ANALYSIS OF DISCRETE WAVELET TRANSFORM FOR OPTIMUM MACHINE INSTRUCTION OF DLX MICROPROCESSOR

Shafitri Nurhanifa[1], Nyoman Bogi Aditya Karna[2], Raditiana Patmasari[3]
[1,2,3]Telecommunication Engineering, School of Electrical Engineering, Telkom University
[1]**shafitrinurhanifa@telkomuniversity.ac.id**, [2]**aditya@telkomuniversity.co.id**,
[3]**raditiana@telkomuniversity.ac.id**

Abstract

The application of monitoring over Wireless Sensor Network (WSN) is highly demanded to be implemented in the Internet of Things (IoT). The problem that appears in IoT is the general purpose microprocessor is still highly used, which causes more energy used than it is needed. Although, an Application Specific Integrated Circuit (ASIC) can be used to make a more efficient energy application, it is more expensive and permanent, which means it can't be changed or reconfigured. This thesis presents a method to design a specific purpose microprocessor by compressing an image in DLX microprocessor, which can still be reconfigured by optimizing machine instruction needed in the microprocessor. Prior to DWT process, an image will go through pre-processing stage. The stage will be done in Matlab to turn an RGB image into a grayscale image, and the matrix of the grayscale image will be obtained. This matrix will be the input for Haar DWT machine instruction. The machine instruction is simulated in WinDLX, a simulator for DLX microprocessor. After the simulation has finished, the statistics of the simulation will be analyzed to conclude whether the machine instruction is optimum enough. The result of Haar DWT machine instruction is the same as the result obtained from Matlab, which means the machine instruction is capable to do the image compression. Out of 92 kinds of instruction, Haar machine instruction only needs 20 kinds of instructions used. This shows that the program will not waste energy for unused instruction. From the statistics obtained, the total cycles executed from the pipelined DLX microprocessor is 1239 cycles, where a non-pipelined microprocessor would need 2755 cycles to execute the program. This means the program is a more efficient method to run a Haar DWT compression.

Keywords: optimum machine instruction, DLX microprocessor, DWT image compression, internet of things (IoT), wireless sensor multimedia networks.

1. Introduction

As the technology grows more and more in human life, the phrase "Internet of Things" is not an uncommon phrase to be involved in the growth. Kevin Ashton was the first person to use the term “Internet of Things” in 1999. At that time, Kevin and his team were developing an extension of the internet to accommodate things and it inspired him to the term “Internet of Things” [1]. The idea of IoT was developed in parallel to WSNs [2].

Wireless Sensor Network or WSN is a network of a large number of nodes that cooperatively sense the environment. The application of the WSN has been done since the 1980s but then became more common to use in 2001 for industrial and research purposes [2]. The WSN is largely applied in many applications, such as environmental monitoring, industrial and infrastructure, and military surveillance.

Although WSN is very useful for the convenience in the society, this technology also comes with some issues [3], such as the minimum exposure path [4] and the energy sink-hole [5], [6] in WSN. The main problem discusses in this thesis is the energy lifetime of the WSN itself, which people have been paying attention as well. Sensor nodes are usually powered by limited lifetime batteries. Changing the batteries frequently become very inefficient for a long use of WSN. There are many suggestions to this specific problem, such as wireless-powered sensor networks [7] and harvesting solar energy as a wireless charging for the WSN [8]. However, even if the additional power can be harvested to the WSN, the resource is still limited for frequent use.

Image compressing is a more detail strategy to reduce excessive energy consumption of WSN. There are many methods of image compressing used for this problem [9], [10]. This thesis uses the image compression strategy by creating DWT machine instruction to be inserted in DLX microprocessor so that the processor will run the specific instructions. This strategy will be efficient to get the most ideal microprocessor to be implanted in the WSN. Furthermore, the WSN will not be wasting energy on other microprocessor instructions that will be left unused.

