8 REFERENCES

[1]. "Ace Controls," Ace Controls, 2014. [Online]. Available: http://www.acecontrols.co.uk/news/understanding-shock-absorbers/. [Accessed 2014].

[2]. D. B. Howad and Geoffrey, in *Car Suspention and Handling 3rd edition*, Ed. London, Pentech Press, 1993.

[3]. A. A. Aly, F. A. Salem, "Vehicle Suspension Systems Control: A Review," *International journal of control, automation and systems*, vol. Vol.2, no. No.2, pp. 2165-8277, 2013.

[4]. K. Yuki, T. Murakami, K. Ohnishi, "Vibration Control of 2 Mass Resonant System by Resonance Ratio Control," *IEEE Int. Con/. Industrial ElectroniCS*, Vo1.3, pp.2009-2014, 1993.

[5]. Y. Ohba, M. Sazawa, K. Ohishi, T. Asai, K. Majima, Y. Yoshizawa, and K. Kageyama, "Sensorless Force Control for Injection Molding Machine Using Reaction Torque Observer Considering Torsion Phenomenon," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 8, pp. 2955– 2960, Aug. 2009.

[6]. K. Ohishi, "Realization of fine motion control based on disturbance observer," *in 10th IEEE International Workshop on Advanced Motion Control*, 2008. AMC '08, Mar. 2008, pp. 1–8.

[7]. K.-s. Kim, K.-H. Rew, and S. Kim, "Disturbance Observer for Estimating Higher Order Disturbances in Time Series Expansion," *IEEE Transactions on Automatic Control*, vol. 55, no. 8, pp. 1905–1911, Aug. 2010.

[8]. Y. Tsukamoto and C. Ishii, "Estimation of the grasping torque of robotic forceps using the robust reaction torque observer," in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Dec. 2014, pp. 1650–1655.

[9]. Y. Ohba, K. Ohishi, S. Katsura, Y. Yoshizawa, K. Kageyama, and K. Majima, "Sensor-less force control for injection molding machine using reaction torque observer," in IEEE International Conference on Industrial Technology, 2008. ICIT 2008, Apr. 2008, pp. 1–6.

[10]. Y. Ohba, S. Katsura, and K. Ohishi, "Sensor-less Force Control for Machine Tool Using Reaction Torque Observer," in *IEEE International Conference on Industrial Technology, 2006. ICIT 2006*, Dec. 2006, pp. 860–865.

[11]. E. Saito and S. Katsura, "A filter design method in disturbance observer for improvement of robustness against disturbance in time delay system," in *2012 IEEE International Symposium on Industrial Electronics (ISIE)*, May 2012, pp. 1650–1655.

[12]. J. Lei and G.-Y. Tang, "Optimal vibration control for active suspension sampled-data systems with actuator and sensor delays," in *10th International Conference on Control, Automation, Robotics and Vision, 2008. ICARCV 2008*, Dec. 2008, pp. 988–993.

[13]. K. Boo, J. Park, D. Lee, B. Cho, and S. Son, "Development of a vehicle height sensor for active suspension," in *International Conference on Control, Automation and Systems, 2007. ICCAS '07,* Oct. 2007, pp. 278– 282

[14]. M. van de Wal and B. de Jager, "Selection of sensors and actuators for an active suspension control problem," in , *Proceedings of the 1996 IEEE International Conference on Control Applications*, 1996, Sep. 1996, pp. 55–60.

[15]. W. Sun, Y. Zhao, J. Li, L. Zhang, and H. Gao, "Active Suspension Control With Frequency Band Constraints and Actuator Input Delay," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 1, pp. 530–537, Jan. 2012.

[16]. A. Chamseddine, H. Noura, and M. Ouladsine, "Sensor location for actuator fault diagnosis in vehicle active suspension," in *IEEE International Conference on Control Applications*, *2008. CCA 2008*, Sep. 2008, pp. 456–461.

[17]. Tongue, Benson, *Principles of Vibration*, Oxford University Press, 2001,Inman.

[18]. Daniel J., *Engineering Vibration*, Prentice Hall, 2001.

[19]. H. Li, Z. Gong, W. Lin, and T. Lippa, "A New Motion Control Approach for Jerk and Transient Vibration Suppression," in *2006 IEEE International Conference on Industrial Informatics*, Aug. 2006, pp. 676–681.

[20]. B. Chandekar and H. D. Lagdive, "Design of Electro-Hydraulic Active Suspension System for Four Wheel Vehicles," *International Journal of Emerging Technology and Advanced Engineering*, vol. 4, no. 4, pp. 885-889, 2014.

[21]. W. Evers, A. Teerhuis, A. van der Knaap, I. Besselink and H. Nijmeijer, " The electromechanical low-power active suspension: Modeling, control, and prototype testing," *Journal of Dynamic Systems, Measurement, and Control* , vol. 133, no. 4, p. 041008, 2011.

[22]. M. J. Thoresson, P. E. Uys, P. S. Els and J. A. Snyman, "Efficient optimisation of a vehicle suspension system, using a gradient-based approximation method, Part 1: Mathematical modelling," *Mathematical and Computer Modelling*, vol. 50, no. 9–10, p. 1421–1436, 2009.

[23]. Fischer D., and Isermann R., "Mechatronic Semi-Active and Active Vehicle Suspensions," *Control Engineering Practice*, vol. 12, no. 11, p. 1353–1367, 2004.

[24]. J. H. E. A. Muijderman, "Flexible Objective Controllers for Semi-Active Suspensions with Preview," Eindhoven University of Technology, Eindhoven, The Netherlands, 1997.

[25]. A. Conway and R. Stanway, "The Influence of Magneto-Rheological Fluid Time Response on the Performance of Semi-Active Automotive Suspensions," in *ASME 2006 International Mechanical Engineering Congress and Exposition*, Chicago, Illinois, USA, 2006.

[26]. J. Martins, I., Esteves, J., Marques, G. and Pina da Silva, F., "Permanent-Magnets Linear Actuators Applicability in Automobile Active Suspensions," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 1, pp. 86-94, 2006.

