapter 1. Introduction

## 1.1. Motivation

We are today's fully aware of the global climate change, the global warming mainly coming from the carbon dioxide ($CO_2$) emission through the urbanization of human. Too much $CO_2$ causes natural disasters such as drought, flooding, tsunami, hurricane, diseases that destroy the earth and human life. Even though more and more developed countries have planned to use green technologies, renewable energy sources and better energy usage monitoring, most of developing countries accept air pollution for economic reasons. In Figure 1.1, the $CO_2$ emission in fossil fuel energy almost remains from 1990 to now in Organization for Economic Cooperation and Development (OCED) countries. However, this value increases three times in non-OCED countries and mainly in coal. The annual energy outlook 2013 of US Energy Information Administration [1] also shows that from 2010 to 2040, coal-fired power plants are still generating about 40% global electrical power and emit over 40% of $CO_2$ in the earth and these $CO_2$ emissions are still mainly linked to the use of energy of human. The amount of $CO_2$ emission are also shown in the greenhouse gas equivalencies calculator tool of the Environmental Protection Agency of United States [2] that 15.873 MWh electricity consumption – the average energy usage in house for one year, is equivalent up to 10 tons $CO_2$ emission to the environment. Consequently, changing people's awareness of using energy effectively and economically becomes very urgent.

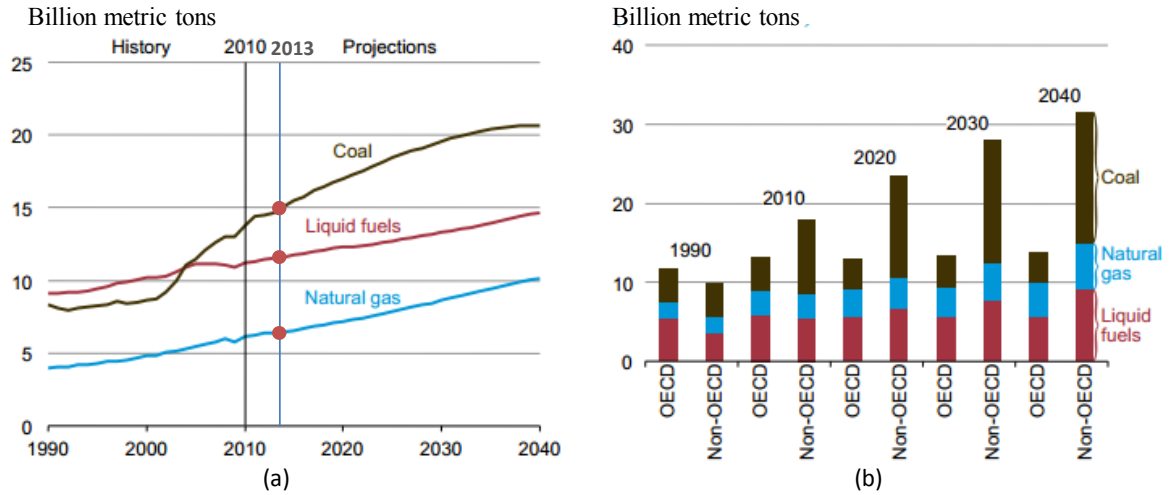
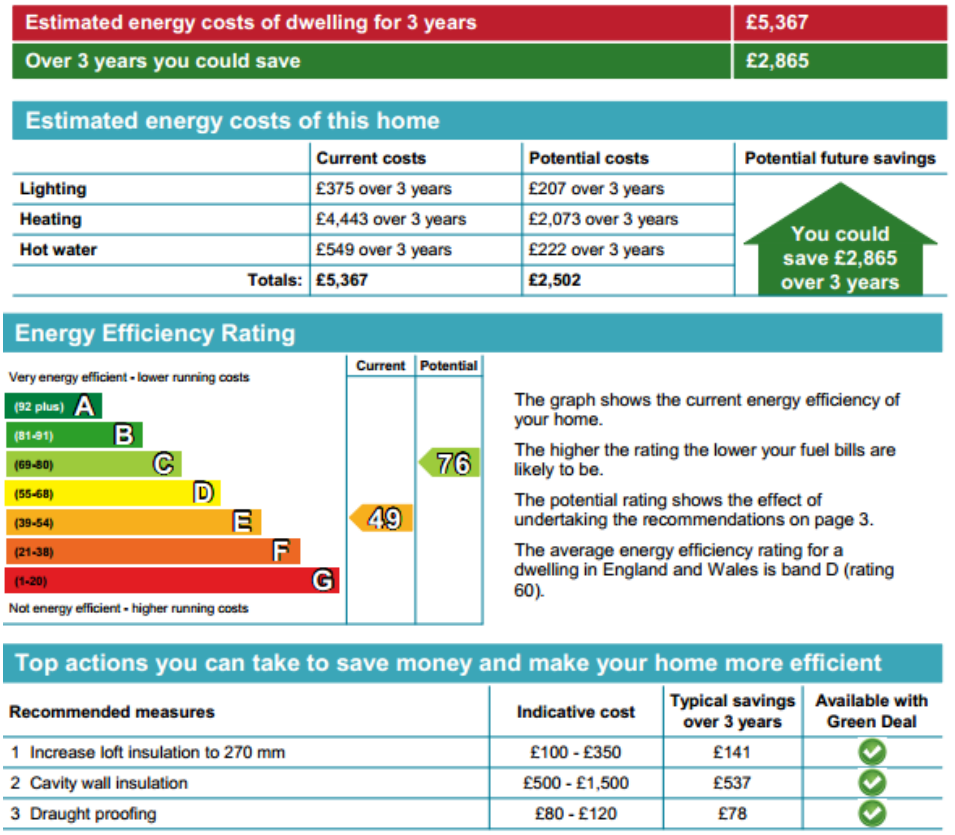



Figure 1.1 (a) World energy-related carbon dioxide emissions by fuel type, (b) OECD and non-OECD energy-related carbon dioxide emissions by fuel type [1]

Although European (EU) countries are leading in using renewable energy sources. Their objective aims to replace 20% of EU's energy consumption by using renewable energy sources in 2020 [3]. Thus, economic energy usage management is still vital in many decades and people want to decrease environmental pollution. In 2007, England introduced the Energy Performance Certificate (EPC), which contains the rating of energy efficiency of a house and recommendations for potential improvements. As illustrated in Figure 1.2, the EPC gives customers the energy consumption level of the house and the potential future saving money, if

they follow such recommendations. With better information in monthly electricity bills, the EPC can engage people to reduce the energy usage in the house.



| Estimated energy costs of dwelling for 3 years | £5,367 |
|---|---|
| Over 3 years you could save | £2,865 |

**Estimated energy costs of this home**

| | Current costs | Potential costs | Potential future savings |
|---|---|---|---|
| Lighting | £375 over 3 years | £207 over 3 years | You could save £2,865 over 3 years |
| Heating | £4,443 over 3 years | £2,073 over 3 years | |
| Hot water | £549 over 3 years | £222 over 3 years | |
| Totals: | £5,367 | £2,502 | |



**Top actions you can take to save money and make your home more efficient**

| Recommended measures | Indicative cost | Typical savings over 3 years | Available with Green Deal |
|---|---|---|---|
| 1 Increase loft insulation to 270 mm | £100 - £350 | £141 | ✓ |
| 2 Cavity wall insulation | £500 - £1,500 | £537 | ✓ |
| 3 Draught proofing | £80 - £120 | £78 | ✓ |

Figure 1.2 Some information in an EPC sample [4]

However, EPC has impact on only 18% of people and 17% of people follow its recommendations [4]. It is probably due to the recommendations are too general which recommend them to repair the house or to replace better appliances. They are not convenient for most of the customers who just need to know the relation between the increasing on the electrical bills and their daily or weekly activities. That is why smart meters, the emerging technology to help people to know their monthly energy consumption, are gradually replacing mechanical power meters nowadays. Some new modern smart meters even offer real-time feedback on the in-home display with detailed daily records of activities of customers in appliances and their effect on the total energy consumption as well as the ratio of energy usage in individual appliance. This individual appliance-monitoring feature at the heart of those smart meters gives customers many advanced benefits:

- Real-time feedback about the effect of their behavior in energy consumption. For example, the water heater is, sometimes, still turned-on while no one is taking a bath or the air conditioning in the living room is still turned-on while there is no one there.
- Awareness of energy performance of their appliance in comparison to the standard energy-saving products. Some customers will recognize that after three years, they have to pay the same money (including buying cost and yearly bills) for the new modern

energy-saving refrigerator and the old one while it is certain that using a new modern refrigerator is much more comfortable and convenient.

- Suggestions about energy usage based on the Time-of-Day Electricity Pricing. Consumers will be indicated about variations of electricity prices during a day, and they can then keep using the same amount of energy while saving money by using it at the right time.
- Estimating the equivalent $CO_2$ emission of energy consumption can engage consumers in changing their energy usage behaviors for environmental protection.

## 1.2. NIALM Technology and Applications

### 1.2.1 Introduction

There are two solutions for monitor consumption on individual appliances: using one sensor per appliance as in a conventional approach or using only one sensor to monitor the whole loads as in the Non-Intrusive Appliance Load Monitoring (NIALM) approach. Intrusive sensors work similarly to a normal current meter that needs to be connected in serial with the load to measure its current. In contrast, non-intrusive current sensors can convert the magnetic field around the electrical wire into analog voltages without intrusive connections. Such sensors are more advanced than intrusive sensors because of a very fast, safety and flexible installation.

As illustrated in

Figure 1.3, NIALM technology introduced by Hart in 1992 [5] aims to use only one non-intrusive sensor to monitor all appliances in the electrical network. Because of using fewer hardware components, NIALM has lower cost because it is easier to setup and maintenance than conventional method. Unfortunately, NIALM requires more complex algorithms to analysis, classify, breakdown and assign power consumption to appropriate appliances while intrusive method is able to define exactly from where the power usage events come. However, intrusive method will be very expensive and it has to use complex communication network to transfer data that limits the bandwidth if there are too many nodes in the network. NIALM method, which has only one node, can use full bandwidth of communication channel. The limited number of appliances, which can be detected and monitored, depends only to the efficiency of NIALM algorithms. Some NIALMs can also do many computations at sensing nodes for examples computing power consumption, detecting transitions to decrease data size in communication channels.

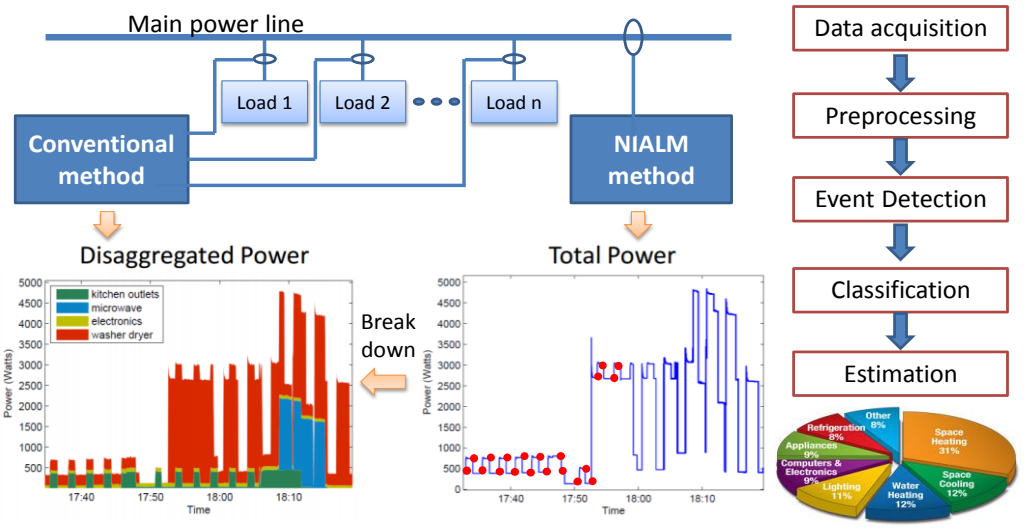



Figure 1.3 Conventional and NIALM method in power monitoring application

In order to be able to monitor consumption on individual appliances in an electrical network using only one sensing node, NIALM systems must have five basic processes:

- **Data acquisition:** NIALM system collects values from current and voltage sensors in a defined sampling frequency.
- **Preprocessing:** This is an important step to handle in noise filtering and electrical features extracting such as total real power, re-active power and apparent power. It can contain some advanced tasks such as calculating the phase of electrical signal, harmonics data and power factor. More electrical features the processing extracts, more accuracy the step classification is.
- **Event Detection** works on the change detection in aggregated current or power to detect an ON-turning or OFF-turning from an appliance in the electrical network. This step also extracts the transition signatures after a detected event.
- **Classification clusters** events after Event Detection step and matches ON events to OFF events to classify appliances.
- **Estimation** is the last process to summarize the total power consumption and ratios power consumption in each appliance.

### 1.2.2. State of the art

In 1992, Hart [5] introduced the topic NIALM which can monitor a number of switches ON and OFF from appliances. His algorithms based on 1-second sampling average power and Root Mean Square (RMS) voltage was able to compute the normalized real and reactive power. The normalized powers were used by the edge detection algorithm to find out the time of the change in normalized power. A fixed 15 Watts tolerance was used to define steady periods and transition periods to create the time-stamped step-change p-vectors. These p-vectors are clustered to find out the quantity of clusters and their centroids to be able to pair the ON-OFF or Finite State Machine (FSM for multi-state appliances) models of appliance using the rule Zero Loop-Sum

Constraint. Last function is to set the name of the model of appliance for the system able to track out in the future. However, his algorithm has still need to be improved to work well with under 100 W appliance, continuously variables, and multi-state appliances. Moreover, it cannot distinguish appliances who have the similar real power and reactive power.

Leeb et al. 1995 [6] used higher sampling frequency to extract power information of third harmonic for classification. Leeb also used three sampling rate levels to detect both fast, medium and slow transitions. However, the classification cannot process in real-time and both overlapped transitions and continuous variable appliances are still intractable. Three sampling rate levels in detection are not enough for all appliances in commercial and industrial application so that Laughman [7] proposed the use of shape information to detect transitions. However, his system is complex and needs a tedious training with all kinds of appliance that it should be able to work with; and his method can still not trace overlapped events.

Baranski et al. in 2004 [8] presented a new approach for NIALM system. His algorithm also uses power data sampled in 1-second rate to detect events by considering the change between two sequential active powers with a threshold value and the direction of the change. That means two changes in the same increasing direction belong to the same event. Then, he used Fuzzy-clustering algorithm to cluster all the detected events. His paper clearly presents Genetic Algorithm (GA) to find out both ON-OFF and FSM appliances up to five different states. However, his algorithm limits the length of detected event sequence to be processed in the limitation in maximum process time.

Instead of using both current sensors and voltage sensors in standard NIALM system, Cox et al. proposed, in 2006 [9], a NIALM system to identify the operation appliances using only the distortion information of total voltage. Removing the need of a current sensor gives a more economical NIALM system solution. The preprocessor will estimate the coefficients information of the input voltage in some order harmonics. Then software will analyze each array of spectral envelopes to detect the transitions. Cox said that system now could detect and identify ON-OFF appliances with small power consumption from 50W. Moreover, his result also shows the ability to track the FSM type appliances. Although his work does not use power measurement, tracking the appliance using only voltage sensor can cooperate with all kind of current power meter to give more energy consumption information for customer in the electricity bill. However, Cox needs to evaluate the accuracy of his system in all household appliances.

Suzuki et al. in 2008 [10] used integer programming approach, based on current waveform in one period, to classify appliance, then to disaggregate appliances. No complex preprocessing algorithms and steady based NIALM are advantages of his algorithm. This algorithm also works with FSM appliances, which makes one more integer equation for each appliance. However, it needs to have a database of appliance waveform and quite sensitive with unknown noises.

Patel et al. in 2009 [11] and Froehlich et al. in 2010 [12] used only single plug-in sensor to detect events of some home appliances from electrical noise in power line in operation of appliances. They found that there are transition noises and continuous noises. The transition

noise lasts only some microseconds and comes from switching mechanism and load characteristics of appliances while continuous noises come from the internal switching mechanism. He developed the power-line interface hardware to filter and collect a standard 60 Hz AC power signal, 100 to 100 kHz filtered signals and 50 kHz to 100 MHz filtered signals. After that, a USB oscilloscope interface (EBest 2000) was used to amplify signals, convert them to digital data, and send data to a computer in sampling frequency up to 100 MHz. One shifting window FFT algorithm will extract harmonic information for classification using Support Vector Machine (SVM) algorithm. However, his work focuses only on the domestic environment especially with a weak noisy outside environment. Other limit of their works is the dependence of noises onto length and type of electrical wires so that the training databases for the same appliances in different houses are different. Moreover, this method cannot be used in real-time with complex Digital Signal Processing (DSP) card, computer and pre-trained database. The system has not been tested for slow transitions and continuously variable appliances.

Many current researches [13] [14] focused on exploring new disaggregation algorithm on 1 second sampling aggregated power consumption using algorithm Hidden Markov Model (HMM). HMM algorithm based NIALM is developed to work with most of current power meters and quite high accuracy. However, slow sampling rate can cause losing the “peak” states sometimes, which can be an important signature to distinguish same power appliances. Moreover, training phases can meet problem if there are new appliances, which do not exist, in the pre-trained database.

The main objective of most of current research in NIALM is to improve the accuracy of this technology by exploring more complicate disaggregation algorithm. However, such algorithms need a powerful computer such as a server to process. Most of researches also use supervised learning or well-known database to classify appliances. Therefore, some challenges are still valid to solve in NIALM are:

- The need to extract more electrical signatures to distinguish similarity appliances.
- The problem of variable load appliances which creates slow transition that is not detected by most of event detect algorithms.
- A low sampling frequency data can meet problems with overlap transitions and can lead to missing event.
- There are still not any effective methods to detect multi-state load appliances, which can have many states in their operation.
- The NIALM system should be low cost, compact and similar price as current power meter. The low cost system can help NIALM to gain popularity in every home in the finance support program of the electricity company or the government.

### 1.2.3. Applications of NIALM

#### Energy monitoring

People applied NIALM technology in many fields of the life. The first application of NIALM is home or building energy monitoring. Mario E. Berges et al. 2010 [15] discussed about four types of existing commercial NIALM products: the whole-house meter, the smart meter ARM/AMI, the plug-load meters and the smart home packaged solution. Products of TED company with the Energy Detective Footprints software can track only five home appliances after training their power consumption [16]. Products of Enetics company in USA record over Internet detail information of energy usage such as time, current, voltage, power, power factor, Total Harmonic Distortion (THD) [17]. Then they provide energy usage analysis based on this information in both residential and industrial sites but their system does not provide real-time monitoring feature.

#### Gas and water monitoring

Gas and water monitoring is another application of NIALM which is often integrated to power monitoring activity to give customer a complete energy monitoring. In an extra part of the article [12] in 2010, Jon Froehlich et al. applied a single point sensor approach to develop HydroSense system to monitor water usage and the GasSense system to monitor gas usage in house. There are already real products to monitor gas usage in at home for example Loop Gas product of Navetas company in UK [18].

#### System fault detection and maintenance

System fault detection and maintenance may be the most interesting capability of NIALM technology. The research [19] used NIALM technology to develop a Condition Base Maintenance system in a US Navy Ship in 2005. Because of the difficulty to connect sensors for all the electrical devices in an exist system in the ship, NIALM was used to trace out and monitor the operation of all devices in the system in order to detect and to predict failures in their working. This application is very useful when analyzing all components inside a complex machine is a very hard work. Moreover, we can also apply this capability to home energy monitoring to early detect errors in electrical appliances for the maintenance.

#### Commercial and Industrial domain

However, most researches in the past worked in residential site and there are not adequate researches about using NIALM in commercial and industrial sites. Most of commercial NIALM products are applying in home energy monitoring. The lack of applying NIALM in commercial and industrial sites may come from below reasons. First, in commercial and industrial sites, machine can contains many basic electrical appliances inside so that it is very difficult to recognize what real machines are working. For instance: an industrial oven can contain a magnetron tube that generates heat in di-electric heater type, one or more fans, power controller and time controller, lamps, tri-ac driver circuit and high voltage transformer. Second, many

kinds of noise in industrial site coming from high power motors, fans or inverters can generate errors in current detection algorithm of NIALM. Third, monitoring all basic appliances inside machine in commercial and industrial sites is very difficult so that it takes a lot of time and money to build database for all new coming machines. A NIALM system may be impossible to analyze and disaggregate all electrical machines in an industrial factory or one NIALM system cannot monitor all apartments in a big building. However, in the supporting of other technology for example wireless sensor network, which each NIALM system is a monitoring node, this technology can be used more broadly in commercial and industrial site.

## 1.3. Electrical Signatures

Electrical signatures are characteristic information of electrical appliances such as current, power, power factor, harmonic etc., which can be used to distinguish appliances. As illustrated in Figure 1.4, such signatures can be classified into two main categories that affects the event detection and algorithms in NIALM: *Steady-state signatures* and *Transition-state signatures*. Steady-state signatures are steady-features, which can be derived under the steady-state operation of the appliances, or in other words, steady-state signatures are the changes in interested features in electrical network after ON-turning and OFF-turning occurrences. In contrast, a *transition-state* is the duration between two *steady states*. The differences in structures between appliances cause differences in some electrical features in transitions-state that can be used to classify them.

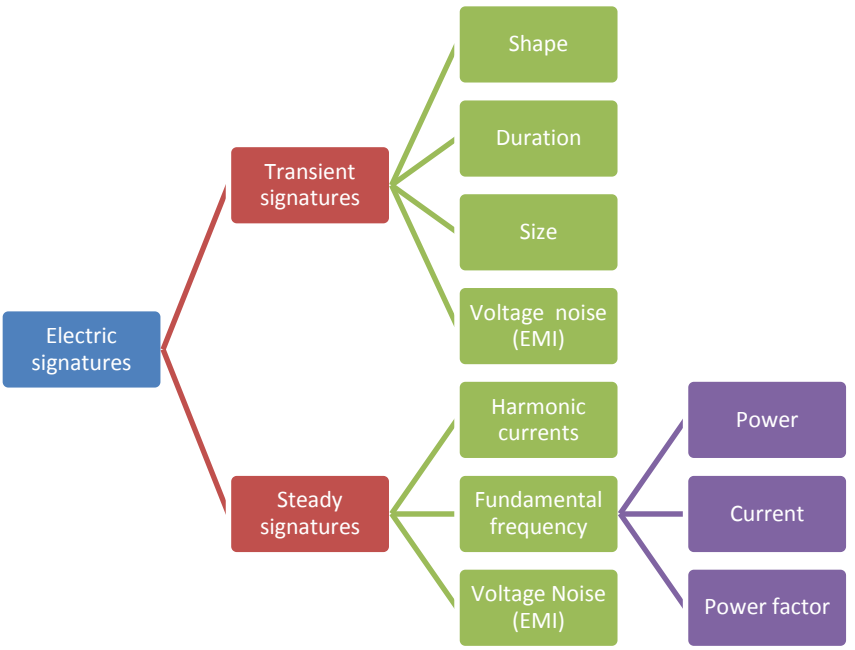



Figure 1.4 Electrical signatures for classification

### 1.3.1. Parallel RLC model

Before analyzing in detail about the steady states and transition states, we present the theory of RLC architecture of electrical appliances, which can explain the difference in electrical signatures between appliances. In theory, an electrical circuit composed by basic elements resistor R, inductor L and capacitor C can represent appliances. Then, this circuit can be simplified to an equivalent parallel RLC circuit as illustrated in Figure 1.5. In this model, power information of appliance then can be computed by traditional equations. $V_S$, the root mean square (rms) of AC voltage, are often used to express voltage of electrical network connected to the RLC circuit. Three elements R, L, C of the circuit generate three kinds of power consumption. In this circuit, $I_R$, $I_L$, $I_C$ are currents flowing in R, L and C branch; and $I_S$ is the total current in parallel RLC circuit.

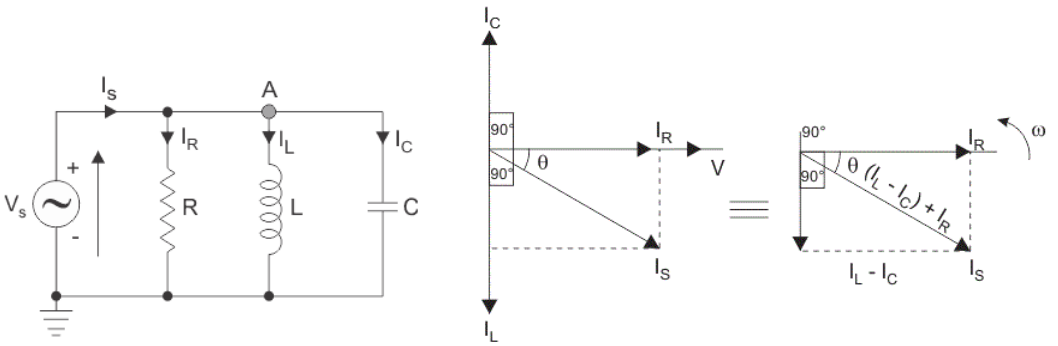



Figure 1.5 Parallel RLC model of an appliance

Real Power P, measured in Watt, is a function of circuit's dissipative element resistance R.

$$P = V_s . I_R \quad (1.1)$$

Reactive power Q, measured in Volt-Ampere-Reactive (VAR), is a function of circuit's reactance X.

$$Q = |V_s . I_L - V_s . I_C| = |Q_L - Q_C| \quad (1.2)$$

Apparent power S measured in Volt-Ampere (VA) is a function of circuit total impedance.

$$S = V_s . I_S \quad (1.3)$$

Applying the Kirchhoff's current law in RLC parallel circuit, we notice that currents in each branch of the circuit are not in the same phase as illustrated in Figure 1.5. Current flowing in resistor R branch is in same phase with voltage. However, current flowing in inductor L branch is later 90 degrees than the voltage and current flowing in capacitor C branch is earlier 90 degrees than the voltage. Apply the Pythagoras's theorem; we have the equation to compute aggregated current.

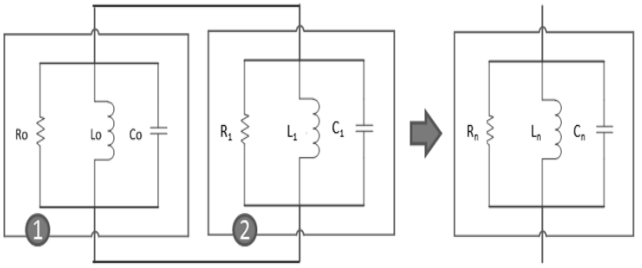
$$I_s = \sqrt[2]{I_R^2 + (I_L - I_C)^2} \quad (1.4)$$

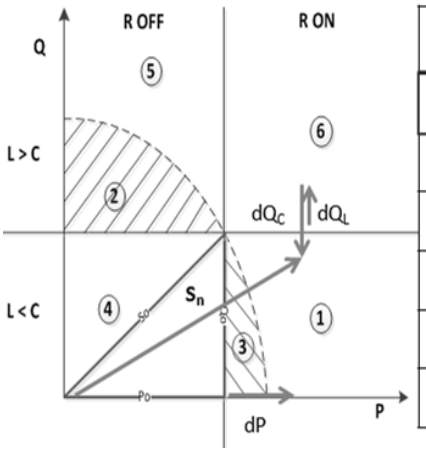
Replace (1.4) to (1.3), we get

$$S = V_s.I_S = V_s.\sqrt[2]{I_R^2 + (I_L - I_C)^2} = \sqrt[2]{V_s^2 I_R^2 + V_s^2 (I_L - I_C)^2}$$

Or $$S = \sqrt[2]{P^2 + Q^2} \qquad (1.5)$$

When a new electrical appliance is turned on, its represented RLC circuit is connected to the electrical network to create a new circuit in the electrical network, which contains new parallel RnLnCn electrical circuit with new power information Qn, Pn, Sn as illustrated in Figure 1.6. Based on the relation between represented RLC of the new electrical appliance with differences in power Qn, Pn, Sn from the point ($P_0$, $Q_0$) of working appliance 1, there are six types of event with three turning on events and three turning off events as illustrated in Figure 1.6. Moreover, events ON can be matched with events OFF in five pairs: (ON-1, OFF-2), (ON-1, OFF-5), (ON-3, OFF-2), (ON-3, OFF-5) for electrical appliances that have $X_{Ln}$ smaller than $X_{Cn}$, and pair (ON-6, OFF-4) for electrical appliances that have $X_{Ln}$ larger than $X_{Cn}$.




| Type | Events | R event (dP) | LC event (dQ) |
|---|---|---|---|
| 1 | dS > 0, dP> 0, dQ < 0 | ON | $XC_{ON} > XL_{ON}$ |
| 2 | dS < 0, dP< 0, dQ > 0 | OFF | $XC_{OFF} > XL_{OFF}$ |
| 3 | dS < 0, dP> 0, dQ < 0 | ON | $XC_{ON} > XL_{ON}$ |
| 4 | dS < 0, dP< 0, dQ < 0 | OFF | $XC_{OFF} < XL_{OFF}$ |
| 5 | dS > 0, dP< 0, dQ > 0 | OFF | $XC_{OFF} > XL_{OFF}$ |
| 6 | dS > 0, dP> 0, dQ > 0 | ON | $XC_{ON} < XL_{ON}$ |

Figure 1.6 (a) Parallel RLC model of electrical network after turning on the second appliance; (b) Six types of appliance based on power change dS, dP, dQ = $dQ_L$ - $dQ_C$

## 1.3.2. Steady-state signatures

### Power change

Most common examples of steady state signatures are the changes of real power (ΔP) and reactive power (ΔQ) after a changing status in appliance. Some appliances having only two

states ON and OFF are called ON-OFF appliances. Some other appliances having many states in their operation are called multi-state appliances or finite-state appliances. As shown in Figure 1.7, the difficulty in classifying finite-state appliances is that such appliance can work as either an ON-OFF appliance or a multi-state appliance depending on its operation mode. Another complex finite-state appliance example is a fan with three levels of the speed forms a 4-states appliance. However, this appliance has many complete operation modes by many different combinations of transition powers.

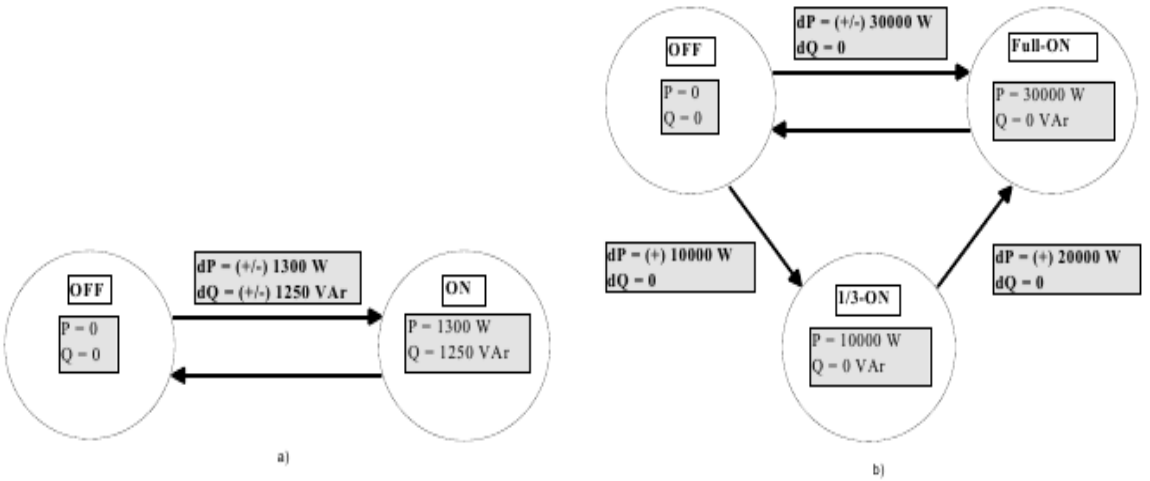



Figure 1.7 (a) ON-OFF appliance water pump and (b) 3-states appliance water boiler

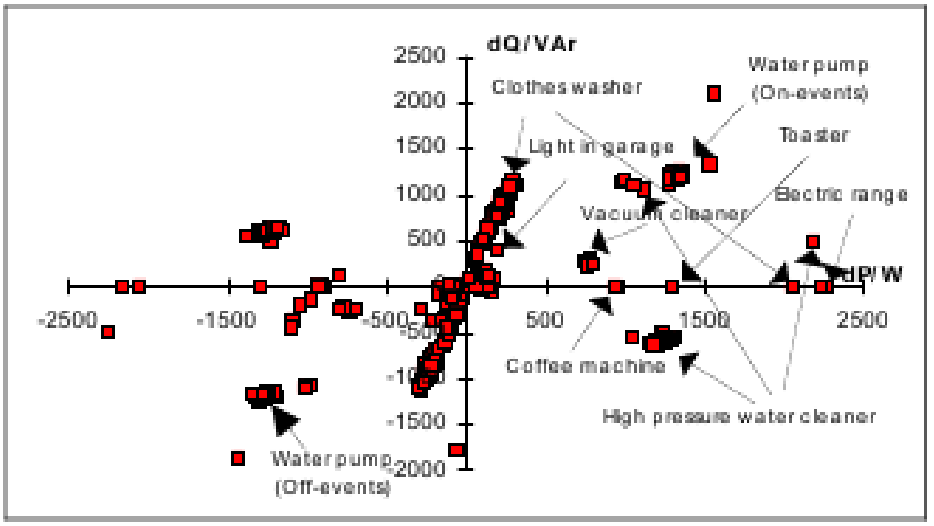



Figure 1.8 ΔP-ΔQ plan of some house appliances

Some NIALM algorithms only need the change in real power to disaggregate total power and classify appliance. Parson et al. [13] and Kolter et al. [14] used only the real power changes to classify appliance using the statistic HMM algorithm. Baranski et al. [8] proposed using Genetic Algorithm (GA) for pattern detection based on a series of real power in 1-second time resolution. His method can also work on finite-state appliances. However, only real power method is not

enough to distinguish appliances that have the similar real power. In order to solve this problem, many researches [6] [7] used both real power and reactive power in ΔP-ΔQ plan to classify appliances. Figure 1.8 shows the principle of this approach. In order to enhance the NIALM accuracy, Leeb et al. 1995 [6] proposed using power information in third harmonics to enhance the accuracy of appliance classification. C. Laughman et al. [7] even proposed using ΔP-ΔQ plan of up to seventh harmonic to distinguish loads.

**Current and voltage**

Waveform of current has been used as a steady-state electrical signature. In a research in 2004, Lee, W. et al. [20] explored that current waveforms are different between appliances and they can be used for classifying appliance. Moreover, Suzuki, Kosuke, et al. [10] used current waveform to disaggregate power base on integer programming method. First, they collected a period current waveform of many appliances including their operation modes. Next, overall load circuit is represented by a current waveform (in 1 period) which is the total of all current waveform of working appliances and a disturbance value generated by noise or unknown appliance. Finally, an integer quadratic programming problem is solved to estimate the working appliances, which minimizes the disturbance value. Cox, Robert et al. [9] used only voltage measurement in high frequency to classify appliance. Ting, K. et al. [21] also proposed using 2-dimensional voltage-current trajectories under start up and steady-state condition to distinguish appliances.

Many NIALM researches worked on voltage noises in very high frequency [11], [12], [22]. These steady-state signatures are potentially the most useful information to classify appliances accurately. That can solve the problem similarity in power based NIALM approaches. However, this approach needs expensive hardware, which is able to extract small signal in high order harmonics. Moreover, voltage noises caused by appliance are very small and easy to be interfered by noises from environment.

### 1.3.3. Transition-state signatures

Transition state signatures are another type of electrical features extracted from electrical network. Most of transition signatures occur in a very short instant (milliseconds) or in high order harmonics. Such signatures are more difficult to be extracted than steady state signatures because measuring them requires high frequency data sampling in acquisition hardware. However, they add more supplement information to classify more appliances.

**Shape of transitions**

Some examples of transition-state signatures are start-up current and power transients with features current spikes, size, duration, shape of the transitions. Wang et al. in 2012 [23] decomposed power consumption to two basic shapes triangle and rectangular. These basic

shapes then can be represented by data such as start-time, peak-time, peak value, steady-time and steady power in a two graphic units, which is call a schematic diagram.

**Voltage noises**

Patel et al. [11] developed a device to measure electrical voltage noises in high frequency sampling in both steady and transition states. Their research shows that electrical noises in transition state have rich frequency spectrum varying from 10 Hz to 100 kHz as illustrated in Figure 1.9. This frequency spectrum depends not only on the load characteristic but also depends on the mechanic architecture of the switches, and the length of electrical lines. Therefore, this is also a potential approach to distinguish same kind appliance located in different rooms. Moreover, this noise can be measured from any electrical outlets at home.

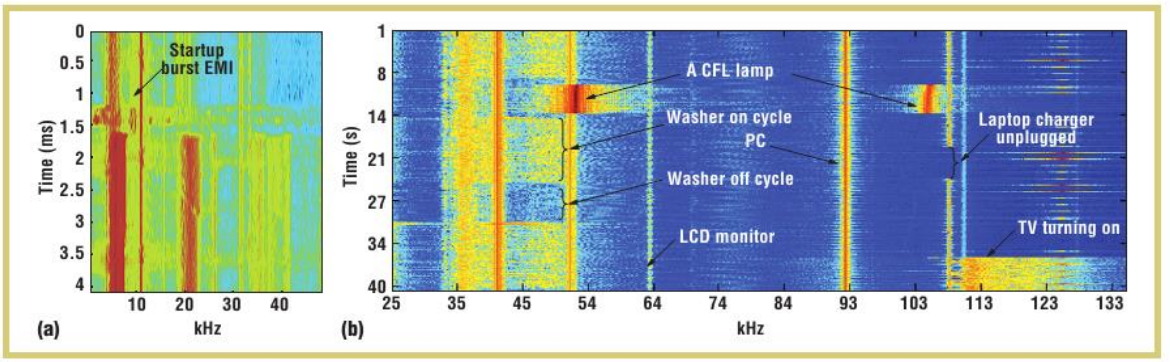



Figure 1.9 (a) Transient voltage noise signatures of turning ON event of a light switch, (b) Steady-state continuous voltage noise signatures of some home appliances [11]

## 1.4. The trend of NIALM technology

The revolution of electronics and telecommunications is changing the application and the architecture of NIALM system. In this section, we will discuss about some important technologies and their impacts on the NIALM then figure out the perspectives of future NIALM smart meters

**Ubiquitous computing (ubicomp)**

This concept means that computing can appear anywhere in human life in any devices, any format. Some involved technologies are Internet of Thing (IoT) and Wireless Sensor Network (WSN). Ubicomp can have underlying technology sensors, microprocessor, network, user interface etc. An example of ubicomp is the LG Smart ThinQ LFX31995ST Refrigerator [24], which can monitor its performance, energy consumption, weather and track the food freshness, daily receipt and notes. When all home appliances support ubicomp technology, they can record their power consumption and customer behavior by them-selves. NIALM can be applied there to disaggregate power consumption of all electrical devices inside that appliance for monitoring

and maintenance purposes for example the pumps, lights and fans inside the smart refrigerator. Information about their power consumption and the temperature inside the refrigerator can give us health status of the refrigerator. So that, for the smart appliance like that, a small stand-alone low cost NIALM meter is more reasonable than the expensive NIALM system.

### Cloud computing

**Cloud computing** technology has been applied into many commercial smart products. Instead of using a computer to process the complex applications and store database, cloud-computing devices can send data to be processed and to be stored in the server. This technology helps the service supplier to upgrade and maintain the system much more easily and more economically. Unfortunately, in NIALM domain, storing power consumption data for millions houses for months or a year is not practical so that current smart meters only send very slow sampling rate aggregated power consumption to reduce the size of transmitted data. This limits the capacity of NIALM to disaggregate more appliances. The reasonable solution is that future NIALM meter can do the preprocessing and event detecting by themselves to extract more electrical signatures with time-stamped events. Then cloud computing is applied to let the server disaggregates total power using event data and do other utility services such as creating detail electricity bill, data statistic.

### System on Chip (SoC)

The last recommended technology for the future of NIALM is System on Chip (SoC). This is single special integrated circuit, which is embedded inside all components to be able to run as a computer. SoC now may contain not only memory, processor unit, input output interface, timing sources, power manager unit; but also communication interface units such as USB, Ethernet, UART, SPI, Wi-Fi module, and real world interface such as temperature sensor, Analog to Digital converter, GPS etc. Some new SoCs also have special reprogrammable hardware named Field Programming Gate Array (FPGA). System on Programmable Chip (SoPC), which can be constructed to complex custom Digital Signal Processing (DSP) hardware to accelerate the computing for instance Zynq-7000 family of Xilinx [25]. Such a SoC includes a Processing System part and a Programmable Logic part. Processing system part includes dual-core ARM Cortex A9 with cache, memory, communication interfaces etc. Programmable Logic FPGA is programmed by using Hardware Programming Language (HDL). HDL is the special language to create the electronic circuit inside FPGA such as a filter, a microprocessor, a video processing hardware unit etc. Therefore, SoC is potential real future for NIALM and all algorithms can be implemented inside a chip. While the microprocessor processes the simple function, the hardware part (FPGA or DSP) will handle and accelerate the hard constraint and complex algorithms. SoC is also used to name the methodology that develops hardware and software in parallel in a system.

## 1.5. Thesis contributions

### 1.5.1. Context

This research belongs to the CoCoE project (**Co**ntrôle de la **Co**nsommation **E**lectrique dans les bâtiments) of ARCSIS (Pôle de compétitivité Solutions Communicantes Sécurisées) and CIM-PACA Design Platform. Partners of this project are EpOC (URE UNS 006) and IM2NP (UMR AMU 7234) laboratories, Qualisteo and RivieraWaves companies. The objective of this project is to develop an innovative, non-intrusive and communicating solution for electrical energy measurement in the building. Today, optimizing energy in building is based on the total electrical power consumption, with no detailed information about power consumed in individual appliances. Thus, solution so far needs complex systems with many meter and sensors, which is incompatible and inconvenient with existing building.

The problematics, which must be solved by the CoCoE project, are that the electric power consumption information is less accessible, not in real-time without any information about kind of appliances and usage, even with the Linky future smart meter. This can be solved by using NIALM technology implemented into FPGA and with the development of remote wireless sensors. There are two other PhD students working on this project in developing these new sensors to measure electrical current and other electrical signatures (Cifre thesis with Qualisteo company) and modeling and optimizing wireless communication systems (Cifre thesis with RivieraWaves company). The CoCoE project received an award during the World Efficiency Congress (Paris, October 2015): "Lauréat du Trophée de la Recherche Publique Energie-Climat-Environnement", given by ADEME and the two magazines EnergiePlus and Mesures, in the topic of Energy efficiency. Moreover, CoCoE is a building block of the "Smart Campus Nice Sophia Antipolis" project, which has been labialized by the French ministry of economy on the national industrial plan on smart grid.

### 1.5.2. A real-time innovative NIALM proposal

The thesis proposes a development of an innovative NIALM system based on SoC, which can solve some challenges in electrical power measurement and optimization. The performance of system was analyzed and validated according to the NIALM public data set REDD (http://redd.csail.mit.edu) to prove that they can solve well above technical challenges in NIALM technology. The system can enhances the precision or accuracy of the energy estimation to reach more than 80% of classification of the total power by solving several challenges including:

- ✓ Extraction of more electrical signatures to improve distinguishing similar appliances. They are changes in real power, reactive power and Total Harmonic Distortion (THD)

of current as well as the shape information of transitions including maximum and minimum values of real and reactive power in the transition and the duration of the transition.

- ✓ Detection of slow transitions in variable load appliances.
- ✓ Detection of simultaneous transitions to avoid missing event
- ✓ Detection of multi-state load appliances, which can have many states in their operation.

Moreover, attempting to solve these challenges requires carefully examination of the timing constraints of the system because we have mentioned the possibility for the user to get a real-time feedback of the effect of his behavior. Typical timing constraints are 5 seconds maximum response time or maximum delay between a real event and its display in the Graphical User Interface (GUI) and 200 milliseconds minimum duration between the detection of two events. Apart of performance constraints, technological and economic constraints cannot be ignored. To get a stand-alone, low cost, compact system, the implementation (hardware platform design) must also cope with other constraints including a low cost (more or less than $150), a low power consumption (more or less than 80 Watt), the ability to disaggregate 80% total power consumption and dimensions compatibles to fit on the breaker panel. Regarding of these constraints, SoC technology is a good candidate to solve that big challenge, except that hardware and software system design in SoC is not an easy task to be achieved.