The purpose of this thesis is to analyze the energy efficiency WSN by reconstructing the machine instruction. The benefit is this method can be successfully implemented on WSN in multimedia sector. The problem can be formulated as how effective DWT for optimum machine instruction affects the WSN energy efficiency in multimedia monitoring system. This thesis uses DLX microprocessor and Haar DWT algorithm in DLX assembly language. The parameters for this paper are the compression result, the power consumption, and

the speed of simulation. The completion of this thesis uses several methodologies, such as literature study, designing the system, and simulation.

2. Basic Concepts

2.1 DLX Microprocessor

DLX microprocessor is a 32-bit RISC CPU designed by John Hennessy and David Patterson [11]. The microprocessor was intended for teaching purpose at first, but now it is widely used in university-level courses. DLX is basically a simplified and modernized MIPS CPU. DLX has 32 general purposes registers and a hard-wired zero in R0.

Pipelining is a technique where several instructions overlapped in one execution. DLX features a 5 stages of instruction pipeline. The stages are Instruction Fetch (IF), Instruction Decode (ID), Executrion (EX), Memory Access (MEM), and Write Back (WB).

Hazards prevent the next instruction from being executed during its clock cycle. Hazards reduce the performance from the ideal speedup gained by pipelining. The 3 classes of hazards structural hazards, control hazards, and data hazards. These hazards will cause stall in the pipeline. Eliminating a hazard often causes some instructions to be allowed to proceed while others are delayed. When an instruction is stalled, all the following instructions are also stalled. Instructions prior to the stalled instruction must continue, otherwise the hazard will never clear.

Instructions/cycles	1	2	3	4	5	6	7	8
instruction 1	IF	ID	EX	MEM	WB			
instruction 2		IF	ID	EX	MEM	WB		
instruction 3			stall	IF	ID	EX	MEM	WB

Figure 1. An example of stall by a structural hazard.

2.2 RGB to Gray

An RGB to Grayscale conversion can be done in several softwares including Matlab. Matlab provides rgb2gray function for the conversion. The function converts RGB values to grayscale values by forming a weighted sum of the R, G, and B components, based on the calculation on Equation 1. The R, G, and B will be presented in matrix that eventually creates gray matrix.

$$Gray = 0.2989 * R + 0.5870 * G + 0.1140 * B \quad (1)$$

2.3 Haar Discrete Wavelet Transform

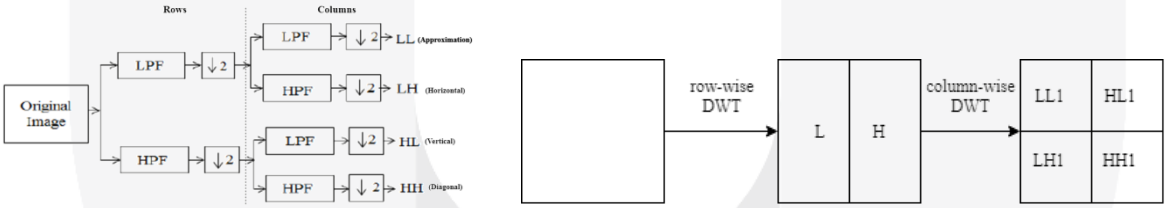

Figure 2. 2D DWT decomposition.

In 2D image, the decomposition is processed in rows and columns in two dimensional array, each of which corresponds to the horizontal and vertical direction of an image. The decomposition yields are divided into four sub-bands which are LL; the approximation coefficient and the low frequency resolution; and the detail coefficients with higher frequency resolution which are LH, HL, and HH.

In decomposition process, wavelet Haar implements Filter Bank with h(0)= h(1)= 1/√2 as the low-pass coefficient that yields to the approximation image, and g(0)= g(1)= -1/√2 as the high-pass coefficient that yields to detail images. The input should be a $2^n x 2^n$ matrix for the transformation. To calculate a level 1 Haar Transform of an array of n sample [12][13]:

1. Based on Figure 2, the first transformation will be row-wise. This transformation is illustrated on Figure 3 for an example. A 4x4 matrix has an array of values from a through p. Row-wise, these values will be calculated for the averages and the differences. The result is of the averages is placed on half-left side of the column and the differences on the other half. This produces a lowpass subband and a highpass subband. The result will be the input for the column transformation.