[27]. Gysen, B., van der Sande, T., Paulides, J. and Lomonova, E., "Efficiency of a Regenerative Direct-Drive Electromagnetic Active Suspension.," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 4, pp. 1384-1393, 2011.

[28]. Boo K., Park, J., Lee D., Cho B. and Son S., "Development of a vehicle height sensor for active suspension," in I*CCAS '07*, Seoul, 2007.

[29]. P. Intani, P. Boonwong, and C. Mitsantisuk, "Study on sensorless force control based on disturbance observer with friction force compensation," in *2013 13th International Conference on Control, Automation and Systems (ICCAS),* Oct. 2013, pp. 593–598.

[30]. S. Katsura, Y. Matsumoto, and K. Ohnishi, "Modeling of Force Sensing and Validation of Disturbance Observer for force Control," *IEEE Transactions on Industrial Electronics*, vol. 54, no. 1, pp. 530–538, Feb. 2007.

[31]. K. Iwazaki, K. Ohishi, Y. Yokokura, K. Kageyama, M. Takatsu, and S. Urushihara, "Robust pressure control of electric injection molding machine using automatic parameter switching reaction force observer," in *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, Nov. 2013, pp. 6557–6562.

[32]. E. Sariyildiz and K. Ohnishi, "An Adaptive Reaction Force Observer Design," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 2, pp. 750–760, Apr. 2015.

[33]. N.-V. Truong, "Mechanical parameter estimation of motion control systems," in 2012 4th International Conference on Intelligent and Advanced Systems (ICIAS), vol. 1, Jun. 2012, pp. 100–104.

[34]. E. Wood, "Prevention of the pathophysiologic effects of acceleration in humans: fundamentals and historic perspectives," *IEEE Engineering in Medicine and Biology Magazine*, vol. 10, no. 1, pp. 26–36, Mar. 1991.

[35]. R. Burton, L. Meeker, and J. Raddin, J.H., "Centrifuges for studying the effects of sustained acceleration on human physiology," *IEEE Engineering in Medicine and Biology Magazine*, vol. 10, no. 1, pp. 56–65, Mar. 1991.

[36]. G. Kjell and J. Lang, "Comparing different vibration tests proposed for li-ion batteries with vibration measurement in an electric vehicle," in *2013 World Electric Vehicle Symposium and Exhibition (EVS27),* Nov. 2013, pp. 1–11.

[37]. Y.-D. Yoon, E. Jung, and S.-K. Sul, "Application of a disturbance observer for a relative position control system," in *twenty-Third Annual IEEE Applied Power Electronics Conference and Exposition, 2008. APEC 2008*,Feb. 2008, pp. 377–382.

[38]. M. J. Griffin, *Handbook of Human Vibration*. Academic Press, Dec. 2012.

[39]. S. John, J. Pedro, and C. Pozna, "Enhanced slip control performance using nonlinear passive suspension system," in 2011 *IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*, Jul. 2011, pp. 277–282.

[40]. "Solid-Axle Suspension - History & Technical Specs - Rod and Custom Magazine." [Online]. Available: http://www.hotrod.com/how-to/additional-how-to/135-0312-solid-axle-front-suspension-details/

[41]. P. Christensen, "Division of Mechanics." [Online]. Available: http://www.mechanics.iei.liu.se/edu ug/tmme11/

[42]. "Front Suspensions." [Online]. Available: http://auto.howstuffworks.com/car-suspension.htm

[43]. "Suncore - Coil Spring Conversion Kit." [Online]. Available: http://www.carid.com/1990-toyota-celica-suspension-systems/suncore-air-spring-to-coil-spring-conversion-kit-14253862.html

[44]. "Audiophile hard drive." [Online]. Available: http://www.diyaudio.com/forums/pc-based/194098-audiophile-hard-drive.html

[45]. C. Liao, K. Wang, M. Yu, and W. Chen, "Modeling of Magnetorheological Fluid Damper Employing Recurrent Neural Networks," in *International Conference on Neural Networks and Brain, 2005*. ICNN B '05, vol. 2, Oct. 2005, pp. 616–620.

[46]. A. Ashfak, K. Abdul Rasheed, and J. Abdul Jaleel, "Modeling, simulation and experimental validation of Magneto-Rheological damper," in *2013 International Conference on Advanced Nanomaterials and Emerging Engineering Technologies (ICANMEET)*, Jul. 2013, pp. 267–274.

[47]. S. Choi, M. Gandhi, and B. Thompson, "An Active Vibration Tuning Methodology for Smart Flexible Structures Incorporating Electro- Rheological Fluids: A Proof-of-Concept Investigation," in *American Control Conference, 1989*, Jun. 1989, pp. 694–703.

[48]. G. Long-ming, S. Wen-ku, and L. Wei, "A semi-active suspension design for off-road vehicle base on Magneto-rheological technology," in *2012 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, May 2012, pp. 2565–2568.

[49]. L. H. Nguyen, S. Park, A. Turnip, and K.-S. Hong, "Modified skyhook control of a suspension system with hydraulic strut mount," in *ICCASSICE, 2009*, Aug. 2009, pp. 1347–1352.

[50]. L. Felix-Herran, R. Soto, J. Rodriguez-Ortiz, and R. Ramirez-Mendoza, "Fuzzy control for a semi-active vehicle suspension with a magnetorheological damper," in *Control Conference (ECC), 2009 European*, Aug. 2009, pp. 4398–4403.

[51]. L. H. Nguyen, S. Park, A. Turnip, and K.-S. Hong, "Application of LQR control theory to the design of modified skyhook control gains for semiactive suspension systems," in *ICCAS-SICE, 2009*, Aug. 2009, pp. 4698– 4703.

[52]. A. Ashari, "Sliding-mode control of active suspension systems: unit vector approach," in Proceedings of the *2004 IEEE International Conference on Control Applications*, 2004, vol. 1, Sep. 2004, pp. 370–375 Vol.1.