### 1.5.3. A HW SW co-development methodology for rapid prototype

The second important contribution of this research is to propose a hardware-software co-development methodology aiming to develop the ability to rapidly prototype the hard real-time constraint SoC system using LabVIEW FPGA [26]. The thesis will present the use of Synchronous Dataflow (SDF) model as a user case of applying SDF model for hardware software co-development in FPGA SoC. Such approach proves that it is suit in developing a NIALM system and very effective in analyzing the performance in hard real-time constraint system. This model-based SoC design approach uses libraries supported in LabVIEW FPGA in both sides: the rich C, RTOS libraries for software development and the ready National Instruments™ reused Intellectual Properties (IPs) for hardware development. Thus, software and hardware co-developing using LabVIEW is boosted in time from the system design.

Moreover, customized FPGA IPs can be developed in several ways: using Xilinx System Generator or HDL manual coding. These IPs then can be integrated to the system by two useful tools: the Component-level IP (CLIP) and the IP Integration Node. These tools are especially vital when supported IP libraries do not satisfy hard constraint functions. Another advantage of this methodology is the possibility proceeding quickly to architecture exploration because partitioning of the functions in hardware or software can be easily investigated. A C/C++ software developer can appreciate the hardware interface API, which is created by the C API Interface tool in the C/C++ Eclipse programming environment, while a LabVIEW developer can use the interface in the LabVIEW environment. Whatever the environment, the SW/SW interface, HW/HW interface, and HW/SW interface mainly affects the memory and latency of

the system that can be analyzed in guide of the system constraints. This approach then enables rapid architecture exploration in order to boost the hard real-time constraints satisfaction by the FPGA and the CPU cooperation, reducing the time to market, which is another major constraint for companies.

## 1.6. Thesis Organization

The thesis has four chapters, which presents in the deep research about the NIALM technology from defining the drawbacks of this technology and their future in the support from the evolution of electronics and telecommunications to propose an innovative NIALM system. In order to develop such a system, the thesis presents base knowledge of embedded system in SoC development methodology, Model of Computation (MoC), and languages and tools. The HW SW co-development methodology to prototype rapidly the NIALM system based on SoC technology is also presented in the document. Then, the thesis also presents the application model for the NIALM meter using many MoCs: the dataflow, synchronous dataflow and statechart model. Finally, results of our experiment work will be presented in the end of the thesis. Detailed contents of each chapter are specified:

**Chapter 1** states the motivation of the thesis on engage people in saving electrical power. We then focus on investigating the NIALM technology- an innovative technology in monitoring power usage on individual appliances in the building. The chapter will present quite deeply about the principle of NIALM approach with their advantages, state of the art of NIALM with remain challenges and predicting the trend of this technology.

**Chapter 2** presents about many theories in developing a SoC embedded system. Reader can also find useful knowledge about the architecture independent design and the model of computations (MoCs). MoC can help developer in finding the best architecture for system early at design step. This capability can increase the productivity, optimize resource, and improve the performance to save cost and time from development to market of product. Introduction about design languages and EDA at each design level of system will be presented at the end of this chapter. This chapter also describes a hardware-software codevelopment approach aiming to rapid prototyping SoC application with hardware acceleration using FPGA. In this approach, we also propose to use synchronous dataflow model to model the system because this model can support well allocating memory and scheduling the operation of system in compilation time in multi processors architectures.

**Chapter 3** focuses on the development of algorithms for NIALM application. First, we present a system specification of innovative NIALM system, which can solve many challenges of NIALM technology in extracting more electrical signatures, detecting multi-states appliance and aiming to the development of real-time NIALM system. Then, information about the Cumulative Sum (CUSUM) event detection and Genetic Algorithm-based disaggregation algorithm will be presented. The chapter also presents some functional verification results in processing a public NIALM data set REDD to analyze the accuracy of system.

**Chapter 4** presents the use cases of MoC approach and HW/SW system design methodology in chapter 2 to develop a Real-time NIALM SoC based system, which focuses on analyzing the architecture exploration to satisfy the system requirements. The chapter also presents some detailed experimental prototypes of the system to give us some important conclusions and the road to continue to make a real commercial NIALM product.

# CHAPTER 2. SYSTEM MODELING FOR EMBEDDED SYSTEM

**Contents**




**Abstract:**

*Most of complex DSP systems are Reactive Process Network (RPN) systems, which contain both event control processes and data streaming processes. The objective of this chapter is finding out a development methodology to develop a Reactive Process Network system in SoC with FPFA acceleration. Therefore, we investigated various development approaches, many MoCs and development tools. Finally, we proposed new hardware and software co-development approach aiming to rapid prototyping a RPN system.*

# Chapter 2. System Modeling for Embedded System

## 2.1. SoC, SoPC and FPGA

### SoC

Today, SoCs systems contain most of essential elements to be able to develop a complete system just in a single chip: one or many programmable processors, memories (cache, RAM, Flash, DMA), complex buses (processor bus, peripheral bus, communication bus) and many kinds of real world interfaces (temperature sensor, ADC blocks, DAC blocks etc.). Moore's law is still valid in stating that the number of transistor in a chip doubles every 18 months period. The integrated circuit has passed ultra-large-scale integration with more than 1 million transistors in a tens nanometer technology chip and now turns to the stage of three-dimensional integrated circuit (3D-IC) [27]. Thus, complex SoCs with multi-core processors are used broadly to increase the system performances. Some SoCs may also have special functional units optimized for specific applications such as energy metering integrated HW, audio decoding/encoding, graphic acceleration, and power management unit. However, the higher integration scale the SoCs are, the higher active power the SoCs consume, that makes temperature in SoCs to increase and increases their leakage power.

### SoPC

The SoPC concept came from FPGA research domain with the possibility to develop a soft-processor based on the huge programmable hardware elements in FPGA for control applications. Such control applications involve looping and interrupt programming to react to input events that are better processed in software on conventional processor rather than in configurable hardware in FPGA. Thus, two most famous FPGA vendors, Xilinx and Altera, supplied some synthesizable processor cores (soft processor) such as the MicroBlaze of Xilinx and the NIOS of Altera. Moreover, there are also various synthesizable processor cores developed by the community of developers, which may open to be used in any applications and can be found in [28]. However, synthesizable processor cores do not satisfy the requirements in most of products with high performance and low power consumption. Thus, newest FPGAs have integrated real processors (hard processor) core into FPGA for example Xilinx has the Zynq IC with dual core ARM processors embedded into the FPGA platform [25].

SoCs are still developed for some objectives including very high-speed interconnection as network-on-chip (NoC), hundreds or more processors, optical communication interfaces, GPS etc. Integrating FPGA into SoC may be the most important evolution of SoC, which provides developers a powerful tool to develop, and analyze the integration of new specific hardware units to accelerate the performance of their application. This feature of course may not give the best solution to get a system with the highest performance and the lowest power consumption but it can help the company to avoid expensive Nonrecurring Engineering (NRE). NRE relates to the total cost in researching, developing, designing and testing a new product, which may be very expensive because of the large number of iteration during the development cycle. NRE

only stops when final product is manufactured and put to market. Unfortunately, NRE in a development of specific applications without standard supported SoCs is still very expensive because some specific hardware IPs are not supported in any current SoCs. Thus, FPGA is necessary to give us a programmable architecture, that supports quickly changes and analyze the customer's hardware functions before prototyping the product.

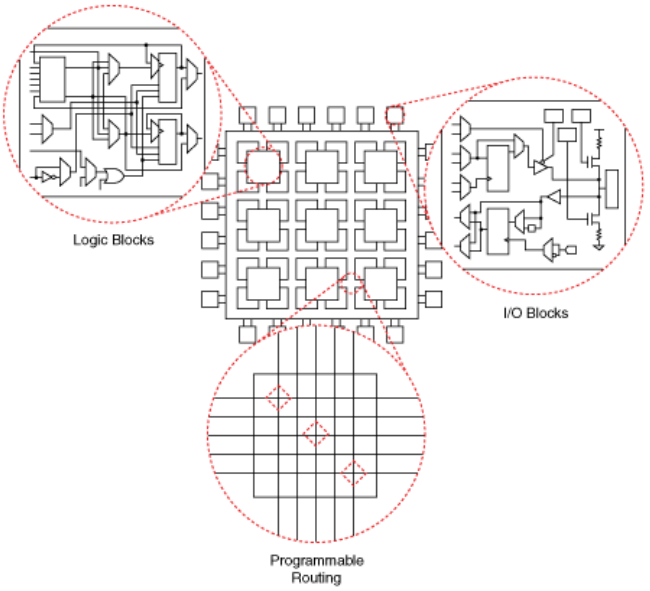



Figure 2.1 A generic structure of FPGA

## FPGA

FPGA is the lowest cost solution in developing specific customer functions in hardware, which are not supported in current SoCs. Figure 2.1 shows the general structure of a FPGA that has a rich hardware resources as configurable logic blocks (CLB), I/O blocks and programmable interconnects. CLBs contain basic logic cells AND, OR, NOT, Flip-flop which are used to create combinatorial or sequential logic circuits. Modern FPGAs also contain Memory (blocks RAM), DSP blocks and hardware-embedded processors to improve the performance of computations. These resources are surrounded by a dense grid network of programmable interconnects which are controlled by configuration memory in FPGA to connect hardware resources together to create functions in digital circuit. Moreover, the configurable I/O blocks in direction and voltage level of this technology gives developers the flexibility in placing and routing PCB boards.

Because hardware functions in FPGA process much faster than in processor, FPGA functions are often used in specific high performance applications that require very high speed processing. In NIALM system, some complex features must be extracted from electrical appliances in very high sampling frequency: high order harmonics or information of events. The strict timing constraints of such system surely require a high throughput processing that cannot be achieved by conventional microprocessors. Figure 2.2 shows an example where parallel processing in FPGA hardware improves performance 19x compared to a standard DSP processor. That is

because DSP processor needs many loops to process an algorithm while FPGA can express in parallel this algorithm in using more hardware resources but improving performance dramatically.

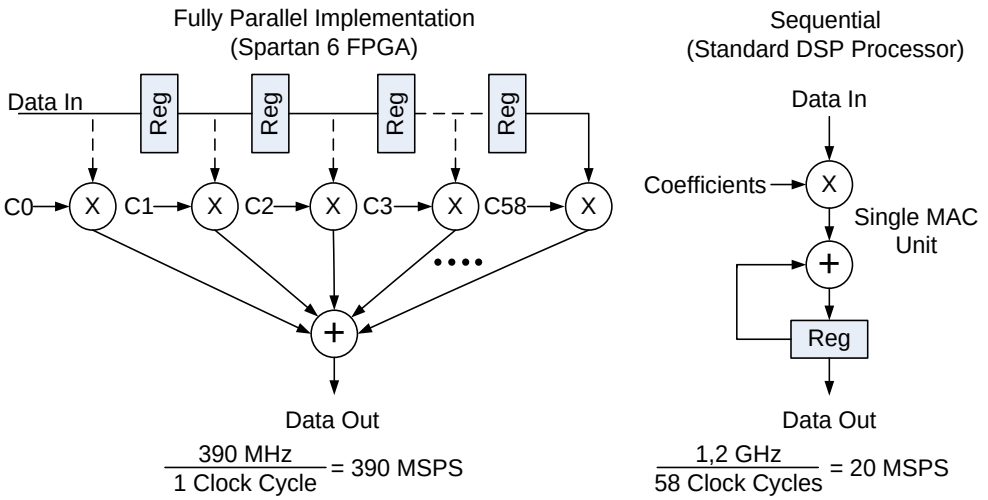



Figure 2.2 Performance comparison between DSP processor and FPGA [29]

The main idea behind FPGA is the ability to program hardware resources using HDL languages. These languages allow developers to program the configuration memory to connect hardware resources in FPGA device together to create a complex digital circuit only by writing codes. Popular HDL languages for "hardware coding" so far are VHDL, VHDL-AMS, Verilog, Verilog-AMS, SystemC, and SystemC-AMS. Moreover, there are also many model-based approaches to develop FPGA functions such as MATLAB HDL coder, Xilinx System Generator in Simulink, LabVIEW FPGA. These languages will be discussed in detail in coming section.

## 2.2. System development of SoC

### 2.2.1. Algorithm optimization

In many high-level analysis tools such as MATLAB, Scilab, and Octave, developers can use matrix mathematics, built-in signal processing or graphic libraries to quickly model and analyze algorithms. MATLAB can process a matrix array with memory size up to 8 Gigabytes in 64-bit Window XP running 64-bit MATLAB and this value is up to about 260 MB in Scilab. However, such original algorithms often consume a lot of memory and they do not suit to be implemented into SoCs. In most of DSP systems, developers always have to collect processes, present and analyze data in various formats to get more knowledge about the object. Such complex systems may contain some parts, which have requirements to process and transmit data stream in a very short time. Developers can solve these requirements easily if they select powerful platforms to implement these complex algorithms. However, SoCs do not support well matrix computing on their limited resources so that developers cannot use the same algorithms developed in

MATLAB. Developers then need to analyze and select algorithms, which satisfy system requirements in resources usage, power dissipation, latency of each function and global timing constraints.

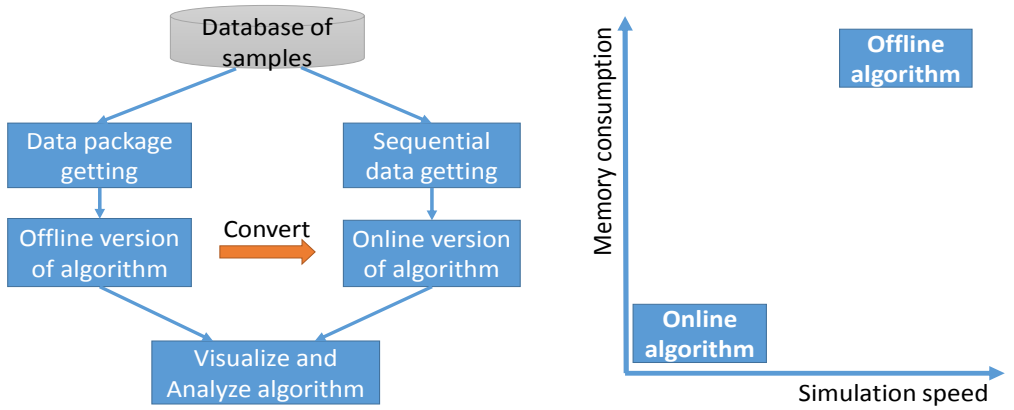



Figure 2.3 Online algorithm and offline algorithm

Figure 2.3 shows two algorithm types: offline and online algorithms; and processing online algorithm requires less memory and power processor than their offline versions. Thus, online algorithm can be used to optimize both timing and resource usage requirements in SoC system. A following simple example of computing average value in online and offline algorithm is shown below:

$$AM = \frac{1}{n}\sum_{i=1}^{n} a_i = \frac{1}{n}\left(a_1 + a_2 + \cdots + a_n\right)$$

```
%% generate database
size = 100000;
data = rand(1,size)*100;
```

```
%% MATLAB offline code
tic();
a1 = data;
b1 = mean(a1);
t1 = toc();
```

---

b1 = 50.1136;

**Memory size: 800016 bytes**

**Processing time: t1 = 0.026 seconds**

```
%% MATLAB online code
tic();
s = 0;
for i =1: size
   s = s + data(1,i)*100;
end
b2 = s/size;
t2 = toc();
```

---

b2 = 50.1136

**Memory size: 32 bytes**

**Processing time: t2 = 0.1543 seconds**

It shows that memory size usage in online version is only 32 bytes comparison to 800016 bytes in offline version. However, time processing for this online algorithm is about 59 times slower than in offline version when tested on a standard computer. The interesting thing is that

this online algorithm needs only 0.1543/100000 or 1.543 microseconds to process each input signal measured in a defined sampling rate about 648 kHz. Such sampling rate is very high in most of SoC applications. Therefore, the main objective of algorithm optimization is to find the compromise algorithms that are able to convert to online version for processing stream data in a resource limited SoC system.

### 2.2.2. SoC design flow

The design process of a SoC has changed to move from a *hardware-oriented* perspective to be close to the *software-dominated* concern in system level design, hardware-software partitioning decision and system architecture design. SoPC technology becomes a powerful tool not just in developing specific hardware functions but also in exploring a system in various architectures (one core, multi core or multi core with FPGA acceleration) to find the best one for the product. So that, this technology may reduce the spin iteration in development works then increasing the productivity and reducing the price of product. Figure 2.4 shows a generic system design flow for SoC with three main steps: *system specification*, *architecture exploration* and *architecture design*.

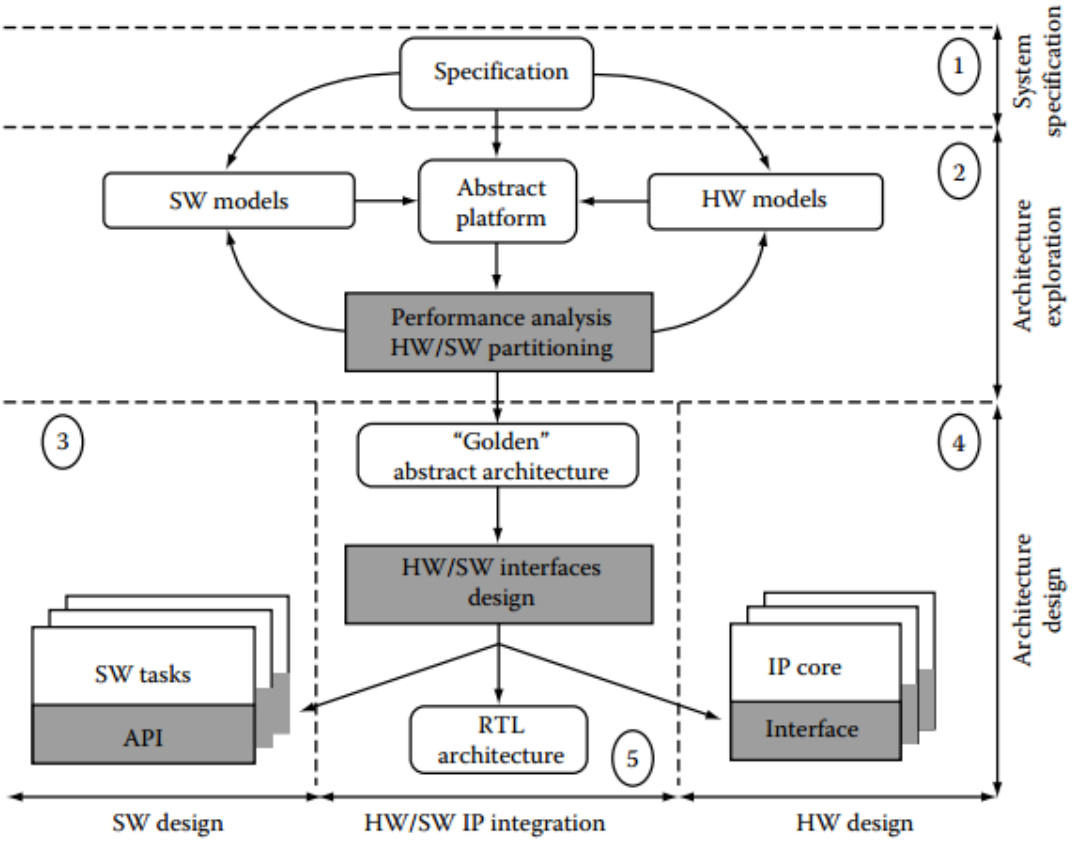



Figure 2.4 Generic design flow of SoC [30]

## System Specification

System specification is the first step to convert the system requirement of customers to a formal document, which sets a contract between customers and developer about the system to design. This process often starts from requirements defined as an abstract description of customer about general functionality, performance, and budget of the system they want to design. Thus, a system specification must formally exhibit the overall activity of the system and then describes objects, data, events definition, and the functional decomposition with selected algorithms for each function. Although the detailed algorithms may not be described in system specification, some results of algorithm's analysis should be shown to prove that they satisfy customer's requirements. Then this document is validated by the end-customer to decide if some modifications in the requirement are needed, involving changes in the functions or improvements in the algorithms. One the agreement from the customer got, system developers can continue their works to develop the system.

## Architecture exploration

Architecture exploration is a major step in SoC system level design to decide the final architecture of the system, which includes modeling functionality of system, selecting the best hardware architecture and hardware software partitioning. Partitioning hardware and software will create an abstract platform of the system. This abstract platform should be executable to be able to analyze the input-output relation and to validate the operation and communication between functions with the system specification. From results of this analysis, developer can find out the best architecture for system that can satisfy all required criteria relating to the performance of the system. This selected architecture will be the "golden" abstract architecture to drive the architecture design.

## Architecture design

Architecture design contains implementation works: design hardware, software and HW/SW interface in selected architecture from architecture exploration step. In hardware design step, hardware designers have to transform hardware functions from the abstract level model into Register-Transfer Level (RTL) model, which can be synthesizable. Designer can reused hardware IPs supported by SoC vendors to optimize hardware resource and performance or they can use RTL generated by automation generation tools. However, software design is often developed when the design of hardware is completed because just at that time, hardware designers can supply to software designer the API (Application Program Interface) to communicate to the hardware design. In some modern SoC approaches, software and hardware can be developed in parallel if hardware designer and software designer reuse standard communication channels. Architecture design often produces a prototype, on which system designers can validate system in real environment to assure that final products will satisfy constraints enumerated in the system specification.

### 2.2.3. SoC development approaches

Three main SoC development approaches are based-board design, virtual platform-based design, and model-based design approach. However, both three approaches generally contain three main processes: hardware, software and application developments, which are responsible by system, application and implementation designers [31]. In order to handle these tasks, there are many specific requirements for each designer:

- ***Application designers*** must have a good knowledge of structures and algorithms of the appliance to be able to decompose it into tasks and sub-tasks. As we mentioned in previous section, application designers are the ones who can investigate various algorithms and do the algorithm optimization to design the best algorithms for SoC system. Therefore, they are often required to have good skills in logistics, mathematics, statistics, and analysis languages such as MATLAB, Scilab, Octave, …

- ***Implementation designers*** are hardware designers and software designers who have knowledge of specific platforms and implementation methodologies with software programming languages (e.g. C, C++, C#, Java, Basic) or HDL languages (e.g. Verilog, VHDL, SystemC) or designing platform using CADs tool. They can implement algorithms of application designers in a specific architecture designed by system designers.

- ***System designers*** often have a good knowledge of system organizations, architecture of microprocessors, hardware components, and the communication between hardware and software. They can design architectures or platforms for the system by selecting suitable hardware components such as CPU, UART, Ethernet etc. and the interface buses between them and they can design the firmware to handle the communication between hardware and applications. System designers must be able to explore and to analyze the performances of the system in many selected architectures to select the best one, which satisfies all system requirements. In order to do these tasks, system designers must have strong skills in system modeling languages (e.g. Dataflow, Finite State Machine, Kahn Process Network etc.), and system design languages (e.g. Verilog, VHDL, SystemC etc.).

SoC development approach will define the importance and the role between these three designer types and that will have effect on system quality. Moreover, selecting development approach may depend on the characteristic of the system we want to design.

#### *2.2.3.1. Board-based design*

Board based design is a traditional system design approach, which the development of system is done systematically through many processes. It starts with the platform designed by system designer; then implementation designer will develop a real prototype for application designer able to test applications. However, we cannot verify the system before prototyping the real

platform, which can causes high risk in timing error between process threads in the real platform. Some problems in hardware, software and hardware software cooperation, which have not been covered completely in the platform design step, may appears. All designers have to work together to fix all bugs, but they sometimes have to rebuild other real platforms. This cycle may repeat many times and take long time to finish the product.

This method is still useful in some cases when considering an application-oriented design in a reused HW and firmware platform, when the constraints are not too strict and when it is just a product update for the next software version. However, such methods cannot apply in designing complex systems with hard constraints in resource and timing of real-time processing and high performance computing applications. That sets a big challenge in system development for such applications from system design, hardware and software implementation, verification, prototyping, system on chip verification to market product.

As illustrated in Figure 2.5, traditional SoC development methodologies often require hardware designers and software designers to convert system design of system designer to HW synthesizable codes and SW compilable codes. System designers can only test the system running on target when hardware designers and software designer finish their work. This final test is more difficult to define where bugs come from HW designers, SW designers or system designers. Recognizing the bug and correcting process may require some designers to work while the others have to wait. In other words, although software and hardware can be developed in parallel, the complete system development is still in waterfall design flow.

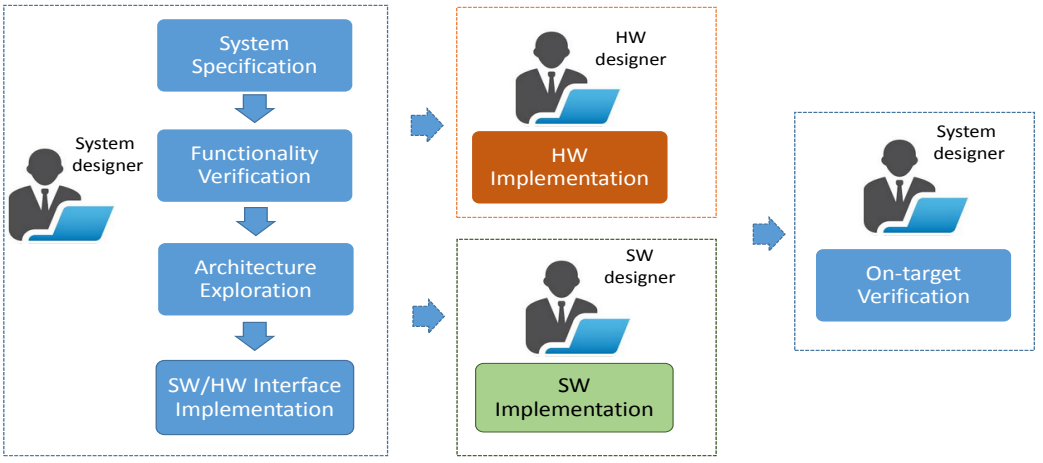



Figure 2.5 Tasks of each designer in the traditional SoC design flow

### *2.2.3.2. Virtual platform-based design*

In architecture exploration, instead of developing real code of HW and SW in each platform to evaluate the performance between them, developers use a special model named “abstract platform”. Abstract platform is a system model where software elements in microprocessor, hardware elements such as hardware IPs, FIFO memory, and communication channels can be

modeled at a very abstract level. Such models give developers the capability to model a technological independent system, which helps to change system architecture quickly and flexibly. Moreover, the abstract level also helps to speed up running the system simulation depending on the accuracy level of designed model. The specification model is the lowest accuracy model but its simulation is the fastest one so that this model is often used to validate the functionality of system. Simulating the cycle-accurate model is slowest but the results is closest to real architecture so that this model is always used to exploring the architecture for system.

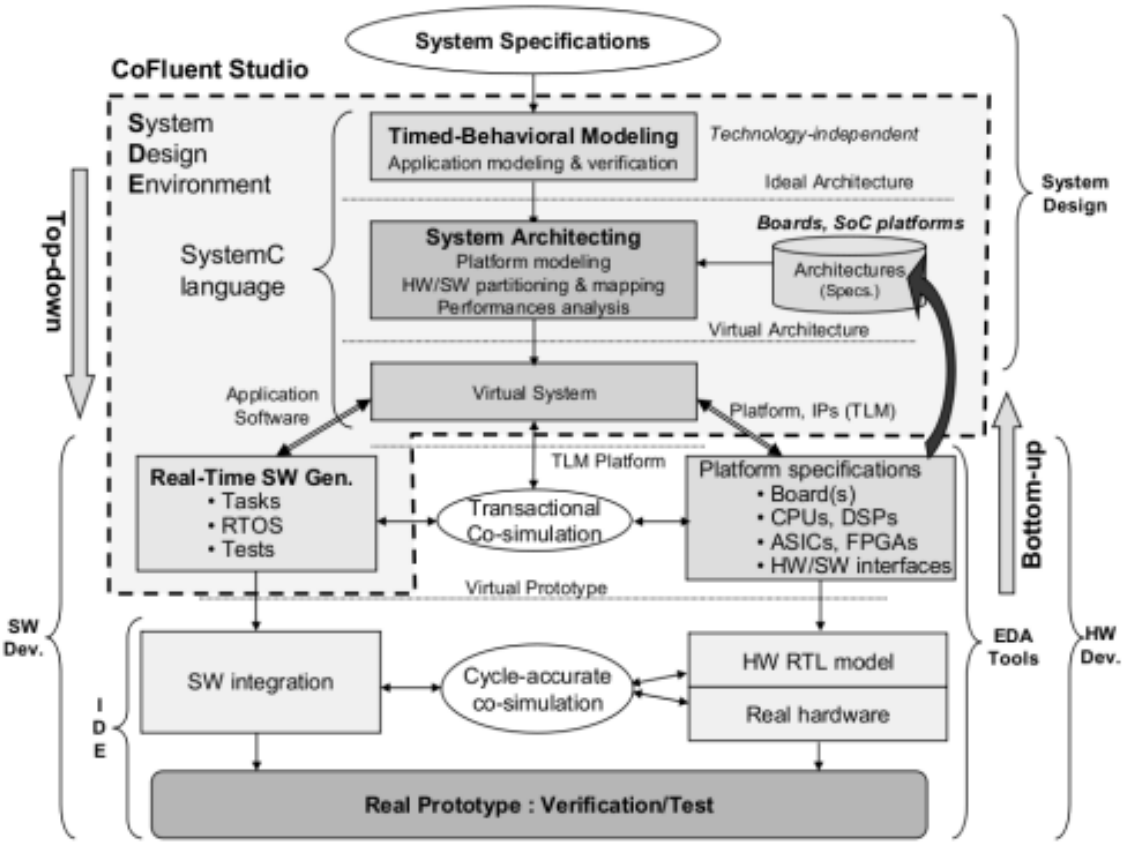



Figure 2.6 Design flow of CoFluent SoC design methodology [32]

Other architecture exploration approach follows the "virtual platform based design" but with a risk to increase the time when changing the model of function to adapt to the selected "virtual platform". For instance, a function modeled in C running in processors must be converted to RTL format using HDL to be able to run in hardware FPGA. However, "virtual platform based design" has an advantage in that the real platform is ready after the architecture exploration and the final step in hardware software development can be boosted from the work with "virtual platform". Cofluent Studio gives the excellent solution for "virtual platform based design" approach. As show in Figure 2.6, initially, SystemC is used to develop the technology-independent architecture, which can be executed for application verification. Thanks to the capability of SystemC, the SystemC specification model then can be quickly converted to

variation levels of architecture model from time-functional model, Transaction-Level Model (TLM) and cycle-accurate model.

The virtual platform can speed up the development process by giving hardware developer and software developer a unique simulation environment to co-design hardware and software. Limitation of this method is that the cycle-accurate model takes a long time to simulate complex systems. Moreover, it may take a huge time to optimize system with hard constraints because this method still cannot verify application at platform modeling step. Designers have to reprocess all steps if some constraints are not satisfied. This method is fast when the constraints in real-time and resource are not too strict and there are not real platforms. An advantage of this approach is that designer can analyze system in different supported platforms to find out the best one for system.

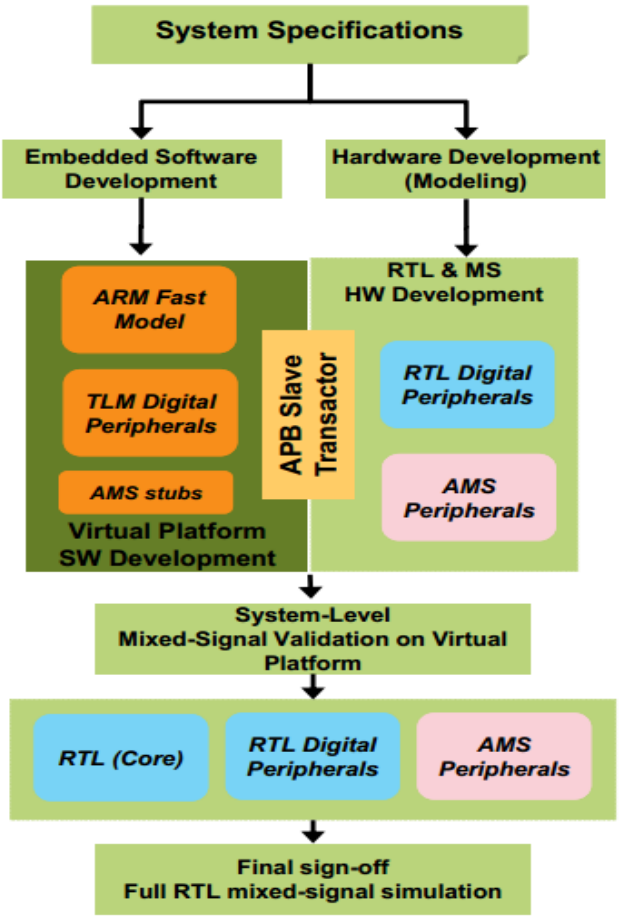



Figure 2.7 System design flow on Cadence Virtual system platform

An example of the virtual platform based design approach is Cadence virtual system platform. This is a powerful commercial software, which supports rich libraries model of hardware components [33]. They are most basic components for embedded system development such as UART, keyboard/mouse controller, real-time clock, programmable timer, interrupt controller, audio codec interface, programmable Led, LCD; to complex component such as

Ethernet controller, I2C, SPI, bus controller, serial interface, buffer, memory, touch screen input, flash memory, router, etc. This approach also supports virtualize hardware such as terminal, Ethernet, etc. Thus, system designer can quickly design a suitable platform that application designer will use to early develop and test the software application. In advance, system designer and application designer can cooperate better in developing complex hardware/software system with automated modeling and faster hardware/software debugging. Cadence virtual system uses various languages from system design language SystemC to RTL languages Verilog, VHDL, System Verilog, and software development languages C, C++, Assembly. As illustrated in the design flow of this approach (Figure 2.7), software can access both emulated RTL digital peripherals and Analog Mixed Signal (AMS) peripherals, that make simulated system works almost exactly the same with the real system.

In open source area, popular instances of virtual platform approach are OVP, ArchC. Open Virtual Platforms (OVP) supplies many architectures for software developers test their applications [34]. Such platform has three main components: Open Source Models, OVPsim simulator and Modeling APIs. Similar to Cadence Virtual system, Open Source models contains many processor models as ARC, ARM, MIPS, PowerPC, NEC v850, and OpenRisc families and system component models such as RAM, ROM, trap, cache, bridge, Direct Memory Access (DMA), UART, FIFO, etc. These models can be put together to create required platform from single to multi-core system with complex memory, cache system. ArchC is an interesting other open source SystemC-based language [35]. It supports system developed in various instruction set architectures such as MISP, PowerPC, 8051, Sparc, PIC16F87. ArchC can also generate an executable SystemC model, so that designers can integrate it with other SystemC model using transaction-level interface for the complete design process from specification, modeling and simulation steps.

#### *2.2.3.3. Model-based design*

Model-based design is an attractive design approach, which system developer can specify a complex system in very high-level model using built-in blocks or using system design languages. System can be even modeled in continuous time and discrete time domain. Instead, writing complex structure and extensive software code, which should be done in the implementation phase after selecting a detail architecture, developers just figure out the abstract specification for the system. Model-based design environment is also call a high-level executable specification because it can simulate, and verify the functionalities of system in abstract level design. Some EDAs also have automated code generation feature, which can generate both hardware and software detail codes to be implemented on specific architecture.

The speed of system simulation varies on different system level models [31]. As we mentioned about the level of abstraction in Figure 2.8, this verification may generate more detailed model step-by-step for example generating RTL model from the system level design. So, the designers can run their verification scenarios in each level of the system with an advantage is that the system simulation runs faster in higher abstract level as illustrated in Figure

2.8. Highest-level specification model takes shortest simulation time because this model is untimed in both computation and communication. In contrast, cycle-accurate model (RTL model) is cycle-timed model in both communication and computation so that system simulation is very slow in this level.

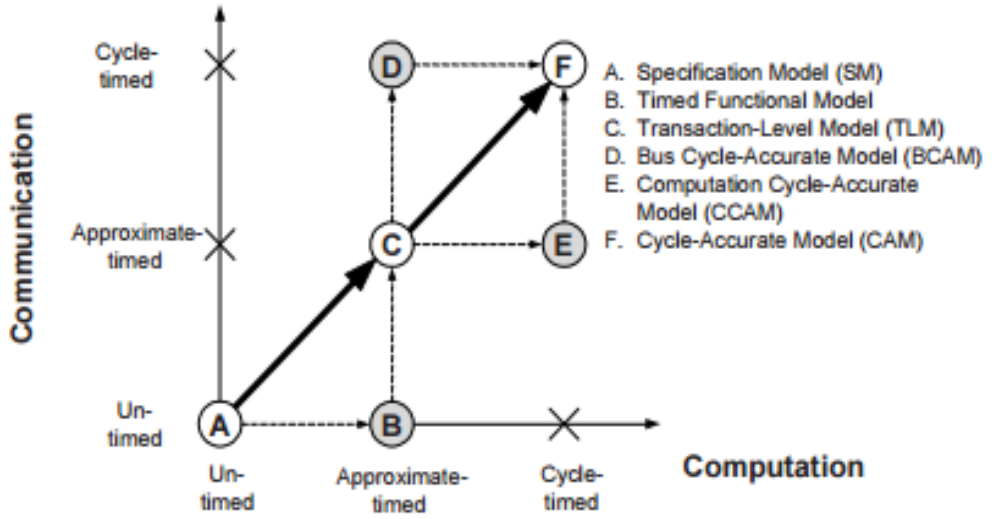



Figure 2.8 System level models [31]

Minimizing hardware detail or technical independent in a model-based design gives developers many important benefits:

- Developer can reuse the stable model libraries: algorithms, communication protocol, peripheral interface, etc. from other developer groups as a powerful base infrastructure for his design. One when the system verification is finished, developer can quickly automated generate lower level of system for the test. Some EDA tools also support testing system design model on real hardware directly (Simulink + System Generator).

- Model-based design approach may help software developers can study to develop a hardware system in a short time then manager can save a lot resource usage. For example, data flow programming with supported model in LabVIEW can help developers who do not know about C, C++ programming, can quickly develop an embedded system. In other example, LabVIEW FPGA can help developer, who have not studied HDL languages, to develop RTL systems.

- Execution specification is a dynamic specification of system, which may increase the productivity of a design. Moreover, there are more and more researches in automated code generation from system model that allows developer a rapid prototyping system to minimize the time to market constraint.

## 2.3. Model of Computation

The development of electronic design methodology is much slower than the development of hardware capability [36] that sets a big challenge in system design of complex systems with difficult constraints in resources usage and timing of real-time processing and high performance applications. That challenge goes through all steps: system design, hardware and software implementation, verification, prototyping, system on chip verification to market product. Therefore, a system design must also be able to help developers to find the best architecture for the system for examples: the number of CPUs, processor speed, power and memory budget etc. Such architecture should be reliable, resource-efficient and support a fast and predictable system development.

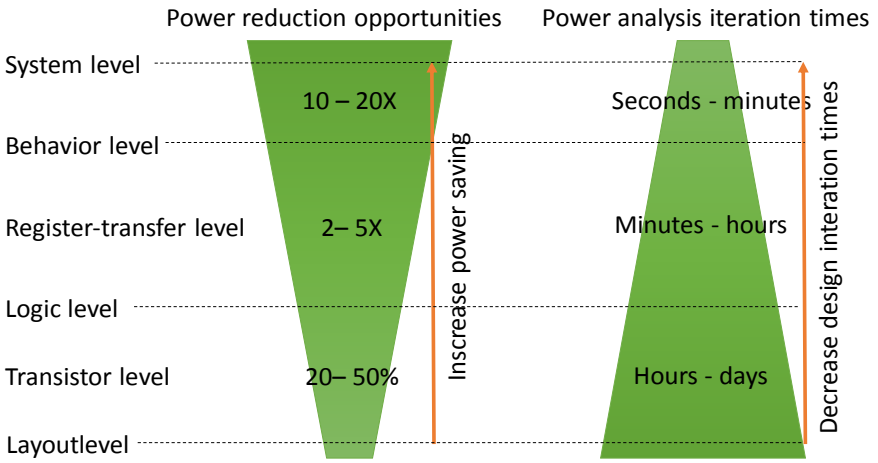



Figure 2.9 High-level power analysis and optimization [37]

Model of Computation is an abstract specification of a computation that is critical for functional validation, performance analysis, and then driving the system development more productively. Abstraction level in a system depends to its complexity. From 1980s to now, abstraction level of system model changed from Boolean expression to represent basic logic gates; to Register transfer level (RTL) to represent medium-scale integration (MSI) circuit and to System Level to represent complex hardware and software system. Designers can save power consumption by optimizing system in each model level of designed system but the system level gives designers highest opportunity to save power consumption and power dissipation on system [37]. As illustrated in Figure 2.9, system level has 10-20x opportunity to increase the power saving compared to 2-5x in register-transfer level and 20-50% in transistor level.

A good MoC has several requirements [38]. As shown in Figure 2.10, input parameters are specified system requirements in resources, environment, and real-time. System resources must include memory and energy consumption, performance of processing and communication. Output system metrics are quality aspects and resource usage in system for analyzing system in reliability, energy efficiency, security, memory usage and throughput.

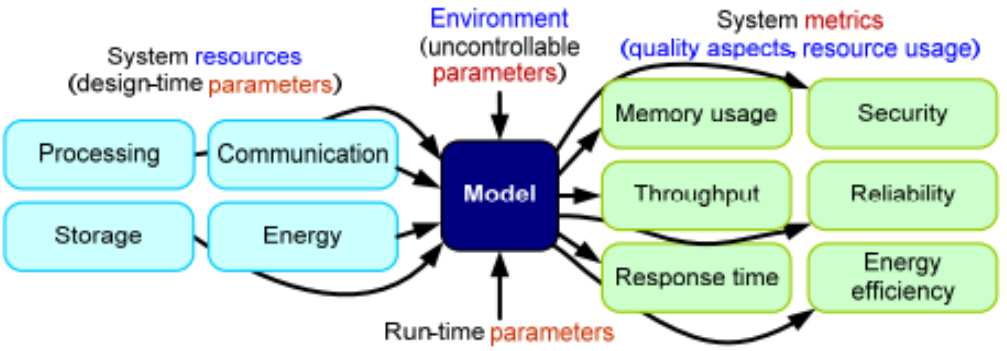



Figure 2.10 System resources and System metrics for performance analysis [39]

Initial MoC of system should be purely to describe behavior of system in a technological-independent way to overcome the gap between technology development and design method improvement. Thus, a MoC should be implementation independence, composability, and analyzability [40]. It should describe in overall the activity of system, which is composed by many complex functions in less detail about the platform, or architecture of the system. The implementation independence property helps the system to be clearly designed for performance analyzing and validation. This property is also very vital to upgrade system to the new architecture in the next generation version. The composability of a MoC decides the possibility in analyzing amount of resources consumed by sub-tasks and timing influence between them. Therefore, with the MoC, we can allocate abstract resource budgets to require the implementation follow timing and resources constraint to improve the system. The other requirement of a MoC, the analyzability, may vary between MoCs, which depends on the restriction of the system. For instance, Synchronous Data Flow model allows only a constant amount of input and output token at each activity cycle so that such MoC is efficient for static scheduling and resources allocation.

### 2.3.1. Finite State Machine

Operation of a system means a series of actions, which can occur regularly in specific time or in responding to input triggers. If such actions are finite, each action can represent to a state of the system and a finite number of states can be considered as an abstract model of the machine-Finite State Machine (FSM). In FSM, a state is a status of a system that is keeping the current operation and waiting to execute the next transition. A transition is executed when system need to process a new set of actions which is caused by a fulfilled condition or by an external event. An example of FSM is the heating, ventilation and air conditioning (HVAC) systems shown in Figure 2.11. Such a HVAC has two state named **cooling** and **heating** depending on the temperature sensor from input source. System is initially in cooling state and the next transition to the heating mode when temperature value is less than $18^0$C. In heating state, only heatOn signal is active to heat the air in the building. Next state transient from heating to cooling occurs and heatOff is active to turn of the heater source when temperature value is higher than $22^0$C.

