

IMPLEMENTATION OF COLLISION CONTROL SYSTEM IN AUTOMOBILES

Muthumanikkamoorthy R¹, C.Jagadeesh Vikram², Abdul Khadar Basha Mohammad³, Anthonyraj J⁴,
Sakthi Narayan D⁵ & Raviteja⁶

^{1,3,4,5,6} B.Tech Student, ² Assistant Professor,

Department of Automobile Engineering, BIST,BIHER,, Bharath University, Chennai, India.

jagadeeshvikram@bharathuniv.ac.in

Abstract— A collision avoidance system is an automobile safety system designed to reduce the severity of a collision. It is also known as a precrash system, forward collision warning system, or collision It uses radar in all weather conditions and sometimes laser and camera (employing image recognition to detect an imminent crash.gpsensors can detect fixed dangers such as approaching stop signs

Once the detection is done, these systems either provide a warning to the driver when there is an imminent collision or take action autonomously without any driver input (by braking or steering or both). Collision avoidance by braking is appropriate at low vehicle speeds (e.g. below 50 km/h), while collision avoidance by steering is appropriate at higher vehicle speeds. Cars with collision avoidance may also be equipped with adaptive cruise control and use the same forward-looking sensors.

Keywords — Aluminum piston, Low Heat-Rejection Engines, Air gap insulated piston, Structural Analysis and Thermal Analysis.

I. INTRODUCTION

Early warning systems have been attempted as early as the late 1950s. Cadillac for instance, developed a prototype vehicle named the Cadillac Cyclone which used the new technology of radar to detect objects in the front of the car with the radar sensors mounted inside the "nose cones"[1-6]. It was deemed too costly and the model was subsequently dropped. Features such as this were typically mentioned as marketing tools[7-15], and likely never actually worked.

The first modern demonstration of a truly functional forward collision avoidance was performed in 1995 by a team of scientists and engineers at Hughes Research Laboratories in Malibu, California. The project was funded by Delco Electronics, and was led by HRL physicist Ross D. Olney. The technology was labeled for marketing purposes as "Forewarn". The system was radar based a technology that was readily available at Hughes

Electronics but not commercially elsewhere. A small custom fabricated radar-head was developed specifically for this automotive application at 77 GHz. The forward radar-head, plus the signal processing unit and visual-audio-tactile feedbacks were first integrated into a Lexus LS400, and shortly thereafter into a Cadillac STS[16-21].

Benefits

By the Insurance Institute for Highway Safety examined how particular features of crash-avoidance systems affected the number of claims under various forms of insurance coverage[22-26]. The findings indicate that two crash-avoidance features provide the biggest benefits: (a) autonomous braking that would brake on its own, if the driver does not, to avoid a forward collision, and (b) adaptive headlights that would shift the headlights in the direction the driver steers. They found lane departure systems to be not helpful, and perhaps harmful, at the circa 2012 stage of development. A 2015 Insurance Institute for Highway Safety study found forward collision warning and automatic braking systems reduced rear collisions[27-30].

In the 2016 Berlin terror attack a truck was driven into the Berlin Christmas market and was brought to a stop by its automatic braking system Collision avoidance features are rapidly making their way into the new vehicle fleet. In a study of police-reported crashes, automatic emergency braking was found to reduce the incidence of rear-end crashes by 39 percent.[A 2012 study suggests that if all cars feature the system[31-36], it will reduce accidents by up to 27 percent and save up to 8,000 lives per year on European roads.

Pedestrian Detection

Honda has developed since 2004 a night vision system that highlights pedestrians in front of the vehicle by alerting the driver with an audible chime and visually displaying them via HUD. Honda's system only works in temperatures below 30 degrees Celsius (86 Fahrenheit). This system first appeared on the Honda Legend to assist in pedestrian safety as well as driver safety, Volvo implemented a pedestrian airbag in the Volvo V40 introduced in 2012[37-41].

III. COMPONENTS AND DESCRIPTION

Microwave Sensors

Microwave sensors, also known as Radar, RF or Doppler sensors, detect walking, running or crawling human targets in an outdoor environment. Southwest Microwave developed the industry's first bi-static microwave sensor in 1971, and has pioneered the development of flexible, reliable microwave links and transceivers for the protection of open areas, gates or entryways and rooftop or wall applications.