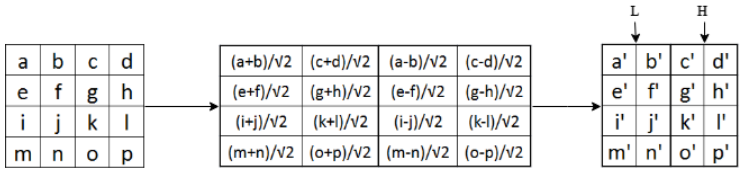

Figure 3. Row transformation.

2. Column transformation is illustrated based on Figure 4. Values from a' to p' are the result of the row transformation. The values will be calculated again for the averages and the differences but column-wise. This calculation resulted to a 4x4 matrix with an array of a'' to p'' values containing the 4 subbands expected from DWT, which are LL, LH, HL, and HH subbands.

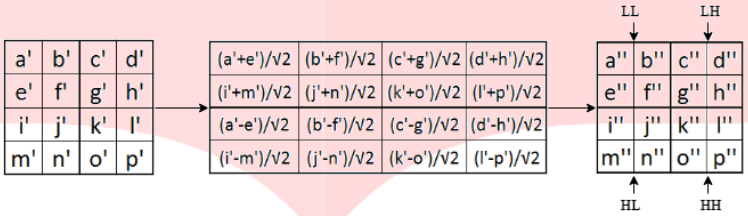

Figure 4. Column transformation.

3. Repeat the process with LL subband only to continue to level 2 transformation.

2.3 System Design

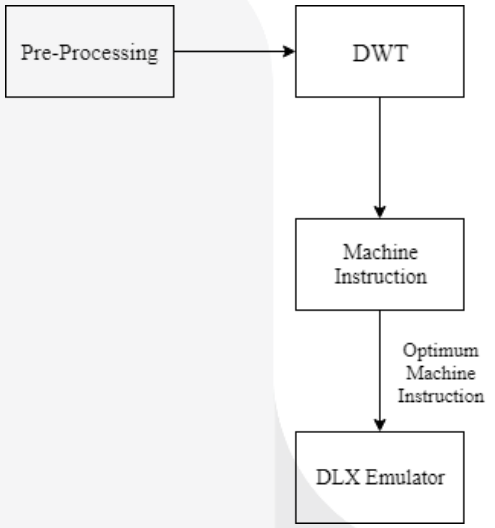

Figure 5. System model block diagram.

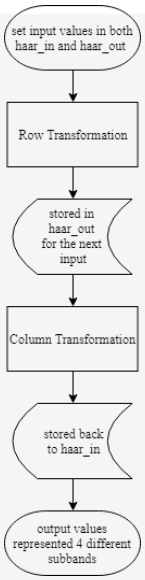

Figure 6. System design plot flowchart.

1. Pre-processing process
   As shown in Figure 5, before going to DLX process, first, the image will go to pre-processing process. This process will be done in Matlab with *rgb2gray* function which is already provided and the syntax is *I = rgb2gray(RGB)* as I is the output image and RGB is the input image. This function converts truecolor red, green, blue (RGB) image to grayscale image. A 4x4 matrix will then sampled from this image. The sampled matrix will be obtained from the far left-top of the whole matrix of the image. The 4x4 matrix will be the input for Haar DWT machine instruction.
2. Haar DWT for Machine Instruction
   This stage presents 6 as to design machine instruction for haar DWT process. This process initializes with the understanding of Haar DWT algorithm and finding the most compatible one for the machine instruction program. To start the program, 2 arrays of haar_in and haar_out will be made to be assigned to it's own calculation for both row and column transformations. Variable *s* will define square root of 2. The calculation of row transformation is done based on the calculation below:

$$v[i + j * m] = (u[2 * i + j * m] + u[2 * i + 1 + j * m])/s$$
$$v[k + i + j * m] = (u[2 * i + j * m] - u[2 * i + 1 + j * m])/s \quad (2)$$

v[] array represents haar_out array, and u[] represents haar_in array. haar_in array calculate which 2 values in the matrix that will be calculated, and haar_out array decide which array on haar_out that will be placed by that calculation. The result of the row transformation will be the input for the column one. The calculation of column transformation is done based on the calculation below:

$$v[i + j * m] = (u[i + 2 * j * m] + u[i + (2 * j + 1) * m])/s$$
$$v[i + (k + j) * m] = (u[i + 2 * j * m] - u[i + (2 * j + 1) * m])/s \quad (3)$$