The mathematical model of computation of FSM can be defined by five elements [41] $(Q, \Sigma, \Delta, \delta, q_0)$ where

- Q is a finite set of states
- $\Sigma$ is a set of input symbols
- $\Delta$ is a set of output symbols
- $\delta$ is a transition function mapping $Q \times \Sigma$ to $Q \times \Delta$
- $q_0$ is the initial state, $q_0 \in Q$

If $\omega$ is a function of output $q_0$, two FSM types can be classified depending to the relation between output and input. If only current state effects on output or ($\omega$: Q → Δ), this defines a Moore machine. If output is a function of both current state and inputs ($\omega$: Q x Σ → Δ), this is a Mealy machine. The main difference between such two models is that Mealy machine responds to the input immediately. Conversely, the Moore machine does the respond when system changes to the next transition. In diagram, Mealy machine labels the outputs in arcs (transitions) while Moore machine labels the outputs in nodes (state). In fact, designer can transform a Mealy machine into Moore machine. However, system uses loops to check current state and input value to determine the next state so that it repeats the action in current state every iteration. Thus, Moore machine consumes more resources than Mealy machine.

| Current state | Input condition | Next state | Respond action |
|---|---|---|---|
| Cooling | Temperature > 18 $^0$C | Cooling | - |
| | Temperature ≤ 18 $^0$C | Heating | heatOn |
| Heating | Temperature < 22 $^0$C | Heating | - |
| | Temperature ≥ 22 $^0$C | Cooling | headOFF |

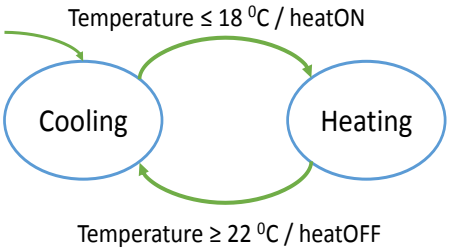



Figure 2.11 Mealy FSM model of a HVAC system

| Current state | Action | Input condition | Next state |
|---|---|---|---|
| Cooling | heatOff | Temperature > 18 $^0$C | Cooling |
| | | Temperature ≤ 18 $^0$C | Heating |
| Heating | headOn | Temperature < 22 $^0$C | Heating |
| | | Temperature ≥ 22 $^0$C | Cooling |

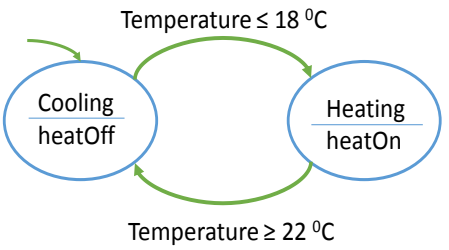



Figure 2.12 Moore FSM model of a HVAC system

Figure 2.11 and Figure 2.12 are the refined Mealy and Moore FSM diagrams of this HVAC system to model the operation of the heating, ventilation and air conditioning (HVAC) systems, where circles denote the states and arcs denote the transients [42]. We can see a transient $\delta_1$

associate with {temperature ≤ 18/ heatOn} where temperature ≤ 18 is a guard and heatOn is an action. That means the condition for the transient $\delta_1$ occurrence is temperature ≤ 18 and the action in this transient activate signal heatOn to turn on the heater. If there are not guards, the transient occurs automatically in the next operation cycle of the FSM. Such FSMs with guards for the transition are Mealy machine; others are Moore machines. The mathematics model of HVAC system described above is Q = {cooling, heating}; Σ = {temperature}; Δ = {heatOn, heatOff}; $\delta_1$ (cooling, temperature ≤ 18) = (heating, heatOn), $\delta_2$ (heating, temperature ≥ 22) = (heating, heatOff) where cooling is the initial state of the system.

### 2.3.2. StateChart

In a complex application, system specification using traditional Finite State Machine model may be not manageable and comprehensible. David Harel in 1980s [43] invented an innovative formalism of state machine and state diagram named StateChart with three notations of hierarchy, concurrency and communication in complex discrete-event system. Two key innovations of StateChart are OR state and AND state. OR state, illustrated in Figure 2.13, is used to describe hierarchical states A and C in the transition to state B. AND state, illustrated in Figure 2.14, is used to describe concurrent states. Because state B and C can run at the same time with states E, G and F, then AND notation divides main block to two side A and D. Transitions of states in each side are in sequence but two FSM A and B can run in parallel. This feature is especially important in developing system in multi-processors architecture. It ensures that all capabilities of parallel computing are used in the system. While conventional FSM is neither hierarchical nor concurrent, StateChart keeps the usefulness of FSM still clearly in complex states and transitions then improve both expressive and analyzable of a FSM specification. In Figure 2.14, replacing eleven arcs (transitions) by only six arcs (transitions) makes the machine much more expressive.

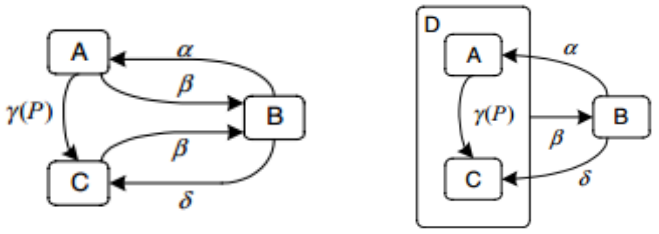



Figure 2.13 A basic FSM and hierarchical equivalent in StateChart with OR states

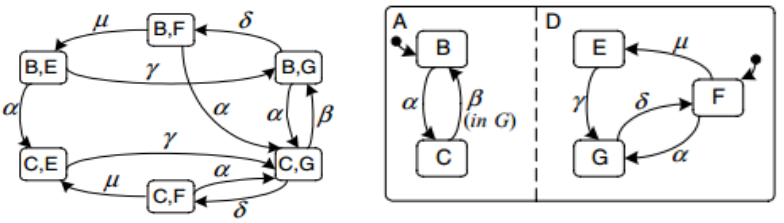



Figure 2.14 A basic FSM and concurrency equivalent in StateChart with AND states

In 1986, Statemate is the first software, which used statechart to model complex reactive systems [44] [45]. This software has graphic editor environment, which designer can draw statechart to model a real system. The model in Statemate can be executed step by step, interact to the environment emulation in designed execution of scenarios. During the execution, current states and activities are highlighted and the results are animated to verify the activity of the system. In addition, Statemate can produce the document, and can automatically translate statechart into code in high-level programming languages Ada and C. This software can also generate hardware description languages VHDL and Verilog [44]. Then, these codes can be synthesized to run in real target as a prototype of system or in a simulation version of the target. Although the generated codes are not always as efficient as final code, this approach satisfies requirements in speeding up the system level verification by reducing the iteration of design flow.

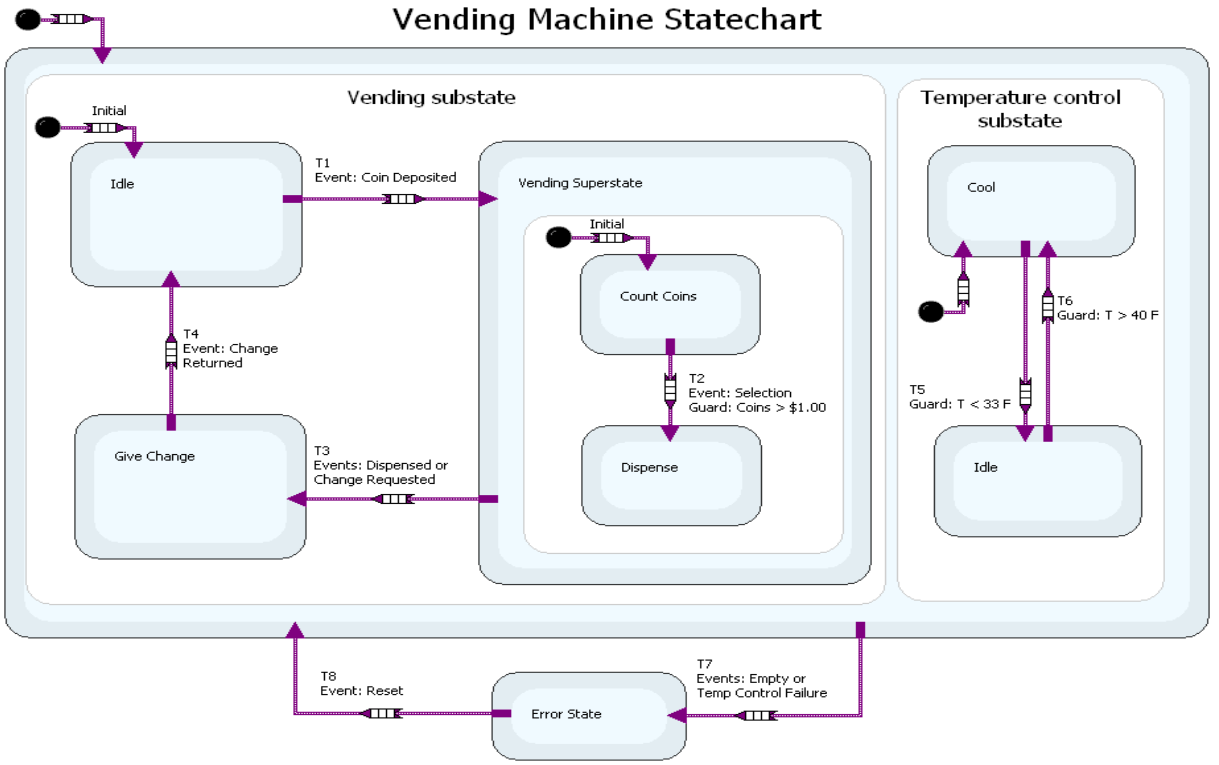



Figure 2.15 Statechart describes Soda Vending Machine with Dispensing Logic and Temperature Control AND state [46]

National Instruments corp. has LabVIEW StateChart module that uses to develop system functionality in statechart model [47]. LabVIEW StateChart supplies a great environment for developer from building executable documents for system specification to LabVIEW code generation. LabVIEW StateChart can deploy system in various architectures from desktop PC, micro controller to FPGAs. Figure 2.15 shows an example in using LabVIEW StateChart to model a vending machine. A state is noted by a square block, which can contain many levels of FSM by hierarchy capability for example the Vending Superstate is inside the Vending substate. Two AND FSMs express concurrent feature: Vending substate and Temperature control substate in two white square block. Actions can be launched in the transition or in a state which makes Statechart can model system in a mixture Mealy and Moore FSM type.

In 1990s, StateChart became one diagram standard to model the behavior of system within Unified Modeling Language OMG-UML V1.3, section 3.73 – 94, 2000. With advantages of a very popular open modeling language in visualize the design of system; UML Statechart diagram can model a state machine with emphasizing states of system and transition among these states. Development tool Rhapsody supports executing a UML-statechart in many modes: regular mode, trace mode or animation mode [48]. Developer can test in high accuracy the behavior of system in a simulation environment, which uses the codes generated by Rhapsody for real product.

### 2.3.3. Dataflow modeling

Dataflow Diagram (DFD) developed by Larry Constantine in 1974 [49] is a popular model to develop digital signal processing (DSP) systems. This model allows visualizing the behavior of system in a very abstract model level that emphasizes on the flow of data through many main processes.

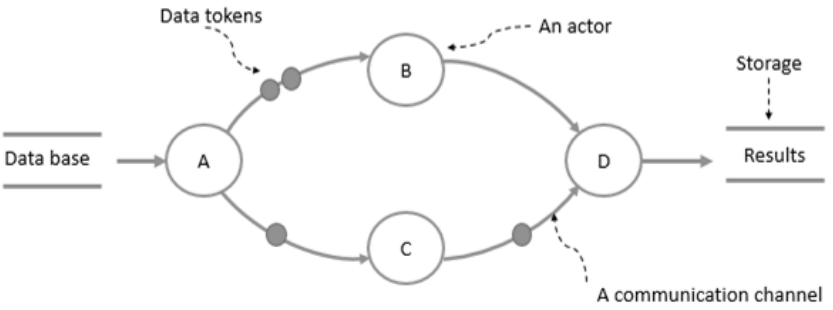



Figure 2.16 A simple example about dataflow diagram

A dataflow model uses only some basic symbols to present the operation of the system as illustrated in Figure 2.16. Functions are call actors denoted by circles which may be data sources or data sinks. File and database are storages to store data denoted by a special symbol in DFD. A transmission or a **firing** is a data transmission from a data source to a data sink (other actor) through a communication channel denoted by an arc in the DFD. Communication channels are

always FIFO queues (buffers) to synchronize data between actors. All data transmitted by actor in a communication channel (arc) in each computation is call a token and an actor can fire only one token but consume many tokens in one computation. As shown in Figure 2.16, actor function B needs two tokens from actor function A to generate only one token.

A characteristic of Dataflow model is the ability in sufficiently allocating the size of buffer memory in communication channels and scheduling operations between actors. This feature is very useful because time delays for computations in actors can cause the full and loose data in communication channel. There are also some fire rules in special DFD to manage size of buffer and scheduling the queue of flow for example the rule in KPN (Kahn Process Network) and SDF (Synchronous Data Flow) which we will present in next sections.

### 2.3.4. Kahn Process Network

Kahn Process Network (KPN) introduced by Gilles Kahn in 1974 [50] is the most general model in expressing streaming-based media and signal processing application. This is one form of process networks, which models a system as a collection of concurrent communication streams of data through FIFO channels. However, nodes KPN (actors) are arbitrary sequential programs that communicate through FIFO channel using **blocking read** and **non-blocking write** rule. Thus, during a processing in KPN nodes can either be waiting for the input or doing computations. When a process reads data from a channel, it will create a read request and block its computing until the data is available. In contrast, **non-blocking write** means the writing immediately works and returns data to output channel. Such a simple mechanism ensures the correctness of the mathematical condition of the process, which allows a KPN process does the computation and writes out data right after it receives input data.

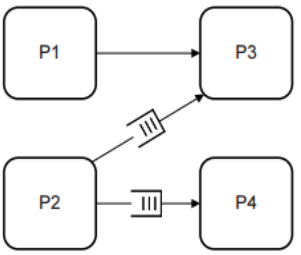



Figure 2.17 An example of Kahn Process Network

A KPN is not broken during the computation so that it is suitable for continuously executing application. However, non-blocking writing may cause the unboundedness problem in FIFO channels. For example in Figure 2.17, processes P2, P3, P4 in this KPN are connected via unbounded FIFO queues created by infinite buffers. If processes P1 and P2 produce tokens at a faster rate than they can be consumed in processes P3 and P4, tokens will accumulate on the FIFOs (buffers) that may unbound the limited physical memory in the system. Therefore, the

rate and the order of processes in network should be determined to reserve enough memory needed for system.

### 2.3.5. Synchronous Data Flow

Synchronous data flow (SDF) introduced by Edward A. Lee in 1987 [51] is a special case of dataflow which the number of tokens produced and consumed in each node are fixed numbers. Thus, in such a data flow model, buffer size in each communication channel and the schedule can be set at compiling time. Figure 2.18 is a simple example of SDF model where function A transmits data to function B. SDF diagram adds to basic dataflow diagram with some important information. The number of tokens (or data) consumed or produced when the node fires, are noted at the start and the end of arcs. Delay unit in each communication is noted in the middle of arcs and the execution time of the function is noted inside each node. These values may be not a fixed number because each function (actor) may be complex and has some conditional processes inside its algorithm. However, in scheduling purposes, the worst value can be investigated and be indicated on the SDF mode. Finally, name of the channel (arc) may be presented in a square node beside the arc.

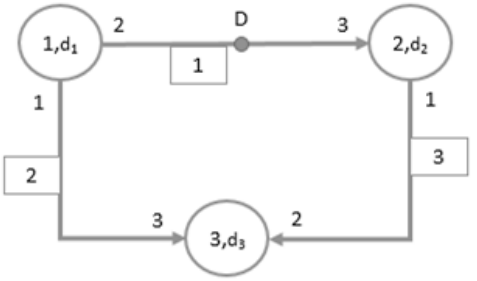



Figure 2.18 An example of a SDF diagram

Additional information does not make the SDF more expressive but they are useful for computing the buffer size and creating the static schedule at the compiling time. Moreover, the independence between actors helps for exploring many architectures for the system from one processor system to multi-processors system [52] [53] [54]. Verifying unboundedness on communication buffers, optimizing memory resource and optimizing performance of system can be statically set in compiling time in SDF development environments.

#### Unbound memory requirement

A schedule of system should define the optimum order of functions to be invoked for the unbound memory requirement. Lee and Seshia [42] mentioned a simple rule to define a satisfied schedule for an SDF model system by balance equation: $q_A.M = q_B.N$.

With assuming that node A produces M tokens and node B consumes N tokens each time they are invoked. In addition, $q_A$ and $q_B$ are the iteration numbers of node A and B invoked in the schedule. This equation must be correct for all arcs in SDF diagram. Such equation was also introduced by Lee in 1987 [51] in more general theory to verify the correctness and scheduling of SDF model. Lee stated the topology matrix of SDF model where $(i, j)^{th}$ entry in the matrix is the amount of tokens produced by node j in arc i each time it is invoked. For example topology matrix of SDF model in Figure 2.18 is:

$$\Gamma = \begin{bmatrix} 2 & -3 & 0 \\ 1 & 0 & -3 \\ 0 & 1 & -2 \end{bmatrix}$$

A vector q contains the iteration number of each node in the schedule. Thus, a schedule satisfies the unbound requirement when Γ*q gives a zero vector. That means balance equation of LeeSeshia is satisfied in any arcs of SDF model. In example Figure 2.18, q can be easily defined:

$$q = \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$$

### Memory resource optimizing

Two criteria in scheduling of system are optimizing performance and saving memory resources. In order to save memory resource, Bhattacharyya et al. 1999 [55] proposed a way to compute the maximum number of token in a schedule with assuming that all tokens have the same memory size. Such an approach initially finds all possible schedules for a SDF system then compute the necessary buffer size for each communication (arc) in this schedule. In example in Figure 2.18, there are two possible schedules are S1 = {1, 1, 1; 2, 2; 3}, S2 = {1, 1, 2, 1, 2, 3}. Then max-tokens in arcs are computed by: Max-token (1, S1) = 2x3 + 1 = 7 where 3 is the times the actor 1 is invoked; 2 are the amount of tokens it produced each time and one more token in the buffer size to avoid the unbound memory problem. Similarly, max token of channel (2, S1) = 1x3 +1 = 4 and channel (3, S1) = 1x2 + 1 = 3. Then maximum buffer for schedule S1 equals total number of tokens in arcs, which is 14 tokens. However, in the same computing, S2 needs only 12 tokens. Therefore, schedule S2 is more economy than schedule S1 in memory consuming.

### Performance optimizing

Optimizing performance of system depends on both schedule and execution time in each node. Because execution times in nodes do not change by the time so that performance of system implemented in one CPU architecture cannot optimize. However, in multiprocessor system, pipeline technique can be used to accelerate the performance. In such case, sharing memory must be used in multi-processing to exchange data between many processors. Unfortunately,

sharing memory management may limit the number of processing for SDF system. In SDF model, sharing memories are FIFOs or queues in asynchronous mode, which mean the read, and write to FIFOs are not synchronous. However, in order to data integrity, policy of FIFO access does not permit two writing operations or two reading operations at a time. This rule allows when there is a reading to a FIFO; the FIFO will turn to busy status and does not allow other reading until the current reading finishes. Therefore, in example Figure 2.19, we cannot schedule three functions (nodes) 1 to run parallel in three processors because they cannot write data to one sharing memory (arc 1) at a time.

Other requirement in a parallel SDF model is the firing rule of actor. It defines that an actor can fire when and only when there are enough tokens consumed in the input channel. That means actor 2 can fire only when actor 1 fires two times to produce enough three tokens. As the same rule, actor 3 can fire when actor 1 has fired three times and actor 2 has fired two times. Then the possible optimum schedule for example Figure 2.18 can be statically set to six time units as Figure 2.19 assuming execution times of nodes 1 and 3 are one time unit and executing node 2 takes two time units. Two functions 2 can run in parallel on two processors but they cannot start in the same time. That is because reading the FIFO in channel "subtask 1-subtask 2" may take less execution time than one time unit.

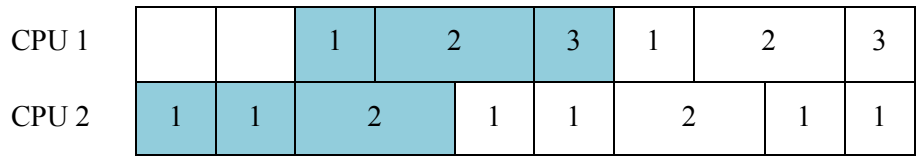



Figure 2.19 A schedule solution for example in dual processors architecture

## 2.3.6. Structured Data Flow

Structured Data Flow is an extended model of Dataflow in order to make DataFlow become a visual programming language, which can be used effectively by a broad range of programmers in different programming skill levels [56]. Such a DF model contains all well-known programming structures such as while loop and for loop structures, case structure, sequential structure or if then else structures as shown in Figure 2.20. These features thus solve the limit of dataflow model in only applying data processing application. Structured DF now can model a mixed control and data processing application in reactive systems. Because of control flow in structured DF, the firing rule in actor functions now can be programmed to fire in complex algorithms. As the principle of KPN, sinking actor only run when there is at least one data token in the communication channel. This process then writes a data token to output channel immediately.

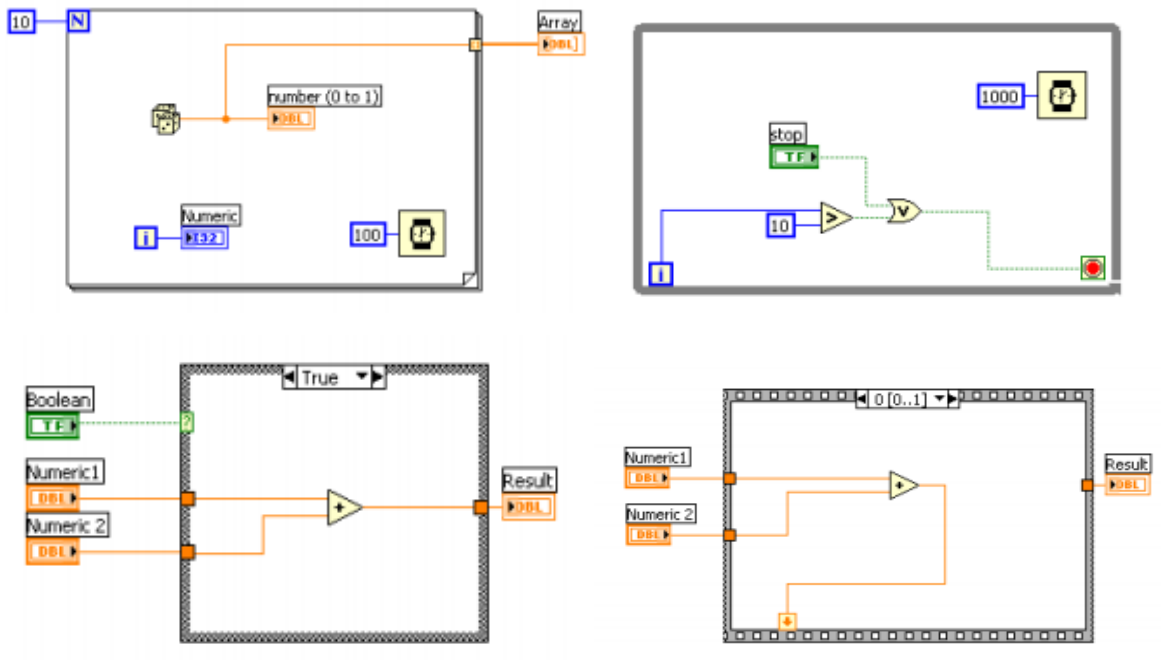



Figure 2.20 Some examples of structured DF model in LabVIEW: for loop, while, if then and case structure model

From 1986, Structured Dataflow became Graphic programming or G programming in LabVIEW- a commercial development environment of Nation Instruments Company for developing software or embedded applications. LabVIEW gives system designer an ideal tool to develop an executable specification with keeping the benefit of dataflow model in modeling the activity of a system. Current version of LabVIEW supports a huge actor functions in library for a large range of application. An interesting feature of LabVIEW is the GUI in its design environment allows developers interact and visual analyze algorithms in the developed system.

### 2.3.7. Reactive Process Network

Requirements of real-life application classify embedded system into two main behaviors: control-dominated and data-dominated behavior. A finite state machine (FSM) which requires very quickly reaction to random time events and irregular control signals is a famous model to define control-dominated system. Such systems mainly focus on how the processes work in sequential automatically (Moore FSM) or relating to input triggers (Mealy FSM). Thus, the operation of system is organized to many states where changing between states presents the algorithm of system. In contrast, data-dominated systems focus on data processing and reaction time is less important. Such systems can be found in DSP systems with Fast Fourier Transform (FFT), Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) functions. Output data of system are the results of input data after passing many transformations, which are implementations of DSP algorithms. Dataflow model is often used to model data-dominated systems. Such model type is can provide a clear and pure view of overall activity of the system

where each DSP process is represented by a node in the model. However, most of real modern systems contain both control-dominated and data-dominated behaviors with a composition of plural processing types: control, event processing and streaming processing, which should be modeled by a specific MoC named Reactive Process Network (RPN) [57]. Some modern applications such as multimedia applications with stream processing audio, video data or our NIALM application in this research are typically objects for using RPN model.

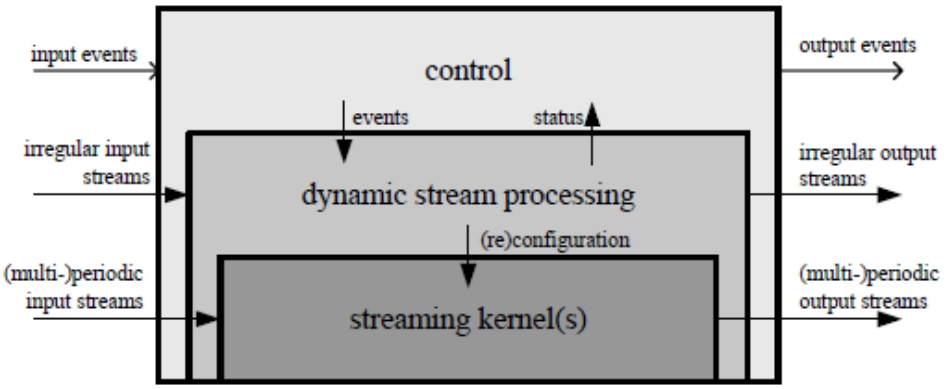



Figure 2.21 Structure of a Reactive Process Network

Figure 2.21 illustrates a structure of such applications. Stream processing is the main part of system. This kernel must process periodic input stream in highest priority. This input stream may be a sequence of audio signal from a microphone sampled at the rate 144 kHz, a video data from a camera with 24 frames per second or sampled current and voltage value in defined sampling rate. These periodic data come infinitively to the system so that they need to finish processing it before the new data coming. The stream kernel will process periodic input data by packet depending on the algorithm. Packet can contain one data or many data, which are temporarily stored in a communication channel (FIFO) between processes, then decide the size of memory consumed by process. Moreover, the duration between two continuous packet data transmissions will affect the timing constraints for this kernel process. All characteristics described above make SDF is a good candidate for this kernel process because this model has efficiently static scheduling and allocating buffer memory.

Irregular input streams and input events are unpredictable data, which is impossible to define the moment they come. They are important control signals or messages to control or reconfigure the system that can stop the streaming process if they are signals for system reconfiguration. Other processing to control signals can run parallel with streaming process if it is necessary. Therefore, KPN can be used in dynamic stream processing layer and FSM can be used in control layer with the requirement in user interaction. Such MoCs are capable to describe a data dependent behaviors and fully asynchronous processing dynamically.

## 2.4. Languages and development tools

In a design flow, system goes through all abstract levels, which require the use of various languages and development tools. As we mentioned earlier, abstract level of an embedded system can be described through untimed model, approximate timed model and cycle-timed model. At the specification level, model relies on untimed models with no consideration of timing or delay during the simulation. The objective of specification model is to give the clear relationship between inputs and outputs for faster functional verification.

High-level programming languages are often used to model the untimed model such as MATLAB, Simulink, and LabVIEW. These languages are widely used in heavy data processing domain and complex reactive system such as FSM, KPN model, SDF computational models. At the low-level, HDLs (Hardware Description Language) such as VHDL, Verilog are very popular in developing VLSI circuits. However, complex SoCs with multi-core and the interconnection between shared resources and components need a special language such as SystemC to specify the architecture of system.

### 2.4.1. System design tools

MATLAB of Mathworks is one of the best mathematical tools to analyze and to visualize data processed in a computation or algorithm. MATLAB uses interpreted code called M or m-code to run directly from the working place in MATLAB environment. It strongly supports vector and matrices mathematic with the accuracy of datatype up to double-precision 64 bit-floating point. Two main cores MATLAB and Simulink and many toolboxes with prebuilt functions can be used to design applications in various domains including filter design control system design, robotic, RF design, etc. The capability of MATLAB in visualizing data help developers quickly to analyze their algorithms; building the gold reference to verify system in lower abstract level. Moreover, MATLAB so far has Instruments Control Toolbox, which allows direct connecting to various instruments including oscilloscopes, function generators, signal analyzers, and analytical instruments. Thus high-level system can be verified the behavior in real input data before the implementation.

Another important tool in MATLAB is embedded code generation. This tool can automatically generate C, C++ code from model designed in MATLAB or Simulink so that it increases the productivity and quality of system by addressing specific architecture and industry certification standards. In order to accelerate algorithms in FPGA or ASIC hardware, HDL coder tool gives developer a very powerful capability in automate VHDL or Verilog code generation from M code. Such worth tools save a lot of time consumed by programming and verifying FPGA or ASIC applications in conventional way. Alternatively, besides using MATLAB, developers can use open-source and cross-platform language named Scilab, which also supports well modeling system level in a syntax close to MATLAB. This interesting language can be used in signal processing, image enhancement, matrix calculation optimization, feedback control system.

### 2.4.2. Model-based design tools

#### Simulink and System Generator

MATLAB Simulink is a model-based design approach for designing and analysis system with rich toolsets and libraries supported by MATLAB. In 1999, Robert D. Turney et al. [58] introduced a new approach to generate HDL netlist design from the system level tool MATLAB Simulink using Xilinx SystemLinx blockset. In Simulink, SystemLinx is an interface to Xilinx Core Generator to generate HDL codes automatically. This interesting tool also supports bit-accurate simulation and even in multi-rate clock system. In 2007, Markovic et al. [59] used Simulink and Xilinx System Generator (XSG) to explore the best architecture for their algorithms. XSG (new version of SystemLinx) is a collection of Xilinx function blocks optimized for DSP design in MATLAB Simulink to model a system. Beside the capability as SystemLinx, this tool also supports the hardware co-simulation. This important feature allows a hardware algorithm implemented in real FPGA board can be controlled and analyzed under Simulink environment [60].

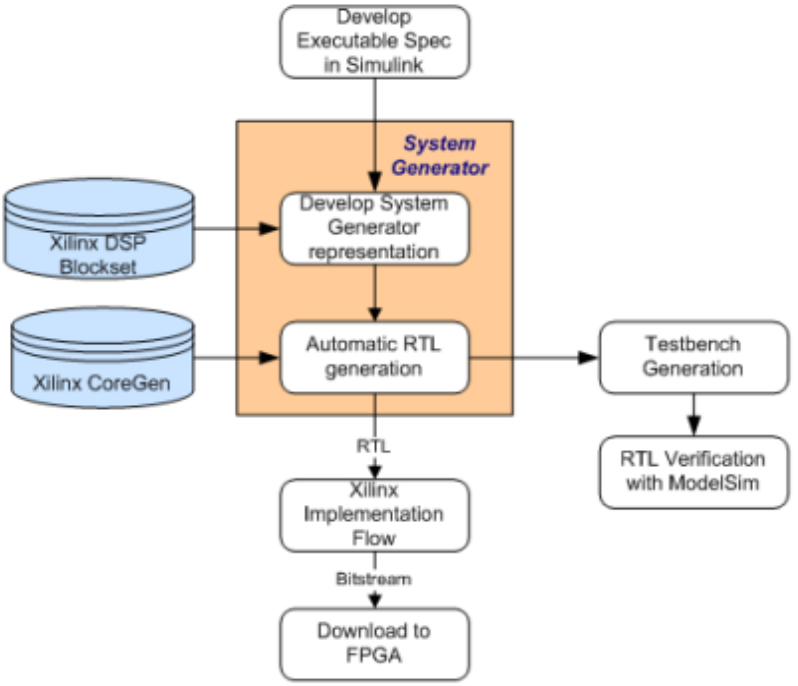



Figure 2.22 System Generator design flow

As illustrated in Figure 2.22, the design flow of System Generator starts from developing the executable specification in Simulink using System Generator blocksets. Logic blocksets supported by System Generator vary from basic logic (AND, OR, NOT) to arithmetic operator blocks, input output interfaces, DSP blocks and even the microprocessor blocks. This dynamic system can also use traditional blocksets of Simulink for visualizing and analyzing functions. One system is modeled and verified in Simulink, RTL codes and testbench codes can be automatic generated to test in RTL simulators such as ModelSim or ISE Simulator. Finally,

Xilinx Implementation Flow creates bit-stream file from these RTL codes, which is able to download to FPGA target to test system in real platforms.

Figure 2.23 shows a simple example of System Generator in Simulink with blocks in blue square are Simulink models and blocks in red square are System Generator models. System Generator symbol is the configuration model for configuring the HDL generation and target implementation. In order to run in the target, the Xilinx synthesis tool (ISE) is invoked to generate the bit stream file for FPGA target implementation.

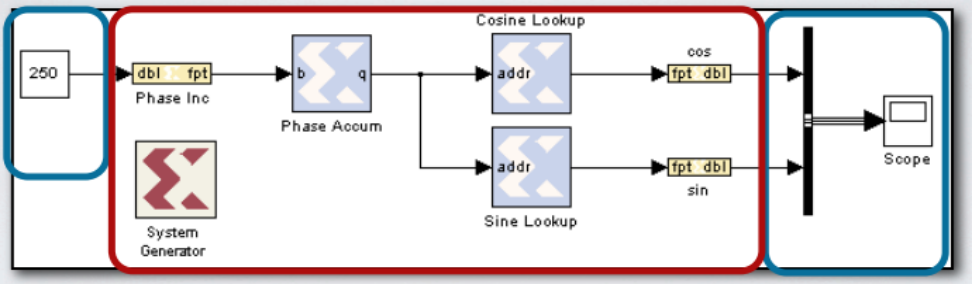



Figure 2.23 A simple example of modeling system with System Generator

System Generator also supports the hardware-software co-simulation mode which is very useful for verify the correct of complex DSP algorithm in real platform. From the designed model, System Generator can generate HDL codes, configuration files and do the implementation to FPGA board. It also manages the interaction between designed model in Simulink and the running system on board by Jtag or Ethernet communication. Moreover, a completed system including microprocessor and peripherals can be modeled in Simulink by importing an embedded design from Xilinx Platform Studio through EDK Processor block. Developer can also monitor signals in input output port and internal signals using verification tool Chip Scope Pro.

Although System Generator can be used in all design levels from specification, system design to architecture and RTL implementation, this software is only useful for specification and system design. In architecture design and RTL implementation (circuit design), this approach focuses only on functional verification then gives developers the feel in algorithm running in real environment so they can proceed to the resources and performance estimation.

### LabVIEW and LabVIEW FPGA

LabVIEW, a term of Laboratory Virtual Instrumentation and Engineering Workbench, is a graphical programming environment based on the dataflow paradigm of National Instruments Corporation. LabVIEW was designed for data acquisition and control systems with a rich set of supported hardware and prebuilt software function blocks. LabVIEW environment can run the simulation in host computer as executable specification or can implement to specific NI target to run the test in real devices. Moreover, LabVIEW keeps the connection of program running in

target with GUI in Front Panel, which provides an attractive feature to make the verification in real device easier in the visual way.

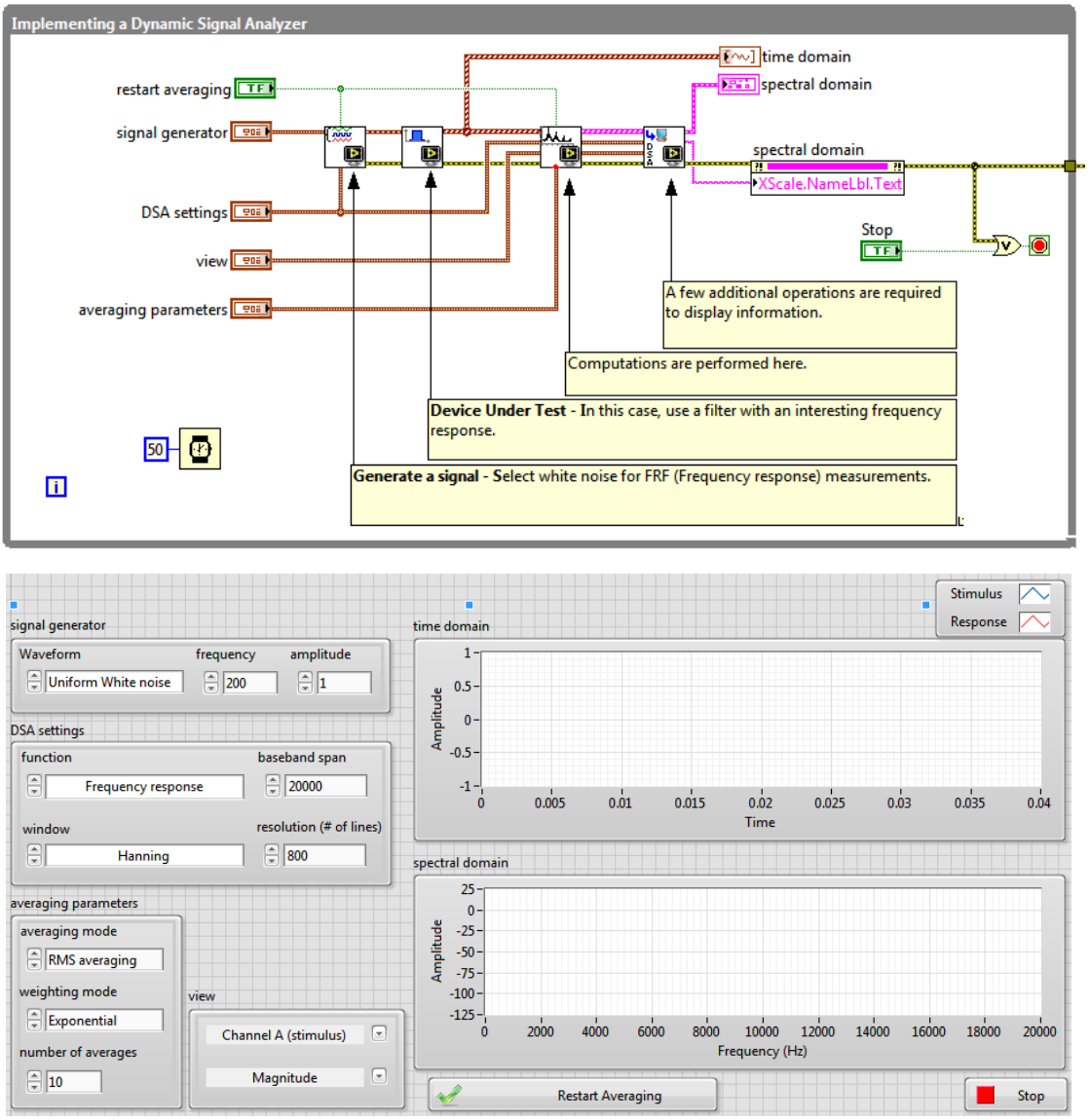



Figure 2.24 A dataflow model in LabVIEW is composed by an activity dataflow model in block diagram and a GUI in Front Panel

As illustrated in Figure 2.24, a LabVIEW model is an integration of two separate components:

- **Block diagram** object specifies the behavior of a model in two main formats: dataflow in G code and MathScript code. G code is the original coding method in LabVIEW. This model-based design approach is more popular than MathScript code because it can reuse thousand functions of LabVIEW that NI optimized for various hardware platforms. Therefore, G codes are often used to prototype the system quickly. In contrast,

MathScript provides a text-based environment to develop the text-based signal processing or mathematic codes, which can work on Windows or on Real-Time (RT) targets. MathScript also has a rich function library in matrix computations, mathematic, DSP, statics and plotting functions etc. as MATLAB.

- **Front panel object** is an advance feature of LabVIEW to support a user interface of Block diagram. The Controls Palette library of LabVIEW has a lot of control components and indicator components, which are used to develop complete application with expert GUI in window, or to test functions. Some examples are control components such as knobs, buttons, dials etc. and indicator components such as graphs, LEDs, meters etc.

LabVIEW FPGA Module is an extra tool of LabVIEW to develop LabVIEW models (VIs) that run on FPGA targets. This feature is actually a high-level synthesis language like the System Generator in Simulink with IPs supported by Xilinx. Some of them are similar to normal functions of LabVIEW but it can implement either in software in desktop or embedded SW in real-time target (MCU) or in hardware in FPGA by selecting the right target to implement. Using a unique environment allows LabVIEW FPGA is the best the hardware software co-development tools to rapid prototype complex system with SW and hardware acceleration. Therefore, accelerating the performance of embedded systems using FPGA is more and more popular in LabVIEW product now.

**Scicos and Scicos-HDL**

Scilab has a package Scicos, which is equivalent to Simulink from Mathworks. This tool has been used as a Rapid Control Prototype environment to develop a real-time motor feedback control system using the open real-time RTAI Linux, RTAI-Lab package [61]. This system can run in several embedded architectures x86, x86_64, PowerPC, ARM, m68k. Scicos has Scicos-HDL package, which is a Digital System Design tool similar to the System Generator in Simulink [62].This tool has basic logic models and some standard combine logics such as Multiplexor, Encoder, Decoder, BUS etc. and can convert the model-based design to Verilog, VHDL or SystemC code. These RTL code then can be synthesized and download to run in FPGA target by any synthesis tools. Unfortunately, Scicos-HDL is limited in Scilab version 5.x.x, and complex logic libraries such as Xilinx IP core library does not support this tool.

### 2.4.3. Architecture design tools

Some conventional tools for architecture exploration are HDL languages such as VHDL, Verilog, System Verilog and SystemC. The following example presents a simple application of Verilog HDL language in modeling a counter in various architectures with time functional model and simple share variables are shared between processes. In the one processor-architecture model 1, delay 10 time-unit (#10) is the estimated time to process one loop. This

value can be get from the estimated instructions generated from the code and the frequency of the processor. Model 2 specifies the "enable" signal (to control the counter) in the CPU 1 and the CPU 2 processes the counter. Different system frequency can cause the missing of the control signal if the "enable" signal switches its status during a counting activity in CPU 2. Last model can solve the problem of model 2 with the counter is accelerated by hardware with very fast system clock and no sequence instructions processes as in processor.

Model 1

```
module adder_single_cpu();
reg [7:0] counter;
reg enable;

//CPU 1
initial begin
   counter = 0;
   enable =1;
   while(1) begin
      if (counter == 256) begin
         counter = 0;
       else
         if (enable = 1) begin
            counter = counter + 1;
          end
       end

       #10   ;  // wait 10 time units
   end
end
endmodule
```

Model 2

```
module adder_dual_cpu();
reg [7:0] counter;
reg enable;

// CPU 1
initial begin
   enable = 0;
   # 100  enable = 1;
   # 100 enable = 0;
end

//CPU 2
initial begin
   counter = 0;
   while(1) begin
       if (counter == 256) begin
          counter = 0;
       else
          if (enable = 1) begin
               counter = counter + 1;
          end
       end
      #10;
   end
end
endmodule
```

Model 3

```
module hw_acceleration();
reg [7:0] counter;
reg enable, clock;

// Hardware acceleration
always @ (posedge clock)
if (counter == 256) begin
      counter = 0;
 else
       if (enable = 1) begin
            counter = counter + 1;
        end
end

// CPU
initial begin
    enable = 0;
    clock = 0;
    counter =0;
    # 100  enable = 1;
    # 100 enable = 0;
end

// Clock generator
always #10 clock = ~clock;

endmodule
```

Multi-cores SoC systems become more and more popular in the requirement of high computing application so far. Modeling such a complex system cannot be handled by standard RTL programming language like Verilog, VHDL because the gap between hardware and software development is so big. Therefore, modeling such a SoC system needs to use more powerful but still lightweight HDL languages which are able to model and run the simulation for both hardware and software components. Moreover, communications between components in multi-cores SoC are no longer a bunch of wire or buffer. Each core in this system must share communication channel and resources for more efficient performance, which is a very hard task and requires a heavy design in RTL programming language. SystemC addresses above problems [63].