[53]. C.-P. Cheng, C.-H. Chao, and T. Li, "Design of observer-based fuzzy sliding-mode control for an active suspension system with full-car model," in *2010 IEEE International Conference on Systems Man and Cybernetics (SMC)*, Oct. 2010, pp. 1939–1944.

[54]. F. Kou and Z. Fang, "An Experimental Investigation into the Design of Vehicle Fuzzy Active Suspension," in *2007 IEEE International Conference on Automation and Logistics*, Aug. 2007, pp. 959–963.

[55]. c. andy, "Ride Analysis and Suspension Control," Nov. 2014. [Online]. Available: http://www.ni.com/white-paper/13019/en/

[56]. Chen Yi, Wang Zhong-Lai and Huang Hong-Zhong, "Hybrid Fuzzy Skyhook Surface Control Using Multi-Objective Microgenetic Algorithm for Semi-Active Vehicle Suspension System Ride Comfort Stability Analysis", *"Journal of Dynamic Systems, Measurement, and Control"*, June. 2012, issue 4, pp. 0022-0434

[57]. Chenghao Han and Dingxuan Zhao, "Multi-objective Static Output Feedback Control for Vehicle Active Suspension", *Proceedings of 2014 IEEE International Conference on Mechatronics and Automation* , Aug. 2014, pp.1526-1532.

[58]. Balazs Nemeth and Peter Gaspar, "Variable-Geometry Suspension Design in Driver Assistance Systems", *2013 European Control Conference (ECC)* , Jul. 2013, pp.1481-1486.

[59]. N. Berg, R. Holm, and P. Rasmussen, "A novel magnetic lead screw active suspension system for vehicles," in *2014 IEEE Energy Conversion Congress and Exposition (ECCE),* Sep. 2014, pp. 3139–3146.

[60]. B. Gysen, J. Paulides, J. Janssen, and E. Lomonova, "Active electromagnetic suspension system for improved vehicle dynamics," in *IEEE Vehicle Power and Propulsion Conference, 2008. VPPC '08*, Sep. 2008, pp. 1–6.

[61]. A. K. S. Ehsan Sarshari, "Selection of Sensors for Hydro-Active Suspension System of Passenger Car With InputOutput Pairing Considerations," *Journal of Dynamic Systems, Measurement, and Control*, vol. 135, no. 1, p. 011004, 2012.

[62]. Bruel and Kjaer, *Measuring Vibration*. Bruel and Kjaer. [Online].Available: https://books.google.lk/books?id=HPoOrgEACAAJ

[63]. J. Lei and G.-Y. Tang, "Optimal vibration control for active suspension sampled-data systems with actuator and sensor delays," in *10th International Conference on Control, Automation, Robotics and Vision, 2008. ICARCV 2008*, Dec. 2008, pp. 988–993.

[64]. ——, "Optimal vibration control for active suspension systems with actuator and sensor delays," in *IEEE International Conference on Systems, Man and Cybernetics, 2008*. SMC 2008, Oct. 2008, pp. 2828–2833.

[65]. Katsuhiko Ogata (2006) Modern Control Engineering. Prentice Hall of India Private Limited, India, (Fourth Edition).
[66]. Sabri Cetinkunt (2007) *Mechatronics*. John Wiley and sons; Inc., USA, (First Edition).
[67]. Wahyudi, (2003) "Friction Identification and Compensation for High Precision Motion Control Sysyem – Part 1: Friction Identification", *Processing of industrial Electronics Seminar (IES).*
[68]. D.P Castle, J.P Bobis (1992). "Velocity model reference adaptive control of the conventional permanent magnet and variable flux DC motors". *Proceeding of the IEEE International Conference on Industrial Electronics, Control, Instrumentation, and Automation*, November, 1992, Australia, 1303-1308.
[69]. C. Jeram, "Cylindrical-shaft linear motors." [Online]. Available: http://machinedesign.com/technologies/cylindrical-shaft-linear-motors
[70]. "Linear Shaft Motor Catalog." [Online]. Available: http://nipponpulse. com/catalog/document/4e1f2c0a01a0f lsm.pdf
[71]. "Nippon Pulse America Technical Notes -," Apr. 2011. [Online]. Available: http://www.electromate.com/news/?c=featurearticles&article id=10180
[72]. "Nippon pulse s250t datasheet." [Online]. Available: http://nipponpulse. com/catalog/document/554d17e86e0a4 S250.pdf
[73]. "Terasic Altera DE1-SOC." [Online]. Available: http://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English& CategoryNo=165&No=836&PartNo=1
[74]. "mbed NXP LPC1768 Getting Started." [Online]. Available: https: //developer.mbed.org/handbook/mbed-NXP-LPC1768-Getting-Started
[75]. "About Servoland." [Online]. Available: http://www.servoland.co.jp/en/
[76]. "Servoland SVF Series (Release version C or later)." [Online]. Available: http://www.servoland.co.jp/reference/SV51-11-en.pdf
[77]. "LM10 LINEAR MAGNETIC ENCODER." [Online]. Available: http://www.rls.si/lm10-linear-magnetic-encoder-system
[78]. "ADXL345 Digital Accelerometer." [Online]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf
[79]. "linear guide - All number - SKF FAG NSK NTN ZWZ LYC TIMKENand so on (China Manufacturer) - Industrial Supplies Stocks – Industrial Supplies." [Online]. Available: http%3A%2F%2Fwww.diytrade.com%2Fchina%2Fpd%2F7588965%2Flinear guide.html
[80]. "Flanged Linear Motion Ball Bearing(LMH Series)." [Online]. Available: http://www.seekpart.com/product/Flanged-Linear-Motion-Ball-Bearing-LMH-Series-1594180.html
[81]. J. Chamberlain, "The difference between linear motors and linear mechanical devices." [Online]. Available: http://www.seekpart.com/product/Flanged-Linear-Motion-Ball-Bearing-LMH-Series-1594180.html
[82]. "Nippon Pulse S250 Linear Shaft Motor." [Online]. Available:http://nipponpulse.com/catalog/document/554d17e86e0a4 S250.pdf
[83]. "Hybrid Hall Effect ICs EW-series." [Online]. Available: http: //www.akm.com/akm/en/file/datasheet/EW-500.pdf
[84]. "Motor Control Algorithms." [Online]. Available: http://in.renesas.com/applications/key technology/motor control/motor algorithms/child/bldc.jsp