Contrary from the row transformation, in the column one, v[] array represents haar_in array and u[] represents haar_out. The row transformation will be stored back in haar_in array so all the result from the calculation is gathered in haar_in array. The haar_in array can be formed into a matrix and this matrix will present the 4 subbands expected for DWT.

3. Implementation to DLX Emulator
   Haar DWT machine instruction will be implemented to WinDLX for the result. If there is no error on the machine instruction, and it reaches trap 0, then the result will be analyzed.

2.4 System Performance Parameter

Based on the result obtained from the simulation, the optimal system parameters are as follows:

1. Haar DWT Result
   In this parameter, the result of Haar DWT machine instruction will be compared to Matlab's function of *haart2*, which is a 2D Haar wavelet transform. For Haar DWT program, the result of the calculation can be seen in a website for DLX simulator. If the the results are the same, then the machine instruction manage to do Haar DWT transformation.
2. Performance Efficiency
   In this parameter, the result's statistics of of Haar DWT machine instruction will be analyzed. The statistics on WinDLX includes the total of cycles and instructions executed, stalls, conditional branches, load/store instructions, floating point stage instructions, and traps used. Because DLX is a pipelined processor, the total cycles obtained will be compared to the total cycles that would be obtained from the non-pipelined processor.

3. Result and Analysis

3.1 Result

1. Pre-processing
   The input for this paper is cactus image with resolution of 1024x1024. This image has 3 layers of RGB and is altered to 1 gray layer. The result from Matlab is shown in Figure 7.

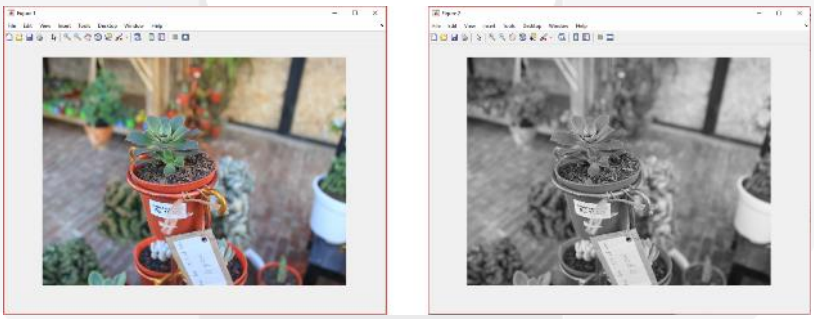

Figure 7. RGB layer.

These 3 layers of RGB image was turned into 1 layer of grayscale. Below is the 4x4 matrix sampled from the image:

$$gray = \begin{bmatrix} 98 & 99 & 98 & 97 \\ 101 & 99 & 100 & 100 \\ 102 & 101 & 102 & 99 \\ 102 & 104 & 102 & 102 \end{bmatrix}$$

the gray matrix is the input for Haar DWT machine instruction. This matrix is equivalent as haar_in array in the program.

2. Haar DWT on DLX

- Global Variable

  The global variables provided for the entire code consist of 6 variables, which are sqrt as the variable for square root of 2, haar_in as the array for input values, haar_out as the array for output values, N as the dimension of row in the matrix, M as the dimension of column in the matrix, and K as the index to get the M or N aligned to the nearest power of 2.

```
.data
sqrt:
.double 1.4142
haar_in:
.double 98, 99, 98, 97, 101, 99, 100,
100, 102, 101, 102, 99, 102, 104, 102, 102
haar_out:
.double 98, 99, 98, 97, 101, 99, 100,
100, 102, 101, 102, 99, 102, 104, 102, 102
M:
.word 4
N:
.word 4
K:
.word 1
```

Figure 8. Global variable.