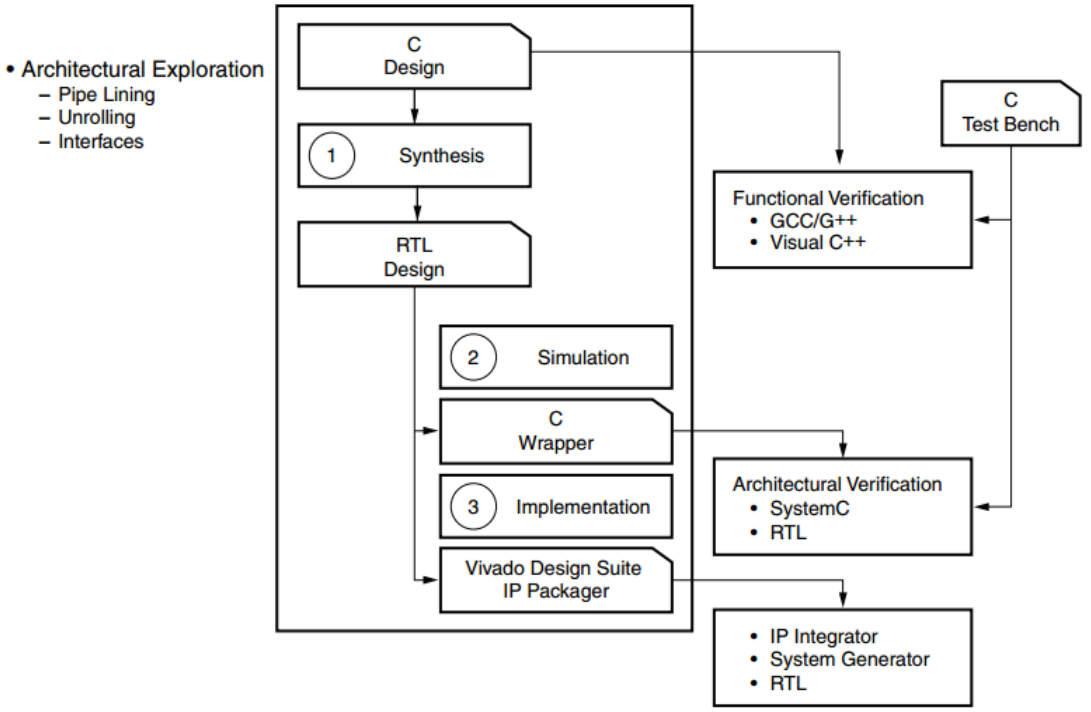



Figure 2.25 Vivado HLS design flow in exploring system in various architectures [64]

Synopsys introduced SystemC in 1990s intending to replace Verilog and VHDL as input language of logic synthesis – a High Level Synthesis (HLS) language [30] [41]. Xilinx uses SystemC in Vivado High-Level Synthesis for their newest device such as seven Series, Zynq and Ultra Scale devices [64]. This tool accelerates IP creation from both C, C++ and SystemC codes without the need of RTL languages. Verifying system in a high-level language gives some benefits. First, developers just need to focus to develop algorithm in very friendly language C, C++ or SystemC. Second, simulation in high-level language is much faster than in cycle accurate level. Third, using Xilinx IP generated from the tool is faster and more reliable than the RTL manual coding. Fourth, testbench in C, C++ can represent the SW task running in processor, which can interact to the HW in the SW/HW co-simulation. This tool helps *architecture definition* in the HW/SW partitioning when we need to decide which processes are in hardware or software. Figure 2.25 illustrates the architecture exploration capability of Vivado HSL design flow in exploring system in various architectures such as pipelining, unrolling and interfaces.

In 2001, SystemC 2.0 introduced the Transaction Level Model (TLM) for a complex SoC model with address management feature for bus model, creating read/write transfers in bus master mode, modeling memory, timer and interrupt controller with a thread in the bus master handling the interrupts. SystemC TLM supports system design and verification in untimed accurate model simulation. Simulating this system in SystemC was around thousand time faster than the equivalent RTL simulation [63]. SystemC is the best language for architecture development. In other words, SystemC can model HW as well as SW components and support the concurrent HW/SW development based on TLM as shown in Figure 2.26.

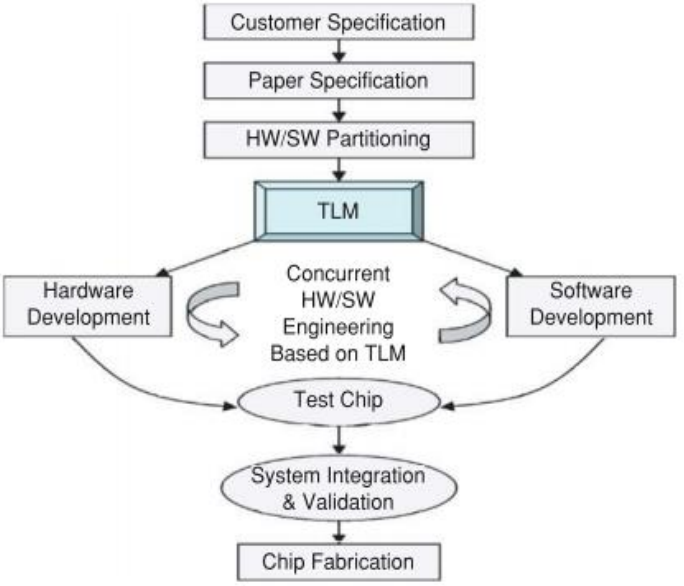



Figure 2.26 A SoC Design Flow using TLM [63]

### 2.4.4. RTL design tools

Schematic CAD tools can be used to develop small-scale integrated circuits (IC) but very difficult to handle for a very large scale integrated (VLSI) circuit design. Hardware description Languages (HDL) is a technological-independent approach in IC design. HDL so far replaced conventional RTL CAD in developing most of digital circuit systems. Such languages can specify a system in many abstract levels for example an one bit ADD operator can be defined in high level c = a +b as C language or directly in low level c = xor(a,b) or c = or(and (a, not (b)), and (b, not(a))). However, high-level designs in HDL, after satisfying the function verification, are converted to RTL low-level circuit. In implementation step, some synthesis software like Synopsys Design Compiler will translate RTL model to gate-level netlist, which is finally placed and routed system to connect CMOS gates to implement the designed RTL model in real hardware.

Verilog [65] and VHDL [66] are the most popular industrial-dominant HDL languages now. Phil Moorby designed Verilog in 1983 in a context that developing large-scale IC using schematic capture software is still a hard work and expensive. Cadence Design System bought Verilog in 1990 and put it become the IEEE Standard 1364-1995 from 1996. At the same time, US Department of Defense developed VHDL and strongly supported it to be standard IEEE 1076-1987. In some applications, system may need to interact with analog parts in the mixed analog and digital systems. In order to simulate some analog peripherals such as sensors, which do conversions between two analog physic parameters, we need to use continuous model. SystemC-AMS, VHDL-AMS, Verilog-AMS are good languages to model some blocks like PLL, A/D converter, etc. Detail discussions about many use-cases of VHDL-AMS and SystemC-AMS can be found in [67], [68].

HDL languages are often used in traditional FPGA design flow as presented in Figure 2.27. The hardware circuit is specified by a digital circuit schematic or by a hardware description language (text-based method). Design then needs to be verified in simulation tools such as Xilinx Simulator, ModelSim etc., which support the functional and timing simulation in various levels: behavior level, RTL level, or timing level.

- Behavioral description is the highest design level, which is used to specify the operation of an algorithm, the activity of a system or the protocol of a communication bus. In more detail, behavior model of a system can be the function description, a FSM with general type of input, output and internal variable such as integer, float, char, etc. A behavioral model may be not synthesizable and only be used to simulate and verify the activity of the system.

- RTL description requires more detailed description in a synthesizable structure where system is described as combinatorial or clocked processes. RTL design contains variables and input, output ports in bit type and defined size such as signals or buses of signal (of combinatorial processes), or registers or buses of register (for clocked processes).

- In FPGA implementation, timing level description is the result of synthesis and mapping process of a design in a specific platform. Synthesis tools convert RTL design to other RTL design, which uses specific component of selected platform such as LUT, BRAM, and DSP blocks to model the circuit. Such model thus contains detail real delay information as in the real system that can be used to do the last verification before implementation. System verification in this level is slowest but it is still faster and signal traceable than on-target testing.

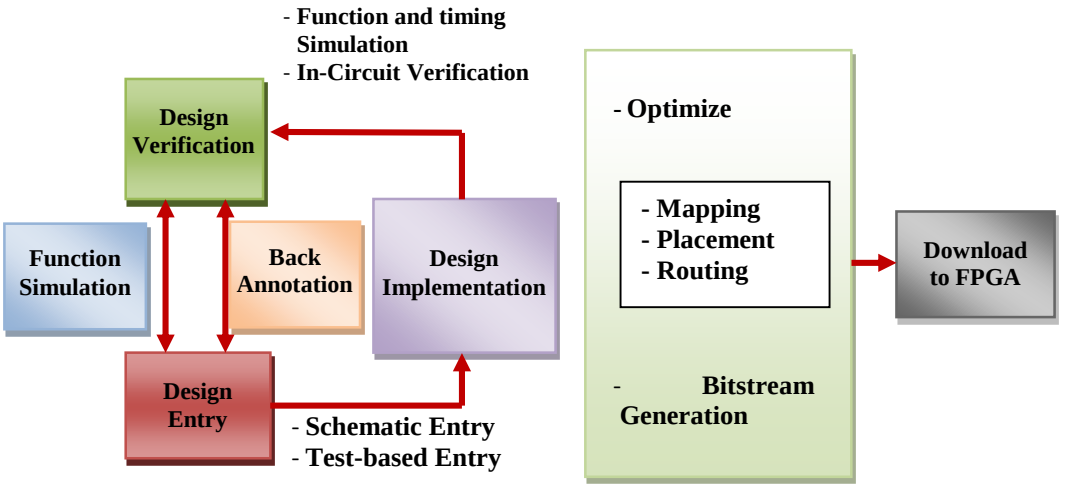



Figure 2.27 Traditional FPGA design flow in RTL

After the functional and timing verification, many optimization steps can be done in mapping, placing and routing using compilation tools. These tools then generate a bit stream file that contains the configuration of the system to be able to download into the FPGA target. Finally, verifying system in real target can be done by specific tool such as ChipScope Pro of Xilinx, which works as the logic analyzer equipment to trace necessary signals.

This FPGA traditional design flow approach is still popular used to design and optimize a system in performance and power usage because developer can manually and directly drive the synthesis, the mapping and routing process. However, in complex DSP applications with hardware acceleration, this method is not convenient because it cannot visualize the analysis's results of DSP's algorithms and hardware-software cooperation.

After investigating many languages and tools, which can be used to develop Reactive Process Network based on SoC, we summarized them according to the design level they can support in developing an embedded system in SoC, as illustrated in

Figure 2.28. It shows that the LabVIEW FPGA may be the best tool for hardware software co-development. This language supports well model-based design approach many analysis functions, GUI, IPs library to quickly develop a system from the specification to prototype.

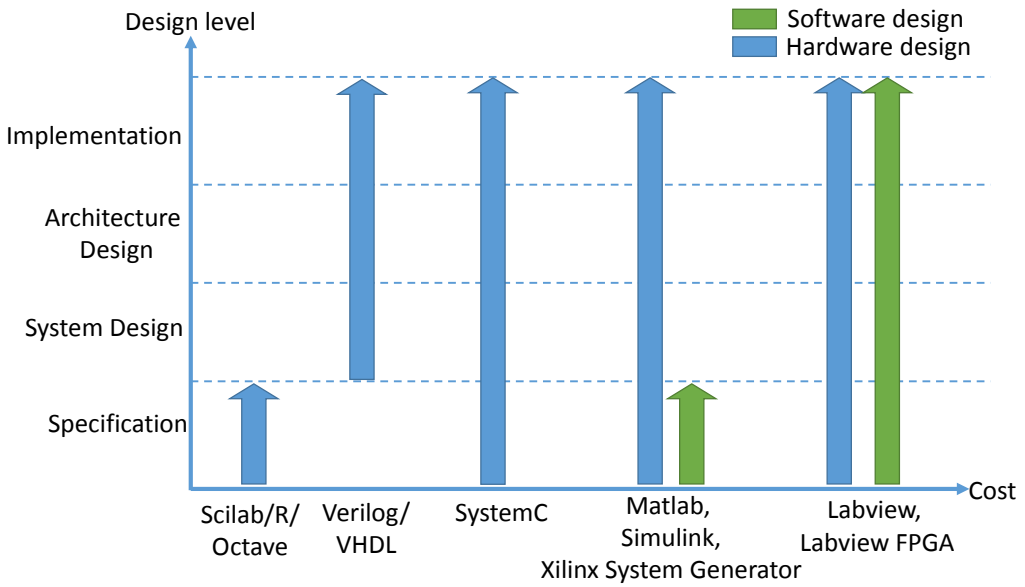



Figure 2.28 Comparison of hardware and software design capacity in some popular languages and tools

## 2.5. HW SW codevelopment approach for rapid prototyping

The board-based approach consumes longest time because it has to process each step of the system development in sequence. That means only when system designer finish their work, hardware designer can develop the real platform for the software designer to implement the

application. Some system problems may only occur after the first version of system is complete and system designer, hardware designer and software designer may have to repeat their process to repair that system problem. Therefore, such traditional approach may be too expensive and it suits only to the simple application. Virtual platform-based approach can overcome the drawback of board-based approach in complex system by using an abstract model of various architectures so that software designer can validate their application running in the virtual platform before running in the real-platform. However, this approach has other drawback. It still takes a lot of time to transform system designed in such approach to the version, which can run in the real-system.

Solving above challenge is the objective of this research. In this section, we propose a model-based design approach to rapid develop a Reactive Process Network system in complex SoC using tools LabVIEW and LabVIEW FPGA. This approach is very interesting because system designers can do all the steps in developing a SoC system by him-self from system specification, architecture design to developing and prototyping the system. This method will be useful and economy for some people:

- Developers and researchers who want to quickly develop, analyze and demonstrate their complex RPN system in real SoC without the need to have hardware/ software design skills (HDL languages, C/C++ languages). With this approach, they just need to focus on developing their algorithms and system architectures that saves a lot of time and money.
- System developers, who are working on projects with difficult requirements in performance, power consumption and short time to market.

As shown in Figure 2.29, the design flow of this approach has three steps:

- System Specification is the first step to specify the overall activity of the system. In order to model the RPN system, both three types of MoC: dataflow, SDF and StateChart model are proposed. In advance, the RPN model developed in Labview can be executable to present better the overall activity of final system.

- The SDF model, which is used to model the data streaming processes in the system specification, can be used for architecture exploration. As shown in Figure 2.29, there are a few modifications in the codes for the Labview able to generate the code, which are runnable in selected architecture SoC. However, it requires changing the valid FIFO communication type in each target. The Target Scope FIFOs are used to transfer data between HW IPs in FPGA; the RT FIFOs are used to transfer data between SW IPs in processors; and the Host to Target DMA FIFOs and Target to Host DMA FIFO are used to transfer data between HW and SW IPs.

- Co-developing software hardware is the prototyping step in this system development approach. This step supports quickly convert selected architecture design from previous

step to prototype a real RPN system. Moreover, it is possible to improve the software of the system by using C text-based programing for software elements and to improve the hardware of the system using manual HDL coding or HDL coding generated by System Generator tool.

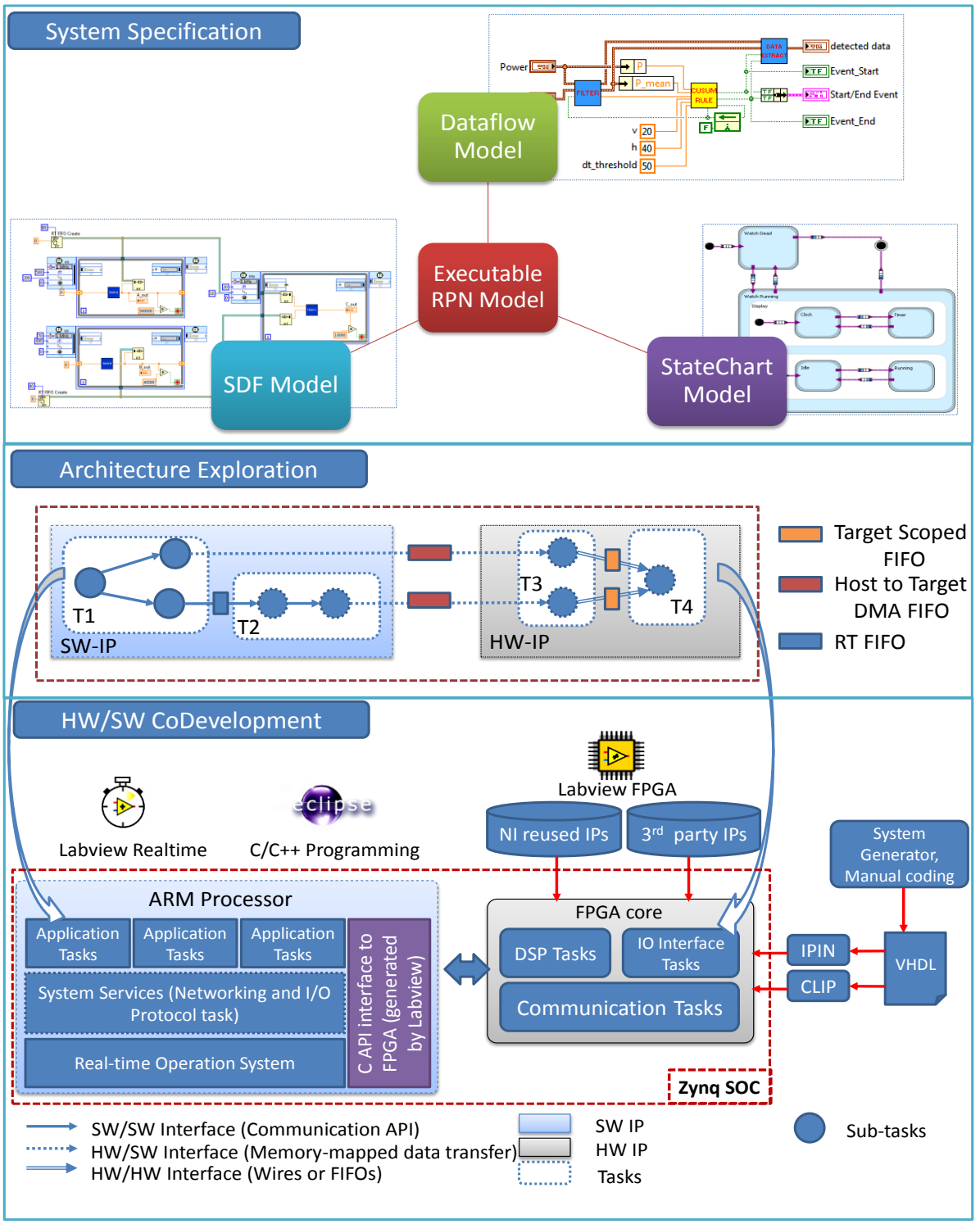



Figure 2.29 SoC development approach aiming to rapid prototyping RPN systems

### 2.5.1. Modeling executable specification of RPN system

#### Modeling an activity dataflow model

LabVIEW is a potential environment for rapid developing of a Reactive Process Network (RPN) as the NIALM system in our research. As shown in Figure 2.29, this approach models the executable specification using three MoCs in system design: modeling activity of system in dataflow, modeling event-based processes of system in StateChart and modeling Synchronous Dataflow model of data streaming process for memory allocation, scheduling and architecture exploration. That thanks to graphic programming approach of LabVIEW, which can also model almost exactly, some dataflow models based MoC such as Process Network (PN), Finite State Machine (FSM), and Synchronous Dataflow (SDF).

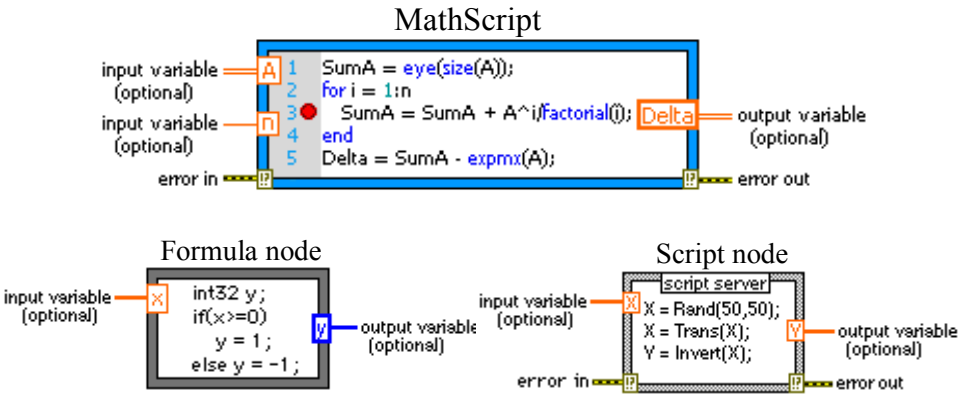



Figure 2.30 MathScript, Formula node and Matlab script node in modeling a function in LabVIEW

At the beginning, dataflow models are used to describe clearly the overall activity of system in processing and transmitting data through processes in the system. LabVIEW is not just a dataflow programming language; it also can create a GUI for the design, which is executable in a visual way. Besides graphic programming with LabVIEW, there are three tools can be used to model a process in text-based code: MathScript node, formula node and script node as illustrated in Figure 2.30.

- LabVIEW MathScript is a text-based environment to model a process using LabVIEW MathScript syntax. The MathScript syntax is an intuitive and logical syntax predominantly based on standard mathematical and computer programming term. We can even deploy MathScript node to real Real-Time target with built-in MathScript functions.
- Formula node is a convenient tool to evaluate mathematics and expressions in LabVIEW. Instead of drawing a For Loop, While Loop, Case structure as diagram, Formula Node uses C-like statement delimited by semicolons as a C code. This tool also supports following built-in functions: abs, acos, acosh, asin, asinh, atan, atan2, atanh,

ceil, cos, cosh, cot, csc, exp, expm1, floor, getexp, getman, int, intrz, ln, lnp1, log, log2, max, min, mod, pow, rand, rem, sec, sign, sin, sinc, sinh, sizeOfDim, sqrt, tan, tanh.

- A script node can connect to other software script servers such as MATLAB, Scilab to execute scripts written in syntax of these languages. The MATLAB script node requires a licensed MATLAB software installed in computer because it will invokes the MATLAB software script server to execute the scripts. However, developer can use open source Scilab script node provided free in LabVIEW website. Developer can also even import .m files to script nodes or export script nodes to .m text files. This is a good feature to validate the script codes in both MATLAB/Scilab and LabVIEW environments.

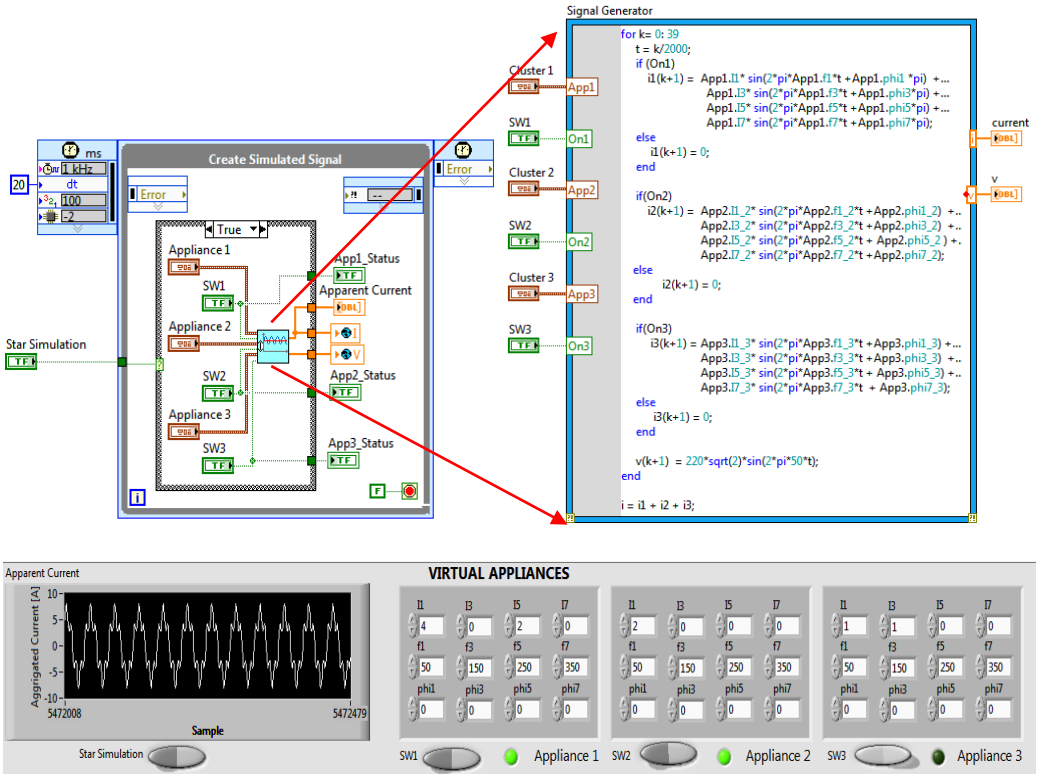



Figure 2.31 Executable electrical network model in LabVIEW MathScript with three virtual appliances

The example in Figure 2.31 presents the executable model of electrical network. Both loop structures in G code and MathScript are used in this model in order to emulate the sampling process in every 20 millisecond (50 Hz) for collecting a full electrical signal data in a cycle. GUI in front panel object is used to change current phase and amplitude of electrical network with three appliances working.

### Modeling a Synchronous DataFlow model

In a structured dataflow in Labview such as the while loop, for, if then, an actor in a dataflow design has to wait previous actors to finish their computation before being able to do its computation. Therefore, all actors in a structure dataflow work in the same sampling rate defined. Maximum sampling rate can be defined by the total latency through all actors in the dataflow.

In order to model a SDF model, LabVIEW uses FIFO buffers to transfer data between two asynchronous processes (actors). This is a special data sharing method using an amount of memory for data transaction between different VIs running on a target or on different targets. The first data written into FIFO will be the first data, which will be read and removed from FIFO. When the FIFO is full, Time Out signal is generated to block writing a new data until a space is available in FIFO after a reading. When the FIFO is empty, it will generate the timeout signal and the FIFO cannot be read until a data is available in FIFO after a writing. Therefore, these basic characteristics of a FIFO can control the writing and reading process so that it avoids the data missing. However, it still needs to define the size of FIFO depending on the operation of functions in the system. As we discussed in previous section, Synchronous Dataflow model can be used to organize the buffer size of FIFO and the schedule of processes in the system.

Example of modeling SDF model in LabVIEW is illustrated in Figure 2.32. Actors A and B working at rate F1=2 kHz and actors C working at rate F2 = 50 Hz are modeled as three single threads, which are working in parallel. This time loop structure of Labview allows designer to define the working rate, the priority level and the processor to execute each thread. The working rate of an actor is the frequency of the actor invoked during a second. Because working in different rates, FIFOs are used to transmit data types between these processes. The buffer size of FIFO is defined based on SDF rule in section 2.3.5: $q_A * M = q_C * N$ and $q_A = 2000$, $q_C = 50$ are number of times task A, and task C are invoked per second. Thus, we must select M=1 is number of token produced by Task A and N = 40 is number of token consumed by Task C. It is safe to select the buffer size 80 for FIFO communications between task A, task B with task C.

Figure 2.32 shows the scheduling of task A, task B and task C running in one processor (CPU0) platform which was monitored and recorded by ***RT Trace Time Viewer*** tool. In order to use this tool, the VIs model *Trace Tool Start Trace* and *Trace Tool Stop Trace and Send* must be used and connected to each process as shown in this figure. Default buffer size of this tool is 250000 bytes, which are used to store all operations of CPU relevant to the operation of these tasks.

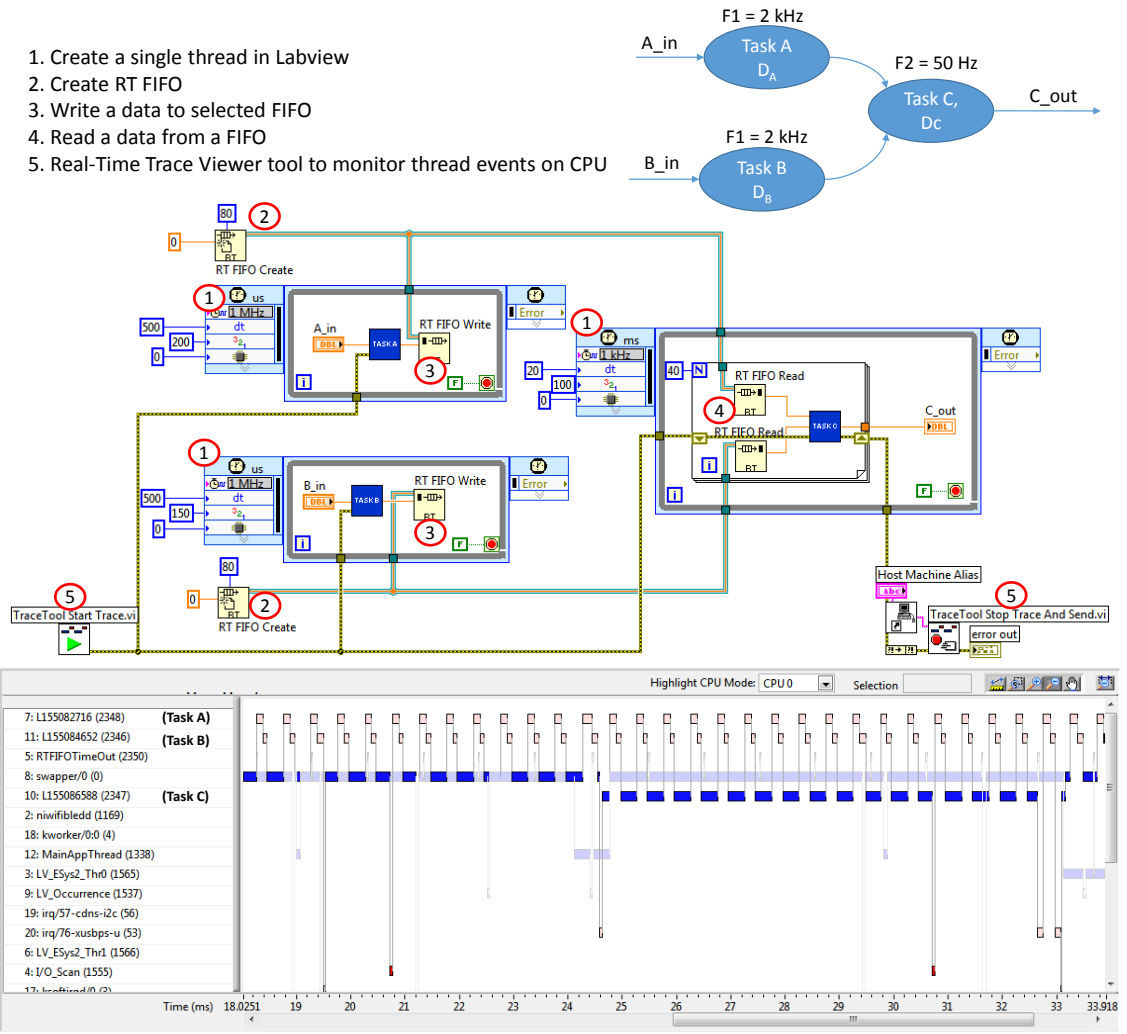



Figure 2.32 The actual activity of actors in a dataflow design

## Modeling a StateChart model

As a complex device, a system has to provide services required by users who intervene not only during its exploitation, like the power estimation service, but throughout its life cycle, from its conception to its dismantling (initialization, local and/or remote configuration, etc.). A service is usually defined as a procedure whose execution results in the modification of at least one datum in the device database, or/and at least one signal on its output interface [69]. In order to define the obtained values (outputs O), one has to describe the computations which are done, its internal configuration (parameters P), the variables on which they are applied (inputs I) and the required resources R (hardware, software). As illustrated in Figure 2.33, we demonstrated how the computations in a service like the estimating power can be expressed using a synchronous data flow diagram formalism following a functional decomposition.

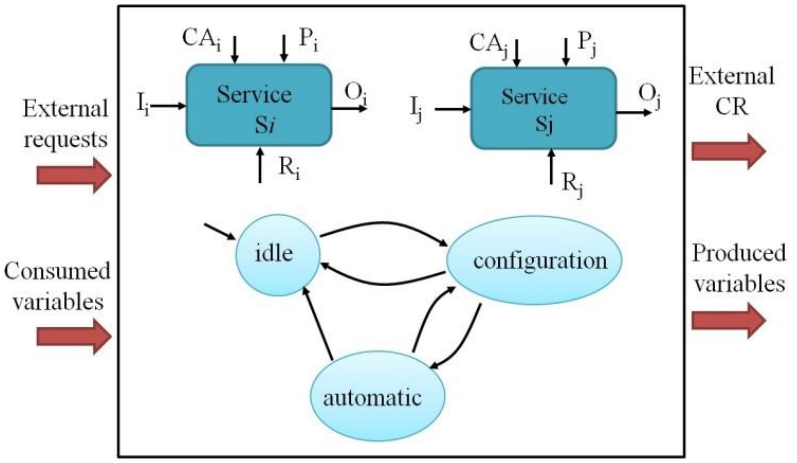



Figure 2.33 External model of the NIALM system

The service executions can be either dependent (precedence, mutual exclusion) or independent and concurrent. Services can have a limited duration or can end on the occurrence of simple or complex events (operator request, emergency alarm, etc.). Finally, these services can be organized according to user operating modes (USOM) is a coherent sub-set of services and contains at least one service and each service belongs at least to one USOM. The expression of such an external model of the instrument requires another computational model different from the one of SDF formalism. Some works in progress aim to express USOM and services using the Statecharts formalism available in LabVIEW [42]. In this formalism, USOM and services can be represented using macro-states. States inside macro states defining services will express the various states of a service (idle, running) with transitions expressing the conditions required for state changes (external events and/or condition guards). Entrance to the running state will launch the SDF procedure to make the computations like for the power estimating service.

Figure 2.34 presents a simple version of our NIALM system with three main services: power monitoring, database management and datalogger. Because services power monitoring and datalogger need to run all the time so that three concurrent states were developed. There is also a small login service to check the user and account of customer. From idle state, if there is a login action, system will move to login state to check input account and password with database. If the account is correct, transient Success will available and system changes to the Main macro state. Both three main services are initialized to idle state and start to run in parallel. Depending on the events from the GUI panel, each service will be activated or deactivated. When entering the Power-monitoring mode, two services Power monitoring and Data-logger run in parallel. However, in Database Management mode, the Data-logger service is disabled while Power Monitoring service is still running inside the system.

Once StateChart is modeled correctly, it can be executed as an executable specification that customer can test and verify according to his requirement. For multi- processing as our NIALM system, StateChart may be the best tool because of its capability in modeling overall activity of complex systems whose services can run in parallel. The second reason to select StateChart

formalism relies on the ability of LabVIEW to generate synthesizable models from StateChart model to get prototype a RPN system in a very short time.

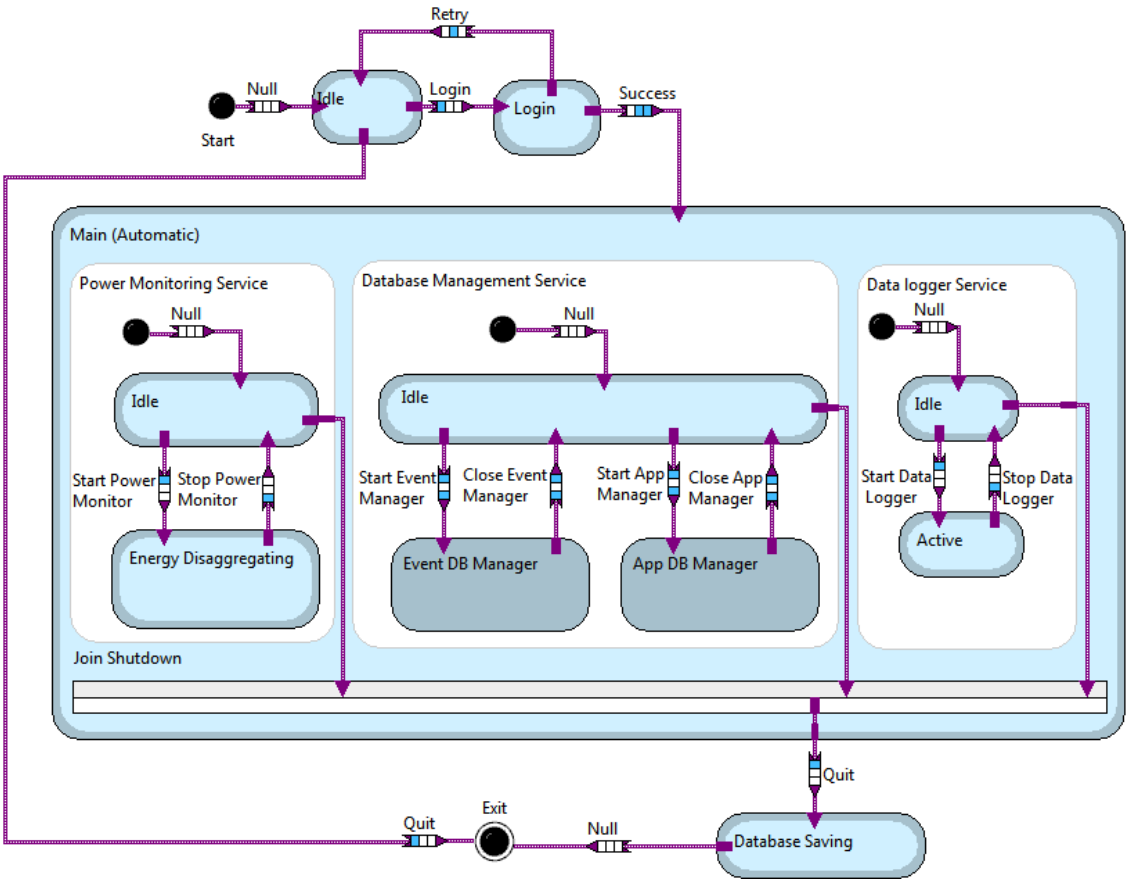



Figure 2.34 StateChart model of NIALM system

## 2.5.2. Architecture exploration

In a RPN application base on SoC, memory usage, response time, throughput, energy efficient are the most important requirements. The requirement to respond in almost real-time to customer behavior in energy usage puts the system into the real-time streaming application class in which data are processed in a streaming way from input (electrical network) to output (user). Following the requirements of Stonebraker et al. [70] for such an application, the data should also be processed throughput in the system. Therefore, selecting suitable Model of Computation (MoC) to model the system for easily analysis of the throughput, the hardware resources usage, and the performance of the system is very critical.

According to the classification tree in [71], both time-driven and event-driven properties exist in NIALM system. As in specification step, the KPN was considered to model the overall activity of this system [38]. However, KPN model does not express the rate and the number of token in each process then it is difficult to analysis the timing and resources consummation to optimize these resources. Basing on data-driven scheduling, the capability in managing FIFO

size at run-time may also cause the memory overflow problem in system. So, SDF is the best candidate to model this system. SDF can solve the KPN problems by making the static schedule and the size of buffer in communication channels for heterogeneous system in compiling time [51], [52] and [55]. However, SDF model requires all processes (or actors) of SDF must fire (or write to communication channels) a constant number of tokens in every firing.

Selecting the best architecture bases on the balance between some criteria: total latency through all processes of the system, hardware resources usage and power consumption. Satisfying timing requirement of system is the most important criteria that requires to measure latency of each process in each architecture. Common method to measure the latency of process is to subtract the time when the process finishes its work to the time when the process starts to run. LabVIEW has high-level function to do this benchmark quickly in supported platform. Figure 2.35 shows that the methods to get benchmark of the worst latency of a process in computer or microprocessor (a) and in FPGA platform (b) are the same. A measured latency will define the maximum working rate of the process and the buffer size of the FIFO, which is needed for an architecture.

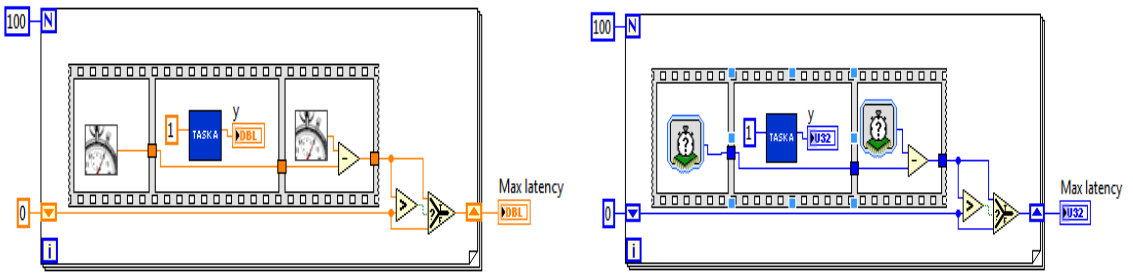



Figure 2.35 Benchmark of the worst latency of each process in (a) computer or microprocessor and in (b) FPGA

**Power consumption analysis**

Similar to hardware usage exploration, power consumption cannot be estimated at system design and must be measured directly in the target by using external equipment. LabVIEW already implemented an operation system on the target to run some default services for examples: myRIO Toolkit, Wireless Certificate Management, System State Publisher that are necessary for running a model based design in target. The formula to estimate power consumption of CPU when it runs functions can be defined as:

$$P_{cpu} = P_{\min} + (P_{\max} - P_{\min}) * CPU_{load} \quad \mathrm{P_{cpu}} = \mathrm{P_{min}} + (\mathrm{P_{max}}\text{-}\mathrm{P_{min}}) * \mathrm{CPU_{load}} \qquad (2.1)$$

Where:

- $P_{min}$ is measured when there is no load in CPU. This is the standby power of system.
- $P_{max}$ is measured when CPU is full of load with no interaction with peripheral elements.

- $CPU_{load}$ is the percentage usage of CPU when it runs our functions. This parameter can be got by using distribute system manager tool of LabVIEW.

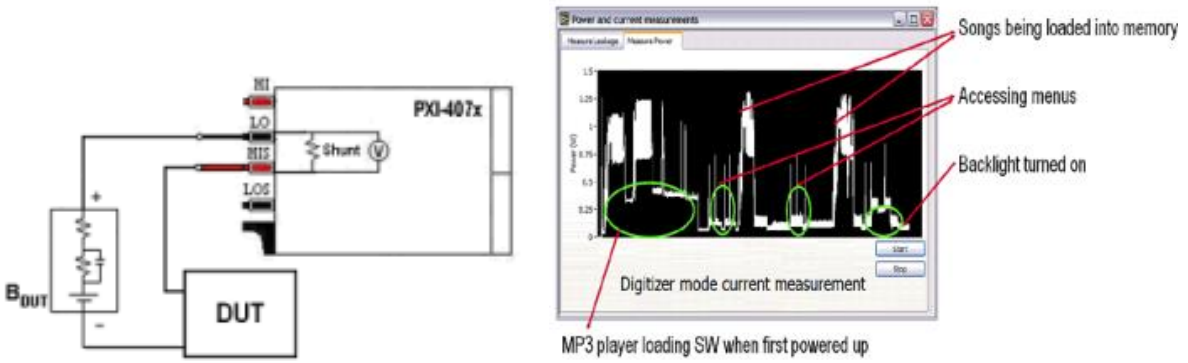



Figure 2.36 Power Consumption measurement using PXI-407x and LabVIEW Software [72]

Power consumption of a RTL function is always supported in RTL development environment for example XPower Analyzer tool of Xilinx, Power Play Power Analyzer of Altera. However, in LabVIEW FPGA, this feature has not been supported and it is impossible to export an FPGA design in LabVIEW FPGA to the VHDL code design that we can estimate its power consumption using power analysis tools of Xilinx or Altera. After that, we have to setup an experiment station as Figure 2.36 to estimate the power consumed by FPGA hardware, which can be defined by function.

$$P_{FPGA} = P_{measured} - P_{CPU} - P_{\min} \qquad (2.2)$$

Where $P_{CPU}$ is defined by (2.2), $P_{min}$ is measured in system where there are not any of our functions running in the system.

### 2.5.3. Hardware Software co-development

In LabVIEW FPGA, software developers and hardware developers can use their favorite test-based approach such as C/C++ for software developing and VHDL for hardware developing to optimize the prototype system initial developed by system developers. While model-based approach supports system developers explore their design in different architectures quickly, the text-based design gives software developer a more flexible tool to optimize the code. In LabVIEW FPGA, the FPGA controls and interfaces to IO ports so that C/C++ software developers must use the hardware interface API to interface to the hardware IPs.