Microwave sensors generate an electromagnetic field between transmitter and receiver, creating an invisible volumetric detection zone. When an intruder enters the detection zone, changes to the field are registered and an alarm occurs. Our microwave sensors are easy to install, provide high probability of detection, low nuisance alarms and resistance to rain, fog, wind, dust, falling snow and temperature extremes. Most operate at K-Band frequency, maximizing detection performance and minimizing interference from external radar sources.

We pointed out that a machine instruction set goes a long way toward defining the processor. If we know the machine instruction set, including an understanding of the effect of each opcode and an understanding of the addressing modes, and if we know the set of user visible registers, then we know the functions that the processor must perform. This is not the complete picture. We must

know the external interfaces, usually through a bus, and how interrupts are handled. With this line of reasoning, the following list of those things needed to specify the

- Operations (opcodes)
- Addressing modes
- Registers
- i/o module interferences

Micro-operations

We have seen that the operation of a computer, in executing a program[42-45], consists of a sequence of instruction cycles, with one machine instruction per cycle. Of course, we must remember that this sequence of instruction cycles is not necessarily the same as the written sequence of instructions that make up the program, because of the existence of branching instructions. What we are referring to here is the execution time sequence of instructions. We have further seen that each instruction cycle is made up of a number of smaller units. One subdivision that we found convenient is fetch, indirect, execute, and interrupt[46-50], with only fetch and execute cycles always occurring.

Constituent Elements of a Program Execution of the smaller cycles involves a series of steps, each of which involves the processor registers. We will refer to these steps as micro-operations. The prefix micro refers to the fact that each step is very simple and accomplishes very little. Depicts the relationship among the various concepts we have been discussing. To summarize, the execution of a program consists of the sequential execution of instructions[51-56]. Each instruction is executed during an instruction cycle made up of shorter sub cycles (e.g., fetch, indirect, execute, interrupt). The execution of each sub cycle involves one or more shorter operations, that is, micro-operations.

Micro-operations are the functional, or atomic, operations of a processor. In this section, we will examine micro-operations to gain an understanding of how the

events of any instruction cycle can be described as a sequence of such micro-operations. A simple example will be used. In the remainder of this chapter, we then show how the concept of micro operations serves as a guide to the design of the control unit. The Fetch Cycle We begin by looking at the fetch cycle, which occurs at the beginning of each instruction cycle and causes an instruction to be fetched from memory. For purposes of discussion, we assume the organization depicted in Figure

Memory address register (MAR):

Is connected to the address lines of the system bus. It specifies the address in memory for a read or write operation. • Memory buffer register (MBR): Is connected to the data lines of the system bus. It contains the value to be stored in memory or the last value read from memory. • Program counter (PC): Holds the address of the next instruction to be fetched. • Instruction register (IR): Holds the last instruction fetched. Let us look at the sequence of events for the fetch cycle from the point of view of its effect on the processor registers. An example appears in Figure 19.2. At the beginning of the fetch cycle, the address of the next instruction to be executed is in the program counter (PC); in this case, the address is 1100100.

The first step is to move that address to the memory address register (MAR) because this is the only register connected to the address lines of the system bus. The second step is to bring in the instruction. The desired address (in the MAR) is placed on the address bus, the control unit issues a READ command on the control bus, and the result appears on the data bus and is copied into the memory buffer register (MBR) Sequence of Events, Fetch Cycle to increment the PC by the instruction length to get ready for the next instruction. Because these two actions (read word from memory, increment PC) do not interfere with each other, we can do them simultaneously to save time. The third step is to move the contents of the MBR to

the instruction register (IR). This frees up the MBR for use during a possible indirect cycle.