- Colum and Row Transformation

  Prior to the transformation, the first thing to do is to get index K to the nearest power 2 aligned with the dimension of M or N. This index is needed because Haar DWT need a $2^n x 2^n$ matrix to be transformed based on Chapter 2. Register 1 holds the value of M and register 2 holds the value of N. Register 3 sets to 0 as a counter for M, or generally called as i, and register 4 sets to 0 as a counter for N or generally called as j. First, set K to 1, and K keeps incremented by x2 until it less than or equal to M or N.

```
; Determine K, the largest power
of 2 for K ≤ M
lw r3, K(r0)
k_less_than_m:
slli r5, r3, 1
slt r4, r1, r5
subi r4, r4, 1
beqz r4, k_loop_m_exit
slli r3, r3, 1
j k_less_than_m
```

```
; Determine K, the largest power
of 2 for K ≤ N
addi r3, r0, 1
k_less_than_n:
slli r5, r3, 1
slt r4, r2, r5
subi r4, r4, 1
beqz r4, k_loop_n_exit
slli r3, r3, 1
j k_less_than_n
```

Figure 9. K for the largest power of 2.

The second loop of K is used to define if the result of the first loop is more than 1, then K will be divided by 2. The purpose for this division is to separate the result for the averages placed in the half-left side of the matrix, and the differences placed in the right-half side. Both of the K loops is done before the row and column transformation.

```
col_transform:
slti r7, r6, 1
bnez r7, col_trans_exit
srli r6, r6, 1
sw K(r0), r6
```

```
row_transform:
slti r7, r6, 1
bnez r7, row_trans_exit
srli r6, r6, 1
sw K(r0), r6
```

Figure 10. K divided by 2.

Row transformation is done before the column one. The calculation of row transformation is done based on the calculation on Equation 2. The machine instruction is shown on Figure 11. Register 9 holds the calculation inside haar_in array, and register 5 holds the haar_out one. The calculation for the averages is done in f4, and the differences in f10. Eventually, the value in f4 and f10 will be stored in a place based the calculation inside of haar_out. The calculation of column transformation is done based on the calculation on Equation 3. The machine instruction is shown on Figure 12. The procedure for column transformation is nearly the same as the row one. The difference lies only for haar_out is now the input and haar_in is the final output.

```
col_trans_k:
slli r5, r3, 1
mult r8, r4, r1
add r9, r5, r8
ld f2, haar_in(r9)
addi r9, r9, 8
ld f6, haar_in(r9)
addd f4, f2, f6
divd f4, f4, f8
subd f10, f2, f6
divd f10, f10, f8
add r5, r3, r8
sd haar_out(r5), f4
slli r6, r6, 3
add r5, r5, r6
sd haar_out(r5), f10
```

Figure 11. Row Transformation.

```
row_trans_m:
slli r5, r4, 1
mult r8, r5, r1
add r9, r8, r3
ld f2, haar_out(r9)
addi r5, r5, 8
mult r9, r5, r1
add r9, r9, r3
ld f6, haar_out(r9)
ld f8, sqrt(r0)
addd f4, f2, f6
divd f4, f4, f8
subd f10, f2, f6
divd f10, f10, f8
srli r8, r8, 1
add r5, r3, r8
sd haar_in(r5), f4
slli r6, r6, 3
add r5, r8, r6
add r5, r5, r3
sd haar_in(r5), f10
```

Figure 12. Column Transformation.

3. Result Comparison

The results obtained for the comparison between Haar DWT machine instruction and Haar DWT in Matlab is shown in Figure 13 and Figure 14. Because of the same result is obtained from both program, thus concluded that Haar DWT machine instruction is capable to do Haar DWT image compression.

$$gray = \begin{bmatrix} 198.503831 & 197.503812 \\ 204.503947 & 202.503908 \end{bmatrix}$$

Figure 13. Approximation result in the program.

$$gray = \begin{bmatrix} 198.5000 & 197.5000 \\ 204.5000 & 202.5000 \end{bmatrix}$$

Figure 14. Approximation result in Matlab.