As illustrated in Figure 2.29, customized FPGA IPs are developed with LabVIEW FPGA or HDL manual coding. These IPs then can be integrated to the system by two useful tools: the Component-level IP (CLIP) and the IP Integration Node. These tools are especially vital when supported IP libraries do not satisfy hard constraint functions. Another advantage of this methodology is the possibility proceeding quickly to architecture exploration because partitioning of the functions in hardware or software can be easily investigated. A C/C++

software developer can appreciate the hardware interface API, which is created by the C API Interface tool in the C/C++ Eclipse programming environment, while a LabVIEW developer can use the interface in the LabVIEW environment. Whatever the environment, the SW/SW interface, HW/HW interface, and HW/SW interface mainly affects the memory and latency of the system that can be analyzed in guide of the system constraints. This approach then enables a rapid architecture exploration in order to boost the hard real-time constraints satisfaction by the FPGA-CPU cooperation and reduce the time to market, which is another major constraint for companies.

## 2.6. Conclusion

In this chapter, we presented many theories about SoC and approaches to develop a SoC embedded system. Then, MoC-based approach is selected as the best candidate in this research because of two reasons: first, it suits model RPN system like the NIALM; second, this approach can help developer in finding the best architecture for system early at design step. Then, many MoCs relating to the RPN development are investigated and languages and development tools are studied in detail. Finally, this chapter describes a hardware-software codevelopment approach aiming to rapid prototyping SoC application with hardware acceleration using FPGA. In this approach, we also propose to use synchronous dataflow model to model the system because this model can support well allocating memory and scheduling the operation of system in compilation time in multi processors architectures. This capability can increase the productivity, optimize resource, and improve the performance to save cost and time from development to market of product.

In the next chapter, we will develop an application model of a real-time NIALM system. Before implementing system using the methodology presented in this chapter, the overall activity of the system must be designed and algorithms for its functions must be selected.

# CHAPTER 3. APPLICATION MODEL FOR A REAL-TIME NIALM SYSTEM

**Contents**




**Abstract:**

*In order to develop our NIALM system on SoC, we need to develop an application model of the system for functional verification. This chapter presents this high-level model of the NIALM system. The system specification starts with an activity model giving an overview of the global functionalities of the NIALM system then it goes in detail in describing functions and the communications between them. The three next sections describe the behavior of the basic operations by presenting the mathematical or algorithmic of the signature extraction activity, the CUSUM real-time event detection and the power disaggregation based on Genetic Algorithm. This chapter also presents functional validation results when processing a public NIALM data set named REDD.*

# Chapter 3. Application Model for a Real-Time NIALM System

## 3.1. Activity model of system in dataflow

### 3.1.1. System requirements

Although processes in a system such the NIALM system are often known in advances, system requirement is still a major decision step, which will have strong effects on the selection and on the development of algorithms inside processes. Therefore, system requirement specification aims to create important input documents useful for the development and verification steps. It may include requirements from customer or requirements from system developer. Developer often plays a crucial role in verifying the initial requirements from customer and transforms them to a system requirement specification, which can satisfy customer's requirements. After a cooperation during a lap of time that depends on the complexity of the project, a final system requirement specification will establish a contract between customer and developer about the system that the customer wants to develop.

As discussed in chapter 1, the innovative NIALM meter we want to develop must satisfy new requirements including precision, timing, technological and economic constraints. System can detect appliances with power consumption from 80W and can classify more than 80% total power in electrical network. Timing constraints are particularly stringent. Our system should be able to display the status ON/OFF in appliances in a GUI (Graphic User Interface) in 5 seconds from a real event. However, technological and economic constraints state that system must be compact enough to be easily located inside the electrical panel while having a price less than 150 euros.

After reviewing many NIALM researches and the trend of this technology, we transformed these basic requirements above into a formal technical requirement in the part "Thesis's contributions" of chapter 1 that aims to be used for the development process. We propose the development of a NIALM based on SoC technology because the evolution of SoC is very interesting and can satisfy requirements relating to the size and price of product and real-time disaggregation. In order to enhance the precision or accuracy of the energy estimation to reach more than 80% of classification of the total power, we arrived to the conclusion that the system has to solve some challenges of NIALM technology such as:

- Extraction of more electrical signatures to improve distinguishing similar appliances. They are the changes in real power, reactive power, coefficients of current and Total Harmonic Distortion (THD) of current as well as the shape information of transitions including maximum and minimum values of power and the duration of the transition.
- Detection of slow transitions in variable load appliances.
- Detection of simultaneous transitions to avoid missing event.
- Detection of multi-state appliances which can have many states in their operation.

Let us now present how to translate these requirements into a specification document to describe the complete external behavior of the system to be designed which operates in the environment explained in the requirements.

### 3.1.2. Entity analysis and modeling

In order to formalize the understanding of the problem and the objective to be achieved, an intermediate document between the system requirement and the solution (to be developed by designers), called specification, should be provided to express functional and non-functional specifications. In this step, developer must list out all entities of the system, analyze their operation and characteristics, and model them at an abstract level. After this step, the system can be delineated fully and correctly.


Figure 3.1 Context diagram of the system

The system which needs to be designed is presented in Figure 3.1 with only two entities: the *electrical network entity* and the *user entity*. The *electrical network* entity is characterized by current (I) and voltage (V). This entity provides *current (I)* and *voltage (V)* data infinitely using specific sensors and Analog to Digital converter modules, which are hidden to simplify the design at the system level. Then system processes I, V values to compute all necessary electrical signatures and to detect ON-OFF transitions of appliances rely on monitoring electrical energy usages. Then, these signatures are used to classify appliances and disaggregate the total power usage in electrical network. System from the view of electrical network entity must do the data-stream processing as in DSP systems. Data of electrical network entity will be monitored and processed by main DSP algorithms in the system so that the behavior must be described for the functional verification purpose of the system. In continuous-time domain, apparent voltage and current sources in an electrical network are represented by standard equations:

$$i(t) = \sum_{k=1}^{\infty} I_k \sin(k\omega t + \varphi_k) \quad and \quad u(t) = \sum_{k=1}^{\infty} U_k \sin(k\omega t)$$

with k = 1,3,5,7… is the harmonic order, $\varphi_k$ is the shifting phase between current order k and voltage order k. $\omega = 2.\pi.F_0$ with $F_0$ is fundamental frequency of electrical system (50 Hz in European Standard or 60 Hz in American Standard).

In order to model this entity, we have to represent them in the discrete domain. If $F_s$ is the sampling frequency of the system, we will have $N = F_s / F_0$ is the number of samples during a period of fundamental frequency signal. Thus, we can model the electrical network entity in the system as shown in Figure 3.2.

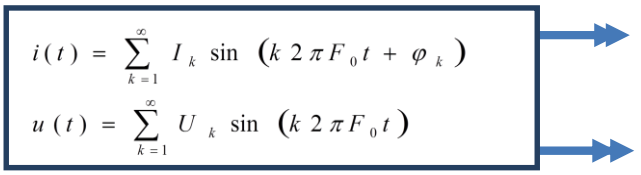



Figure 3.2 Behavioral model of electrical network entity

The *user entity* can make various requests to select modes to see or to modify the information. More precisely, the user can request some services like: Start or Stop the system, turn on the monitoring mode or maintenance mode with actions: to load, modify and save the database. Therefore, user can be the NIALM client or the man in charge of the installation of the system.

### 3.1.3. Activity model of system

The aim of functional specifications is to describe the complete external behavior of the NIALM system to be designed, which operates in the environment explained in the previous paragraph. Due to the lack of a single model containing all required features to express specifications, a coherent description of the external model usually requires three complementary views.

- Object and data modeling to describe the static characteristic of each component or item.
- Activity modeling to describe the internal activities of the modeled component and all the relations with its environment.
- Behavior modeling involving the description of temporal dependencies between the occurrence of events and the execution of actions.

Such a specification of the NIALM system can be provided starting by the activity-modeling viewpoint as illustrated by Figure 3.3. As stated earlier, the aim of the activity modeling is to define the relationship between data, events, information items and the internal activities of an entity or the system. Then, starting with a global approach and the description of the activities rather than the behavioral viewpoint is a strategy that we adopted to cope with the complexity of the NIALM system. In this kind of viewpoint, a data-flow diagram (DFD) decomposition hierarchy is first developed as an intermediate model used to make the behavioral modeling easier. In addition to the DFD, a control specification expresses the temporal evolution of each activity in order to specify the global behavior of the NIALM system. For this, operating modes can be established according to the functionalities that are available at a particular step in the life cycle of the instrument. For example, in a maintenance-operating mode, the disaggregated total apparent power service cannot be launched. If this mode can express, the maintenance functionality is not available by the customer. System from the view of user entity works as a

reactive process network, which events generated from activities of user, requires the reaction from the system. In system level, user entity can be modeled by a message source where each message represents a command of this entity. Control logic part then enables or disables each function and shows it in the GUI of the system to respond to user's interaction.

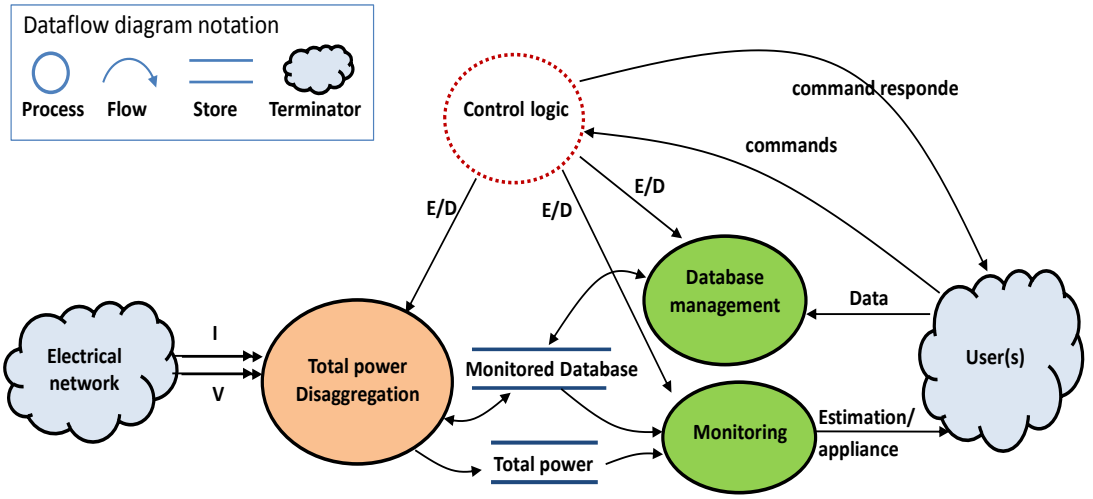



Figure 3.3 Global activity diagram of the NIALM system

As illustrated in DF model in Figure 3.3, in power estimation mode, total power disaggregation activity and monitoring activity are active. Current and voltage are measured at a specific sampling rate and are used to compute the total real power in electrical network. Then, this total power is disaggregated to power consumed in each appliance. In communication between activities, total power disaggregation periodically sends the total power value to activity monitoring. In addition, it can detect an event of appliance (ON-OFF transition) and send electrical signature data of detected appliances to the Monitoring activity. However, in maintenance mode, control logic disables total power disaggregation and monitoring activity and actives data management activity. Then user can download, upload or modify the database for system configuration.

This main activity of the NIALM system is a composition of some entities and description of their operations:

- Preprocessing nodes do the preprocessing step in NIALM system including computing real power P, reactive power Q and coefficients of harmonics $I_{ak}$, $I_{bk}$ from input current I and voltage. The coefficients are needed to compute the change in THDi because it is a non-linear value and cannot be calculated directly [73]. Grady et al. [74], Gupta et al 2007 [22] and Patel et al. [11] proved that coefficient and THDi can be used to classify home appliances. Moreover as illustrated in Figure 3.4, the three nodes can be processed in parallel to enhance the performance of system, if needed.

- Event detection node to detect the change (event) in total power usage in NIALM system. After detecting an event, this activity can extract many useful electrical signatures and duration of the transition and send them to Event classification node.
- Event classification to early classify events to three basic event types: resistor, motor and lighting. Coming section will present more details about the activity and results of this algorithm.
- Disaggregation process will classify appliance from the detected events using three tasks: Event clustering, Genetic Algorithm and Appliance Finding. Although Genetic Algorithm can process the power disaggregation task, Event Clustering can improve GA results and Appliance Finding is responsible to database searching.
- The Monitored Database supports the storage of information relevant to detected events and applications. The requirement of this database is small, fast and possible to embed in small resource architecture in the embedded system.

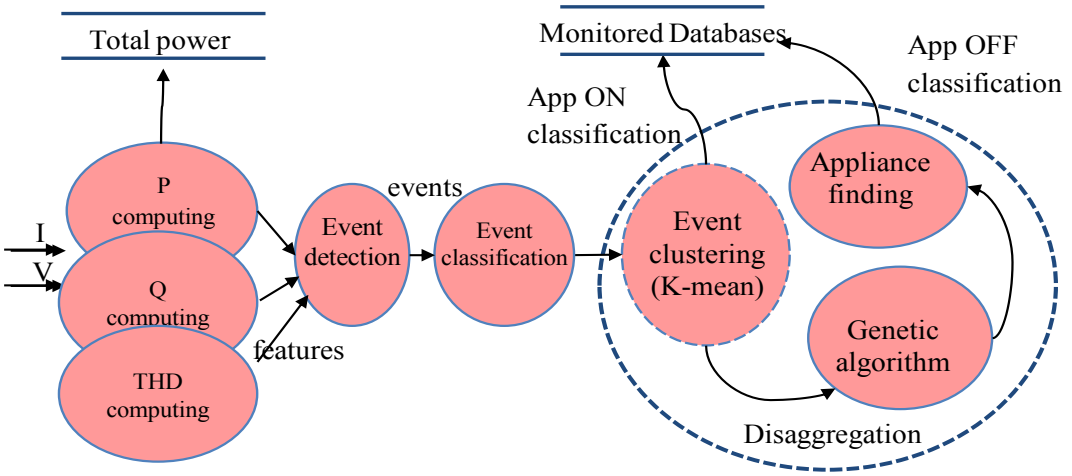



Figure 3.4 Data flow model of the total power disaggregation function

Dataflow model works well in modeling the activity model of system because firstly, it is easily and visually in verifying the flow of data in a data streaming system. Second reason is the expectation in refining the dataflow model to concurrent dataflow model for architecture exploration and throughput analysis. The main advantage of such a modeling approach is the ability to express the functionality through a global approach without modeling each entity. The main disadvantage is the difficulty to know when stop the decomposition process.

In coming parts, we will present systematically the behavioral model part of the NIALM system. For basic activities, the data transformations (relationship between inputs and outputs) operated by elementary activities can be expressed in a formal way using transfer function or any other mathematical formalisms. For control logic, state chart like formalisms are more suited.

## 3.2. Electrical signatures extraction: Event-based approach

Actually, current energy measurements ICs do not support well NIALM technology. In some popular energy measurement ICs in Table 3.1, they measure only basic information such as apparent power S, real power P, reactive power Q, average voltage, and average current. The advanced information Total Harmonics Distortion (THD) can also be extracted in product of NXP and Analog. MSP430 of TI and EM773 of NXP are interesting smart meters ICs because they contain Microprocessor inside to be able implemented a complete low cost smart meter system. All extracted information of current meter ICs are the aggregated information of electrical network. However, these IC does not support extracting information of each appliance especially with nonlinear information. This feature need customer algorithms implemented in NIALM software. In order to solve that challenge, we proposed a new generation of energy measurement based on event detection approach, which can embed into new kind of energy measurement IC. A new event detection algorithm specific, which is able to extract both linear and non-linear electrical signatures, will be presented in the next section. This event detection can run in real-time and can extract most of necessary information for NIALM technology.

| **Vender** | **Texas Instrument** | **Cirrust Circuit** | **NXP** | **ANALOG** | **MICROCHIP** |
|---|---|---|---|---|---|
| Typical product | **MSP430F471xx** | **CS5480** | **EM773** | **ADE7880** | **MCP 3905/06** |
| ADC Resolution | 16 bits | 24 bits | NA | 24-bits | 16 bits |
| ADC Sample rate | 3.2768 kHz | 500 kHz | NA | 1.024 MHz | 1MHz |
| MCU inside | MSP430CPUx | No | ARM Cortex | No | No |
| Monitor parameter | P, S, Q, F, Irms, $I_p$, Vrms, Vp, power factor | P, S, Q, F, Irms, $I_p$, Vrms, Vp, power factor | P, S, Q, Vrms, Irms and THD | Harmonics, P, S, Q, Vrms, Irms, power factor | Active power |
| Accuracy | 1% | 0.1% | 1% | 0.2% | NA |
| Firmware/ Hardware | 32-bit x 32-bit HW multiplier | On chip calculation | Metrology Engine FW | DSP hardware | Energy to Frequency HW |
| Price | 2-4$ | 5.25$ | 2.5$ | 9.45 $ | 2$ |

Table 3.1 Some popular energy measurement ICs and their features

As we discussed in chapter 1, electrical signatures can be classified into two categories: steady-state signatures and transition-state signatures. Steady-state signatures of appliances, which are in stable ON or OFF state in the electrical network, are electrical signatures that we can use to distinguish appliances. Thus, such signatures must be unique for a given appliance. Some examples of steady state signatures of appliances are real power, reactive power, average current etc. In the research conducted by Froehlich, J. et al. [12], continuous voltage noise is also a steady-state signature. As illustrated in Figure 1.9 in chapter 1, each appliance has its own noise with a specific frequency. For example, LCD monitor has a specific noise at a 64 kHz frequency, but PC's noise is located at 93 kHz. This high frequency feature can also identify some always-ON appliances like telephone, router, and other electronics. Unfortunately, there

are no standards for the noise on appliances. Moreover, extracting high frequency noise requires expensive equipment. Therefore, in our research, we focus on the development of a NIALM system based on transition-state signatures leading to an event-based NIALM.

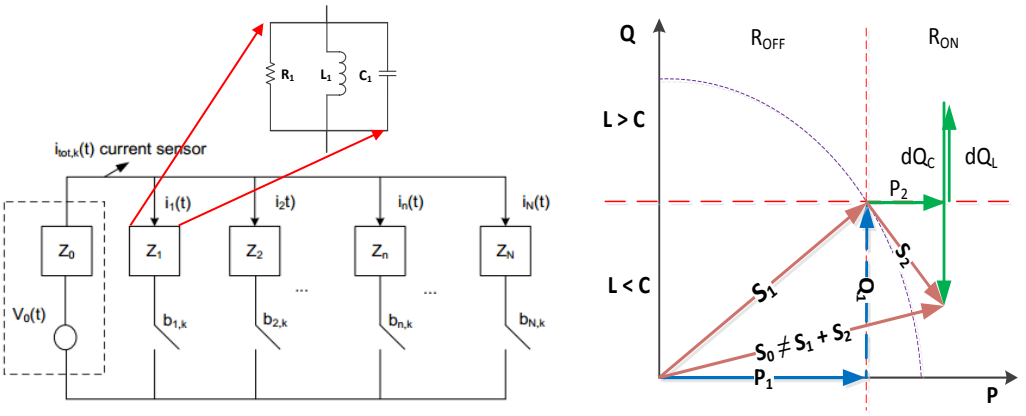



Figure 3.5 Electrical network with represent RLC model of appliances

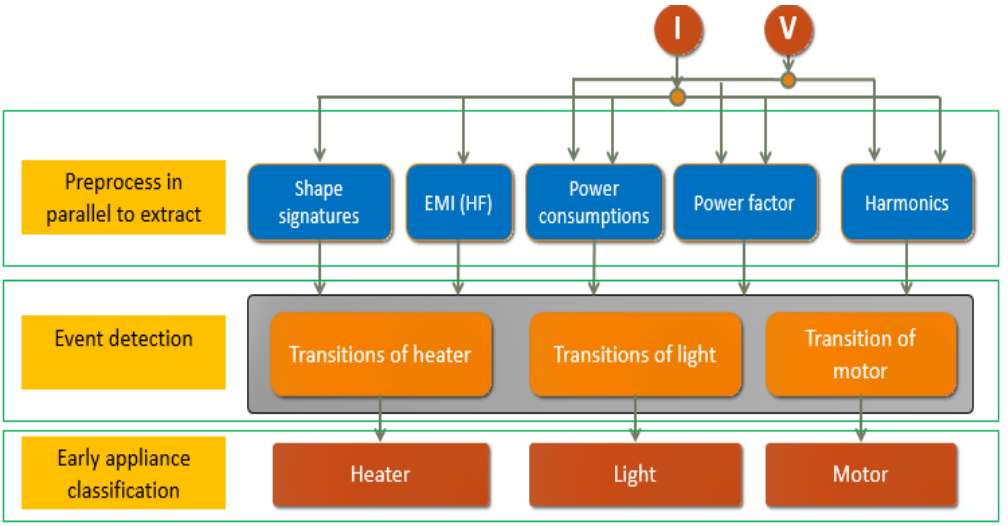



Figure 3.6 Event-based signature extraction and early appliance classification methodology

The event-based approach for electrical signature extraction can detect status changing of appliances based on the change of aggregated information in an electrical network. However, only changes in linear features can be used to compute appliance's signature. In chapter 1, we assumed a representation of appliances in electrical network by RLC models connected in parallel, as illustrated in

Figure 3.5. When a turning ON event occurs, caused by a switch $b_{N,k}$, changes in real power P, reactive power Q and apparent power S will occur in response. However, only changes in active power and reactive power measured in main electric are linear and can be used to calculate active and reactive power of the appliance. As illustrated in

Figure 3.5, aggregated apparent power S, a non-linear data, is not the sum of apparent power of new turning on appliance and apparent power of circuit before the event. We will present below the behavioral model of the preprocessing activity for extracting the power signatures, shape of power signature and harmonics information. Moreover, Figure 3.6 presents the early appliance classification based on extracted signature and event detection algorithm.

### 3.2.1. Power signatures

Power signatures are very common signatures of electrical appliances, which are not only information for classifying appliances but also the main information to build electrical bills. Among three types of power, active power (true power) is the actual amount of power consumed by the electrical devices. Reactive power is generated by the inductive or capacity elements in appliance. Although this kind of power does not exist in electrical bill but it has effects on the power factors, which is represented to power efficiency of appliances. In order to describe the behavior of real power and reactive power computing, we assume that input voltage and current can be represented by formulas below.

#### Active power

Active power is a very basic information of appliance, which presents how much energy that an electrical appliance can perform actual work in resistance part of appliance. It is different to reactive power, which is cause by the inductive and capacitive part of the system. The following general formula describes the computation of the voltage and current in time continuous domain:

$$u(t) = \sum_{k=1}^{\infty} U_k.\sqrt{2}\ .\cos(k\omega t\ +\ \varphi_k) \qquad (3.1)$$

$$i(t) = \sum_{k=1}^{\infty} I_k.\sqrt{2}\ .\cos(k\omega t\ +\ \gamma_k) \qquad (3.2)$$

Where $U_k$ = Average amplitude of voltages
$I_k$ = Average amplitude of currents
$\omega$ = Angular frequency
$\varphi_k, \gamma_k$ = Phase of voltages and currents at the time t = 0
k = 1, 2, 3…N are harmonics of current and voltage.

Formal formula to compute the active power is given by:

$$P = \sum_{k=1}^{\infty} U_k.I_k.\cos(\varphi_k - \gamma_k) \qquad (3.3)$$

From (3.1) and (3.2), we have:

$$u(t).i(t) = \sum_{k=1}^{\infty} U_k.\sqrt{2}\ .\cos(k\omega t\ +\ \varphi_k).\sum_{l=1}^{\infty} I_l.\sqrt{2}\ .\cos(l\omega t\ +\ \gamma_l)$$

As product-to-sum identity of trigonometric theory, we know that:

$$\cos(x).\cos(y) = \frac{\cos(x-y)+\cos(x+y)}{2}$$

So, we get

$$u(t).i(t) = \sum_{k=1,k=l}^{\alpha} U_k I_k.\cos(\varphi_k - \gamma_k) - \sum_{k=1,k=l}^{\alpha} U_k I_k.\cos(2k\omega t + \varphi_k + \gamma_k) + \sum_{k,l=1,k\neq l}^{\alpha} U_k I_l.\{\cos[(k-l)\omega t + \varphi_k - \gamma_l] - \cos[(k+l)\omega t + \varphi_k + \gamma_l]\} \quad (3.4)$$

Then, we can have the definite integral of equation (3.4) in a period of the normalized frequency defined by equation.

$$\frac{1}{nT}\int_0^{nT} u(t).i(t).dt = \sum_{k=1}^{\infty} U_k.I_k.\cos(\Delta\varphi_k) \quad (3.5)$$

Thus, from (3.3) and (3.5) we have the equation to compute active power defined as

$$P = \frac{1}{nT}\sum_{t=1}^{nT} u(t).i(t) \quad (3.6)$$

Where T is the amount of samples during a period of signal and n is an integer value.

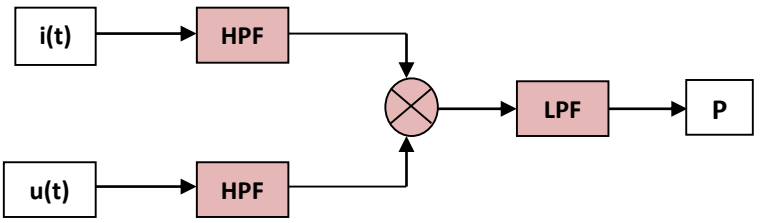



Figure 3.7 Filter-based active power computation approach in analog domain

Moreover, from above equation (3.4), there is other method to calculate the active power in analog domain by using analog multiplier for input current and voltage and a low-pass filter as illustrated in Figure 3.7. Output data from the analog multiplier then is fed to a low pass filter to extract only the DC component from signal, which is the final equation to compute active power in (3.3).

## Reactive power

Reactive power, which is a useful data for evaluating the power efficiency (power factor) of appliances, becomes important in billing activity. That is because there are more and more non-

linear loads from end-customers profile especially in industrial area. Thus, the reactive power measurement is required to be more and more accurate. There are many researches in computing reactive power consumption.

Kezunovic, M. in 1991 [75] proposed to represent power measurement algorithms as 2D FIR filter applied on current and voltage samples. The advantage of such a method is the possibility to apply on both sinusoidal and non-sinusoidal signals. The author did a comprehensive test the accuracy of real power and reactive power computing when there is the frequency variation in the electrical network. Various reactive power algorithms such as Budeanu, Fryze, and Kuster for inductive reactive power and capacitive reactive power computing were represented as 2D digital filters. The advantage of this research is that it gives a very high accurate result when the highest relative error is only 0.00277% in +0.5% frequency variation case. However, the requirement of N samples in each channel current and voltage to run the convolution operation is too expensive in resource especially when signals are sampled at high frequency. For example, 10 kHz needs about 30000 samples stored in FIFO or RAM to compute a value of reactive power.

Other research [76] proposed an approach to measure reactive and distortion power using the **wavelet transform**. Power analysis in wavelet domain can give us a view in both frequency and time domains for both real and reactive powers. Such approach still needs to shift voltage $90^0$ earlier than current to be able to compute the reactive power. However, the traditional wavelet transform approach needs N storage cells for N cascaded filter so that it is still a resource expensive method. Such drawback does not allow a real-time power measurement in compact system. Ejaz, Waleed et al. [77] reformulated a new algorithm named pyramid recursive algorithm (PRA) which consumes only L*log(N-L) storage cells with L-length of the filter much less than N. However, investigated PRA varied significantly when they changed parameters algorithm, the resolution of ADC and the sampling frequency.

In other research [78] in 2009, Grigorescu and al. have reviewed some methods for measuring reactive power including Hilbert transform method, power triangle method, quarter period time delay method and one pole low pass filter method. This research contributed to give a comparison in performance between above methods under various testing conditions like current and voltage in the same phase, the frequency variation test cases, and some harmonic test cases. The results proved that Quarter period time delay is the less accurate method and Hilbert transform method gave the best results. Hilbert transform is a digital filter circuit that is able to shift phase input signal for exactly $90^0$ without underlying an amplitude attenuation in every frequency. Quarter period time delay method, that merely shifts signal N/4 samples ($\pi/2$ equivalent), has high accuracy in the ideal signal without any frequency variation.

Ludek Slosarcik in 2014 [79] proposed the use of using Fast Fourier Transform for power computing in an application note of Freescale Semiconductor. Such algorithm first transforms signal from time domain to frequency domain including coefficients of harmonics in current and voltage data. Then, these data are fed to the formula that computes the complex power in

Cartesian form. Such algorithm is potentially the best method to implement a power measurement system in a resource limitation embedded system.

After reviewing many researches, we propose the use of Budeanu algorithm to represent reactive power the use of Hilbert transform approach to get the most accurate results. We would like to make the comparison between these methods and the Fast Fourier Transform method because no comparison between them can be found. In this comparison, the accuracy and the resources consumption of algorithms will be investigated under various testes relating to frequency variation and the sampling rate.

**Budeanu definition**

As a brief introduction of [75], Budeanu approach defined computing apparent power by orthogonal formula:

$$S = P \pm j.Q \tag{3.7}$$

Such an algorithm can represent the sinusoidal waveform:

$$Q = U.I.\sin(\varphi) \tag{3.8}$$

Therefore, for the non-sinusoidal waveform, reactive power can be defined as:

$$Q = \sum_{k=1}^{M} \frac{U_k.I_k}{2}.\sin(\varphi_k - \gamma_k) = \sum_{k=1}^{M} \frac{U_k.I_k}{2}.\sin\Delta\varphi_k \tag{3.9}$$

Where k is harmonic of signal; $\Delta\varphi_k$ is the different phases between current and voltage in each harmonic. M = N/2 - 1 is the maximum harmonics order and N is the number of samples per second, following the sampling theory.

In comparison to equation (3.3), we have basic idea to compute Q:

$$Q = \frac{1}{nT}\int_0^{nT} u(t).i'(t).dt = \frac{1}{nT}\int_0^{nT} u(t).i_{shift-90}(t).dt \tag{3.10}$$

**Proof:**

$$i'(t) = \sum_{k=1}^{\infty} I_k \cos(k\omega t + \gamma_k - \frac{\pi}{2}) = \sum_{k=1}^{\infty} I_k \sin(k\omega t + \gamma_k)$$

$$u(t).i'(t) = \sum_{k=1}^{\infty} U_k.\sqrt{2}\ .\cos(k\omega t + \varphi_k)\ .\sum_{l=1}^{\infty} I_l.\sqrt{2}\ .\sin(l\omega t + \gamma_l)$$

As product-to-sum identity of trigonometric theory, we know:

$$\cos(x).\sin(y) = \frac{\sin(x-y)+\sin(x+y)}{2}$$

So that, we get

$$u(t).i'(t) = \sum_{k=1,k=l}^{\alpha} U_k I_k.\sin(\varphi_k - \gamma_k) - \sum_{k=1,k=l}^{\alpha} U_k I_k.\sin(2k\omega t + \varphi_k + \gamma_k) + \sum_{k,l=1,k\neq l}^{\alpha} U_k I_l.\{\sin[(k-l)\omega t + \varphi_k - \gamma_l] - \sin[(k+l)\omega t + \varphi_k + \gamma_l]\} \quad (3.11)$$

Then from (3.10) and (3.11) we have

$$Q = \sum_{k=1}^{\infty} U_k I_k.\sin(\Delta\varphi_k) = \frac{1}{nT}\int_0^{nT} u(t).i'(t).dt = \frac{1}{nT}\sum_{\tau=1}^{nT} u(\tau).i'(\tau) \quad (3.12)$$

Similar to active power, from (3.11), reactive power can be computed in analog domain by a low-pass filter shown in Figure 3.8.

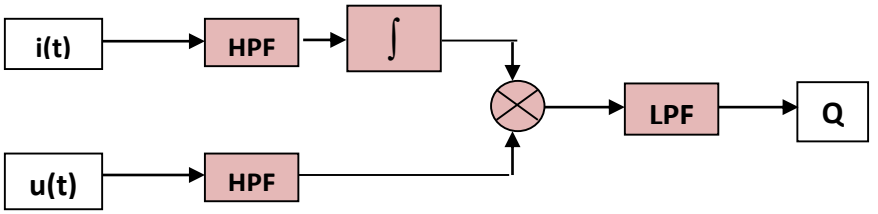



Figure 3.8 Filter-based reactive power computation approach in analog domain

In order to shift current to 90 degree to use equation (3.10), the research proposes using two methods for comparison: the quarter period time delay, low pass filter and Hilbert transform. Three approaches will be presented in detail bellow.

### Quarter Period time delay

This method may be the simplest approach base on the sampling frequency. The objective of this method is using a suit number of delay unit to get the delay π/2 which equivalent to T/4 with T is the period value of fundamental frequency Fp. If the sampling frequency is Fs, number of sample of a full period signal is N = Fs/Fp. Thus, reactive power is an average value of the multiplication of digitized current with the voltage delayed T/4.

$$Q = \frac{1}{nT}\int_0^{nT} u\left(t-\frac{T}{4}\right).i(t).dt = \frac{1}{nN}\sum_{t=1}^{nN} u\left(t-\frac{N}{4}\right).i(t) \quad (3.13)$$

However, this method is not too accuracy depending on the sampling frequency as shown in below table assuming fundamental frequency varies in 50Hz ± 1% with period 19.8 - 20.20 milliseconds. That means a quarter period time delay should create a delay 4.95 ms (at worse case 50Hz + 1%) and 5.05 ms (at worse case 50Hz - 1%). Below Table 3.2 is the accuracy evaluation in these two worst cases. It means that this method can be used in high sampling frequency from 20 kHz with very small error (<1%). In lower sampling rate less than 2 kHz; this method can cause error in reactive power up to 9%.

| **Sampling frequency** | **1 kHz** | **2 kHz** | **5 kHz** | **10 kHz** | **20 kHz** | **100 kHz** |
|---|---|---|---|---|---|---|
| Samples per Period | 19 – 20 | 39 – 40 | 99 – 101 | 198 – 202 | 396 – 404 | 1980 – 2020 |
| Samples in [Period/4] | 4 – 5 | 9 -10 | 24 – 25 | 49 – 50 | 99 – 101 | 495 - 505 |
| Max Error | 19.19% | 9% | 3% | 1% | 0% | 0% |

Table 3.2 Accuracy of quarter period time delay in different cases of sampling rate

**Hilbert transform**

Hilbert transform is a special filter, which can shift the input signal a phase 90 degrees relative to the original signal [80].The Hilbert transformer of a signal g(t) is defined as the convolution of g(t) with the signal 1/(πt).

$$\mathcal{H}[g(t)] = g(t) * \frac{1}{\pi t} = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{g(\tau)}{t-\tau}d\tau = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{g(t-\tau)}{\tau}d\tau$$

In Fourier transform, the signal 1/(πt) is defined as

$$-j.sgn(f) = \begin{cases} -j, & if\ f > 0 \\ 0, & if\ f = 0 \\ j, & if\ f < 0 \end{cases}$$

From the convolution property of the Fourier transform, if G(f) is the Fourier transform of g(t), the Hilbert transform of g(t), the ĝ(t) has the Fourier transform :

$$\hat{G}(f) = -j.sign(f).G(f)$$

Thus, the Hilbert transform does not change the magnitude of G(f) but it changes the phase. Above formula shows that, in the positive frequency f >0, $\hat{G}(f) = -j.G(f)$ which corresponds to the phase change of –π/2. In other words, we have the Hilbert transform of cos(t) is sin(t).

The impulse response h(n) of the Ideal Hilbert transform thus can be defined as the following equation :

$$h(n) = \begin{cases} \frac{2}{\pi} \frac{sin^2(\frac{\pi n}{2})}{n}, & n \neq 0 \\ 0, & n = 0 \end{cases}$$

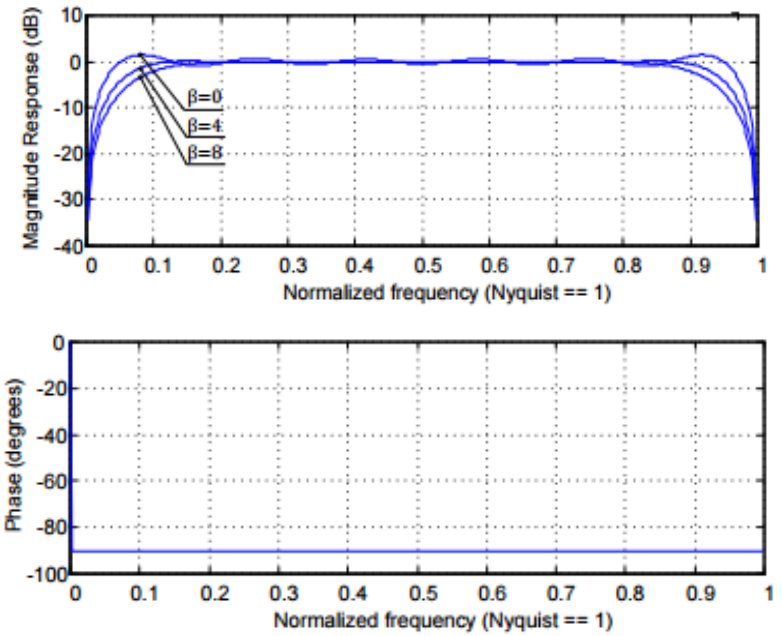



Figure 3.9 FIR approximation block magnitude and phase response

Unfortunately, ideal Hilbert transform has an infinite length of the impulse response so that implementing it is not practically. Thus, window technologies are often used to limit impulse length of the ideal Hilbert transform. Mienkina in [81] proposed Kaiser Window of width N = 2M + 1 to implement a FIR approximation of the Ideal Hilbert Transform Impulse Response. The Kaiser Window coefficients of this Hilbert FIR filter are expressed by equation:

$$w[n] = \begin{cases} \frac{I_0\{\beta\sqrt{1-[(n-n_d)/n_d]^2}\}}{I_0\{\beta\}}, & 0 \leq n \leq N-1 \\ 0, & other\ wise \end{cases}$$

Where, $n_d = M/2$, $I_0$ is the zeroth order modified Bessel function of the first kind, $\beta$ is an arbitrary real number that determines the shape of the Kaiser Window, N = 2M + 1 is the length of the Hilbert FIR filter. As illustrated in Figure 3.9, a drawback of this method is the magnitude of output signal is attenuated when the normalized frequency (signal frequency/sampling Nyquist frequency) is less than 0.1 or more than 0.9. For example: signal 50 Hz and sampling frequency 2 kHz will give the normalized frequency 0.05 which the magnitude response is about -30 to -20 dB. A solution for this problem is downing the sample rate to sufficient value but it

will has effects to other computations, which the accuracy depend to the time resolution. Other solution is amplifying the signal after the Hilbert filter with a value to requite the signal attenuation. In this research, we accepted the accuracy of reactive power around 90% so that we selected the quarter period time delay method for the system.

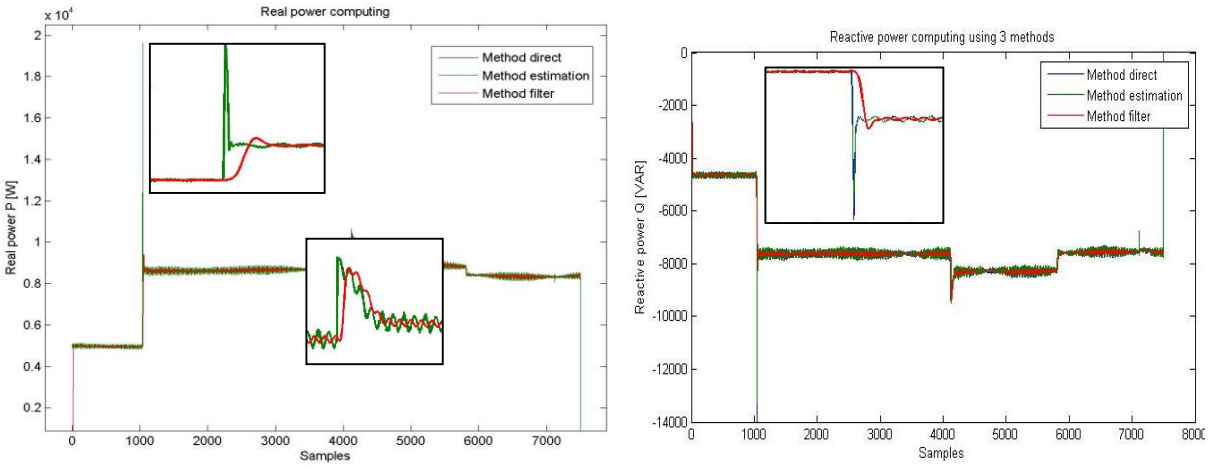



Figure 3.10 Comparison real powers and reactive power computed by three methods

Functional verification of real power and reactive power is shown in Figure 3.10. In this figure, direct method is power computed from original data and two other methods, which were mentioned in this section. The results show that, the filter method has the delays in output data and change the shape of transition. Thus, computing power using filters cause the loss of information for classification. In the contrast, estimating method keeps in phase and information for classification.

### 3.2.2. Shape of transitions signatures

Thanks to the innovative event detection algorithm, which will be presented in coming section, we can extract more information about the shape of transition. Because of differences in electrical architecture between appliances, each appliance generates a specific shape on the ON-Turning transition and even on the OFF-turning transitions. As illustrated in Figure 3.11**,** shape information includes max value and min value of real power P, reactive power Q and the time duration of a transition dt. This information is useful for early classifying appliance that makes the NIALM can work in real-time.

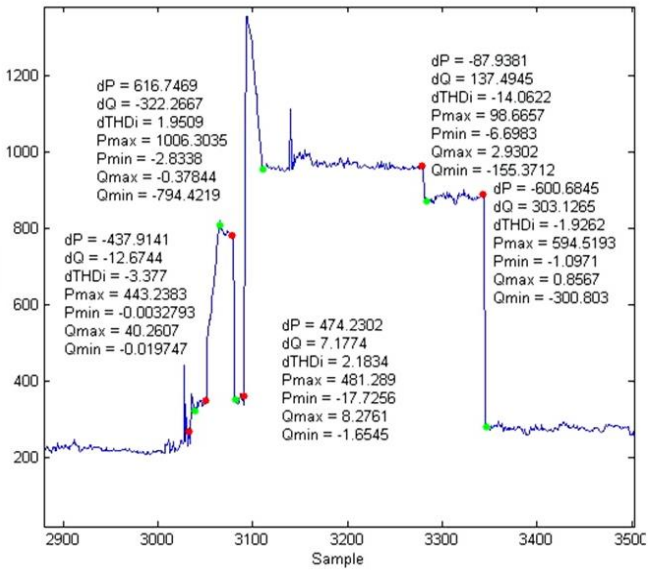



Figure 3.11 Extracting shape information of a transition including Pmax, Pmin, Qmax, Qmin, dt in a real data

### 3.2.3. Harmonic signatures

In real-world signal, there are not any ideal sinusoidal signals. The ideal signals are always interfered by many kind of signal: noise, harmonics from structure of appliance. This distorted signal if still periodic can be deconstructed into two or more sinusoidal signals at different harmonic frequencies by Fourier technique as illustrated in Figure 3.12.