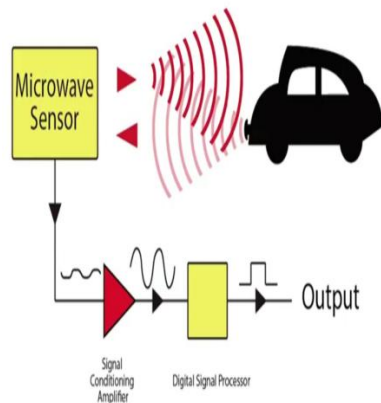
Thus, the simple fetch cycle actually consists of three steps and four micro-operations. Each micro-operation involves the movement of data into or out of a register. So long as these movements do not interfere with one another, several of them can take place during one step, saving time. Symbolically, we can write this sequence of events as follows: $t1: MAR \leftarrow (PC)$ $t2: MBR \leftarrow \text{Memory}$ $PC \leftarrow (PC) + I$ $t3: IR \leftarrow (MBR)$ where I is the instruction length. We need to make several comments about this sequence. We assume that a clock is available for timing purposes and that it emits regularly spaced clock pulses. Each clock pulse defines a time unit. Thus, all time units are of equal duration.

Each micro-operation can be performed within the time of a single time unit. The notation ($t1, t2, t3$) represents successive time units. In words, we have • First time unit: Move contents of PC to MAR. • Second time unit: Move contents of memory location specified by MAR to MBR. Increment by I the contents of the PC. • Third time unit: Move contents of MBR to IR. Note that the second and third micro-operations both take place during the second time unit. The third micro-operation could have been grouped with the fourth without affecting the fetch operation: $t1: MAR \leftarrow (PC)$ $t2: MBR \leftarrow \text{Memory}$ $t3: PC \leftarrow (PC) + I$ $IR \leftarrow (MBR)$ the groupings of micro-operations must follow two simple rules: 1. the proper sequence of events must be followed. Thus ($MAR \leftarrow (PC)$) must precede ($MBR \leftarrow \text{Memory}$) because the memory read operation makes use of the address in the MAR. 2. Conflicts must be avoided. One should not attempt to read to and write from the same register in one time unit, because the results would be unpredictable.

For example,

The micro-operations ($MBR \leftarrow \text{Memory}$) and ($IR \leftarrow MBR$) should not occur during the same time unit. A

final point worth noting is that one of the micro-operations involves an addition. To avoid duplication of circuitry, this addition could be performed by the ALU. The use of the ALU may involve additional micro-operations, depending on the functionality of the ALU and the organization of the processor. We defer a discussion of this point until later in this chapter. It is useful to compare events described in this and the following.



The Indirect Cycle Once an instruction is fetched, the next step is to fetch source operands. Continuing our simple example, let us assume a one-address instruction format, with direct and indirect addressing allowed. If the instruction specifies an indirect address, then an indirect cycle must precede the execute cycle. The data flow differs somewhat from that indicated

t1: MAR $\leftarrow$ (IR(Address))

t2: MBR $\leftarrow$ Memory

t3: IR(Address) $\leftarrow$ (MBR(Address))

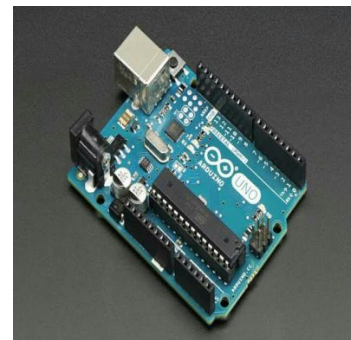
The address field of the instruction is transferred to the MAR. This is then used to fetch the address of the operand. Finally, the address field of the IR is updated from the MBR, so that it now contains a direct rather than an indirect address. The IR is now in the same state as if indirect addressing had not been used, and it is ready for the execute cycle. We skip that cycle for a moment, to consider the interrupt cycle. **The Interrupt Cycle** At the completion of the execute cycle, a test is made to determine whether any enabled interrupts have occurred. If so, the

interrupt cycle occurs. The nature of this cycle varies greatly from one machine to another. We present a very simple sequence of events, as illustrated

t1: MBR $\leftarrow$ (PC)

t2: MAR $\leftarrow$ Save_Address PC $\leftarrow$ Routine_Address t3: Memory $\leftarrow$ (MBR)

In the first step, the contents of the PC are transferred to the MBR, so that they can be saved for return from the interrupt. Then the MAR is loaded with the address at which the contents of the PC are to be saved, and the PC is loaded with the address of the start of the interrupt-processing routine. These two actions may each be a single micro-operation. However, because most processors provide multiple types and/or levels of interrupts, it may take one or more additional micro-operations to obtain the Save_Address and the Routine_Address before they can be transferred to the MAR and PC, respectively. In any case, once this is done, the final step is to store the MBR, which contains the old value of the PC, into memory. The processor is now ready to begin the next instruction cycle.