9 APPENDIX

9.1 Equipment Details

9.1.1 Linear motor

Motor specifications

Linear Motor [72]

Continuous Force	60N (13.5lbs)
Continuous Current	1.3Arms
Acceleration Force	240N (54.0lbs)
Acceleration Current	5.1Arms
Force Constant (Kf)	47N/Arms (10.67lbs/amp)
Back EMF (Ke)	16V/m/s
Resistance 25°C	12Ω
Inductance	15mH
Electric Time Constant	1.25ms
Rated Voltage (AC)	240V
Fundamental Motor Constant (Km)	11.19N√W
Magnetic Pitch (North-North)	90mm (3.54lbs)
Usable stroke length	250mm
Shaft length	515mm
Shaft mass	1.6kg

Dimensions of selected motor as shown in the figure 6.5,

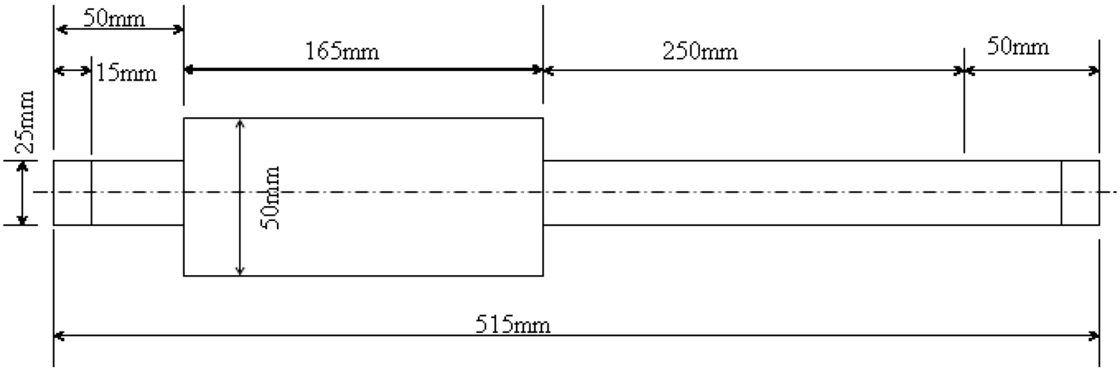
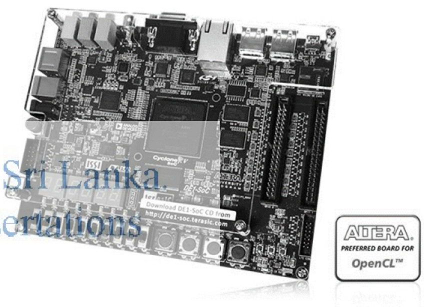


Figure 9.2: Motor Dimensions

9.1.2 FPGA



Terasic Altera DE1-SOC [73]

Altera Terasic DE1 Soc field programmable gate array is used as the controller to provid high speed processing with following specifications;

Cyclone V SoC 5CSEMA5F31C6 Device

85K Programmable Logic Elements

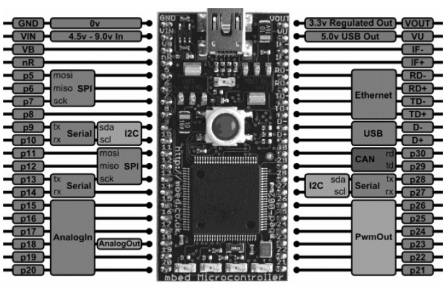
4,450 Kbits embedded memory

6 Fractional PLLs

This controller provides 100MHz sampling and processing with a minimum amount of latency. Controller desing is included in the appendix.

9.1.3 Mbed Microcontroller

Mbed NXP LPC1768 microcontroller is used as secondary controller to control cam assembly. Mbed microcontroller is taking input signals mainly from accelerometer and rotary encoder and giving output PWM signals to rotary motor driver.



Mbed Microcontroller[74]

Mbed 32-bit ARM Cortex-M3 core is running at 96MHz. It includes 512KB FLASH, 32KB RAM and lots of interfaces including built-in Ethernet, USB Host and Device, CAN, SPI, I2C, ADC, DAC, PWM and other I/O interfaces.
NXP LPC1768 MCU

- High performance ARM® Cortex™-M3 Core
- 96MHz, 32KB RAM, 512KB FLAS
- Ethernet, USB Host/Device, 2xSPI, 2xI2C, 3xUART, CAN, 6xPWM, 6xADC, GPIO

Prototyping form-factor

- 40-pin 0.1" pitch DIP package, 54x26mm
- 5V USB or 4.5-9V supply
- Built-in USB drag 'n' drop FLASH programmer

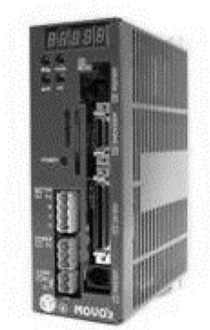
Mbed.org Developer Website

- Lightweight Online Compiler
- High level C/C++ SDK
- Cookbook of published libraries and projects

9.1.3.1.1 Linear Motor Driver

Linear Motor Driver is one of the key components of the system. It communicates with host controller (FPGA) and control linear motor according to signals. SVFH3-H3-DSP*SRI Servo land linear motor driver is selected as linear motor driver. It is capable to control motor current of the linear motor. Serial 2 phase current command type was added for the driver for that purpose.