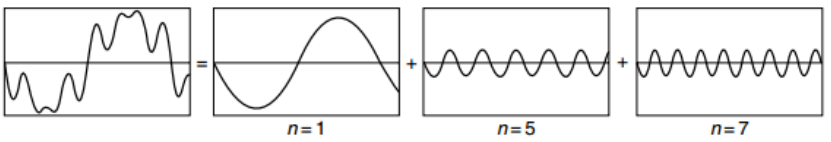



Figure 3.12 An example of the decomposition of a distorted waveform into fundamental, fifth and seventh components

***Linear Coefficients information***

Thus, applying Fourier technique for electrical network, which contains only working appliance, the electrical current signal can be represented as equation (3.14).

$$i(t) = \sum_{k=1}^{\infty} I_k \sin(k\omega t + \varphi_k) = \sum_{k=1}^{\infty} [Ib_k \cos(k\omega t) + Ia_k \sin(k\omega t)] \text{i (t)} = \sum_{k=1}^{\infty} I_k \sin(k\omega t + \varphi_k) = \sum_{k=1}^{+\infty} [Ib_k \cos(k\omega t) + Ia_k \sin(k\omega t)] \quad (3.14)$$

$$Ib_k = I_k \sin(\varphi_k) \qquad Ia_k = I_k \cos(\varphi_k) \quad (3.15)$$

$$I_k = \sqrt{Ia_k^2 + Ib_k^2} \qquad \varphi_k = \arctan(\frac{Ia_k}{Ib_k}) \tag{3.16}$$

$$\mathrm{Ib_k} = \frac{2}{\mathrm{T}}\int_0^{\mathrm{T}} \mathrm{i}\ \ (\mathrm{t})\cos(\mathrm{k\omega t})\mathrm{dt} = \frac{2}{\mathrm{N}}\sum_{\mathrm{t=1}}^{\mathrm{N}} \mathrm{i}\ \ (\mathrm{t})\cos(\frac{2\pi}{\mathrm{N}}\mathrm{kt})$$

$$\mathrm{Ia_k} = \frac{2}{\mathrm{T}}\int_0^{\mathrm{T}} \mathrm{i}\ \ (\mathrm{t})\sin(\mathrm{k\omega t})\mathrm{dt} = \frac{2}{\mathrm{N}}\sum_{\mathrm{t=1}}^{\mathrm{N}} \mathrm{i}\ \ (\mathrm{t})\sin(\frac{2\pi}{\mathrm{N}}\mathrm{kt}) \tag{3.17}$$

*Where, k = 1, 3, 5, 7… M are order of harmonics of current, and M = N/2 - 1 with N is the number of samples in a cycle of signal*

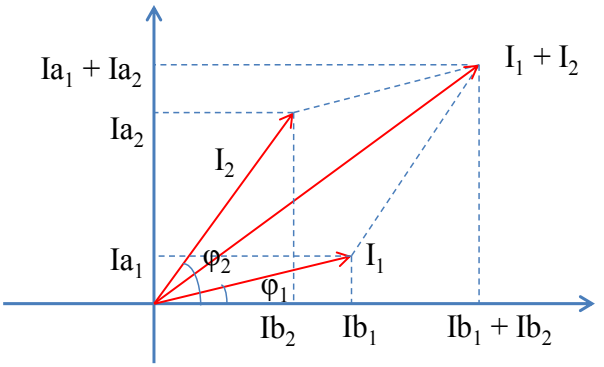



Figure 3.13 Non-linear characteristic of current in fundamental element

When a second appliance is turned on in the network, the current of the network is now defined by the total of two working appliance. However, as illustrated in Figure 3.13, the current depends on the phase so that total of this non-linear variable is not simply the sum of currents from two appliances. Fortunately, the coefficients of the current do not depend on the phase so that total coefficients can be calculated directly by adding the coefficients in the same axis of two appliances. Linear variables $Ia_{2k}$ and $Ib_{2k}$ of the second appliance are now the changes of aggregated data $Ia_k$ and $Ib_k$ after the detected event. Then equation (3.16) can be used to estimate the amplitude and the phase of the current consumed by the detected appliance.

$$\mathrm{i(t)} = \mathrm{i_1(t)} + \mathrm{i_2(t)} = \sum_{\mathrm{k=1}}^{+\infty} \mathrm{I_{1k}}\sin(\mathrm{k\omega t} + \varphi_{1\mathrm{k}}) + \sum_{\mathrm{k=1}}^{+\infty} \mathrm{I_{2k}}\sin(\mathrm{k\omega t} + \varphi_{2\mathrm{k}})$$

$$= \sum_{\mathrm{k=1}}^{+\infty} [(\mathrm{Ia_{1k}} + \mathrm{Ia_{2k}}).\cos(\mathrm{k\omega t}) + (\mathrm{Ib_{1k}} + \mathrm{Ib_{2k}}).\sin(\mathrm{k\omega t})]$$

Finally, THDi of detected appliance is calculated by equation (3.18).

$$THD_i = \frac{\sqrt{\sum_{k=2}^{\infty} I_k^2}}{I_1} .100\% \quad (3.18)$$

Moreover, other harmonic information, individual harmonic distortions (IHD), can be also computed by equation (3.19):

$$IHD_{ik} = \frac{I_k}{I_1} .100\% \quad (3.19)$$

In an article of Analog Device [82], this approach is better than other methods such as Fast Fourier Transformer, Goertzel, band-bass filter when these methods cannot compute in real-time, require more DSP memory, and have lower accuracy. Other advantage is that the developers or customers can select the harmonics that they want to extract dynamically. So this method is effective when the number of coefficient of harmonics need to be computed is less than the total number of harmonics. The algorithm is correct as shown in Figure 3.14. In real work, depending on the quality of power in electrical network [83], the supply voltage frequency can vary between 49.5 and 50.5 Hz (± 1%) during 99.5% of the year and can vary between 47 and 52 Hz (± 4%) some times during a year. Nordel system (Finland, Sweden, Norway, and part of Denmark) have best power quality when standard deviation of 0.03 Hz. However, this value is 2.5% (1.25 Hz) for the grid network and 5% (2.5 Hz) for the island network. Frequency variation can cause error in standard calculation of power information. Figure 3.15 shows an experiment in the context that voltage fundamental frequency varies between 49.5 and 50.5 Hz. We compute some harmonics using two methods: our coefficient estimation and Goertzel algorithm. Result shows that our approach has same harmonics result in all tested frequency with very small relative error in coefficient type b. However, Goertzel algorithm is only closed when frequency varies from 49.9 Hz to 50.1 Hz.

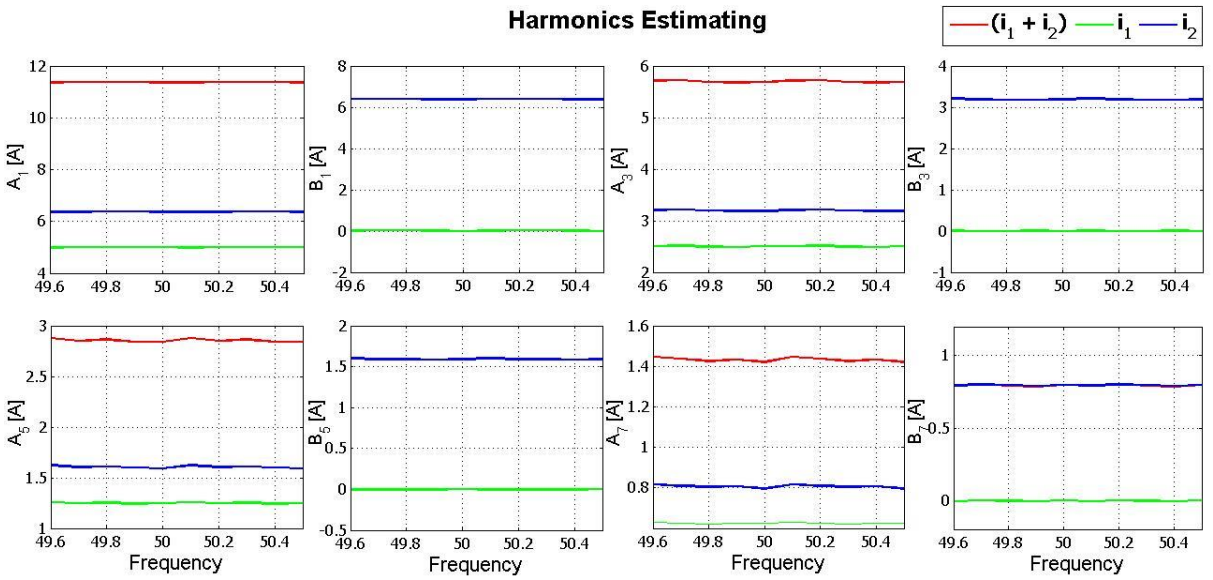



Figure 3.14 Estimate (49.6-50.5) Hz signal with (49.6-50.5) in harmonic estimating algorithm

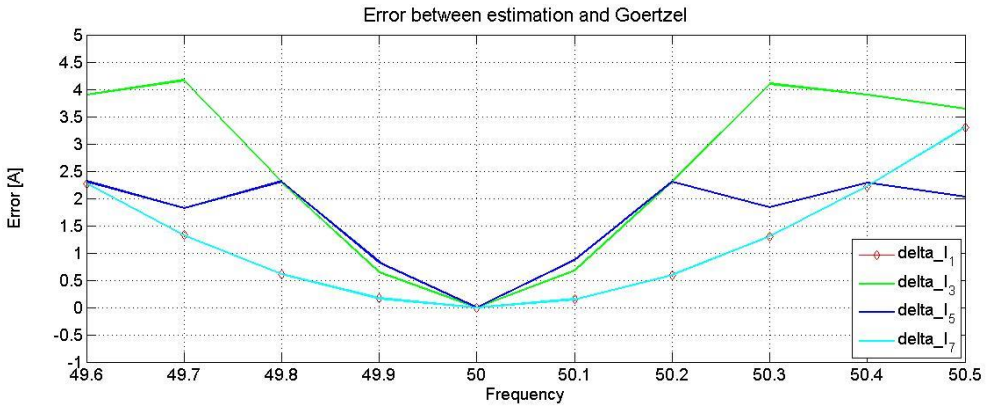



Figure 3.15 Harmonics calculated by coefficient estimation and by Goertzel algorithm

### 3.2.4. Early application classification

After investigating many power data, we believe that there are three main basic electrical appliances in all machines in industrial and commercial site: motors or fans, lighting appliances and heating appliances. An electrical machine is composed of one or many basic these electrical appliances as in below industrial oven example.

**Motor and fan:** There are four kinds of motor [84]. ON-OFF motors have only two states: ON and OFF. Multi-speed driver motors use windings as voltage transformer to change voltage level to vary speed of motor manually or automatically. Multi-motor driver motors are similar with multi-speed driver motors but they use more than one motor with different speeds to increase energy efficiency. Last type of motor is the most complex motor but they are widely used in industry application: the Variable Frequency Driver (VFD) motor. Frequency variation is used to vary the power consumption of VFD motors continuously. VFD motors are used for three main purposes [85]:

- Varying torque load by varying power to keep some parameters constant in applications such as centrifugal fans, blowers and pumps.
- Keep constant torque at all load by varying power in applications such as conveyors, mixers, and compressors.
- Keep constant power consumption to optimize speed in applications such as lathe machines, milling machines. The power consumption of third type VFD motors is similar with ON-OFF motors while the two others are different.

**Lighting appliances:** Three basic lighting types are incandescent light, fluorescent light and High Intensity Discharge (HID) light. Incandescent lights work exactly as a pure resistor sources in which the current waveform is in phase with the voltage waveform. Fluorescent light and HID light use ballasts to control the power factor so that they distort current waveform and generate harmonics.

**Heating appliances**: Industry applications generate heat from four main sources: resistance, inductance, infrared and dielectric heaters. Resistance heaters use small resistance to though high current and generate heat while inductance heaters generate Foucault current to heat metal material inside the electrical coil. Infrared heaters use electromagnetic radiation from light source to generate heat. Di-electric heaters use radio wave or microwave electromagnetic radiation to heat di-electric material.

Table 3.3 illustrates the composition of real appliance, on which the main power consumption part is mainly used for classifying appliances. However, control and electronics parts may cause noise in high frequency, which can be used for appliance classification. When all of basic electrical appliances working on the electrical network are known, we can find out what kinds of machine are working. Then, we can estimate the power consumption of the machine based on total of power consumption of its basic electrical appliances.

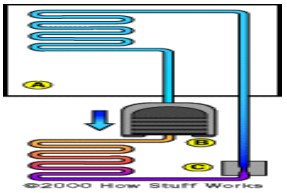
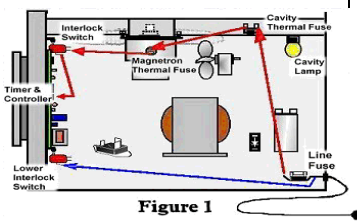
| Appliance | Electrical parts | Characteristic |
|---|---|---|
|  Refrigerator | Compressor<br>Fans<br>Control system<br>Pipe<br>Refrigerant | Main power consumption<br>Specific parts |
|  Microwave Oven | Fan<br>DC lamp<br>Timer controller<br>Power controller<br>Triac driver circuit<br>High voltage transformer<br>The magnetron tube | Specific parts<br>Main power consumption part |

Table 3.3 Illustration about the composition of some real appliances

Thus, although there are various kinds of appliance in real life, but we can classify them into three main categories: the motor (or fan), the lighting and the heating appliance as below table. After investigating database about energy usage of customer, we found that it is possible to early classify the category of appliance based on the shape information of the Turning-ON transition, as illustrated in Table 3.4. From three basic categories, we can develop complex algorithms to recognize specific appliances.

| dP | $P_{max}$ | $P_{min}$ | dt | dQ | $Q_{max}$ | $Q_{min}$ | Event type | Basic Appliance type |
|---|---|---|---|---|---|---|---|---|
| $< 0$ | - | $\approx 0$ | short | - | - | - | OFF | Undefined Appliance |
| $> 0$ | $> dP$ | $\approx 0$ | short | $> 0$ | $> dQ$ | $\approx 0$ | ON | Motor |
| $> 0$ | $\approx dP$ | $\approx 0$ | short | $\approx 0$ | - | - | ON | Heating |
| $> 0$ | - | $\approx 0$ | long | X | $\neq 0$ | $\neq 0$ | ON | Lighting |

Table 3.4 Some basic shape information for early classify appliances

## 3.3. CUSUM - An online Event Detection

An event is a change in any electrical signatures to indicate a changing state of an appliance in the electrical network. Detecting an event is necessary to compute changes in electrical signatures after an event occurrence. Then NIALM system must use this event information to classify appliance and disaggregate energy. Most of NIALM researches used Hart's approach [5], which used the changes of mean input data to detect events. Such an approach defines a steady state to exist if the input value does not vary more than a specific threshold value during defined samples. In contrast, if input value varies more than that threshold, an even occurs. The mean value in steady stage is useful in minimizing noise however; such a method cannot detect slow transitions. In addition, data sampled at a slow rate limits the possibility of detecting short steady states and a fixed threshold makes the system potentially assimilate a large signal noise as an event.

Marisa Figueiredo et al. 2012 [86] improve Hart's method by using adaptive threshold values depending to each steady period in order to detect the event if the next input value was larger than the adaptive threshold. Such an adaptive threshold is defined between (min value + a constant threshold) and (max value - the constant threshold). Even if their approach improves the detection of small power appliances and is better resistant to noise, they still cannot detect appliances with slow transitions.

Men-shen Tais, 2012 [87], proposed another method to detect events based on waveform of current. His method subtracts current waveform in 3 continuous cycles to define the different waveforms and the change rates. Then, change rates are compared to a threshold value to define if they are events. However, three continuous cycles are not enough to detect a slow transition and the research has not tested the system in noise environment.

In order to solve both above challenges, we proposed to use an online event detection using the Cumulative Sum (CUSUM) algorithm. This algorithm was first introduced by Basseville and Nikiforov [88] to detect transients in sequence data. This algorithm has many applications such as monitoring petrol in tanks, tracking GPS signal, filtering noise in earphones, monitoring fault in DC motors etc. The basic CUSUM algorithm for both positive and negative transient detections is presented below:

$$g_k^- = \max\left[0\,,\, g_{k-1}^- - \left(y_k - \hat{\theta}_{k-1}\right) - v\right] \quad \text{and} \quad g_k^+ = \max\left[0\,,\, g_{k-1}^+ + \left(y_k - \hat{\theta}_{k-1}\right) - v\right]$$

$$t_a = \min\left[k : \left(g_k^+ \geq h\right) \vee \left(g_k^- \geq h\right)\right] \qquad \text{and} \quad g_{ta}^+ = g_{ta}^- = 0$$

Where $\widehat{\boldsymbol{\theta}}_{\boldsymbol{k-1}}$ is the estimated value after a low pass filter and $\boldsymbol{y_k}$ is the input value. The *drift parameter* $\boldsymbol{v}$ and the *threshold* $\boldsymbol{h}$ are design parameters. While threshold value $\boldsymbol{h}$ defines the minimum change in time series data that CUSUM can detect, drift parameter $\boldsymbol{v}$ controls the

latency from a real input event to the detected event or the speed of the detector. Finally, $t_a$ is time when the event occurs.

In theory, a standard CUSUM algorithm is one kind of adaptive filter where a low pass filter such as a least square approximation is used to predict the next value. In real life, a measured value $\mathbf{y_t}$ is the total of real input value $\boldsymbol{\theta_t}$ and very small noise $e_t$ as defined in equation (3.20). Thus, when there is a change in measured value $\mathbf{y_t}$, there should be a big change in this noise but we call it now the residual value $\boldsymbol{s_t}$ defined in equation (3.21) where $\widehat{\boldsymbol{\theta}}_{\boldsymbol{t}}$ is the estimated value of real input value. Because the noise $\boldsymbol{e_t}$ is very small so that this residual value $\boldsymbol{s_t}$ can represent the transition event in input data. This value is monitored to detect the transition in a time series data if it is larger than a threshold. The behavior of this event detection can be described by Figure 3.16.

$$y_t = \theta_t + e_t \tag{3.20}$$

$$s_t = y_t - \hat{\theta}_t \tag{3.21}$$

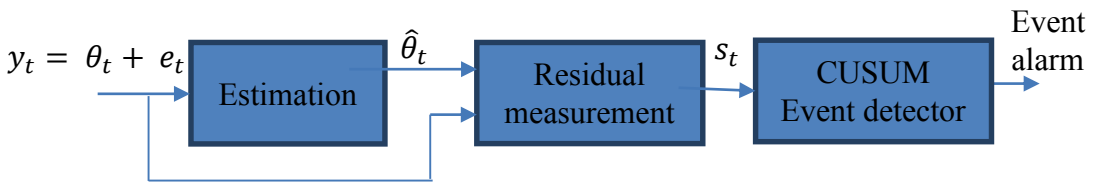



Figure 3.16 General event detection system

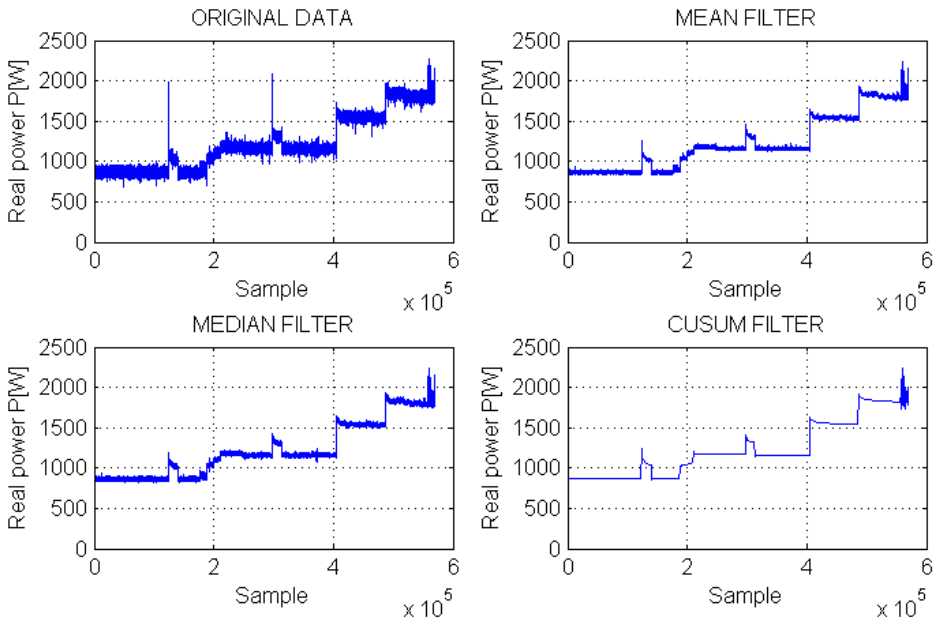



Figure 3.17 Comparison in removing noise between mean filter, median filter and CUSUM filter in a power profile of a lighting sector

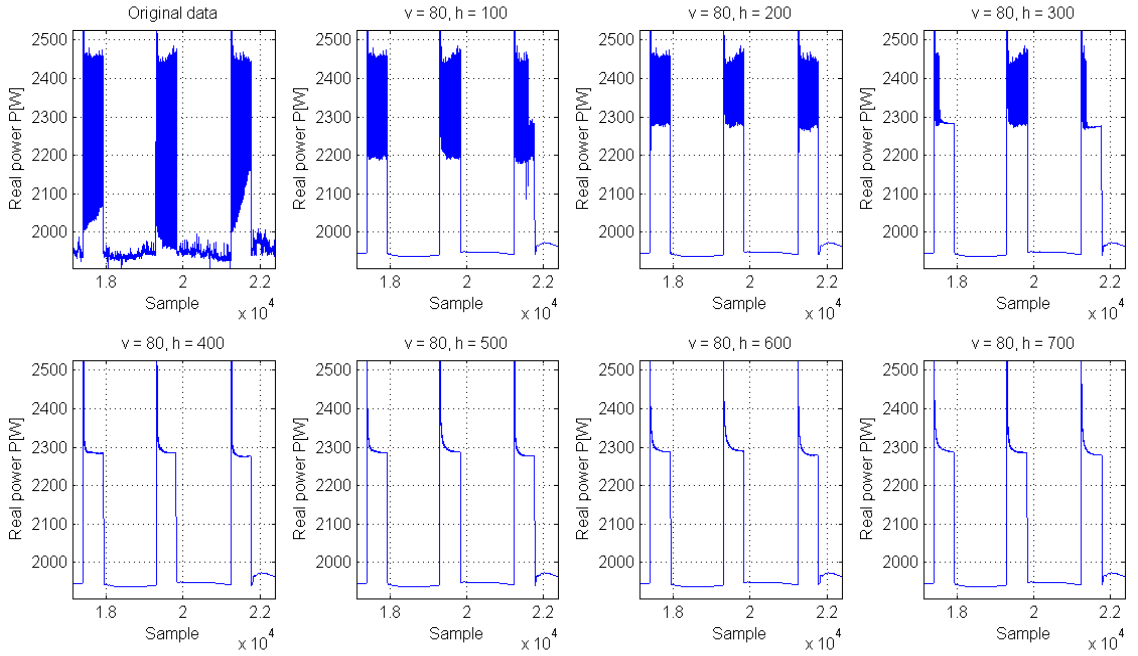



Figure 3.18 Comparison CUSUM filter with different values of threshold h in power profile of an office sector (with v=80)

In our previous researches [89] [73], the CUSUM event detection algorithm was able to detect fast transients but not the beginning and the end of a transient-state. That version was still using fixed threshold value that makes large noise signals in high power consumption appliance to potentially be seen as an event. However, as shown in Figure 3.17, first tests in power profile of some real appliances proved that it is better than the mean and median filters in removing noises in data. However, in Figure 3.18, CUSUM filter in very high noise signal with drift parameter v = 80 and threshold h = 300 cannot remove all the noises so that we need to tune v and h for better result and we found that all values h > 300 give us better filter. Because the threshold of detection algorithm depends on h so that the lower value of h, the lower power consumption appliance the algorithm can detect. In this case, value h = 400 is the best parameter in removing noise and lower power consumption detectable appliances. This is a drawback of this algorithm when it detects well only appliance with power from 400.

We have improved CUSUM event detection algorithm with the adaptive threshold feature and the detection now can detect not only smaller power consumption appliance (from 60W) but it also can define both the beginning and the end of the transient-state signature as shown in Figure 3.19. This capability can solve the slow transient detection problem. The adaptive threshold is defined as the difference between the max value and min value of a signal in each steady state. This feature offers the better resistance to the noise. Moreover, CUSUM event detection can extract the shape information in any linear electrical signatures as we introduced in section 2 about the event-based approach. The detailed algorithm is defined below.

***The modified CUSUM Algorithm***

***Initial:*** $\widehat{\theta_1} = y_1;\ t_1 = 1;\ \ d_1 = 0;$

***Repeat:***

$$\hat{\theta}_{k-1} = \frac{1}{t-t_a} \sum_{k=t_a+1}^{t} y_k$$

$$g_k^+ = max[0,\ g_{k-1}^+ + (y_k - \hat{\theta}_{k-1}) - v\ ]$$

$$g_k^- = max[0,\ g_{k-1}^- - (y_k - \hat{\theta}_{k-1}) - v\ ]$$

$$y_{max_k} = max\ [y_{max_{k-1}},\ y_k]\ \ ;$$

$$y_{min_k} = min\ [y_{min_{k-1}},\ y_k]$$

$$h_k = max[40\ , y_{max_k} - \ y_{min_k}]\ ;\ v = \frac{h_k}{2}$$

$$d_k = d_{k-1} + 1$$

***Start of event rule:***

$$t_s = min\ \{\ k : [(g_k^+ \geq h_k)\ \cup (g_k^- \geq h_k)]\ \ \cap\ (d_k > d_{thr})\}$$

***End of event rule:***

$$t_e = \{\ k :\ (g_k^+ \leq h_k)\ \cap\ (g_k^- \leq h_k)\ \cap\ (d_k = d_{thr})\} - d_{thr}$$

$$d_{ts} = 0;\ \ y_{maxt_e} = y_k\ ;\ y_{mint_e} = y_k$$

***Stop rule:***

$$t_a = min\{\ k: (g_k^+ \geq h)\ \cup\ (g_k^- \geq h)\ \}$$

$$g_{ta}^+ = \ g_{ta}^- = 0$$

$$d_k = 0$$

***Output:*** *Mean value* $\hat{\theta}_t$

*Start time of transient-state* $t_s$

*End time of transient-state* $t_e$

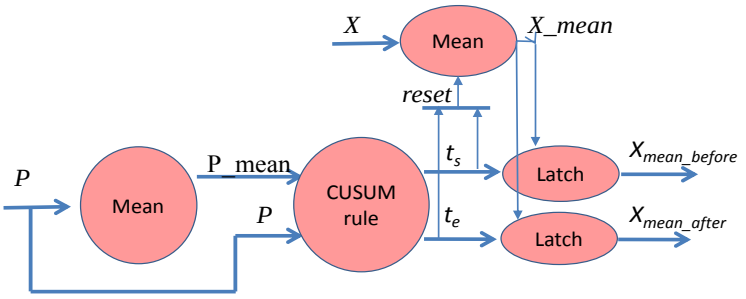



Figure 3.19 The refinement activity model of the CUSUM detection

In above algorithm, $d_k$ is the number of samples of current status of an appliance. $t_s$ is the beginning of a transient that happen when there is a change of value y in current steady state. Steady state is defined when $d_k$ is larger than a threshold population $d_{thr}$. This threshold depends on the sampling frequency by dividing minimum time of a valid steady state to period of

sampling frequency. $t_e$ is the end of a transient. $t_e$ is defined when appliance is in transient state and the condition of a steady state is satisfied. The input signature x will be latched its average value before $t_s$, its average value after $t_e$ and the min value and max value during $t_s$ and $t_e$. We then can extract changes in real power dP, reactive power dQ, coefficient $dI_{ak}$ $dI_{bk}$ , dTHDi; max values Pmax, Pmin, Qmax, Qmin and the duration of the transient dt.

Moreover, differences in $dI_{ak}$ and $dI_{bk}$ are linear variables that can compute easily by subtracting their mean values after and before the event then we can compute $I_k$ of new appliance who causes the event by equation $I_k = \sqrt{dIa_k^2 + dIb_k^2}$. The harmonic information of new appliance after an event then is recomputed by equation below. Finally, Total Harmonic Distortion of the new appliance is computed by equation:

$$THD_i = \frac{\sqrt{\sum_{k=2}^{\infty} I_k^2}}{I_1} .100\% \qquad (3.18)$$

Other harmonic information is individual harmonic distortions (IHD) which are computed by equation:

$$IHD_{ik} = \frac{I_k}{I_1} .100\% \qquad (3.19)$$

A functional verification of this algorithm has been conducted with the data set REDD [90] in MATLAB environment using data sampled at a 15 kHz rate as illustrated in Figure 3.20 and Table 3.5. It shows that the CUSUM works well in detecting the event as well as extracting many electrical signatures of appliances as stated in the requirements. The accuracy of CUSUM event detection is very high for high power appliance for example 100% in Bath-gfi 1600W, and quite low for low power appliance such as microwave oven 165W. Such an error may come from the compressed algorithm of the REDD data set which forms data by [time, quantity, 275 samples of a cycle]. However, the accuracy constraints for some main appliances are satisfied. After detecting the transient, the changes in real power, reactive power and the THDi as well as their max values, min values and the duration of the transient are calculated. Table 3.5 proves the capability of detecting the multi-state appliances such as, for example, the furnace, which has three, states with averaged power consumption of Furnace App 1 (620W), Furnace App 2 (440W) and Furnace App 3 (100W).

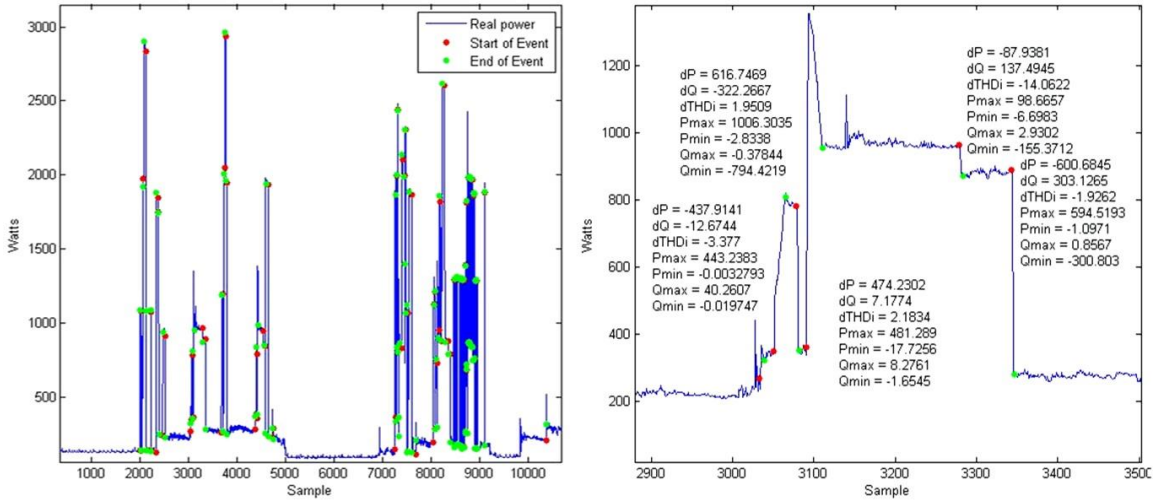



Figure 3.20 Results of CUSUM event detection algorithm in extracting the beginning, and the end of transients for both fast and slow transients

| Circuit - Appliance | Sub-appliances | Total events (1Hz data) | Detected events (15kHz data) | Accuracy [%] | P_max ON |
|---|---|---|---|---|---|
| Washer-Dryer | 2250W | 140 | 138 | 98.57 | 2640W |
| Furnace | 620W<br>440W<br>100W | 70<br>66<br>66 | 65<br>62<br>65 | 92.85<br>93.93<br>98.48 | 620W<br>450W<br>190W |
| Microwave | 1750W<br>165W | 62<br>16 | 54<br>10 | 87.1<br>62.5 | 1730W<br>160W |
| Bath-gfi | 1600W<br>1300W<br>950W | 13<br>11<br>8 | 13<br>10<br>8 | 100<br>90.9<br>100 | 1650W<br>1300W<br>1000W |
| Electronics | 1100W | 130 | 100 | 76.92 | 1120W |
| Kitchen outlet | 1250W<br>950W<br>380W | 17<br>42<br>44 | 17<br>41<br>33 | 100<br>97.6<br>75 | 1250W<br>950W<br>380W |

Table 3.5 Results of CUSUM event detection for main appliances in 1-Week in the House 3 in REDD Data Set

## 3.4. Genetic Algorithm-based power Disaggregation

The main task of the NIALM system is to disaggregate total power usage to a set of power usage for every appliance. As discussed in section 1, a latency constraint of a disaggregation processing time has to be satisfied with a maximum of 5 seconds to display the associated output to an input event. It requires a disaggregation process that can classify appliances in a time as short as possible.

Michael Baranski and Jurgen Voss in 2004 [8] have proposed the use of a fuzzy clustering and a genetic algorithm to disaggregate appliance based on power information for NIALM system. Fuzzy clustering initially clusters all events detected upon a change of power larger than a threshold value of 80W. After that, a genetic algorithm will determine the appliance from best-fit clusters. A first limitation with such an elementary detector arises when it is confronted with the detection of slow transient appliances. Moreover, their disaggregation Genetic Algorithm method does not process input data in real-time and, thus, a fuzzy algorithm must be used to select the best appliance from detected appliances.

Suzuki, Kosuke et al. in 2008 [10], used an integer programming method based on the cycle-period of the current waveform to estimate the condition of electrical appliance even with multi-state load appliance. Such a method, based on steady-state disaggregating, is not effective because the non-linear characteristic of current makes the aggregated current waveform undefined based upon the sum of known currents waveform. The necessity to operate on a known database is also a major drawback of this method. There are some other studies, also based on trained database such as the works of Figueiredo et al. 2012 [86], Patel et al. 2007 [11] and Froehlich, Jon et al. 2011 [12].

Akshay Uttama Nambi, S.N., et al. 2013 [91] have employed a probabilistic Hidden Markov Model (HMM) to model the time series of aggregated power but the knowledge of the number of appliances with their average and peak power consumption as well as their number of states is needed. This steady-stated method also has the big drawback to require a number of iterations that is an exponential function of the number of appliances. Moreover, collecting information about all appliances in every house is not practical.

While most previous researches require a database and training step for appliance disaggregation, we propose a new adaptive disaggregation algorithm that can learn new appliances and then, can build the database by it-self. As illustrated in Figure 3.4, the disaggregation algorithm includes two main algorithms: Event Clustering (EC) algorithm and the Genetic Algorithm (GA). Event Clustering – a sequential version of K-mean algorithm, will group events to clusters where their centroid will represent events including dP, dQ, dTHDi, dt information. Then, the Genetic Algorithm (GA) processes a sequence of detected events to discover groups of events that can possibly match an appliance based on a fitness function thereby enabling the detection of multi-state appliances. In addition, the Appliance Finding (AF) algorithm needs to determine the appliance name in the database thanks to its list of events. If the detected appliance does not exist in the database, Appliance Finding will automatically assign a new name and will save it in the database. Finally, we can also predict at an early stage the appliance name after Event Clustering using a statistic and probabilistic event information in the database.

### 3.4.1 Sequential clustering K-mean

The main purpose of clustering algorithm is to collect the centroid data of the events when their values vary in each measurement. This algorithm acts as a low pass filter to remove noise in measured data. The sequential clustering algorithm was proposed by Konstantinos Slavakis and Sergios Theodoridis [92] for real-time applications. In contrast to the hierarchical clustering algorithm that proceeds once all data available, the sequential clustering algorithm runs whenever there is a new input data and, thus, works in a straightforward and an efficient manner. The algorithm is defined:

**Sequential clustering K-mean algorithm**

Given initial set of k means value $m_1$, $m_2$ … $m_k$, with $m_i$ is the mean value of cluster i of transients. This algorithm has two steps:

1. Assignment step: compute distance from an event $x_p$ (dP, dQ, THDi, dt) to all value $m_k$ of a cluster of events.

$$S_i^{(t)} = \{x_p: \left(\left\|x_p - m_i^{(t)}\right\|^2 \le h\right) \cap \left(\left\|x_p - m_i^{(t)}\right\|^2 \le \left\|x_p - m_j^{(t)}\right\|^2\right) \forall\, 1 \le j \le k\}$$

2. Update step:

- If the minimum distance from event $x_p$ to cluster is less than a threshold distance h, it will be assigned to cluster $S_i$ and the centroid value of this cluster will be updated to a new value

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

- If the new value $x_p$ does not satisfy equation (*) in any clusters, it will belong to a new unknown cluster.
  - $k = k + 1;$
  - $m_k = x_p;$

The threshold *h* plays an important role in deciding if a data belongs to a list of known events or if it is a new event. The parameter represents the space around the center of a cluster, meaning that if the new event is inside this space, it will belong to the cluster and vice versa. However, the size of this space should not be the same for all clusters because 80W cluster has a measured value ranging from 40W to 120W but the 4kW cluster has measured values ranging from 3,8kW to 4,2kW. In order to overcome this problem, the threshold h is dynamically set in the below function.

$$\mathrm{h} = \sqrt{(\mathrm{Kp}.\,\mathrm{dP})^2 + (\mathrm{Kq}.\,\mathrm{dQ})^2 + (\mathrm{Kthd}.\,\mathrm{dTHDi})^2 + (\mathrm{Kt}.\,\mathrm{dt})^2}$$

Where Kp, Kq, Kthd and Kt are tuning parameters for the variation of dP, dQ, dTHDi and dt.

### 3.4.2. Genetic Algorithm

From the refined event list obtained after sequential clustering, some ON and OFF events can be easily matched to pair ON-OFF based on the rule that the total changes of real and reactive power of events in every pair is approximately zero. However, some appliances have multi-states that this simple pair-matching algorithm cannot detect. Such an issue leads to the new definition below:

(1) Find the groups of events that satisfy the total of changes in real, reactive power and harmonic is approximately zero.
(2) The group of an event must start with ON event and end with OFF event.
(3) If there is an OFF event $E_k$ in the found group [$E_1$, $E_2$... $E_N$] with $1 < k < N$, the total of changes in real, reactive power and harmonic from the first event to event k must be larger than zero.

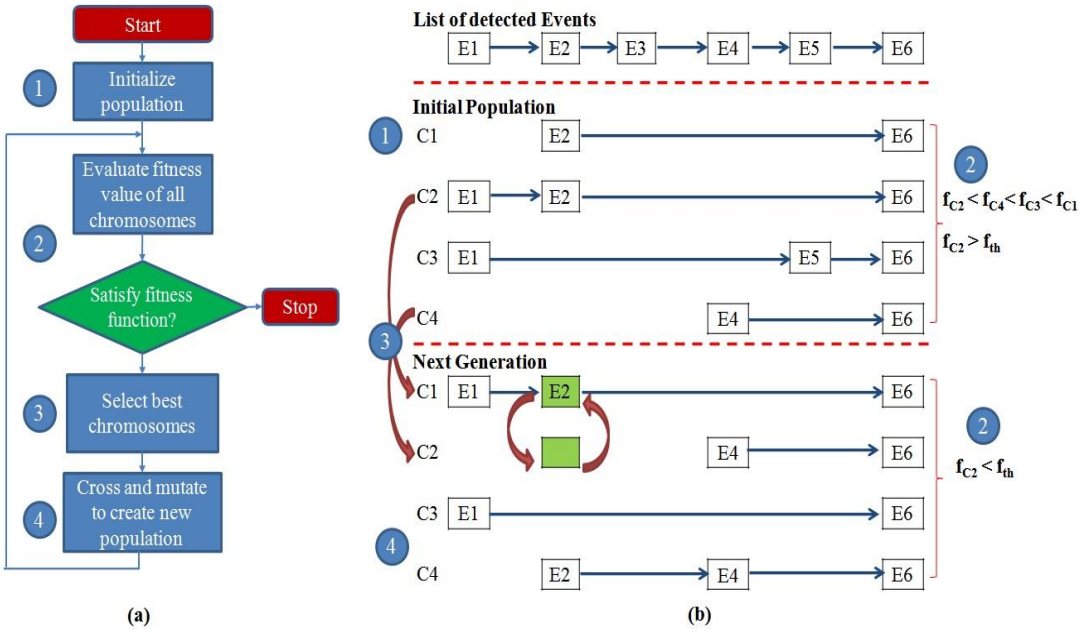



Figure 3.21 GA algorithm (a) and illustration of GA operation in NIALM system (b)

A simple search algorithm can solve the problem described above; however, a long sequence of events will require a long computation time with a large number of iterations. A Genetic Algorithm [93] can solve this difficult problem. The algorithm, with terms population, chromosome, fitness function, selection, crossover and mutation, simulates the evolution of creatures in nature to select and create the best generation progressively in time. A chromosome is the representation of the problem in the way it can be solved in programming. The best chromosome is the best solution of the search. In GA, searching works based on grouping of chromosomes or in other words, the population of chromosomes. Crossover and mutation are special operations that exchange and change some characteristics between two chromosomes to create new chromosomes. New chromosomes are the next generations of two old chromosomes.

Such a function is a special means to compute how close the chromosome is to the best solution. The fitness function helps in selecting the best chromosomes to create the new generator. In this way, chromosomes in new generations are always better than their parents so that the finding is able to converge much faster. Bandyopadhyay, Sanghamitra [93] proved that GA always converges if the iteration quantity is sufficiently large and the convergence of GA is independent of a crossover operation but the probability in process the mutation operation should be greater than 0 and in the [0, ½] interval. As shown in Figure 3.21 (a), GA processes the searching with random groups to speed up the convergence of the fitness function following the listed tasks:

(1) Generate a first random population of four chromosomes.

(2) Evaluate the fitness function of each chromosome in this population. The fitness function is a formula of changes in real power, reactive power and total harmonic distortion.

(3) Select two chromosomes that have the best fit.

(4) Proceed to crossover and mutation to generate a new population from two selected chromosomes. This population potentially contains a chromosome that satisfies the fitness condition better than the previous one.

(5) Repeat steps (2) to (4) until the fitness function is satisfied or until a maximum number of iterations is reached. In example of Figure 3.21 (b), convergence is reached in only two iterations. In this example, we assume that there is a list of detected events data (ON/OFF) from E1 to E6 and group {E2, E4, and E6} is the full operation cycle of an appliance. That means this appliance has three states: E2-ON, E4-S2, E6-OFF and the summary of event information F(E2) + F(E4) + F(E6), which is called the fitness function, should be approximately equal to zero. In this way, regular searching algorithms can solve this problem but can require 26 or 64 iterations in the worst case and the long event lists consume huge computation time to iterate.

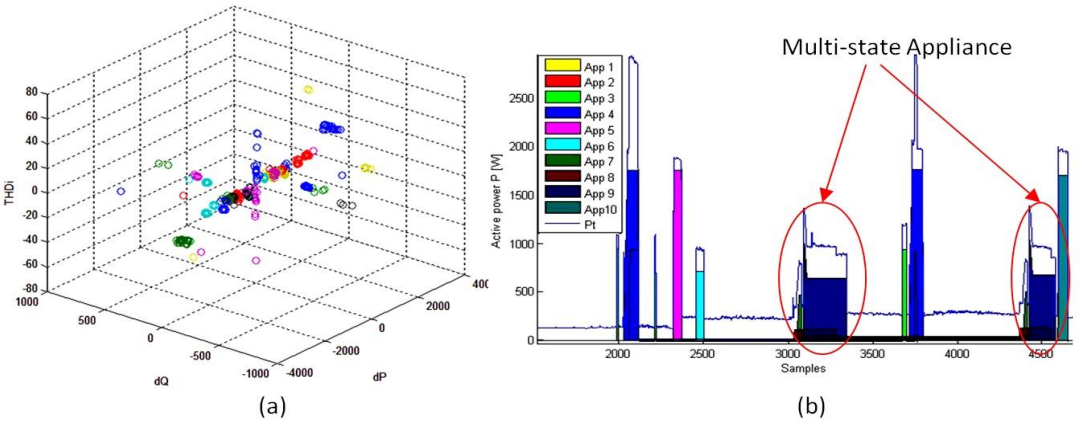



Figure 3.22. Sequential clustering results (a) and disaggregated pattern by GA on REDD 16/04/2011 (b)

| Appliance type | P [W] | Q [W] | THD [%] | dt_ON [samples] | Accuracy [%] |
|---|---|---|---|---|---|
| Washer Dryer 1 | 2264 | 29 | 2 | 11->17 | 95.59 |
| Bathroom gfi 1 | 1693 | -11 | 3 | 2->4 | 100 |
| Microwave Oven 1 | 1690 | -319 | 40 | 23->29 | 85.19 |
| Outlet 3 App 3 | 1293 | -2 | 3 | 2 | 76.47 |
| Electronics App 1 | 1133 | 0 | 2 | 2 | 95 |
| Bathroom GFI App 2 | 1065 | -18 | 2 | 28 | 66.67 |
| Lighting 1 App 1 | 1014 | -120 | 18 | 25 | 100 |
| Outlet 3 App 1 | 942 | -13 | 3 | 3 | 87.1 |
| Furnace App 1 | 666 | -320 | 2 | 1 | 100 |
| Furnace App 2 | 442 | 4 | 2 | 14 | 96.55 |
| Outlet 3 App 2 | 387 | 2 | 2 | 10 | 90.91 |
| Lighting 2 App 1 | 197 | 2 | 7 | 3 | 57.14 |
| Lighting 5 App 1 | 180 | -62 | 21 | 6 | 62.5 |
| Lighting 4 App 1 | 132 | 29 | 51 | 4 | 72 |
| Refrigerator App 1 | 123 | -25 | 8 | 16->17 | 94.46 |
| Furnace App 3 | 91 | -128 | 12 | 5 | 93.55 |

Table 3.6 Disaggregation results in 1-week in house 3 in REDD data set.