The Execute Cycle The fetch, indirect, and interrupt cycles are simple and predictable. Each involves a small, fixed sequence of micro-operations and, in each case, the same micro-operations are repeated each time around. This is not true of the execute cycle. Because of the variety of opcodes, there are a number of different sequences of micro-operations that can occur. The control unit examines the opcode and generates a sequence of micro-operations based on the value of the opcode. This is

referred to as instruction decoding. Let us consider several hypothetical examples.

First, consider an add instruction: ADD R1, X which adds the contents of the location X to register R1. The following sequence of micro-operations might occur: t1: MAR $\leftarrow$ (IR(address)) t2: MBR $\leftarrow$ Memory t3: R1 $\leftarrow$ (R1) + (MBR) We begin with the IR containing the ADD instruction. In the first step, the address portion of the IR is loaded into the MAR. Then the referenced memory location is read. Finally, the contents of R1 and MBR are added by the ALU. Again, this is a simplified example. Additional micro-operations may be required to extract the register reference from the IR and perhaps to stage the ALU inputs or outputs in some intermediate registers. Let us look at two more complex examples. A common instruction is increment and skip if zero: ISZ X The content of location X is incremented by 1. If the result is 0, the next instruction is skipped. A possible sequence of micro-operations is

t1: MAR $\leftarrow$ (IR(address))
 t2: MBR $\leftarrow$ Memory
 t3: MBR $\leftarrow$ (MBR) + 1
 t4: Memory $\leftarrow$ (MBR)
 If ((MBR) = 0) then (PC $\leftarrow$ (PC) + I)

The new feature introduced here is the conditional action. The PC is incremented if (MBR) = 0. This test and action can be implemented as one micro-operation. Note also that this micro-operation can be performed during the same time unit during which the updated value in MBR is stored back to memory. Finally, consider a subroutine call instruction. As an example, consider a branch-and-save-address instruction: BSA X The address of the instruction that follows the BSA instruction is saved in location X, and execution continues at location X + I. The saved address will later be used for return. This is a straightforward technique for supporting subroutine calls. The following micro-operations suffice:

t1: MAR $\leftarrow$ (IR(address)) MBR $\leftarrow$ (PC) t2: PC $\leftarrow$ (IR(address)) Memory $\leftarrow$ (MBR) t3: PC $\leftarrow$ (PC) + I The address in the PC at the start of the instruction is the address of the next instruction in sequence.

This is saved at the address designated in the The Execute Cycle The fetch, indirect, and interrupt cycles are simple and predictable. Each involves a small, fixed sequence of micro-operations and, in each case, the same micro-operations are repeated each time around. This is not true of the execute cycle. Because of the variety of opcodes, there are a number of different sequences of micro-operations that can occur. The control unit examines the opcode and generates a sequence of micro-operations based on the value of the opcode. This is referred to as instruction decoding. Let us consider several hypothetical examples. First, consider an add instruction: ADD R1, X which adds the contents of the location X to register R1.

Control of the Processor

Functional Requirements As a result of our analysis in the preceding section, we have decomposed the behaviour or functioning of the processor into elementary operations, called micro-operations. By reducing the operation of the processor to its most fundamental level, we are able to define exactly what it is that the control unit must cause to happen. Thus, we can define the functional requirements for the control unit: those functions that the control unit must perform.