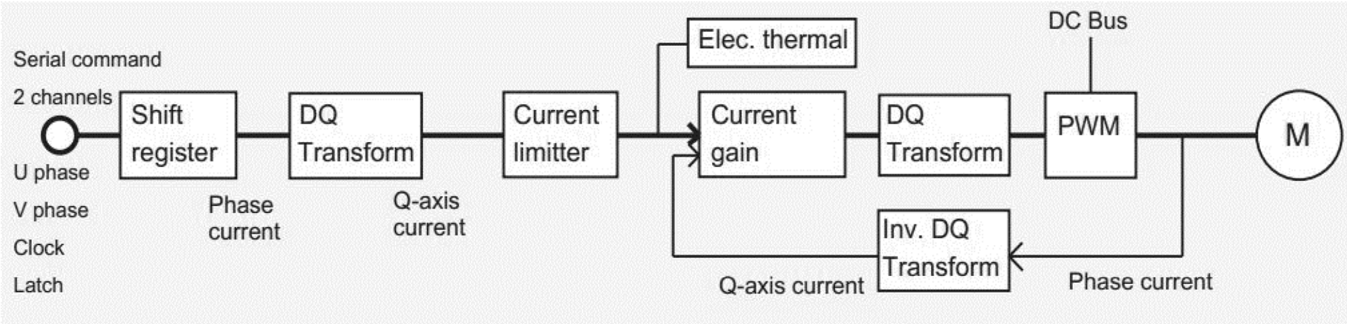
Principal specifications



Linear Motor Driver [75]

- Motion mode - SRI Serial 2 phase current command mode
- Analog input port - 2 channels, Differential / Single end
- Input command data - 16 bit
- Control type - DQ Vector control
- A/D Resolution of current detection - 16 bit fast sigma-delta type
- PWM frequency - 15 to 40 kHz
- PWM Resolution - 7 ns
- Current control cycle - Synchronized with PWM frequency
- Current response - 3 kHz (DSP model), 1.5 kHz (Base model)
- Output - 436W

Internal block diagram of the motor driver is as shown in the figure 6.9,

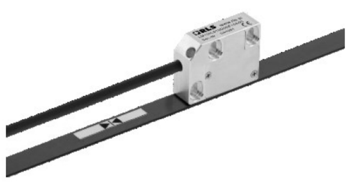


Internal Block Diagram of Linear Motor [76]

9.1.3.1.2 Linear Encoder

Linear encoder is the main and the only sensor which take measurements for controlling the system. Linear encoder signals are directly connected with FPGA. From the linear Encoder, relative displacement between sprung mass and un-sprung mass is measured. LM10 magnetic type linear encoder is selected from Renishaw Company for the application. System data

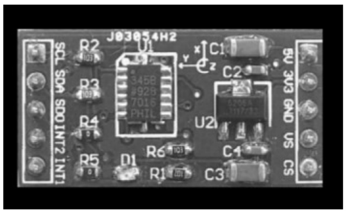
- Maximum length for MS scale 50 m (100 m special order)
- Pole length 2 mm
- Available resolutions 1 µm, 2 µm, 5 µm, 10 µm, 20 µm and 50 µm
- Sinusoidal period length 2 mm
- Maximum speed For a Precision class for MS scales ±20 µm/m and ±40 µm/m
- Linear expansion coefficient for MS scale ~ 17 × 10-6/K
- Repeatability Better than unit of resolution for movement in the same direction
- Hysteresis < 3 µm up to 0.5 mm ride height
- Sub divisional error ±3.5 µm for < 0.7 mm ride height ±7.5 µm for 1 mm ride height



Linear Encoder [77]

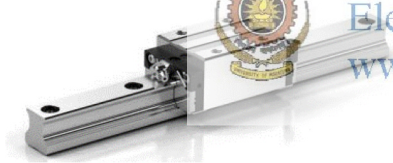
9.1.3.1.3 Accelerometer

Accelerometer is used for data collection purposes. Accelerometer is connected to Mbed microcontroller. For that ADXL345 3-Axis digital accelerometer is used. The ADXL345 is a small, thin, ultralow power, 3-axis accelerometer with high resolution (13-bit) measurement at up to ±16 g. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI (3- or 4-wire) or I^2C digital interface.



Accelerometer [78]

9.1.4 Mechanical equipment



Linear Guide [79]

Linear Bearings [80]