We validated this disaggregation algorithm in the REDD data set and got the results in Figure 3.22. Figure 3.22 (a) shows that the correction in functional of sequential clustering algorithm and Figure 3.22 (b) visualizes the feature disaggregation of Genetic Algorithm without the use of an initial database for the appliance. Detected appliances are initially given a default name; however, the customer can reassign the name for each appliance by using database management feature. Besides the capability to disaggregate and classify ON-OFF appliance, the Figure 3.22 also shows the ability in disaggregating multi-state appliances. Moreover, the Disaggregation results in 1-week in house 3 in REDD data set on Table 3.6 has quite high classification accuracy which satisfy the system requirement.

## 3.5. Conclusion

In this chapter, we have presented the development of algorithms for NIALM application. Firstly, we defined system specification of innovative NIALM system, which should overcome many challenges of NIALM technology in extracting more electrical signatures, detecting multi-states appliance and aiming to the development of real-time NIALM system. Then, overall

activity of system is presented based on Dataflow model. We also presented a studying about designing algorithms to extract electrical signature. Then, detail algorithms Cumulative Sum (CUSUM) for event detection and Genetic Algorithm for power disaggregation were presented. Finally, some functional verification results in processing a public NIALM data set REDD were presented to analyze the accuracy of system.

In the next chapter, we will present the implementation of these NIALM algorithms into a SoC with FPGA acceleration. This FPGA acceleration is needed for system to satisfy hard constraints in timing and hardware resources as defined in this chapter. In the following chapter, we will also present applying hardware software co-development approach proposed in chapter 2 to prototype the NIALM system quickly.

# CHAPTER 4. SOC IMPLEMENTATION OF NIALM SYSTEM

**Contents**




**Abstract:**

*This chapter presents a use case of SoC HW SW codevelopment methodology proposed in chapter 2 for the development of the NIALM system. Many experimental results will be presented to prove that implementing a SoC system with FPGA acceleration is supported well with this approach. With HW SW cooperation, the system even satisfies much more than the hard constraints of the system relevant to timing and hardware resources usage.*

# Chapter 4. SoC Implementation of NIALM system

## 4.1. The Zynq-7000 platform

In the scope of this research, we selected SoPC is the Zynq®-7000 family of Xilinx because its innovative architecture allows a high-level synthesis approach based on high-level synthesis language such as C/SystemC, Labview FPGA. With both FPGA and processor core inside the chip, Zynq is a very important evolution of FPGA technology, which allows applying various hardware software co-development approaches. Although, there are some similar SoPC technologies before Zynq such as the PowerPC in FPGA Virtex family of Xilinx, the low power and high performance of ARM processors in Zynq bring a really low cost and powerful platform in variety complex applications.

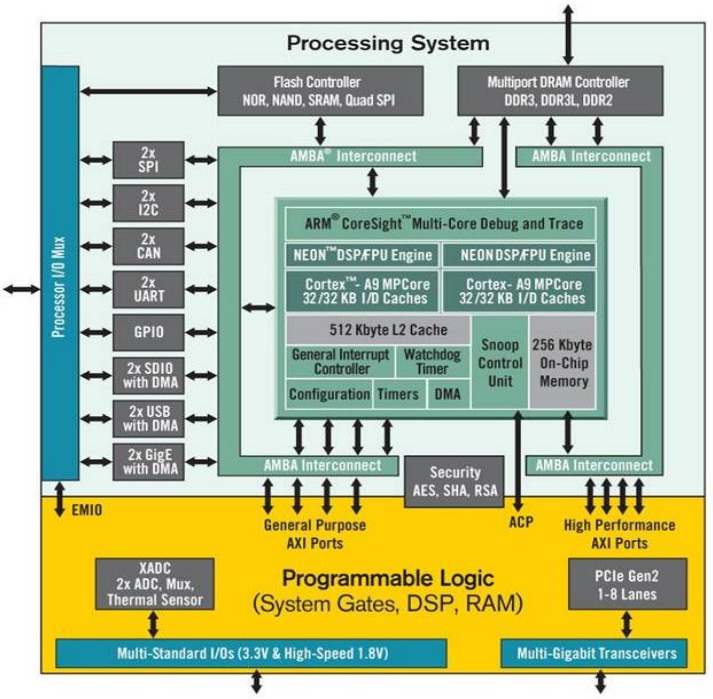



Figure 4.1 Overview of the architecture of Zynq-7000 SoC [25]

As shown in Figure 4.1, some highlights of Zynq®-7000 are this Xilinx's products contain a Processing System (PS) based on ARM core and the Programmable Logic (PL) based on FPGA in a single device in 28 nm, high-performance, and low power (HPL) technology. Moreover, this IC also contains most of the necessary hardware for all embedded applications including on-chip memory, external memory interface and a rich set of I/O peripheral.

### Central Processing Unit (CPU)

The Processing System of Zynq is composed of dual-core ARM® Cortex™–A9 MPCore working in high frequency from 667MHz that can be configured to run the application in single processor or dual-core processor architecture. Moreover, each processor has its own media-processing engine (NEON) which extends the functionality of Cortex-A9 in processing vector

floating-point instruction set. Other important elements for PS in Zynq are the memory management unit (MMU) and separate 32 KB level one (L1) instructions and data caches.

### Programmable Logic

As general FPGAs, Programmable Logic of Zynq has some components such as Configurable logic blocks (CLB), RAM blocks, DSP48E1 Slices, Clock Management, Configurable I/Os. Moreover, there are also other extended integrated elements: low-power gigabit transceiver, dual 12-bit 1 MSPS analog-to-digital converters (ADCs), and interface block for PCI Express communication standard that are specialized for monitoring and controlling applications. In Zynq's applications, CLBs are commonly used to implement the high performance through put data processing algorithms. DSP48E1 slices contain high-resolution 48 bits multiplier/accumulator that provides an ability in optimizing the performance and power consumption of DSP functions. Configurable I/O technology allows setting IO port to work as input, output or bi-direction ports in many voltage levels 1.2 V, 1.8 V or 3.3 V that adapts quickly to external peripherals with no need the logic level shifter ICs.

### Interconnections

In order to manage inside communications, Zynq-7000 uses some interconnect technologies such as AXI High Performance Data path Switches for PS interconnections; AXI_ACP connects to the snoop control unit for cache coherency between the CPUs and the PL; AXI_HP with four high performance/bandwidth master ports in the PL and AXI_GP for four general-purpose ports. There are about 3000 connections between PL and PS, which allows hardware acceleration function designed in PL can access memory resources in PS effectively. Moreover, the DMA controller with four channels for PS (memory copy to/from any memory in system) and four channels for PL (memory to PL, PL to memory) plays an important role in data transactions inside PS and PL.

#### HW/SW Partitioning

Although the architecture of Zynq is quite complex, it is not difficult to develop a system based on Zynq with LabVIEW and LabVIEW FPGA. Figure 4.2 illustrates a FPGA project in LabVIEW base on Zynq SoC architecture in NI myRIO-1900 board. HW/SW partitioning a system in Zynq can be done by locating designed model into the right folder in the project. As example in Figure 4.2, module RT-Processes will be implemented in processor while module FPGA-Processes will be implemented in FPGA. These modules will communicate together using FIFO channels. In this figure, the FIFO channels created in FPGA Target folder transfer data between RT Processes function running in Processor and Event Detection and Preprocessing functions running in Program Logic. The left figure shows all hardware resources

connecting to the processor system and the programmable logic with the built-in interface under the FPGA target as shown in the right figure.

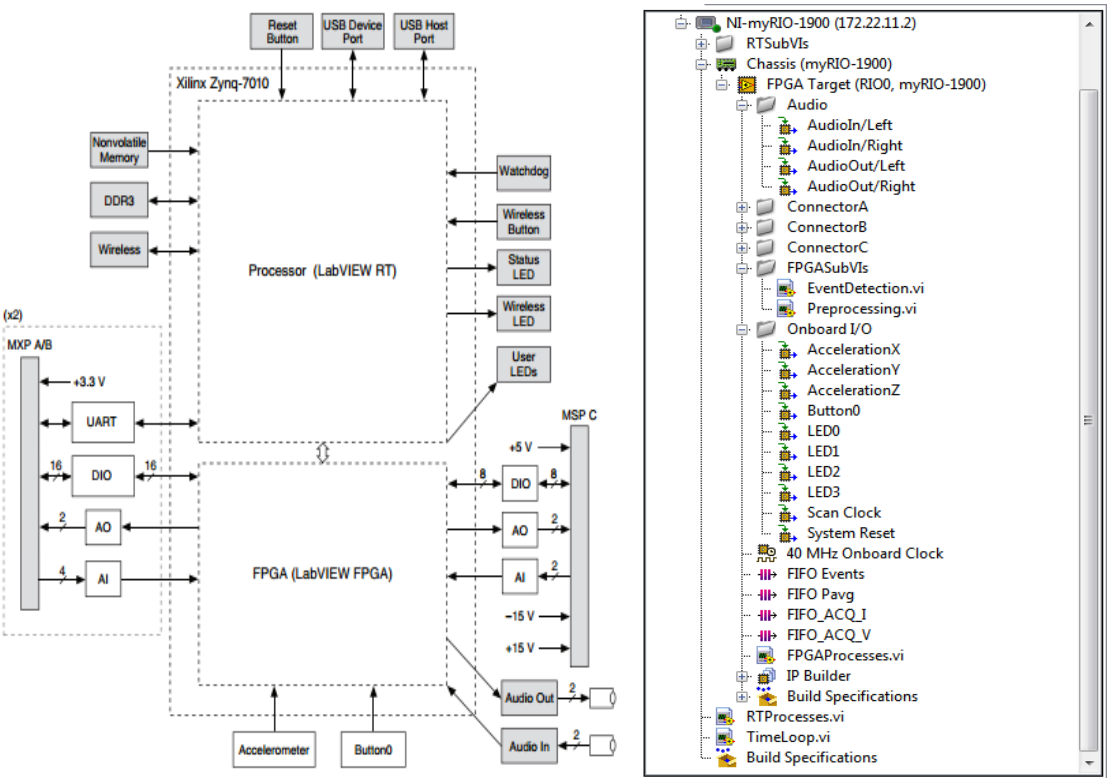



Figure 4.2 SoPC Zynq architecture in myRIO-1900 platform and its LabVIEW project

In the next section, we will present the overall system design work of our NIALM system using this tool and development approach proposed in chapter 2 from executable specification of the system to architecture exploration and prototyping the system.

## 4.2. Executable specification

### 4.2.1. Modeling Virtual Appliances

Virtual Appliance model is used to emulate an electrical network to test the design NIALM system. As illustrated in Figure 4.3, three virtual appliances are modeled in MathScript codes from four order of harmonics and their shifting phases. We issued that the total voltage is only in fundamental frequency and 230V amplitude. The Time Loop structure means in each 20 ms, the system can acquire 40 samples of current and voltage signals as designed SDF model of NIALM in chapter 3. FIFOs variables I and V are used to store these acquisition samples. This model starts to run when switch Start Simulation is turned ON and electrical information such as amplitude, frequency of harmonics in each active virtual appliance are set.

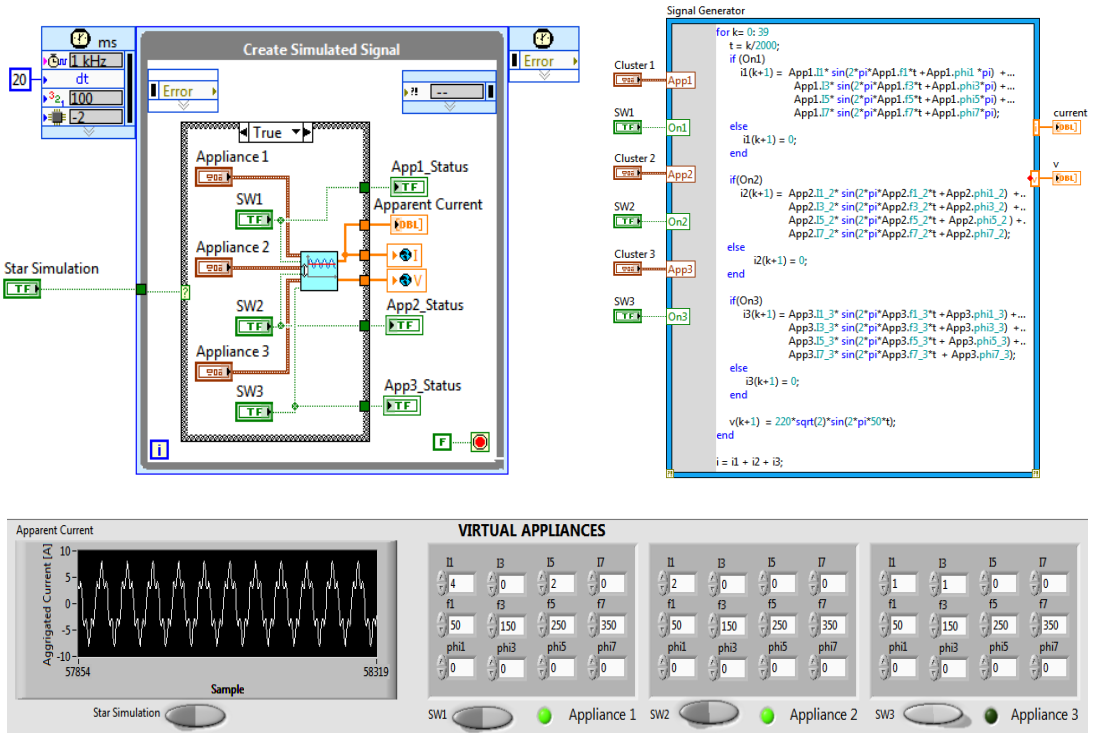



Figure 4.3 Electrical network model in LabVIEW MathScript with three virtual appliances

## 4.2.2. Modeling the NIALM process

The dataflow model of NIALM process is modeled as in Figure 4.4, with Preprocesing connected to the sample acquisition through the communication channel FIFO I and V. As designed in SDF model in chapter 3, both three processes Preprocesing, Event Detection and Disaggregation work in 50 Hz rate and transmit only one package to each other in every cycle. Therefore, a Time Loop 20ms is used to control this model. Start Power Monitor is a control signal of the ***Control Logic*** to activate the power monitoring service.

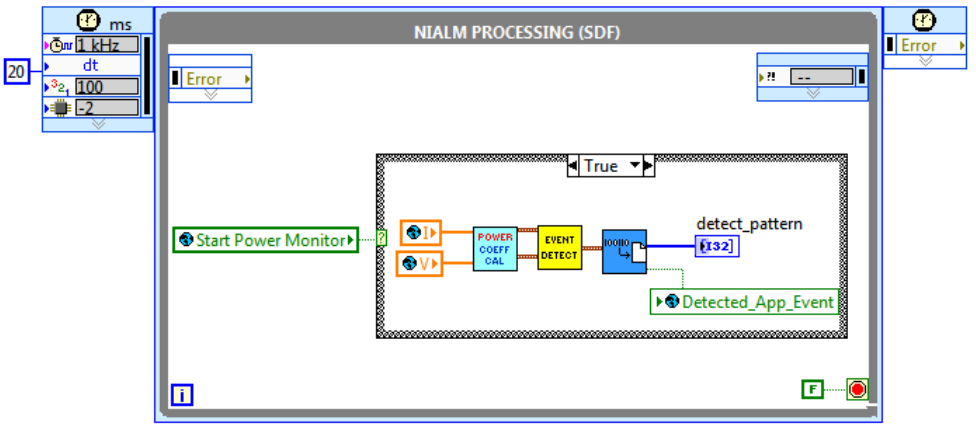



Figure 4.4 Modeling the NIALM dataflow in LabVIEW

In order to model the Preprocessing processes, which are used to compute power, and harmonics data, MathScript models are used as shown in Figure 4.5. However, LabVIEW can compile MathScript models to the embedded codes that are able to run in processors.

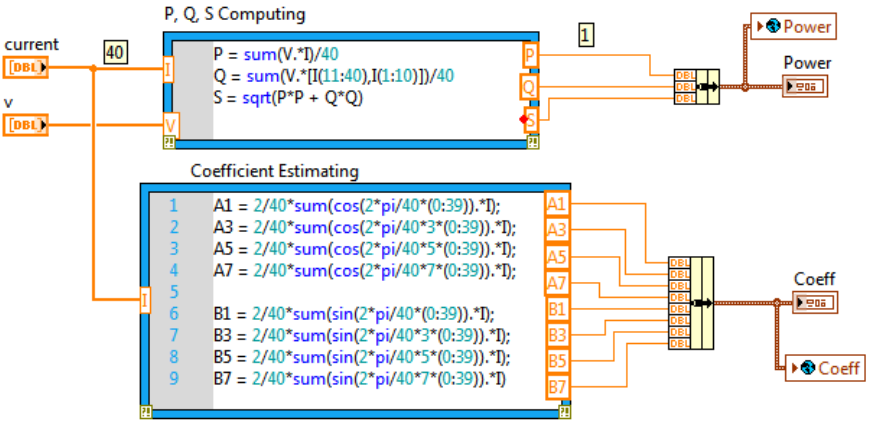



Figure 4.5 Modeling the Preprocessing using MathScript

Figure 4.6 is the model of CUSUM Event detection. This complex DSP algorithm is composed by three functions Filter, CUSUM rule and Data Extract. This Event data includes signals Event Start, Event End, the changes in real power, reactive power and the Total Harmonic Distortion. Disaggregation based on Genetic Algorithm is modeled in MathScript in Figure 4.7 to keep the algorithm from MATLAB code for quickly functional validation the system.

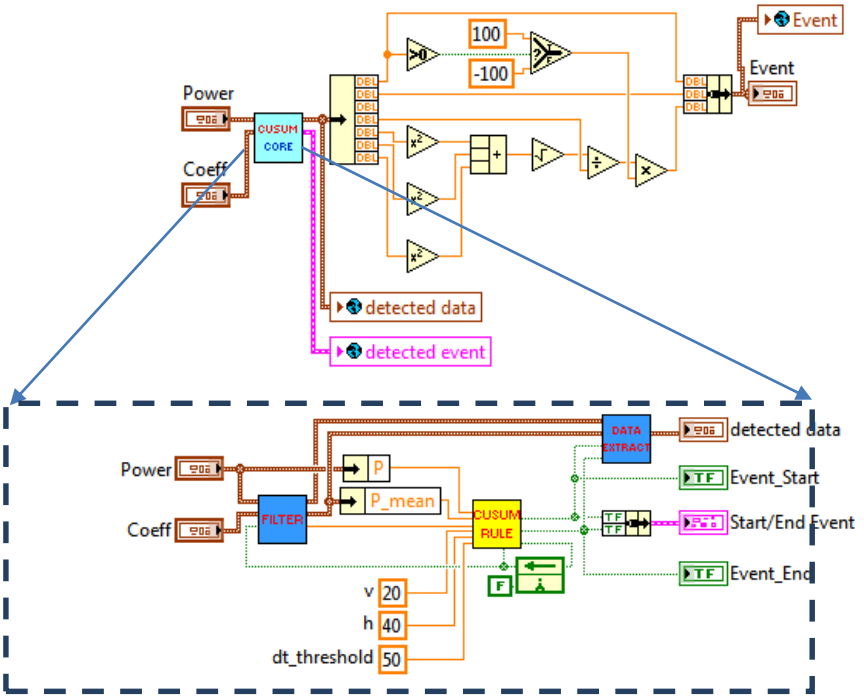



Figure 4.6 Modeling the CUSUM event detection in LabVIEW

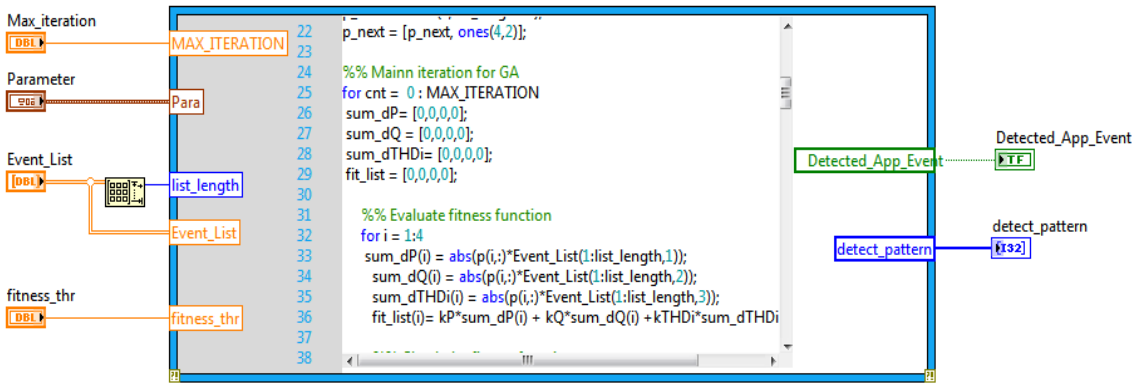



Figure 4.7 Modeling the Genetic Algorithm in MathScript

### 4.2.3. Modeling Control Logic

As we developed the activity model of NIALM system in chapter 3, Control Logic handles the interaction between system and users who can access the system for to two main services: power monitoring service and database management service. LabVIEW StateChart was used to model this Control Logic as illustrated in Figure 4.8.

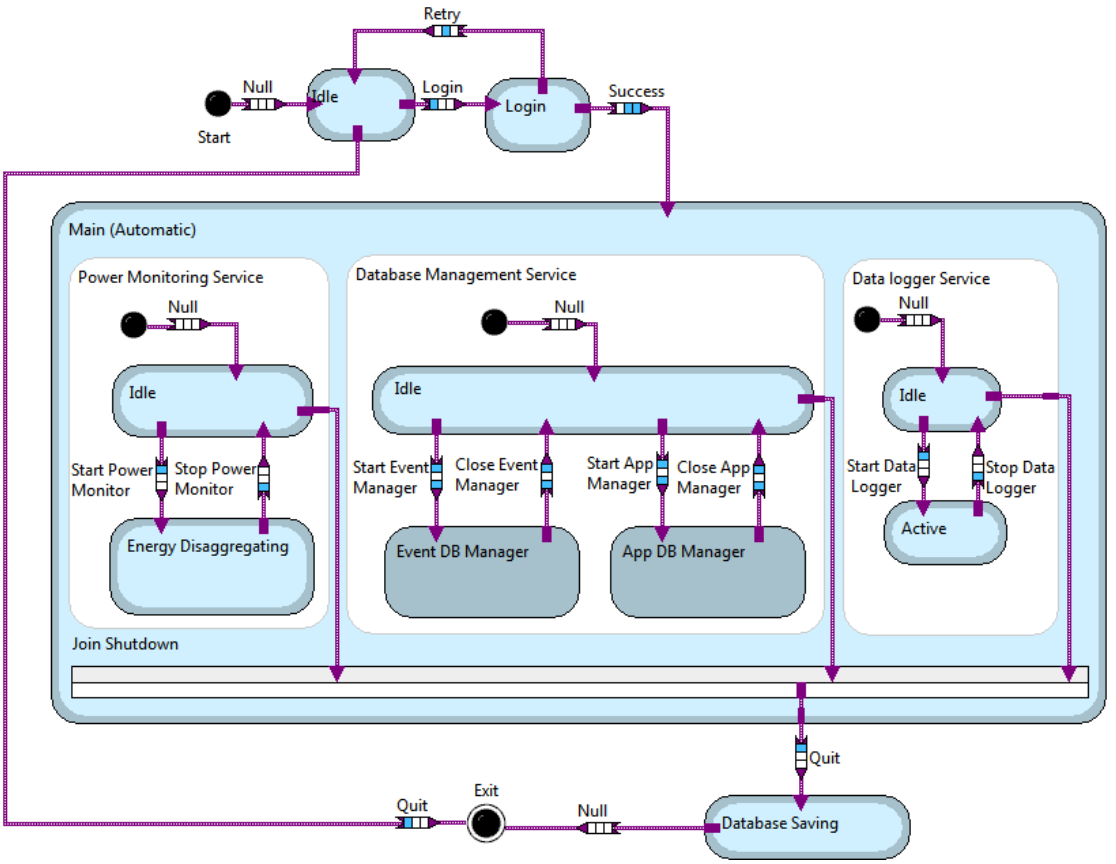



Figure 4.8 The StateChart model of the system with three main services Power Monitoring, Database Management and Data Logger Service

This StateChart model ( Figure 4.8) controls the other processes by using the control message shared variable. Depending on state transition, StateChart model generates different controlling messages:

- "Active Event Manager": when the system login is successful (Figure 4.9).
- "Start Power Monitor": when the StateChart change to Main (macro state) and the Power Monitoring Service starts automatically and turns from Idle state to Energy Disaggregating state. After activate this service, NIALM processes start to run.
- "Stop Power Monitor": when users active Database Management Service.
- "Start Event Manager" and "Stop Event Manager", "Start App Manager" and "Stop App Manager": to active and de-active managing database of Events and Appliances

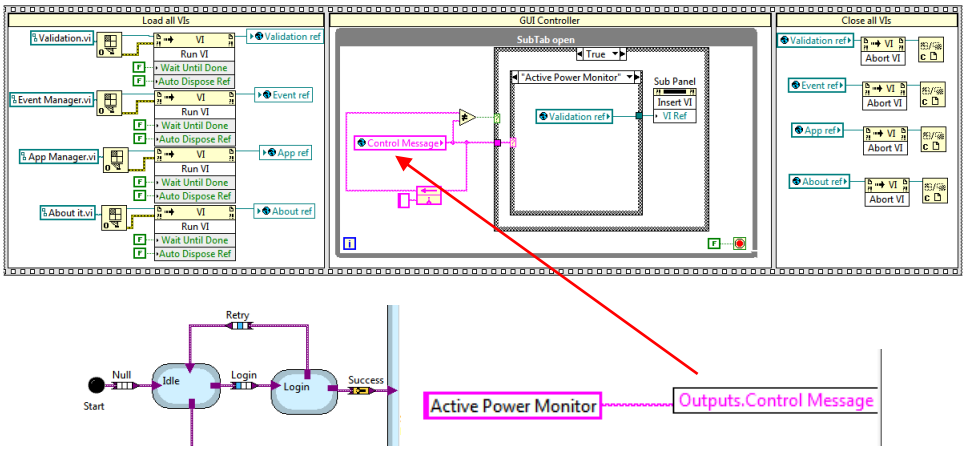



Figure 4.9 StateChart sends the message "Active Power Monitor" to global shared control message to active the NIALM function when the login process is successful

### 4.2.4 Disaggregation functional validation

As an executable specification, it presents better the activity of the system. The power-monitoring service works quite well when it can detect and recognize all known appliance. As demonstrated in Figure 4.11, the GUI can display information of total real power, reactive power, detected event and then match OFF transient to ON transient to detect the appliance. The results show that LabVIEW is a good mode-based design approach, which is able to model a complex Reactive Process Network system on both dataflow model for streaming processes and StateChart model for event-based processes.

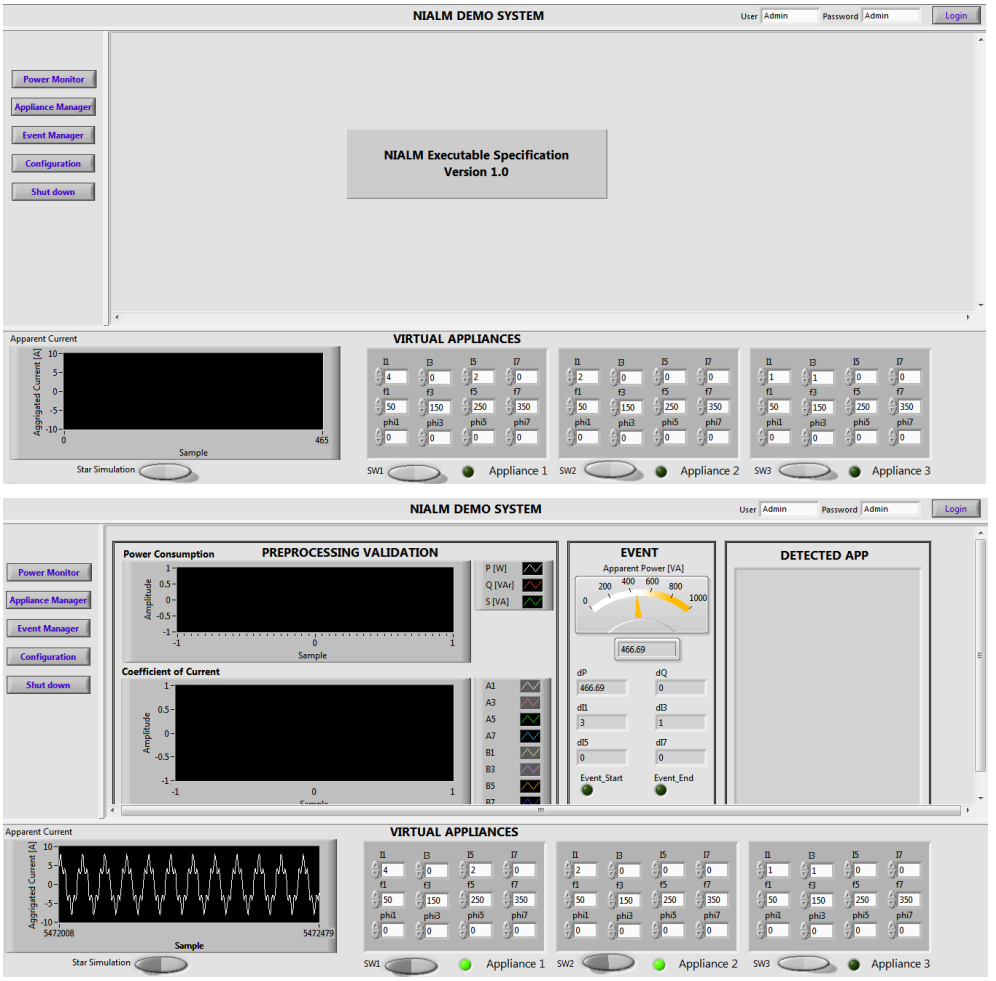



Figure 4.10 Start GUI of the executable Specification before and after the successful login with NIALM function activate

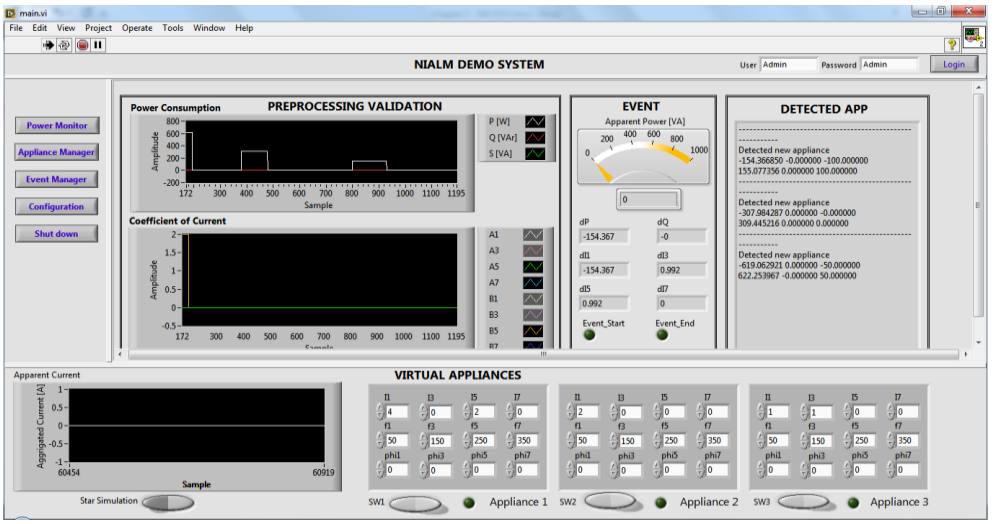



Figure 4.11 An executable specification starts the main NIALM function when it receives command Power Monitor from pushing button Power Monitor of user

## 4.3. FPGA development approaches

In comparison to memory consumed for processor functions, hardware resource consumed of functions in FPGA is much more important because it is much expensive than memory for software code. SDF model can optimize buffer size of FIFOs in communication channels and improve the throughput process of system with static scheduling. However, this model is not an efficient approach for optimizing the FPGA hardware resources and power consumed in each process. Fortunately, LabVIEW FPGA supports prototyping the system to measure these criteria directly and quickly. Because this tool supports develop FPGA hardware in some approaches, we investigated development a CUSUM algorithm using these approaches for comparison. As illustrated in Figure 4.2, a VI function will be converted to assembly code if it is located in the Chassis folder (example RTProcess.vi) or it will be converted to RTL code if it is located in the "FPGA target" folder (as EventDetection.vi and Preprocessing.vi).

### VHDL manual coding

Figure 4.12 presents the CUSUM model in traditional method and detail register level after synthesizing in Xilinx ISE. The design verification in Xilinx Simulator loads input data to the system and records the output data after the CUSUM. Then the results are compared to the results of functions designed in MATLAB, illustrated in Figure 4.13. We then used the Xilinx Timing Analyzer tool to evaluate the performance of system and the Xilinx XPower Analyzer to analyze the power consumption of system. The results were collected in the case implementing system in the target Spartan xc3s500e in automatic synthesis without manual optimization.

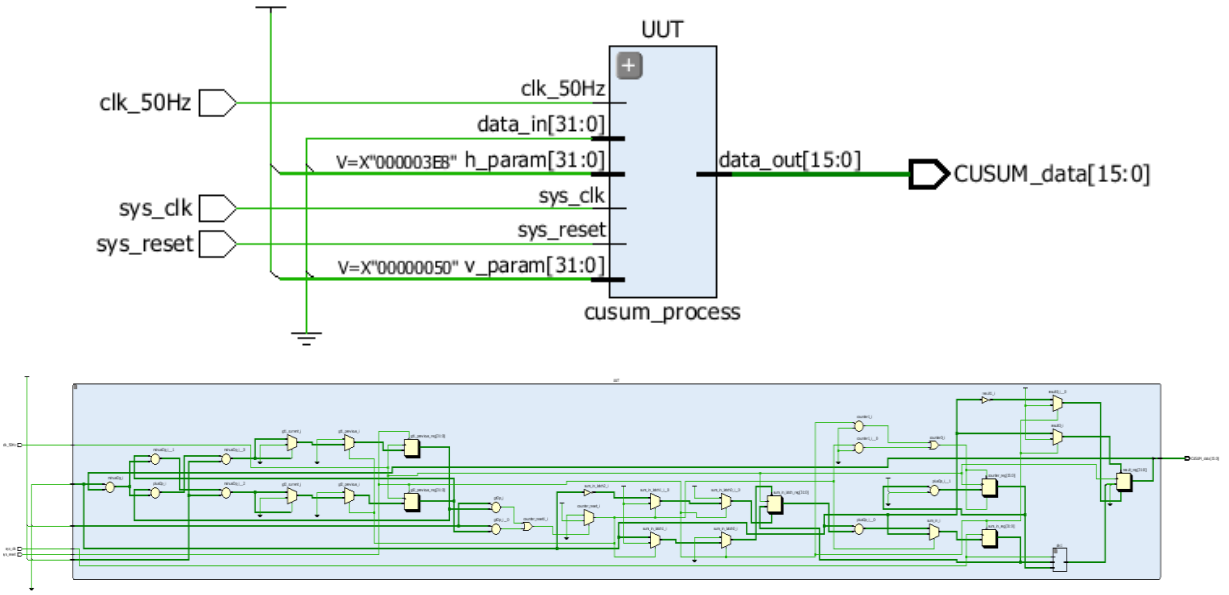



Figure 4.12 Schematic of CUSUM VHDL module (above) and its detail registers level after synthesizing (below) in Vivado

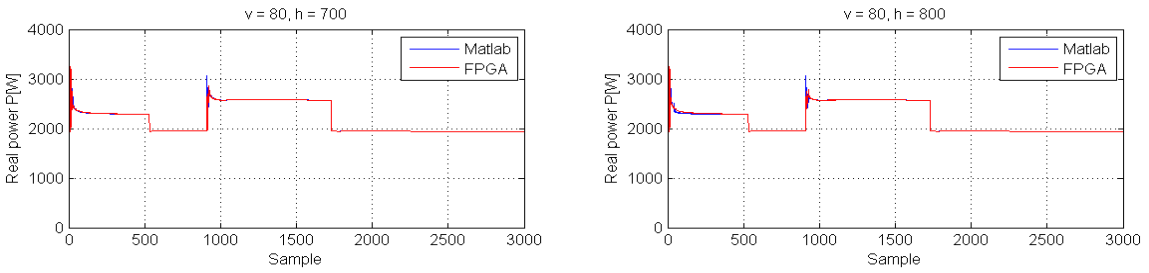



Figure 4.13 Comparison result of CUSUM filter of MATLAB and ModelSim

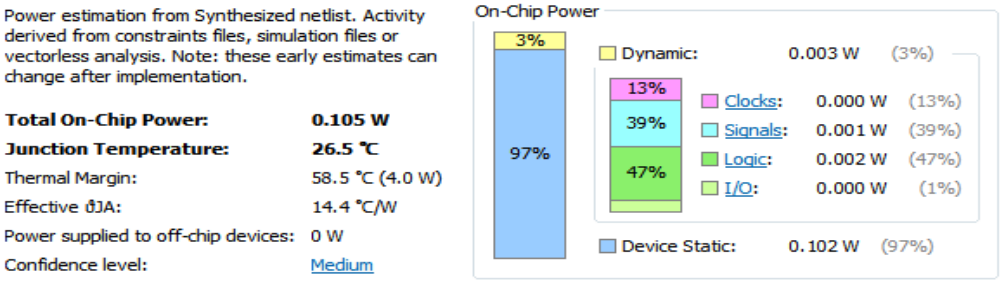



Figure 4.14 Static power estimation of CUSUM in Vivado Synthesis for target Zynq XC7z010

## System Generator

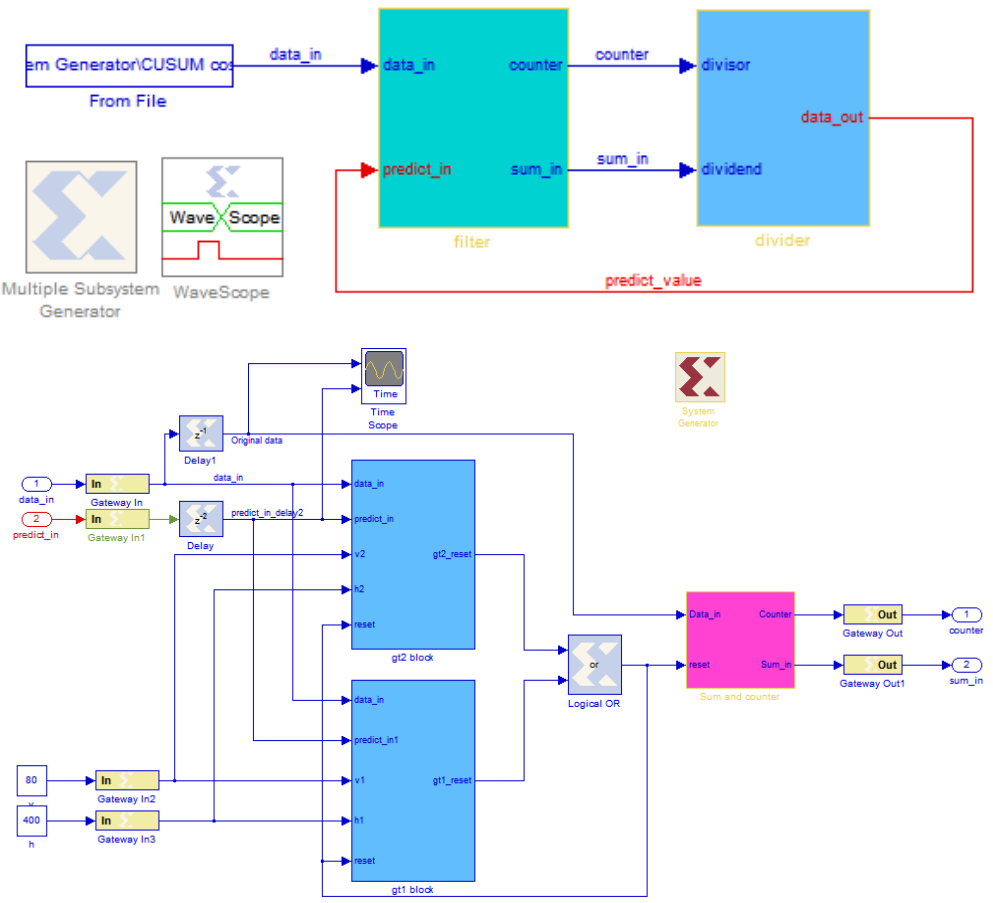



Figure 4.15 CUSUM algorithm implemented by Simulink and Xilinx System Generator

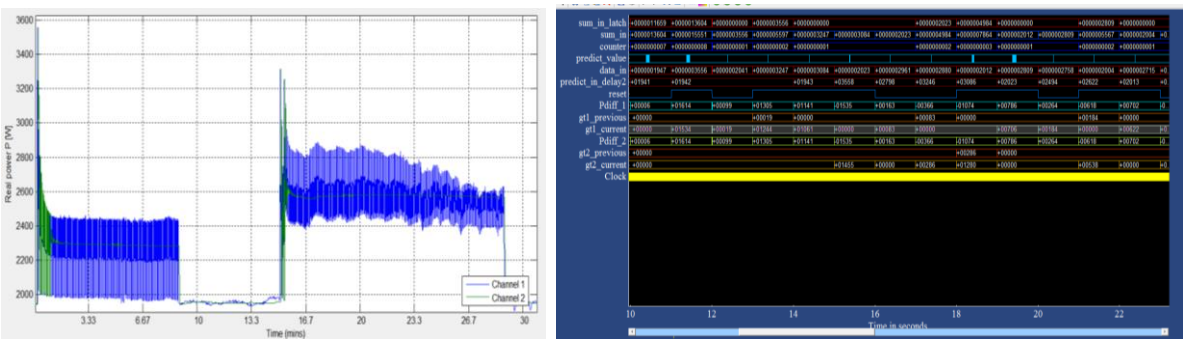


Figure 4.16 Comparison functional result of CUSUM in a signal design in System Generator (left) and result in bit accuracy and cycle accuracy verification in WaveScope (right)

### LabVIEW FPGA

Implement results of Event Detection in Figure 4.17 shows that model-based FPGA IP design sometimes consumes more hardware resources. The VHDL manual coding of CUSUM event detection function consumed only 125 slices of Flip-Flop (FF) and 586 slices of LUT in FPGA. However, it consumed 3016 slices of Flip-Flop (FF) and 3924 slices of LUT by IP generated by System Generator. Implement results in LabVIEW FPGA consumes 938 slices of Flip-Flop (FF) and 826 slices of LUT. So that, LabVIEW FPGA uses hardware better than System Generator but worse than VHDL manual coding approach.

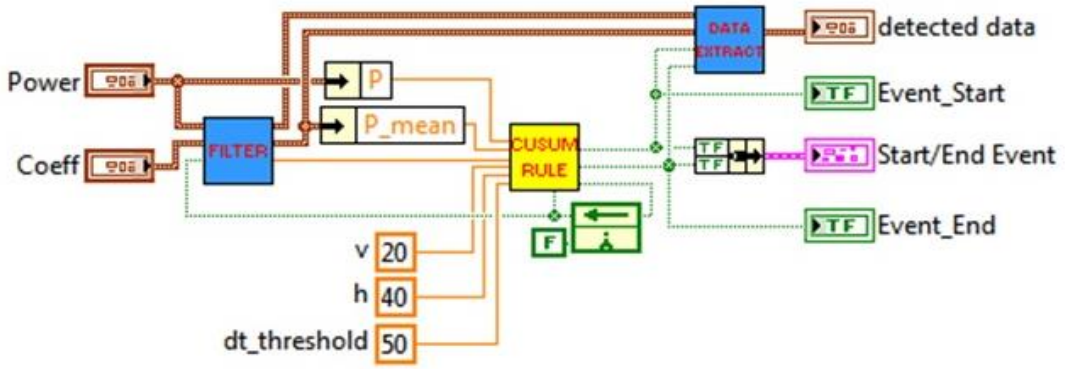



Figure 4.17 Implementation of CUSUM Event Detection in SDF model based design

As shown in Table 4.3 in comparison of the throughput of function running in FPGA architecture, function implemented in LabVIEW FPGA has the best result when it can run in 40 MHz clock or the maximum latency of the computation is only 25 ns. This result is 71 ns in System Generator approach and 76 ns in VHDL manual coding approach. However, model-based FPGA development approach is much faster than VHDL manual coding approach because model validated in high-level can be used to implement hardware function in FPGA without too much changes in model from the system design. Unfortunately, the drawback of LabVIEW

FPGA approach is that it is impossible to estimate the power consumption at system design stage and we have to measure it with method discussed at the end of chapter 2.