A definition of these functional requirements is the basis for the design and implementation of the control unit. With the information at hand, the following three-step process leads to a characterization of the control unit: 1. Define the basic elements of the processor. 2. Describe the micro-operations that the processor performs. 3. Determine the functions that the control unit must perform to cause the micro-operations to be performed. We have already performed steps 1 and 2. Let us summarize the results. First, the basic functional elements of the processor are the

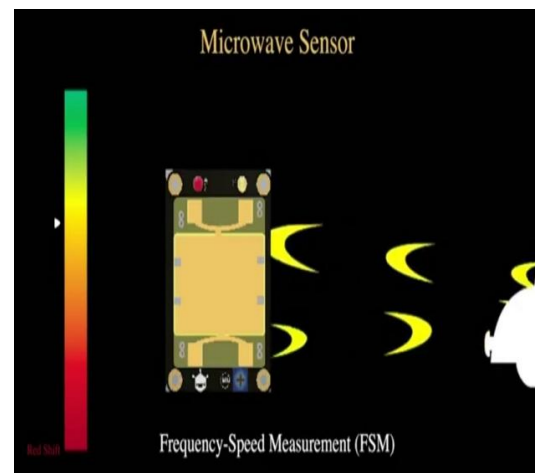
following: • ALU • Registers • Internal data paths • External data paths • Control unit some thought should convince you that this is a complete list. The ALU is the functional essence of the computer. Registers are used to store data internal to the processor.

Some registers contain status information needed to manage instruction sequencing (e.g., a program status word). Others contain data that go to or come from the ALU, memory, and I/O modules. Internal data paths are used to move data between registers and between register and ALU. External data paths link registers to memory and I/O modules, often by means of a system bus.

The control unit causes operations to happen within the processor. The execution of a program consists of operations involving these processor elements. As we have seen, these operations consist of a sequence of micro-operations. Upon review of Section 19.1, the reader should see that all micro-operations fall into one of the following categories:

- Transfer data from one register to another.
- Transfer data from a register to an external interface (e.g., system bus).
- Transfer data from an external interface to a register.
- Perform an arithmetic or logic operation, using registers for input and output. All of the micro-operations needed to perform one instruction cycle, including all of the micro-operations to execute every instruction in the instruction set, fall into one of these categories.

We can now be somewhat more explicit about the way in which the control unit functions. The control unit performs two basic tasks: • sequencing: The control unit causes the processor to step through a series of micro-operations in the proper sequence, based on the program being executed.



Execution: The control unit causes each micro-operation to be performed. The preceding is a functional description of what the control unit does. The key to how the control unit operates is the use of control signals.

CONCLUSION:

It is one of the great choices to do project on Collisions of control systems. I got some exposure, knowledge from the resources. And I know about the peaceful setup how helps the team management and work to succeed in our life. And I thank who all helped me to finish this Journal Report

REFERENCE :

1. Ramamoorthy, R., Kanagasabai, V., Kausalya, R., Impact of celebrities' image on brand, International Journal of Pure and Applied Mathematics, V-116, I-18 Special Issue, PP-251-253, 2017
2. Ramamoorthy, R., Kanagasabai, V., Vignesh, M., Quality assurance in operation theatre withreference to fortis malar hospital, International Journal of Pure and Applied Mathematics, V-116, I-14 Special Issue, PP-87-93, 2017
3. Ramya, N., Arthy, J., Honey comb graphs and its energy, International Journal of Pure and Applied Mathematics, V-116, I-18 Special Issue, PP-83-86, 2017