9.2 Main module code

This only includes the main module code of one version of implementation due to limitations of pages.

```
module ACTIVE_MODULE(

        ////////////// CLOCK //////////

        input                       CLOCK_50,

        input                       CLOCK2_50,

        input                       CLOCK3_50,

        input                       CLOCK4_50,

        output          [6:0]       HEX5,


        ////////////// KEY //////////
```

```
        input                 [3:0]           KEY,
        //////////// LED //////////
        output                [9:0]           LEDR,
        //////////// SW //////////
        input                 [9:0]           SW,
        //////////// GPIO_0, GPIO_0 connect to GPIO Default //////////
        inout                 [35:0]          GPIO_0,
        //////////// GPIO_1, GPIO_1 connect to GPIO Default //////////
        inout                 [35:0]          GPIO_1
); //=======================================================
//  REG/WIRE declarations
//=======================================================
wire res;   //async reset
reg reset = 0;                //sync reset
wire enable;          //enable
//pll pins
reg pll_reset = 0;            //oustside clock sync pll_reset
wire [4:0] pll_cascade;
wire pll_lock;
wire pll_lock_2;
wire CLOCK_1000;
wire CLOCK_500;
wire CLOCK_200;
wire CLOCK_100;
wire CLOCK_50_PLL;
wire CLOCK_40;
wire CLOCK_4;

//reg input Key edge detection
reg [3:0] KEY_prev = 0;
wire [3:0] KEY_edge;
//led register for sync led out
reg [9:0] LED_OUT = 9'b000000000;
//input switch logics
wire SW_12_logic_00;
```

```
wire SW_12_logic_01;
wire SW_12_logic_11;
wire SW_12_logic_10;
wire SW_345_logic_000 ;
wire SW_345_logic_001;
wire SW_345_logic_010;
wire SW_345_logic_011 ;
wire SW_345_logic_100;
wire SW_345_logic_101 ;
wire SW_345_logic_110 ;
wire SW_345_logic_111;
wire SW_6_logic_0;
wire SW_6_logic_1;
//register read SPIO logic
wire GPIO1_1357_logic_0000;
wire GPIO1_1357_logic_0001;
wire GPIO1_1357_logic_0010;
wire GPIO1_1357_logic_0011;
wire GPIO1_1357_logic_0100;
wire GPIO1_1357_logic_0101;
wire GPIO1_1357_logic_0110;
wire GPIO1_1357_logic_0111;
wire GPIO1_1357_logic_1000;
wire GPIO1_1357_logic_1001;
wire GPIO1_1357_logic_1010;
wire GPIO1_1357_logic_1011;
wire GPIO1_1357_logic_1100;
wire GPIO1_1357_logic_1101;
wire GPIO1_1357_logic_1110;
wire GPIO1_1357_logic_1111;
//wires for mbed spi slave GPIO connections
wire MBED_SCLK;
wire MBED_CS;
wire MBED_MOSI;
wire MBED_MISO;
```

```
wire MBED_READY;
wire MBED_FINISH;
wire MBED_BUSY;
reg [15:0]  MBED_DATAOUT = 16'b0;
wire [15:0] MBED_DATAIN;
reg [15:0] MBED_READ_REG = 16'b0;
reg [15:0] MBED_READ_REG_HIGH = 16'b0;
//adc read values
reg [15:0] ADC_DIN_VALUE = 16'b10_00_00_110_000_0000;
wire [15:0] ADC_DOUT_VALUE;
//input wires for movo drive GPIOs
wire movo_data_A;
wire movo_data_B;
wire movo_clk;
wire movo_latch;
wire movo_data_A_not;
wire movo_data_B_not;
wire movo_clk_not;
wire movo_latch_not;
wire data_status ;
//BLDC commutation
wire hall_1;
wire hall_2;
wire hall_3;
wire [15:0] current_U;
wire [15:0] current_V;
wire hall_error;
//quadrature encoder interface
wire encoder_phaseA;
wire encoder_phaseB;
wire [31:0] pulse_position;
wire pulse;
//force output to send to the drivers
reg [31:0] forceOutFloat = 0;
//calculated float values
```

```
wire [31:0] position;
wire [31:0] acceleration;
wire [31:0] velocity;
wire[31:0] Kspring;
wire[31:0] Cdamper;
wire[31:0] Kp;
wire[31:0] Ki;
wire[31:0] Kd;
wire[31:0] mass;
wire[31:0] forceCurrentSense;
wire[31:0] forceRef;
wire[15:0] currentOutDAC;
wire[15:0] currentOutMOVO;
reg [31:0] forceRef_in = 0;
wire[31:0] forceRFOB;
wire[31:0] forceOutRFOB_act;
wire[31:0] forceOutSpring;
//raw integers of values
reg [15:0] forceRefRaw = 0;
reg [15:0] KpRaw = 950;
reg [15:0] KiRaw = 0;
reg [15:0] KdRaw = 0;
reg [15:0] massRaw = 110;
reg [15:0] KspringRaw = 1200;
reg [15:0] CdamperRaw = 0;
assign res = ~KEY[0];
assign enable = SW[0] & pll_lock;
// LED indicator assignmene
assign LEDR = LED_OUT;
//mbed spi slave conncector ---check input pin filters
assign GPIO_1[2] = MBED_MISO;           //output
//movo communication pins
assign GPIO_0[0] = movo_data_A;
assign GPIO_0[2] = movo_data_B;
assign GPIO_0[4] = movo_clk;
```

```
assign GPIO_0[6] = movo_latch;
assign GPIO_0[1] = movo_data_A_not;
assign GPIO_0[3] = movo_data_B_not;
assign GPIO_0[5] = movo_clk_not;
assign GPIO_0[7] = movo_latch_not;
assign GPIO_0[8] = 1'b0;
assign GPIO_0[9] = 1'b0;
//check input filters
//DAC 712 ouput pins
assign GPIO_1[25:10] = currentOutDAC[15:0];
// positive edge detection of the input keys
assign KEY_edge[0] = ~KEY_prev[0] & KEY[0];
assign KEY_edge[1] = ~KEY_prev[1] & KEY[1];
assign KEY_edge[2] = ~KEY_prev[2] & KEY[2];
assign KEY_edge[3] = ~KEY_prev[3] & KEY[3];
//SW 0-9 logic connectors
assign SW_12_logic_00 = ~SW[1] & ~SW[2];
assign SW_12_logic_01 = ~SW[1] & SW[2];
assign SW_12_logic_11 = SW[1] & SW[2];
assign SW_12_logic_10 = SW[1] & ~SW[2];
assign SW_345_logic_000 = ~SW[3] & ~SW[4] & ~SW[5];
assign SW_345_logic_001 = ~SW[3] & ~SW[4] & SW[5];
assign SW_345_logic_010 = ~SW[3] & SW[4] & ~SW[5];
assign SW_345_logic_011 = ~SW[3] & SW[4] & SW[5];
assign SW_345_logic_100 = SW[3] & ~SW[4] & ~SW[5];
assign SW_345_logic_101 = SW[3] & ~SW[4] & SW[5];
assign SW_345_logic_110 = SW[3] & SW[4] & ~SW[5];