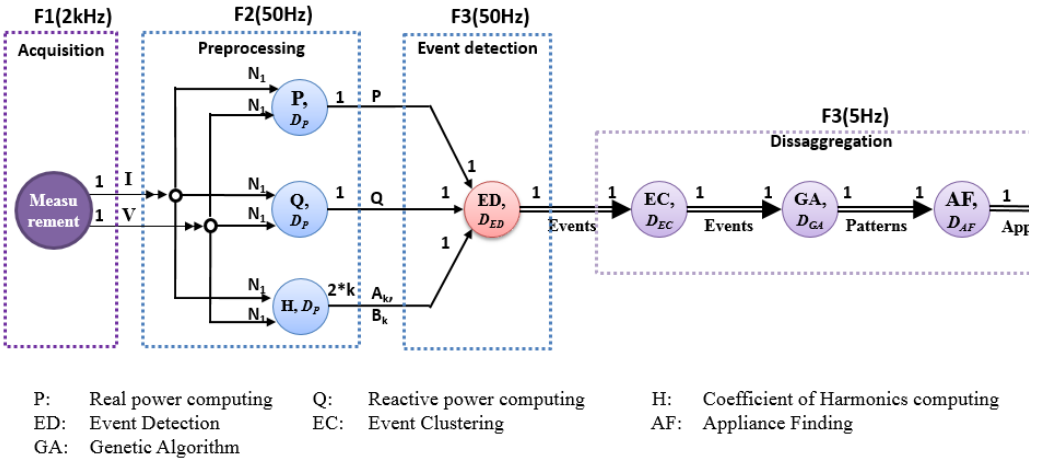



Figure 4.18 SDF model of the Total power Disaggregation function where $D_P$, $D_{ED}$, $D_{EC}$, $D_{GA}$, $D_{AF}$ are times to finish processes Preprocessing, Event Detection, Event Clustering, Genetic Algorithm, and Appliance Finding

## 4.4. Architecture exploration

Architecture exploration is important step to analyze the satisfaction of system in various architectures to select the best architecture for prototyping. Figure 4.18 shows a refinement SDF model for the *disaggregating total apparent power* activity. The acquisition periodically measures current I and voltage V at defined sampling rate. The second periodic part is the preprocessing because it consumes a quantities of samples $N_1$ measured from the acquisition to do the computation. Although preprocessing can compute in the same sampling rat of the acquisition, it should be separated in different sampling rate domains (50Hz) to avoid the difficulty in optimizing the performance of the processing algorithm. If acquisition and preprocessing are in the same sampling rate, there is no requirement about the FIFO memory for communication between two parts but it is mandatory the preprocessing must finish its computation before the next sample is measured. This solution suits to the slow sampling rate system. The next part-Event Detection needs to check every preprocessed data from Preprocessing so that it works in the same sampling rate of Preprocessing. Finally, the disaggregation algorithm needs to process every detected event so far. As we stated in the constraint, the minimum duration between two continuous events is 200 milliseconds, which means the system cannot detect two events occurring in less than 200 milliseconds. However, there is possible no event during hours which mean Disaggregation may be time-driven (every 200 milliseconds) or an event-driven process.

**Memory allocation**

The visual information in SDF model gives developer more information about the performance of each process of the system, and the number of tokens consumed and produced in each process. That speeds up the architecture exploration, effectively static scheduling and memory allocation. In Figure 4.18, the notations 1, 2*k, or $N_1$ are the number of tokens, which processes fire or consume in each operating cycle of system. Therefor SDF model can clearly express the capability of the processes to work in pipeline to improve the throughput of the system. While computing the average values of variables such as real power P, reactive power Q, and coefficient of current requires $N_1$ samples (number of samples in one cycle), other processes need only one token. Therefore, developers can define the FIFO memory sizes necessary for effective data transactions between processes as below Table 4.1.

| **Communication Chanel** | **Memory requirement (Bytes)** |
|---|---|
| Measurement - Preprocessing | ($N_1$ +b)* { Sizeof (voltage) + Sizeof (current)} |
| Preprocessing – Event Detection | (1+b)*{2* Sizeof(Datatype of P/Q) + 2*k* Sizeof(Datatype of $A_k$/$B_k$)} |
| Event Detection - Disaggregation | (1+b)*{Sizeof(Datatype of dP) + Sizeof(Datatype of dQ) + Sizeof(Datatype of dTHDi) } |

Table 4.1 Allocating buffer size for FIFOs in communication channels

Where "b" is the preventive buffer of FIFO for the FIFO-writing can continue when the system waits to the FIFO-reading; however a FIFO-reading must always finish before a FIFO-writing. For example, from SDF model in Figure 4.18, the pseudo codes for a possible case of architecture in one CPU system can be defined by three tasks.

**Task 1**: (Time-driven)
Each 0.5 milliseconds do
Code block measurement;
end

**Task2**: (Time-driven)
Each 20 milliseconds do
for i = 1 to $N_2$ do
Code block preprocessing;
end
Code block event detection;
end

**Task3**: (Time-driven)
Wait 200 milliseconds do
Code block disaggregation;
end

In this architecture, we can statically allocate size of buffer of communication channels as Table 4.2 with N1 = 40 and b = 1.

| **Communication Chanel** | **Memory requirement (byte)** |
|---|---|
| C1. Measurement - Preprocessing | (N1 + b) * (2+2) = 4*(N1+b) bytes = 164 bytes |
| C2. Preprocessing – Event Detection | (1+b)*(2 + 2+ 2*4*2) = (1+b)* 20 bytes = 40 bytes |
| C3. Event Detection - Disaggregation | (1+b)*{2 + 2+ 2} = 6*(1+b) = 12 bytes |

Table 4.2 Memory analysis results for the example architecture

**Throughput analysis**

Moreover, notations $D_P$, $D_{ED}$, $D_{EC}$, $D_{GA}$, $D_{AF}$ present the worst processing tasks latency that can be set bounded values to satisfy the real-time constraints of the system. Because of the periodic, the $D_p$ must be less than $N_1$*Ts where Ts is the sampling period of acquisition process. $D_{ED}$ must be less than 20 millisecond-sampling period of Preprocessing. Finally, total of $D_{EC}$, $D_{GA}$, $D_{AF}$ must be less than 200 milliseconds to avoid the overflow memory trouble when there are too many events detected in every 200 milliseconds. Such algorithms must be able to minimize the processing delay in order for the system to be able to reach the timing, performance, and resource constraints.

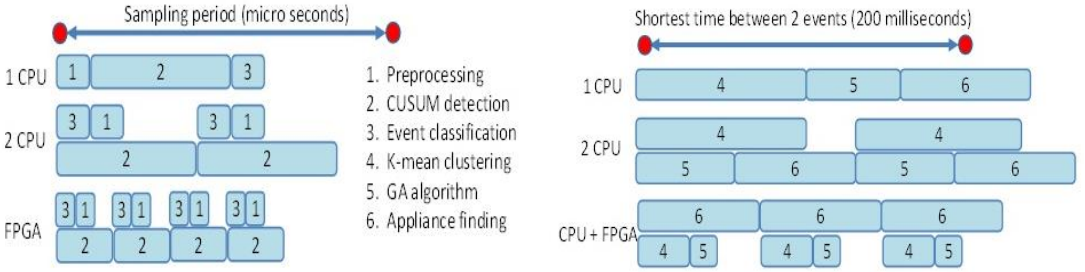



Figure 4.19 Throughput analysis of system implemented in various architectures

In order to understand the capacity of system in different architectures designed for Zynq SoC, we investigated each process in each architecture to evaluate their performance in timing, hardware resource usage and power consumption. As illustrated in Figure 4.18, SDF model defines clearly the order and the rate of each function to be invoked in a static schedule operation. The architecture of Zynq can support various architectures to implement a system using only one processor, using two processors or using two processors with FPGA acceleration. Figure 4.19 illustrates the possible overall timing latency of total power disaggregation function in various architectures of Zynq. As defined in SDF model, processes Preprocessing, Event Detection and Event Classification run in the same rate in every time N1 samples are collected by Measurement function. At that time, the communication buffer in Measurement-Processing has enough $N_1$ data for the function Preprocessing able to consume. Event Detection and Event classification then run in sequence to sense the event, compute information of events. The operation of these sequence processes in different architectures can be presented in Figure 4.19 where latency of the functions running on FPGA is always faster than running on CPU. Therefore, if both processes 1, 2, 3 run on FPGA, the sampling frequency can increase 4 times in comparison to run both 3 processes on one CPU.

Preprocessing in CPU can process 40 samples in 188 microseconds that means it can support about 212765 samples per second. However, as a limitation of Zynq, the maximum data-sampling rate of Acquisition supported in this platform is only 255 kHz, which can acquire more than 5000 samples per a period 20 milliseconds of electrical signal. Moreover, the FIFO controller of Zynq

supports the FIFO's deep maximum 1024 samples per 20ms (or 51200 samples per second) then maximum sampling rate is limited at about 50 kHz.

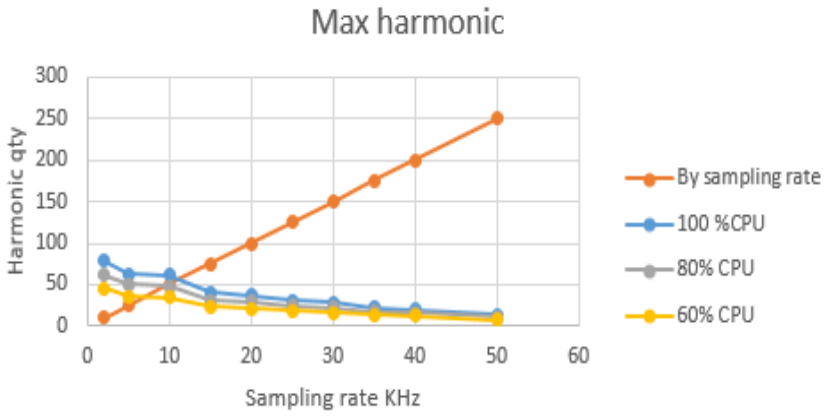



Figure 4.20 Max number of harmonics and optimized sampling rate for a NIALM system in a single processor

In LabVIEW, we can define the processor core (of Zynq) to run the function. When all functions of NIALM run on one processor or multicore, the computation cycle of processes preprocessing, event detection, and disaggregation repeats every 20 milliseconds (or 20.000 microseconds). Thus, the CPU usage is defined by the ratio of active operation time of CPU in 20 milliseconds or the ratio of total latency through NIALM functions per 20 milliseconds. This value presents the limit number of harmonic orders a single CPU can extract and defines the optimized sampling rate for the system. Figure 4.20 shows that the best-selected sample rate should not be larger than 10 KHz because CPU cannot extract the maximum harmonic. It also shows that the maximum number of harmonics system can extract is 50 harmonics at 10 KHz sampling rate. In order to increase these values, we have to implement the preprocessing in FPGA hardware. As shown in Table 4.3, FPGA can accelerate the preprocessing function to process each token in only 250 nanoseconds and the system can run in maximum supported sampling rate of Zynq 50 kHz. The maximum harmonics can be extracted up to 25 kHz.

## 4.5. Prototyping system

There are two possible system architectures relating to the characteristic of function disaggregation as illustrated in Figure 4.21. In the first architecture, processes acquisition, preprocessing and event detection run on smart meters and the disaggregation runs on a server. Events detected in NIALM meter are sent to server by internet communication and server will run the disaggregation algorithm to classify appliances. Such system is complex, expensive in comparison of the system with all NIALM function implemented on a SoC. As shown in Figure 4.22, the first architecture was prototyped in sbRIO-9363 platform and the disaggregation was deployed in C++ application in a standard PC Intel core i5-2410M 4CPUs 2.3GHz. Disaggregation responds to the appliance detection during four milliseconds. This system was

tested to be able to disaggregate three simple appliances: a blender, a kettle and a halogen lamp. After setting the name of these appliances in database, system now can classify the name of appliances.

| **Acquisition** | **Throughput (worst case)** | **Resources usage** | **Power consumption** |
|---|---|---|---|
| FPGA (LabVIEW based) | 3.925 µs (≈255 kHz) | 1106 FFs (3.1%)<br>672 LUTs (3.8%) | |

| **Event Detection** | **Throughput (worst case)** | **Resources usage** | **Power consumption** |
|---|---|---|---|
| 1 ARM Cortex A9 667MHz (LabVIEW RT) | 91 µs (per 20ms average sample) | 16.54 kBs RAM | |
| FPGA (LabVIEW based) | 40 MHz (25 ns) | 938 FFs (2.6%)<br>826 LUTs (4.7%) | |
| FPGA (System Generator based) | 14 MHz (71 ns) | 3016 FFs (8.6%)<br>3924 LUTs (22.3%) | 0.225 W (estimation) |
| FPGA (Manual coding) | 12.9 MHz (76 ns) | 125 FFs (0.4%)<br>568 LUTs (3.2%) | 0.105 W (estimation) |

| **Preprocesing (P,Q + Harmonic Computing)** | **Throughput (worst case)** | **Resources usage** | **Power consumption** |
|---|---|---|---|
| 1 ARM Cortex A9 667MHz (LabVIEW RT) | 188µs + k*333 µs (*) (40 tokens) | 16.8 kBs RAM | |
| FPGA (LabVIEW based) | 250 ns (per token) | 808 FFs (2.3%)<br>735 LUTs (4.2%) | |

| **Disaggregation (Clustering,** | **Throughput (worst case)** | **Resource usage** | **Power consumption** |
|---|---|---|---|
| Computer Intel core i5-2410M 4CPUs 2.3GHz | 4 ms | 847 kBs RAM | |
| 1 ARM Cortex A9 667MHz (LabVIEW RT) | 953 µs | 8.03 kBs RAM | |

(*) k is the number of harmonic.

Table 4.3 Architecture exploration results

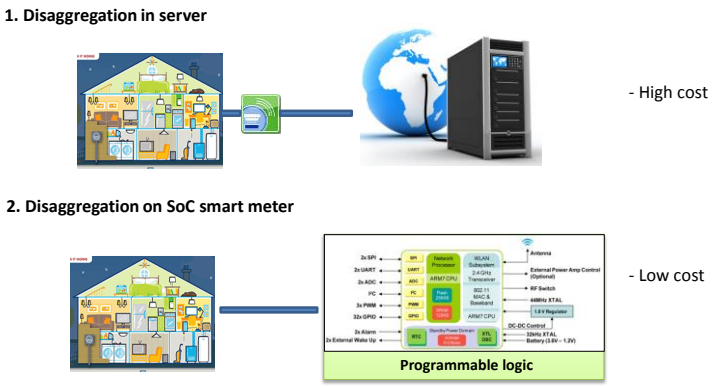



Figure 4.21 Two general models for commercial NIALM system

Implementing all NIALM functions into a single SoC needs to analyze the system in various architectures of the SoC such as: an architecture with all functions implemented in embedded software running in one CPU or multi-cores CPU or a complex architecture with both hardware and software implementations. Results in Table 4.3 show in detail the performance of NIALM function when we analyzed them in various architectures for a simple NIALM as we implemented in the first architecture in Figure 4.22. In the second architecture, we used the capability C/C++ programming in Zynq to bring the C++ window application to run in Zynq architecture as shown in Figure 4.23.

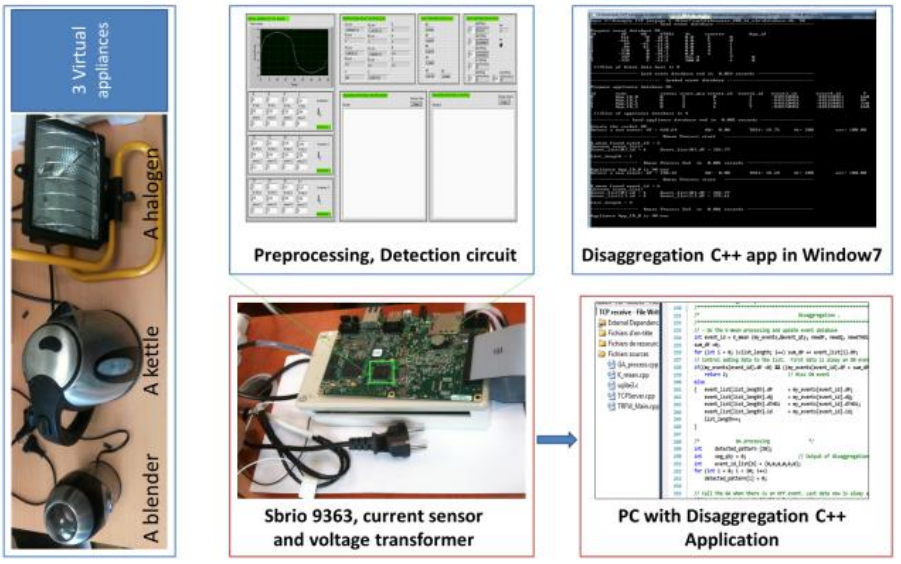



Figure 4.22 Prototyping system in architecture with disaggregation in server

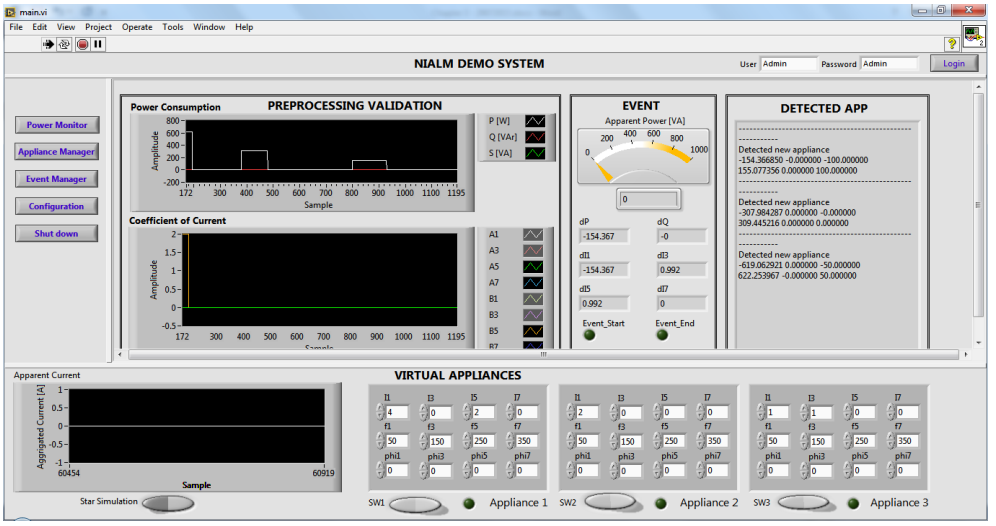



Figure 4.23 Prototyping NIALM in second architecture. Data Acquisition and Preprocessing, Event Detection run in FPGA and Disaggregation runs in processor

## 4.6. Conclusion

This chapter illustrated a use case of development of a RPN system using the hardware software co-development proposed in chapter 2. In order to understand this approach, the myRIO platform with SoC Zynq was selected because this SoC has both processors and FPGA and support exploring NIALM system in various architectures. This chapter showed that our proposed methodology is very interesting and effectively in quickly develop and prototype a complex RPN system like NIALM. First prototype of NIALM can function in real-time the disaggregation algorithms and the architecture exploration results showed that the Zynq can support more a system with more difficult timing constraints. However, this method must use the NI platform to process the architecture exploration and power consumption measurement.

# CONCLUSIONS AND PERSPECTIVES

## Conclusion

In the motivation to engage people saving energy, the objective of the thesis is studying about the NIALM technology and proposing an innovative NIALM system, which is more powerful, compact and low cost and can overcome some challenges of current NIALM meters. The research got some important results:

We developed a real-time event detection CUSUM which is able to detect and extract more electrical in time-real event with low transient appliances based on configuring parameter drift of algorithms. We also added adaptive threshold feature to CUSUM algorithm which can improve the noise and small signal distinction. Functional verification of CUSUM in dataset REDD showed its high accuracy but also its ability to smooth the noise waveform of the signal. Then, we implemented event detection into FPGA in many approach. Experimental results in the first prototype show that it can detect events of signal sampled at up to 40 MHz. This function is has the potential to be integrated to a low cost but powerful NIALM system.

We proposed estimating coefficient harmonics information Ak, Bk because unlike current information Ak, Bk are linear so that they are compatible to CUSUM event detection to extract Ak, Bk of events. Then, Ak and Bk are used to compute the harmonics Ik, Total Harmonic Distortion (THD) and Individual Harmonic Distortion (IHD). The capability of extracting harmonics information of individual appliances has not yet been integrated into existing commercial product. Commercial power meter product can measure only the THD of all the appliance in the electrical network. Then, we continued by implementing this algorithms in many architectures. Maximum harmonics order extracted by one core ARM should be less than $50^{th}$ and optimum sampling rate in this case is about 10kHz because of the limitation of processor performance. Version implemented in FPGA can extract coefficient harmonics of signal sampled up to 40 MHz but that will consume more FPGA resources. First implementation extracts 4 harmonics took 808 (2.3%) Flip-Flop and 735 (4.2%) LUT in the myRIO 1900 architecture.

The disaggregation of our NIALM system was based on Genetic Algorithm to be able to detect multi-state appliances. In this research, the GA version used only three information of detected events: real power, reactive power and THD but we have not yet finished analyzing to find out the best fitness function for this GA algorithm. First results in REDD shows that this unsupervised algorithm can detect multi-state appliance and disaggregate more than 80 % power consumption in the electrical network. The prototype of GA function running on a server PC just takes around 4ms to disaggregate and display information.

In studying about modeling our system, we found the relation between NIALM and the Reactive Process Network model, which is often used to model a complex DSP system containing both the event controlling processes and data streaming processes. RPN model is a

composition of dataflow model and Finite State Model. Moreover, we found that the Synchronous Dataflow model is the best dataflow model, which allows better buffer size allocation and statistic scheduling for the data streaming processes. Then, we proposed the use RPN to model our NIALM system aiming to optimize the performance, memory usage and power consumption of the NIALM system. Various approaches, tools and languages are studied and LabVIEW, LabVIEW FPGA was found to be the best candidate not just in modeling the RPN system but also in prototyping the product quickly. Therefore, in the second contribution of the thesis, we proposed a methodology for quickly developing a RPN system on SoC based on LabVIEW and LabVIEW FPGA. The advantage of this approach is that it can be used for the regular embedded system development as well as the SoC embedded system with the hardware acceleration. This is the first approach to implement RPN complex into SoC based on Labview FPGA and it is a very useful for some specific purposes:

- With system designers, they can quickly develop complex DSP system without the deep knowledge in HDL programming for hardware development and C/C++ programing for software development. The whole system after analyzing algorithms using high-level mathematics languages such as Matlab/Scilab/ Octave can be modeled into the system in either model-based programming or MathScript based programming. The real-time codes or HDL codes generated by LabVIEW are runnable into the selected NI platform.
- With researchers, understanding well how to model RPN model in this approach can give them a very powerful tool to analyze their DSP algorithms in real system quickly.
- With hardware designers, they can apply this approach to optimize the performance and resources consumption in their system applying SDF to model their system and explore them in various architectures.
- With software designers, this approach also gives them the capability to prototyping a system quickly by embedding their C code algorithm with formula node tool or programing system in C coding approach.

Although the designed NIALM worked well in disaggregate power in dataset REDD, verifying the system in real database measured in laboratory shows some limitations:

- The CUSUM works well in detecting ON-OFF appliances but it does not work well with power consumed by some lighting, informatics and electronics appliances. That is because their power usages is small and its transitions are too slow. This event-based method should be used in combining with the steady-based method. This measure the power consumption in network at each period time so that it can detect a large power change between two measurements.
- Disaggregation based Genetic Algorithm matches ON and OFF events if the fitness function of this group is satisfied. However, in industrial domain, there are so many appliances, which have the same ON or OFF events can cause mistakes. Other common limitation is, this algorithm cannot detect and classify two simultaneous events.

Regarding the development of the system, our SoC development methodology has to connect to a real platform to estimate the power consumption and process's latency. Therefore, it is

impossible to explore system without real-platform as virtual platform modeling approach. The solution for this limit is studying about developing an estimation model in Labview, which allows estimating the power consumption and the latency during the system design phase.

## Perspectives

Harmonics may give more information about some appliances like electronics, informatics appliance, which have some particular information in high harmonics. Results in this thesis show that the NIALM system works well at the maximum 50 kHz sampling rate but we believe that interleaving sampling methods can increase the sampling rate. Improving harmonic extraction with FPGA acceleration will keep a low cost system and increase the power NIALM system. Therefore, there is a need to improve the harmonics analysis which can be used to compute real power and reactive power in higher harmonic order.

GA disaggregation algorithm needs to be improved to be able to classify similar and simultaneous events. A probability classification approach such as Hidden Markov Model could be used to improve the appliances classification. However, the algorithm will be more complex and may not satisfy the timing constraint of a real-time NIALM meter. So, the disaggregation algorithm implementation should be considered using hardware.

Another perspective of the thesis is to transform the design in one phase electrical network to a three phases version and integrating the SoC NIALM system into a single current sensor to create an innovative NIALM sensor node that will be able to the monitor power in large buildings using Wireless Sensor Network.

Besides monitoring power usage application of NIALM, there is a potential application in monitoring the electrical power quality. Bad quality electrical power can cause malfunctions and fails in electrical machines which means maintenance. Two most important standards in rating the quality of electrical power are EN50160 standard and IEC 61000 standard based on the limits of voltage, current and frequency in order to prevent maintenance. NIALM technology may be the best solution to find out which appliances cause problems or pollution in electrical power quality, using customer or distributor views. We also will continue apply the HW SW codevelopment methodology to improve the performance of our system in newest sbRIO-9607 platform FPGA with integrating quality power monitoring application. This rich FPGA resources card supported by 667MHz dual-core ARM Cortex A9 can work as an independent NIALM system while last product still need a server PC to execute the disaggregation algorithm. Moreover, this system is needed to executing in real-time the high order Ak, Bk harmonic extraction and HMM algorithm.

# PUBLICATIONS

**2014** Kien Nguyen Trung, Cyril Jacquemod, Eric Dekneuvel, Benjamin Nicolle, Olivier Zammit, Cuong Nguyen Van, Philippe Lorenzini, Gilles Jacquemod,"Innovative Current Sensor and Event Detection Algorithms for NIALM Application", SAME 2014, Nice, France. Best demo and poster award.

Kien Nguyen Trung, Eric Dekneuvel, Benjamin Nicolle, Olivier Zammit, Cuong Nguyen Van, Gilles Jacquemod," Event Detection and Disaggregation Algorithms for NIALM System", Second International Workshop in NILM 2014, Texas, USA

**2013** Cyril Jacquemod; Kien Nguyen Trung ; Asma Chargui; Khalifa Aguir; Olivier Zammit, Eric Dekneuvel, Benjamin Nicolle, Philippe Lorenzini, Gilles Jacquemod, "Power Consumption Monitoring for Smart Building Application", SAME 2013, Nice, France

T. K. Nguyen, E. Dekneuvel, B. Nicolle, O. Zammit, V. C. Nguyen and G. Jacquemod, ""Using FPGA for real time power monitoring in a NIALM system."," in In Industrial Electronics (ISIE), 2013 IEEE International Symposium on, 2013

**2012** Kien Nguyen Trung, Ngoc Nguyen Van, "Pulse Oximetry System based on FPGA", VN-UK International Conference on Bio-Sciences and Bio-Electronics (ICBSBE) , Vietnam, 2012

T. K. Nguyen, O. Zammit, E. Dekneuvel, B. Nicolle, V. C. Nguyen and G. Jacquemod, "An innovative non-intrusive load monitoring system for commercial and industrial application," in In Advanced Technologies for Communications (ATC), 2012 International Conference on, 2012.

# REFERENCES


[1] "The International Energy Outlook 2013," Office of Energy Analysis. U.S. Department of Energy, 2013.

[2] EPA, "United States Environmental Protection Agency," April 2014. [Online]. Available: http://www2.epa.gov/energy/greenhouse-gas-equivalencies-calculator.

[3] A. Zervos, C. Lins, L. Tesnière and E. Smith, "Mapping renewable energy pathways towards 2020," European Renewable Energy Council, 2011.

[4] U. Department of Energy and Climate Change, "Behavior change and energy use," 2011.

[5] G. W. Hart, "Nonintrusive appliance load monitoring," *Proceedings of the IEEE,* pp. 1870-1891, 1992.

[6] S. B. Leeb, S. R. Shaw and J. L. & Kirtley Jr, "Transient event detection in spectral envelope estimates for nonintrusive load monitoring," *IEEE Transactions on Power Delivery,* pp. 10(3), 1200-1210, 1995.

[7] C. Laughman, K. Lee, R. Cox, S. Shaw, S. Leeb, L. Norford and P. Armstrong, "Power signature analysis.," *Power and Energy Magazine, IEEE,* pp. 1(2), 56-63, 2003.

[8] M. Baranski and J. Voss, "Genetic algorithm for pattern detection in NIALM systems," in *In Systems, Man and Cybernetics, 2004 IEEE International Conference on, Vol. 4*, 2004.

[9] R. Cox, S. B. Leeb, S. R. Shaw and L. K. Norford, "Transient event detection for nonintrusive load monitoring and demand side management using voltage distortion.," in *In Applied Power Electronics Conference and Exposition, 2006. APEC'06*, 2006.

[10] K. Suzuki, S. Inagaki, T. Suzuki, H. Nakamura and K. & Ito, "Nonintrusive appliance load monitoring based on integer programming," in *In SICE Annual Conference*, 2008.

[11] S. N. Patel, T. Robertson, J. A. Kientz, M. S. Reynolds and G. D. Abowd, "At the flick of a switch: Detecting and classifying unique electrical events on the residential power line," *Springer Berlin Heidelberg,* pp. 271-288, 2007.

[12] J. Froehlich, E. Larson, S. Gupta, G. Cohn, M. Reynolds and S. Patel, "Disaggregated End-Use Energy Sensing for the Smart Grid," *Pervasive Computing, IEEE,* pp. vol.10, no.1, pp.28-39, 2010.

[13] O. Parson, S. Ghosh, M. Weal and A. & Rogers, "Non-intrusive load monitoring using prior models of general appliance types," 2012. [Online]. Available: http://www.orchid.ac.uk/.

[14] J. Z. Kolter and T. Jaakkola, "Approximate inference in additive factorial hmms with application to energy disaggregation," *In International conference on artificial intelligence and statistics,* pp. 1472-1482, 2012.

[15] M. E. Berges, E. Goldman, H. S. Matthews and L. Soibelman, "Enhancing electricity audits in residential buildings with nonintrusive load monitoring," *Journal of industrial ecology,* pp. 14(5), 844-858, 2010.

[16] TED, "theenergydetective," The Energy Detective, 2015. [Online]. Available: http://www.theenergydetective.com/.

[17] Enetics, "Power Quality Monitoring," Enetics, Inc, 2012. [Online]. Available: http://www.enetics.com/app-PQM.html. [Accessed 2015].

[18] Navetas, "NAVETAS," Maple Tree Energy Management Limited, Company, 2015. [Online]. Available: http://www.navetas.com/products/#gas. [Accessed 2015].

[19] W. Greene, J. S. Ramsey, S. B. Leeb, T. DeNucci, J. Paris, M. Obar and T. J. McCoy, "Non-intrusive monitoring for condition-based maintenance," *In American Society of Naval Engineers Reconfigurability and Survivability Symposium (Vol. 11),* 2005.

[20] W. K. Lee, G. S. K. Fung, H. Y. Lam, F. H. Y. Chan and M. & Lucente, "Exploration on load signatures," in *In International conference on electrical Engineering (ICEE)*, 2004.

[21] K. H. Ting, M. Lucente, G. S. Fung, W. K. Lee and S. Y. R. & Hui, "A taxonomy of load signatures for single-phase electric appliances.," in *In IEEE PESC (Power Electronics Specialist Conference)*, 2005.

[22] S. Gupta, M. Reynolds and S. Patel, "ElectriSense: single-point sensing using EMI for electrical event detection and classification in the home," in *12th ACM international conference on Ubiquitous computing*, 2010.

[23] Z. Wang and G. & Zheng, "Residential appliances identification and monitoring by a nonintrusive method," *Smart Grid, IEEE Transactions on, 3(1),* pp. 80-92, 2012.

[24] LG, "Smart ThinQ™ Super-Capacity 3 Door French Door Refrigerator with 8" Wi-Fi LCD Screen," LG,inc, [Online]. Available: http://www.lg.com/us/refrigerators/lg-LFX31995ST-french-3-door-refrigerator. [Accessed 2015].

[25] Xilinx, "Zynq-7000 Silicon Devices," Xilinx Inc, [Online]. Available: http://www.xilinx.com/products/silicon-devices/soc/zynq-7000/silicon-devices.html. [Accessed October 2015].

[26] NI, "LabVIEW FPGA Module," National Instruments Corporation, [Online]. Available: http://www.ni.com/labview/fpga/ . [Accessed October 2015].

[27] Wikimedia, "Integrated circuit," Wikimedia Foundation, Inc, [Online]. Available: https://en.wikipedia.org/wiki/Integrated_circuit. [Accessed October 2015].

[28] OpenCores, "Opencore Projects," OpenCores.org, [Online]. Available: http://opencores.org/projects. [Accessed October 2015].

[29] Tom Hill, "Advancing the Use of FPGA Co-Processors through Platforms and High-Level Design Flows," Xilinx , 2011.

[30] R. Zurawski, Embedded systems handbook, CRC Press, 2005.

[31] D. D. Gajski, S. Abdi and A. Gerstlauer, Embedded System Design Modeling, Synthesis and Verification, Springer, 2010.

[32] J. P. Calvez, A. Wyche and C. Edmundson, Embedded real-time systems, J. Wiley., 1993.

[33] Cadence, "Cadence Virtual System Platform datasheet," Cadence Design Systems, Inc, [Online]. Available: http://www.cadence.com/rl/Resources/datasheets/virtual_system_platform_ds.pdf. [Accessed October 2015].

[34] "Open Virtual Platforms™ (OVP™) portal," Imperas Software Limited, [Online]. Available: http://www.ovpworld.org/. [Accessed Oct 2015].

[35] S. Rigo, G. Araujo, M. Bartholomeu and R. & Azevedo, "ArchC: A SystemC-based architecture description language," in *Computer Architecture and High Performance Computing, 2004. SBAC-PAD 2004*, 2004.

[36] D. D. Gajski, S. Abdi, A. Gerstlauer and G. Schirner, "Embedded system design: modeling, synthesis and verification," *Springer Science & Business Media,* 2009.

[37] A. Raghunathan, N. K.Jha and S. Dey, High-Level Power Analysis And Optimization, Kluwer Academic Publishers, 1998.

[38] M. Geilen and T. Basten, "Requirements on the execution of Kahn process networks. In Programming languages and systems.," *Springer Berlin Heidelberg,* pp. 319-334, 2003.

[39] T. Basten, "Computational Models for Concurrent Streaming Applications," 2010. [Online]. Available: http://www.asci.tudelft.nl/media/winterschool2010/asci2010_4.pdf. [Accessed 2015].

[40] A. Jantsch, Models of computation for distributed embedded systems, 2009.

[41] I. Radojevic and Z. Salcic, Embedded Systems Design Based on Formal Models of Computation, Springer, 2011.

[42] E. A. Lee and S. A. Seshia, Introduction to embedded systems: A cyber-physical systems approach, Lee & Seshia, 2011.

[43] D. Harel, " Statecharts: A visual formalism for complex systems," *Science of computer programming,* pp. 8(3), 231-274, 1987.

[44] D. Harel and M. Politi, Modeling reactive systems with statecharts: the STATEMATE approach., McGraw-Hill, Inc, 1998.

[45] D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman and M. Trakhtenbrot, "Statemate: A working environment for the development of complex reactive systems," *Software Engineering, IEEE Transactions on,* pp. 16(4), 403-414, 1990.

[46] "Developing Applications with the NI LabVIEW Statechart Module," 2012. [Online]. Available: http://www.ni.com/tutorial/6194/en/. [Accessed 2015].

[47] "NI LabVIEW Statechart Module," [Online]. Available: www.ni.com/labview/statechart. [Accessed 2015].

[48] D. Harel and H. Kugler, "The rhapsody semantics of statecharts (or, on the executable core of the UML).," in *In Integration of Software Specification Techniques for Applications in Engineering*, 2004.

[49] W. P. Stevens, G. J. Myers and L. L. Constantine, "Structured design," *IBM Systems Journal,* pp. 13(2),115-139, 1974.

[50] K. A. H. N. Gilles, "The semantics of a simple language for parallel programming," in *In Information Processing'74: Proceedings of the IFIP Congress Vol. 74*, 1974.

[51] E. A. Lee and D. G. Messerschmitt., "Synchronous data flow," *Proceedings of the IEEE 75.9,* pp. 1235-1245, 1987.

[52] Z. Zhou, W. Plishker, S. S. Bhattacharyya, K. Desnos, M. Pelcat and J. F. Nezan, "Scheduling of Parallelized Synchronous Dataflow Actors for Multicore Signal Processing," *Journal of Signal Processing Systems,* pp. 1-20, 2014.

[53] M. Lattuada and F. & Ferrandi, "Modeling pipelined application with synchronous data flow graphs. In Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIII), 2013 International Conference on (pp. ). IEEE..," in *In Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIII), 2013 International Conference on*, 2013.

[54] A. Bonfietti, L. Benini, M. Lombardi and M. Milano, "An efficient and complete approach for throughput-maximal SDF allocation and scheduling on multi-core platforms," in *In Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2010.

[55] S. S. Bhattacharyya, P. K. Murthy and E. A. Lee, "Synthesis of embedded software from synchronous dataflow specifications," *Journal of VLSI signal processing systems for signal, image and video technology,* pp. 21(2), 151-166, 1999.

[56] Kodosky, Jeffrey, J. MacCrisken and G. Rymar., "Visual programming using structured data flow," in *Proceedings of Workshop on Visual Languages*, 1991.

[57] M. Geilen and T. Basten, "Reactive process networks," in *InProceedings of the 4th ACM international conference on embedded software*, 2004.

[58] R. D. Turney, C. Dick, D. B. Parlour and J. & Hwang, "Modeling and implementation of dsp fpga solutions," in *In 1999 International Conference on Signal Processing Applications and Technology ICSPAT*, 1999.

[59] D. Markovic, C. Chang, B. Richards, H. So, B. Nikolic and R. W. Brodersen, "ASIC Design and Verification in an FPGA Environment," in *InCustom Integrated Circuits Conference, 2007. CICC'07. IEEE*, 2007.

[60] "Xilinx System Generator for DSP User Guide," 2009. [Online]. Available: http://www.xilinx.com/support/documentation/sw_manuals/xilinx11/sysgen_user.pdf. [Accessed 2015].

[61] R. Bucher and S. Balemi, "Some practical experiences with Scilab/Scicos and RTAI-Lab at the SUPSI laboratory," in *Claude Gomez Hangzhou*, 2006.

[62] Z. Dong and K. & Cai, "Targeting the Scicos Code Generator HDL Model Example," in *Claude Gomez Hangzhou*, 2006.

[63] F. Ghenassia, "Transaction-level modeling with SystemC," *Dordrecht, The Netherlands: Springer,* pp. 153-183, 2005.

[64] "Vivado Design Suite," [Online]. Available: http://www.xilinx.com/support/documentation/white_papers/wp416-Vivado-Design-Suite.pdf. [Accessed 2015].

[65] "New York IEEE Standard Hardware Description Language Based on the Verilog Hardware Description Language (1364 - 2001)," IEEE Computer Society, 2001.

[66] "IEEE Standard VHDL Language Reference Manual (1076 - 2002)," IEEE Computer Society, New York., 2002.

[67] P. J. Ashenden, G. D. Peterson and D. A. Teegarden, The system designer's guide to VHDL-AMS: analog, mixed-signal, and mixed-technology modeling., Morgan Kaufmann, 2002.

[68] K. Einwich, J. Bastian, C. Clauss, U. Eichler and P. & Schneider, "SystemC-AMS Extension Library for Modeling Conservative Nonlinear Dynamic Systems," *In FDL,* pp. 113-119, 2006.

[69] E.Dekneuvel, "Intelligent sensors : analysis and design.," in *The industrial Information Technology Handbook*, IEEE CRC PRESS , 2005, pp. 72-1 section V.

[70] M. Stonebraker, U. Çetintemel and S. & Zdonik, " The 8 requirements of real-time stream processing," *ACM SIGMOD Record 34.4,* pp. 42-47, 2005.

[71] C. G. Cassandras, Introduction to discrete event systems, Springer Science & Business Media, 2008.

[72] "Characterize Batteries and Power Consumption Using the NI PXI-4071 7 ½-digit Digital Multimeter," Nation Instrument white paper, 2012.

[73] T. K. Nguyen, E. Dekneuvel, B. Nicolle, O. Zammit, V. C. Nguyen and G. Jacquemod, "“Using FPGA for real time power monitoring in a NIALM system.”," in *In Industrial Electronics (ISIE), 2013 IEEE International Symposium on*, 2013.

[74] W. M. Grady and a. R. J. Gilleskie., "Harmonics and how they relate to power factor," *Proceedings of PQA93,* 1993.

[75] M. Kezunovic, E. Soljanin, B. Perunicic and S. & Levi, "New approach to the design of digital algorithms for electric power measurements," *Power Delivery, IEEE Transactions on 6.2,* pp. 516-523, 1991.

[76] W.-K. Yoon and M. J. Devaney, "Reactive power measurement using the wavelet transform," *Instrumentation and Measurement, IEEE Transactions,* pp. 246-252, 2000.

[77] W. Ejaz, M. K. Atiq and H. S. & Kim, "Recursive Pyramid Algorithm-Based Discrete Wavelet Transform for Reactive Power Measurement in Smart Meters," *Energies 6.9,* pp. 4721-4738, 2013.

[78] S. D. Grigorescu, C. Cepisca, I. G. O. Potirniche and M. & Covrig, "Numerical simulations for energy calculation in power measurements," in *Proceedings of the European Computing Conference (ECC09) and Proceedings of the 3rd International Conference on Computational Intelligence (CI09)*, 2009.

[79] L. Slosarcik., ""FFT-Based Algorithm for Metering Applications" Application note, Freescale Semiconductor, Inc," Freescale Semiconductor, Inc, 2014.

[80] F. R. Kschischang, The hilbert transform., University of Toronto, 2006.

[81] M. Mienkina, ""Filter-Based Algorithm for Metering Applications." An application note of Freescale Semiconductor," Freescale Semiconductor, Inc., 2013.

[82] G. A. Petre Minciunescu, "Novel Harmonic Analysis Method Improves Accuracy, Reduces Computation Overhead in Smart Meters," 2011.

[83] A. B. Baggini, Handbook of power quality. Vol. 520, John Wiley & Sons, 2008.

[84] State_Iowa, "Energy-Related Best Practices: A Sourcebook for the Food Industry," Iowa State University, 2005.

[85] "Variable frequency drives – Energy efficiency Reference Guide," Natural Resources Canada, 2009.

[86] M. Figueiredo, A. De Almeida & B. Ribeiro, "Home electrical signal disaggregation for non-intrusive load monitoring (NIALM) systems.," *Neuro computing 96,* pp. 66-73, 2012.

[87] M. S. Tsai and Y. H. & Lin, "Modern development of an adaptive non-intrusive appliance load monitoring system in electricity energy conservation," *Applied Energy 96,* pp. 55-73, 2012.

[88] M. Basseville and I. V. Nikiforov., Detection of abrupt changes: theory and application, Englewood Cliffs: Prentice Hall, 1993, p. Vol. 104..

[89] T. K. Nguyen, O. Zammit, E. Dekneuvel, B. Nicolle, V. C. Nguyen and G. Jacquemod, "An innovative non-intrusive load monitoring system for commercial and industrial application," in *In Advanced Technologies for Communications (ATC), 2012 International Conference on*, 2012.

[90] J. Z. Kolter and a. M. J. Johnson., "REDD: A public data set for energy disaggregation research," in *SustKDD workshop on Data Mining Applications in Sustainability*, 2011.

[91] S. N. Akshay Uttama Nambi, T. G. Papaioannou, D. Chakraborty and K. & Aberer, "Sustainable energy consumption monitoring in residential settings," in *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, 2013.

[92] K.K. Sergios Theodoridis, Pattern recognition, 2nd edition, Elsevier Academic press, 2003.

[93] B. S. Pal. and a. S. Kumar, Classification and learning using genetic algorithms: applications in bioinformatics and web intelligence, Springer, 2007.