4. Ramya, N., Jagadeeswari, P., Proper coloring of regular graphs, International Journal of Pure and Applied Mathematics, V-116, I-16 Special Issue, PP-531-533, 2017
5. Ramya, N., Karunagaran, K., Proper, star and acyclic coloring of some graphs, International Journal of Pure and Applied Mathematics, V-116, I-16 Special Issue, PP-43-44, 2017
6. Ramya, N., Muthukumar, M., On coloring of 4-regular graphs, International Journal of Pure and Applied Mathematics, V-116, I-16 Special Issue, PP-491-494, 2017
7. Ramya, N., Muthukumar, M., On star and acyclic coloring of graphs, International Journal of Pure and Applied Mathematics, V-116, I-16 Special Issue, PP-467-469, 2017
8. Ramya, N., Pavi, J., Coloring of book and gear graphs, International Journal of Pure and Applied Mathematics, V-116, I-17 Special Issue, PP-401-402, 2017
9. Ramya, P., Hameed Hussain, J., Alteration framework for integrating quality of service in internet real-time network, International Journal of Pure and Applied Mathematics, V-116, I-8 Special Issue, PP-57-61, 2017
10. Ramya, P., Sriram, M., Tweet sarcasm: Peep, International Journal of Pure and Applied Mathematics, V-116, I-10 Special Issue, PP-231-235, 2017
11. Sabarish, R., Meenakshi, C.M., Comparision of beryllium and CI connecting rod using ansys, International Journal of Pure and Applied Mathematics, V-116, I-17 Special Issue, PP-127-132, 2017
12. Sabarish, R., Rakesh, N.L., Outcome of inserts for enhancing the heat exchangers, International Journal of Pure and Applied Mathematics, V-116, I-17 Special Issue, PP-419-422, 2017
13. Sangeetha, M., Gokul, N., Aruls, S., Estimator for control logic in high level synthesis, International Journal of Pure and Applied Mathematics, V-116, I-20 Special Issue, PP-425-428, 2017
14. Sangeetha, M., Gokul, N., Aruls, S., Image steganography using a curvelet transformation, International Journal of Pure and Applied Mathematics, V-116, I-20 Special Issue, PP-417-422, 2017
15. Saraswathi, P., Srinivasan, V., Peter, M., Research on financial supply chain from view of stability, International Journal of Pure and Applied Mathematics, V-116, I-17 Special Issue, PP-211-213, 2017
16. Saravana Kumar, A., Hameed Hussain, J., Expanding the pass percentage in semester examination, International Journal of Pure and Applied Mathematics, V-116, I-15 Special Issue, PP-45-48, 2017
17. Saravana, S., Arulselvi, S., AdaBoost SVM based brain tumour image segmentation and classification, International Journal of Pure and Applied Mathematics, V-116, I-20 Special Issue, PP-399-403, 2017
18. Saravana, S., Arulselvi, S., Dynamic power management monitoring and controlling system using wireless sensor network, International Journal of Pure and Applied Mathematics, V-116, I-20 Special Issue, PP-405-408, 2017
19. Saravana, S., Arulselvi, S., Clustered morphic algorithm based medical image analysis, International Journal of Pure and Applied Mathematics, V-116, I-20 Special Issue, PP-411-415, 2017
20. Saravana, S., Arulselvi, S., Networks, International Journal of Pure and Applied Mathematics, V-116, I-20 Special Issue, PP-393-396, 2017
21. Saritha, B., Chockalingam, M.P., Adsorptive removal of heavy metal chromium from aqueous medium using