assign SW_345_logic_111 = SW[3] & SW[4] & SW[5];
assign SW_6_logic_0 = ~SW[6];
assign SW_6_logic_1 = SW[6];
//GPIO_1 [1-7] Mbed register read connection
assign GPIO1_1357_logic_0000 = ~GPIO_1[1] & ~GPIO_1[3] & ~GPIO_1[5] & ~GPIO_1[7];
assign GPIO1_1357_logic_0001 = ~GPIO_1[1] & ~GPIO_1[3] & ~GPIO_1[5] & GPIO_1[7];
assign GPIO1_1357_logic_0010 = ~GPIO_1[1] & ~GPIO_1[3] & GPIO_1[5] & ~GPIO_1[7];
assign GPIO1_1357_logic_0011 = ~GPIO_1[1] & ~GPIO_1[3] & GPIO_1[5] & GPIO_1[7];
```

```
assign GPIO1_1357_logic_0100 = ~GPIO_1[1] & GPIO_1[3] & ~GPIO_1[5] & ~GPIO_1[7];
assign GPIO1_1357_logic_0101 = ~GPIO_1[1] & GPIO_1[3] & ~GPIO_1[5] & GPIO_1[7];
assign GPIO1_1357_logic_0110 = ~GPIO_1[1] & GPIO_1[3] & GPIO_1[5] & ~GPIO_1[7];
assign GPIO1_1357_logic_0111 = ~GPIO_1[1] & GPIO_1[3] & GPIO_1[5] & GPIO_1[7];
assign GPIO1_1357_logic_1000 = GPIO_1[1] & ~GPIO_1[3] & ~GPIO_1[5] & ~GPIO_1[7];
assign GPIO1_1357_logic_1001 = GPIO_1[1] & ~GPIO_1[3] & ~GPIO_1[5] & GPIO_1[7];
assign GPIO1_1357_logic_1010 = GPIO_1[1] & ~GPIO_1[3] & GPIO_1[5] & ~GPIO_1[7];
assign GPIO1_1357_logic_1011 = GPIO_1[1] & ~GPIO_1[3] & GPIO_1[5] & GPIO_1[7];
assign GPIO1_1357_logic_1100 = GPIO_1[1] & GPIO_1[3] & ~GPIO_1[5] & ~GPIO_1[7];
assign GPIO1_1357_logic_1101 = GPIO_1[1] & GPIO_1[3] & ~GPIO_1[5] & GPIO_1[7];
assign GPIO1_1357_logic_1110 = GPIO_1[1] & GPIO_1[3] & GPIO_1[5] & ~GPIO_1[7];
assign GPIO1_1357_logic_1111 = GPIO_1[1] & GPIO_1[3] & GPIO_1[5] & GPIO_1[7];
//external clock sync reset
always @(posedge CLOCK_50) begin
        pll_reset <= ~KEY[0] & ~KEY[3];
end
//logic
always @(posedge CLOCK_200) begin
        reset <= res;
        KEY_prev[3:0] <= KEY[3:0];
        LED_OUT[0] <= enable;
        LED_OUT[1] <= reset;
        LED_OUT[6:2] <= SW[5:1];
        LED_OUT[7] <= hall_error;
        LED_OUT[8] <= pulse;
        LED_OUT[9] <= data_status;

        //LED_OUT = MBED_DATAIN[9:0];
        if(reset) begin
                forceRef_in <= 0;
                forceOutFloat <= 0;
                forceRefRaw <= 0;
                KpRaw <= 995;
                KiRaw <= 0;
                KdRaw <= 0;
```

```
                KspringRaw <= 1200;
                CdamperRaw <= 1;
                forceOutFloat <= 0;
        end
end
// observations of values using mbed SPI
always @ (posedge CLOCK_200)        begin
        if (reset)begin
                MBED_DATAOUT <= 16'b0;
                MBED_READ_REG <= 16'b0;
                MBED_READ_REG_HIGH <= 16'b0;
        end
        else begin
                if (enable)begin
                        if(MBED_CS) begin
                                MBED_DATAOUT <= MBED_READ_REG;
                                if ( GPIO1_1357_logic_0000 ) begin
                                        MBED_DATAOUT <= pulse_position[15:0];
                                        MBED_READ_REG_HIGH <= pulse_position[31:16];
                                end
                                if ( GPIO1_1357_logic_0001 ) begin
                                        MBED_DATAOUT <= velocity[15:0];
                                        MBED_READ_REG_HIGH <= velocity[31:16];
                                end
                                if ( GPIO1_1357_logic_0010 ) begin
                                        MBED_DATAOUT <= forceRFOB[15:0];
                                        MBED_READ_REG_HIGH <= forceRFOB[31:16];
                                end
                                if ( GPIO1_1357_logic_0011 ) begin
                                        MBED_DATAOUT <= forceOutRFOB_act[15:0];
                                        MBED_READ_REG_HIGH <= forceOutRFOB_act[31:16];
                                end
                                if ( GPIO1_1357_logic_0100 ) begin
                                        MBED_DATAOUT <= forceOutSpring[15:0];
                                        MBED_READ_REG_HIGH <= forceOutSpring[31:16];
```

```
end
if ( GPIO1_1357_logic_0101 ) begin
        MBED_DATAOUT <= Kspring[15:0];
        MBED_READ_REG_HIGH <= Kspring[31:16];
end
if ( GPIO1_1357_logic_0110 ) begin
        MBED_DATAOUT <= Kp[15:0];
        MBED_READ_REG_HIGH <= Kp[31:16];
end
if ( GPIO1_1357_logic_0111 ) begin
        MBED_DATAOUT <= MBED_READ_REG_HIGH;
end
if ( GPIO1_1357_logic_1000 ) begin
        MBED_DATAOUT <= currentOutDAC;
        MBED_READ_REG_HIGH <= 16'b0;
end
if ( GPIO1_1357_logic_1001 ) begin
        MBED_DATAOUT <= ADC_DOUT_VALUE;
        MBED_READ_REG_HIGH <= 16'b0;
end
if ( GPIO1_1357_logic_1010 ) begin
        MBED_DATAOUT <= KpRaw;
        MBED_READ_REG_HIGH <= 16'b0;
end
if ( GPIO1_1357_logic_1011 ) begin
        MBED_DATAOUT <= KdRaw;
        MBED_READ_REG_HIGH <= 16'b0;
end
if ( GPIO1_1357_logic_1100 ) begin
        MBED_DATAOUT <= KiRaw;
        MBED_READ_REG_HIGH <= 16'b0;
end
if ( GPIO1_1357_logic_1101 ) begin
        MBED_DATAOUT <= KspringRaw;
        MBED_READ_REG_HIGH <= 16'b0;
```

```
                                        end
                                        if ( GPIO1_1357_logic_1110 ) begin
                                                MBED_READ_REG <= CdamperRaw;
                                                MBED_READ_REG_HIGH <= 16'b0;
                                        end
                                        if ( GPIO1_1357_logic_1111 ) begin