3.1 Analysis

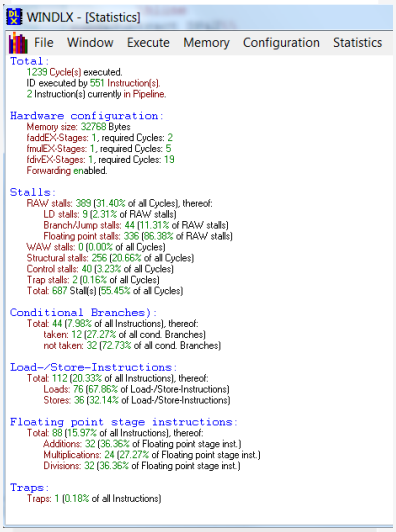

Figure 15. Statistics result of Haar DWT machine instruction.

Figure 14 shows the statistics of Haar DWT program after the simulation has finished. Below is the analysis for the statistics obtained:

1. Total

The total cycles executed are 1239 cycles. Instruction Decode (ID) executed by 551 instructions. It means that out of 775 instructions that need to be executed, it only took 1239 cycles for a pipelined DLX instruction to execute them. However, a non-pipelined microprocessor would require more cycles to run this program. In non-pipelined processor, one cycle can only run one stage out of 5 stages, for an instruction to be fully executed. It means that a non-pipelined processor requires 551x 5 or 2755 cycles to fully executed

Haar DWT machine instruction. With that fact, this program is already a more efficient method to run a Haar DWT compression.

2. Hardware Configuration
   In hardware configuration, DLX needs 32768 bytes of memory. It also requires 1 faddEX-Stages, 1 fmulEX-Stages, and 1 fdicEC-Stages. This configuration is a default configuration in WinDLX emulator.
3. Stalls
   In order to avoid a hazard, a pipelined processor will cause a stall which is a delay of an instruction. For Haar DWT machine Instruction, Read After Write (RAW) stalls occurred with 31.40% of the cycles, with the floating point stalls takes the majority of the percentage because of the instructions involve many floating points calculation, and some calculations also need the result of the prior calculations as their input. Structural stall takes the percentage by 20.66%. The total of all the stalls is 55.45%, which is a reasonable number considering the amount of floating point calculation in the program.
4. Conditional Branches
   The total conditional branches of all instructions is 7.98% with 27.27% branches were taken and the other 72.73% were not taken. This conditional branches is needed to select which instruction to proceed to and which instruction to pass on so it will not cost more memory.
5. Load and Store Instruction
   There are 67.86% of load instructions and 32.14% of the store ones, with the total 0f 20.33% of all instructions. These load instructions were by the loading process of M, N, and K data from the registers. The transformation were also loading and storing the values between haar_in and haar_out arrays.
6. Floating Point Stage Instruction
   The total for this instruction is 15.97% of all instructions. The instruction completed with 36.36% addition, 27.27% multiplication, and 36.36% division. These floating point instructions were used for the calculation involving *sqrt* variable which holds square root of 2.
7. Trap
   Only 1 trap instruction on the program which only took 0.18% of all instructions. The trap #0 instruction used to finish the program.

$$\text{Clock cycle time} = \frac{\text{CPU time}}{\text{CPU clock cycles for a program}} \quad (4)$$

The CPU time obtained from the program is 14.32 s. The clock cycles for the program are 1239 cycles. Both numbers is calculated based on Equation 4 to find the clock cycle time, which yields to 11.56 ms. This means the processor needs 11.56 ms to execute 1 cycle. Compared to modern processors, this period of time is way slower, but considering the time DLX made and the purpose of it, then it is an appropriate time. The execution time per instruction can be concluded as:

- Branch Instruction (6 cycles) = 69.36 ms
- Load and Store Instructions (5 cycles) = 57.8 ms
- R-type Instruction (5 cycles) = 57.8 ms
- Jump Instruction (5 cycles) = 57.8 ms
- Trap instruction (4 cycles) = 46.24 ms

4. Conclusion

This thesis proposes an image compression strategy by creating Haar DWT machine instruction to be implemented in DLX microprocessor so that the processor will run a specific instruction. The simulation of the machine instruction is done in WinDLX application, which is the DLX microprocessor emulator. The result of Haar DWT machine instruction is the same as the result obtained from *haart2* function in Matlab, that means the machine instruction is capable to do the image compression. Out of 92 kinds of instruction, Haar machine instruction only needs 20 kinds of instructions used. This shows that the program will not waste energy for unused instruction, thus to be more efficient. From the statistics obtained, the total cycles executed from the pipelined DLX microprocessor is 1239 cycles, where a non-pipelined microprocessor would need 2755 cycles to execute the program. This means the program is a more efficient method to run a Haar DWT compression.

Blibliography:

[1] Duncan Mcfarlane, “The origin of the Internet of Things,” June 26, 2015, Retreived from https://www.redbite.com/the-origin-of-the-internet-of-things/. [accesed on February 9, 2017].

[2] Internasional Electrotechnical Comission, White Paper “Internet of Things: Wireless Sensor Network.”

[3] S. Sukhwinder, B. Rakesh Kumar, B. Savina, “Issues and Challenges in Wireless Sensor Network,” International Conference on Machine Intelligence Research and Advancement (ICMIRA), December 2013.
[4] Y. Miao, W. Yuping, D. Cai, W. Xiaoli, “A Hybrid Genetic Alghoritm for The Minimum Exposure Path Problem of Wireless Sensor Network Based on Numerical Functional Extreme Model,” IEEE Transaction on Vehicular Technology, Vol. 65, No. 10, October 2016.
[5] R. Ju, Z. Yaoxue, Z. Kuan, L. Anfeng, C. Jianer, S. Xuemin, “Lifetime and Energy Hole Evolution Analysis on Data-Gathering Wireless Sensor Networks,” IEEE Transaction on Industrial Informatics, Vol. 12, No. 2, April 2016.
[6] M. A. Habib, “Investigating the Energy Sink-Hole Problem in Connected k-Covered Wireless Sensor Networks,” IEEE Transactions on Computer, Vol. 63, No. XX, 2014.
[7] S. Min, Z. Mheng, “Energy Efficieny Optimization For Wireless Powered Sensor Networks With Nonorthogonal Multiple Access,” IEEE Sensors Letters, Vol. 2, No. 1, March 2018.
[8] W. Cong, Y. Yuanyuang, Y. Fan, “Combining Solar Energy Harvesting with Wireless Charging for Wireless Sensor Networks,” IEEE Transactions on Mobile Computing, Vol. 17, No. 3, March 2018.
[9] W. Yong, W. Dianhong, Z. Xufan, C. Jun, L. Yamin, “Energy-Efficient Image Compressive Transmission for Wireless Camera Networks,” IEEE Sensors Journal, Vol. 16, No. 10, May 15, 2016.
[10] C. Shin-Lun, W. Ghuei-Shian, “A Cost and Power Efficient Image Compressior VSLI Design With Fuzzy Decision and Block Partition for Wireless Sensor Networks,” IEEE Sensors Journal, Vol. 17, No. 15, August 1, 2017.
[11] B. Fagin, P. Chitrakulchai, “Prototyping The DLX Microprocessor,” IEEE workshop on Rapid System Prototyping, NC, USA, June 1992.
[12] S. Arora, Y. S. Brar, S. Kumar, "Haar Wavelet Transform for solution of image retrieval," International Journal of Advanced Computer and Mathematical Sciences, Vol 5, pp27-31, 2014.
[13] N. Ledya, K. Adrian, "Analisis Perbandingan Kompresi Haar Wavelet Transform dengan Embedded Zerotree Wavelet pada Citra," Jurnal ELKOMIKA, No. 2, Vol. 3, 2015.