- modified natural adsorbent, International Journal of Civil Engineering and Technology, V-8, I-8, PP-1382-1387, 2017
22. Saritha, B., Chockalingam, M.P., Adsorptive removal of brilliant green dye by modified coconut shell adsorbent, International Journal of Pure and Applied Mathematics, V-116, I-13 Special Issue, PP-211-215, 2017
 23. Saritha, B., Chockalingam, M.P., Photodegradation of eriochrome black-T dye from aqueous medium by photocatalysis, International Journal of Pure and Applied Mathematics, V-116, I-13 Special Issue, PP-183-187, 2017
 24. Saritha, B., Chockalingam, M.P., Photodegradation of malachite green DYE using TiO_2 /activated carbon composite, International Journal of Civil Engineering and Technology, V-8, I-8, PP-156-163, 2017
 25. Saritha, B., Chockalingam, M.P., Synthesis of photocatalytic composite Fe-C/TiO₂ for degradation of malachite green dye from aqueous medium, International Journal of Pure and Applied Mathematics, V-116, I-13 Special Issue, PP-177-181, 2017
 26. Saritha, B., Chockalingam, M.P., Removal of heavy X^+X^- from aqueous medium using modified natural adsorbent, International Journal of Pure and Applied Mathematics, V-116, I-13 Special Issue, PP-205-210, 2017
 27. Saritha, B., Chockalingam, M.P., Degradation of malachite green dye using a semiconductor composite, International Journal of Pure and Applied Mathematics, V-116, I-13 Special Issue, PP-195-199, 2017
 28. Sartiha, B., Chockalingam, M.P., Photocatalytic decolourisation of textile industry wastewater by TiO_2 , International Journal of Pure and Applied Mathematics, V-116, I-18 Special Issue, PP-221-224, 2017
 29. Sartiha, B., Chockalingam, M.P., Study on photocatalytic degradation of Crystal Violet dye using a semiconductor, International Journal of Pure and Applied Mathematics, V-116, I-18 Special Issue, PP-209-212, 2017
 30. Shanthi, E., Nalini, C., Rama, A., The effect of highly-available epistemologies on hardware and architecture, International Journal of Pharmacy and Technology, V-8, I-3, PP-17082-17086, 2016
 31. Shanthi, E., Nalini, C., Rama, A., Drith: Autonomous, random communication, International Journal of Pharmacy and Technology, V-8, I-3, PP-17002-17006, 2016
 32. Shanthi, E., Nalini, C., Rama, A., A case for replication, International Journal of Pharmacy and Technology, V-8, I-3, PP-17234-17238, 2016
 33. Shanthi, E., Nalini, C., Rama, A., Elve: A methodology for the emulation of robots, International Journal of Pharmacy and Technology, V-8, I-3, PP-17182-17187, 2016
 34. Shanthi, E., Nalini, C., Rama, A., Autonomous epistemologies for 802.11 mesh networks, International Journal of Pharmacy and Technology, V-8, I-3, PP-17087-17093, 2016
 35. Sharavanan, R., Golden Renjith, R.J., Design and analysis of fuel flow in bend pipes, International Journal of Pure and Applied Mathematics, V-116, I-15 Special Issue, PP-59-64, 2017
 36. Sharavanan, R., Jose Ananth Vino, V., Emission analysis of C.I engine run by diesel, sunflower oil, 2 ethyl hexyl nitrate blends, International Journal of Pure and Applied Mathematics, V-116, I-14 Special Issue, PP-403-408, 2017

37. Sharavanan, R., Sabarish, R., Design of built-in hydraulic jack for light motor vehicles, International Journal of Pure and Applied Mathematics, V-116, I-17 Special Issue, PP-457-460, 2017
38. Sharavanan, R., Sabarish, R., Design and fabrication of aqua silencer using charcoal and lime stone, International Journal of Pure and Applied Mathematics, V-116, I-14 Special Issue, PP-513-516, 2017
39. Sharmila, G., Thooyamani, K.P., Kausalya, R., A schoolwork on customer relationship management with special reference to domain 2 host, International Journal of Pure and Applied Mathematics, V-116, I-20 Special Issue, PP-199-203, 2017
40. Sharmila, S., Jeyanthi Rebecca, L., Anbuselvi, S., Kowsalya, E., Kripanand, N.R., Tanty, D.S., Choudhary, P., SwathyPriya, L., GC-MS analysis of biofuel extracted from marine algae, Der Pharmacia Lettre, V-8, I-3, PP-204-214, 2016
41. Sidharth Raj, R.S., Sangeetha, M., Data embedding method using adaptive pixel pair matching method, International Journal of Pure and Applied Mathematics, V-116, I-15 Special Issue, PP-417-421, 2017
42. Sidharth Raj, R.S., Sangeetha, M., Android based industrial fault monitoring, International Journal of Pure and Applied Mathematics, V-116, I-15 Special Issue, PP-423-427, 2017
43. Sidharth Raj, R.S., Sangeetha, M., Mobile robot system control through an brain computer interface, International Journal of Pure and Applied Mathematics, V-116, I-15 Special Issue, PP-413-415, 2017
44. Sivaraman, K., Sundarraj, B., Decisive lesion detection in digital fundus image, International Journal of Pure and Applied Mathematics, V-116, I-10 Special Issue, PP-161-164, 2017
45. Sridhar, J., Sriram, M., Cloud privacy preserving for dynamic groups, International Journal of Pure and Applied Mathematics, V-116, I-8 Special Issue, PP-117-120, 2017