                                                MBED_READ_REG <= forceRefRaw;
                                                //MBED_DATAOUT <= MBED_DATAIN;
                                                MBED_READ_REG_HIGH <= 16'b0;
                                        end
                                end
                        end
                        else begin
                        end
                end
end
//input pin filters
/*
assign MBED_SCLK =              GPIO_0[24]; //input
assign MBED_CS     =                    GPIO_0[26];             //input
assign MBED_MOSI =              GPIO_0[20];             //input
*/
filter_1_bit filter_1_bitMBED_SCLK(
        CLOCK_500,
        GPIO_1[4],
        MBED_SCLK
  );
filter_1_bit filter_1_bitMBED_CS(
        CLOCK_500,
        GPIO_1[6],
        MBED_CS
  );
filter_1_bit filter_1_bitMBED_MOSI(
        CLOCK_500,
        GPIO_1[0],
```

```
            MBED_MOSI
    );
/*
assign hall_1 = GPIO_0[10];
assign hall_2 = GPIO_0[12];
assign hall_3 = GPIO_0[14];
*/
filter_1_bit filter_1_bithall_1(
            CLOCK_200,
            GPIO_0[30],
            hall_1
    );
filter_1_bit filter_1_bithall_2(
            CLOCK_200,
            GPIO_0[32],
            hall_2
    );
filter_1_bit filter_1_bithall_3(
            CLOCK_200,
            GPIO_0[34],
            hall_3
    );
/*assign encoder_phaseA = GPIO_0[17];
assign encoder_phaseB = GPIO_0[19];
*/
filter_1_bit filter_1_bitencoder_phaseA(
            CLOCK_200,
            GPIO_0[27],
            encoder_phaseA
    );
filter_1_bit filter_1_bitencoder_phaseB(
            CLOCK_200,
            GPIO_0[29],
            encoder_phaseB
    );
```

```
//PLL connection
//module pll_main (    refclk,   rst, 50,40,4, locked );
pll_main pll_main_inst(
                CLOCK_50,
                pll_reset,
                CLOCK_50_PLL,
                CLOCK_200,
                CLOCK_500,
                pll_lock
);
//spi 16bit slave to tranfer data to mbed
spi_slave mbed_spi(
        CLOCK_500,
        reset,
        enable,
        MBED_SCLK,
        MBED_CS,
        MBED_MOSI,
        MBED_DATAOUT,
        MBED_MISO,
        MBED_DATAIN,
        MBED_READY,
        MBED_FINISH,
        MBED_BUSY
);
//encoder interface
quadrature_encoder quadrature_encoder_inst(
        .clk(CLOCK_200),
        .rst(reset),
        .A(encoder_phaseA),
        .B(encoder_phaseB),
        .up(),
        .down(),
        .pulse(pulse),
        .direction(),
```

```
            .pulse_count(pulse_position)
    );
//data converter module
data_converter data_converter_inst(
            .clk(CLOCK_200),
            .rst(reset),
            .enable(enable),

            .positionRaw(pulse_position),
            .accelerationRaw(MBED_DATAIN),
            .KspringRaw(KspringRaw),
            .CdamperRaw(CdamperRaw),
            .KpRaw(KpRaw),
            .KiRaw(KiRaw),
            .KdRaw(KdRaw),
            .massRaw(massRaw),
            .senseCurrentRaw(ADC_DOUT_VALUE),
            .forceRefRaw(forceRefRaw),
            .forceFloat(forceOutFloat),
            .position(position),
            .velocity(velocity),
            .acceleration(acceleration),
            .Kspring(Kspring),
            .Cdamper(Cdamper),
            .Kp(Kp),
            .Ki(Ki),
            .Kd(Kd),
            .mass(mass),
            .forceCurrentSense(forceCurrentSense),
            .forceRef(forceRef),
            .currentOutDAC(currentOutDAC),
            .currentOutMOVO(currentOutMOVO)
    );
//active RFOB connection
activeRFOB activeRTOB_inst(
```

```
            .clk(CLOCK_200),
            .rst(reset),
            .enable(enable),
            .acceleration(acceleration),
            .position(position),
            .velocity(velocity),
            .senseForce(forceCurrentSense),
            .Kp(Kp),
            .Ki(Ki),
            .Kd(Kd),
            .mass(mass),
            .forceRef(forceRef_in),
            .forceRFOB(forceRFOB),
            .forceOutFloat(forceOutRFOB_act)
    );
//active spring damper connection
activeVirtualSpring activeVirtualSpring_inst(
            .clk(CLOCK_200),
            .rst(reset),
            .enable(enable),
            .position(position),
            .velocity(velocity),
            .K(Kspring),
            .C(Cdamper),
            .forceOutFloat(forceOutSpring)
    );
//BLDC commutation
BLDC_commutation BLDC_commutation_inst(
            .clk(CLOCK_200),
            .rst(reset),
            .enable(enable),
            .hall_1(hall_1),
            .hall_2(hall_2),
            .hall_3(hall_3),
            .current_in(currentOutMOVO),
```

```
            .current_out_U(current_U),
            .current_out_V(current_V),
            .hall_error(hall_error)
    );

//Movo interface
movo_interface movo_interface_inst(
            .clk(CLOCK_50_PLL),
            .rst(reset),
            .enable(enable),
            .value_A(current_U),
            .value_B(current_V),
            .clk_movo(movo_clk),
            .clk_movo_not(movo_clk_not),
            .data_A(movo_data_A),
            .data_A_not(movo_data_A_not),
            .data_B(movo_data_B),
            .data_B_not(movo_data_B_not),
            .latch(movo_latch),
            .latch_not(movo_latch_not),
            .status(),
            .data_status(data_status)
            );
endmodule
```