REFERENCES

[1] Rogers, J. and Plett, C., 2010, *Radio Frequency Integrated Circuit Design*, 2nd edition, Boston: Artech House.

[2] Huang, J.F., Liu, R.Y. and Hong, P.S., 2006. An ISM band CMOS power amplifier design for WLAN. *International Journal of Electronics and Communications,* Vol. 60, No. 7, pp. 533-538.

[3] Ho, K.W. and Luong, H.C., 2003. A 1-V CMOS Power Amplifier for Bluetooth Applications. *IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing,* Vol. 50, No. 8, pp. 445-449.

[4] Miliozzi, P., Kundert, K., Lampaert, K., Good, P. and Chian, M., 2000. A design system for RFIC: Challenges and solutions. *Proceedings of the IEEE,* Vol. 88, No. 10, pp. 1613-1631.

[5] Poulin, D., 2009. The III-V vs. silicon battle. *Microwave Journal,* Vol. 52, No. 4, pp. 22-38.

[6] Avenier, G., Diop, M., Chevalier, P., Troillard, G., Loubet, N., Bouvier, J., Depoyan, L., Derrier, N., Buczko, M., Leyris, C., Boret, S., Montusclat, S., Margain, A., Pruvost, S., Nicolson, S.T., Yau, H.K.Y., Revil, N., Gloria, D., Dutartre, D., Voinigescu, S.P. and Chantre, A., 2009. 0.13 µm SiGe BiCMOS Technology Fully Dedicated to mm-Wave Applications. *IEEE Journal of Solid-State Circuits,* Vol. 44, No. 9, pp. 2312-2321.

[7] Naudé, N., Božanić, M. and Sinha, S., 2006. Analogue CMOS direct sequence spread spectrum transceiver with carrier recovery employing complex spreading sequences, *Proceedings: IEEE Mediterranean Electrotechnical Conference*, Malaga, 16-19 May, pp. 1227-1230.

[8] Kazimierczuk, M.K., 2008, *RF Power Amplifiers*, 1st edition, Chichester: John Wiley & Sons.

[9] Niknejad, A.M. and Meyer, R.G., 2002, *Design, Simulation and Application of Inductors and Transformers for Si RF ICs*, 2nd edition, Boston: Kluwer Academic Publishers.

[10] Yue, C.P. and Wong, S.S., 2000. Physical Modeling of Spiral Inductors on Silicon. *IEEE Transactions on Electron Devices,* Vol. 47, No. 3, pp. 560-568.

[11] Gao, W. and Yu, Z., 2006. Scalable Compact Circuit Model and Synthesis for RF CMOS Spiral Inductors. *IEEE Transactions on Microwave Theory and Techniques,* Vol. 54, No. 3, pp. 1055-1064.

[12] Lee, K.Y., Mohammadi, S., Bhattacharya, P.K. and Katehi, L.P.B., 2006. Compact Models Based on Transmission-Line Concept for Integrated Capacitors and Inductors. *IEEE Transactions on Microwave Theory and Techniques,* Vol. 54, No. 12, pp. 4141-4148.

[13] Anon., 2011. Austriamicrosystems AG [Online]. Available from: http://www.austriamicrosystems.com/ [Accessed: 24 February 2011].

[14] Anon., 2011. IBM [Online]. Available from: http://www.ibm.com/ [Accessed: 24 February 2011].

[15] Roberts, G.W. and Sedra, A.S., 1997, *SPICE*, 2nd edition, Oxford: Oxford University Press.

[16] Anon., 2011. The MathWorks - MATLAB and Simulink for Technical Computing [Online]. Available from: http://www.mathworks.com/ [Accessed: 24 February 2011].

[17] Anon., 2011. Cadence Design Systems [Online]. Available from: http://www.cadence.com/ [Accessed: 24 February 2011].

[18] Weststrate, M., 2009. Analysis of a Low Noise Amplifier with LC-Ladder Matching and Capacitive Shunt-Shunt Feedback*, Proceedings: 8th IEEE Africon*, Nairobi, 23-25 September, pp 1-6.

[19] Wibben, J. and Harjani, R., 2008. A High-Efficiency DC–DC Converter Using 2 nH Integrated Inductors. *IEEE Journal of Solid State Circuits,* Vol. 43, No. 4, pp. 844-854.

[20] Opperman, T.A.K. and Sinha, S., 2008. A 5 GHz BiCMOS I/Q VCO with 360° Variable Phase Outputs Using the Vector Sum Method*, Proceedings: IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Cannes, 15-18 September, pp. 1-5.

[21] Lie, D.Y.C., Lopez, J., Popp, J.D., Rowland, J.F., Wang, G., Qin, G. and Ma, Z., 2009. Highly Efficient Monolithic Class E SiGe Power Amplifier Design at 900 and 2400 MHz. *IEEE Transactions on Circuit and Systems — I: Regular Papers,* Vol. 56, No. 7, pp. 1455-1466.

[22] Ramos, J., Francken, K., Gielen, G.G.E. and Steyaert, M.S.J., 2005. An Efficient, Fully Parasitic-Aware Power Amplifier Design Optimization Tool. *IEEE Transactions on Circuits and Systems—I: Regular Papers,* Vol. 52, No. 8, pp. 1526-1534.

[23] Walling, J.S., Lakdawala, H., Palaskas, Y., Ravi, A., Degani, O., Soumyanath, K. and Allstot, D.J., 2009. A Class-E PA With Pulse-Width and Pulse-Position Modulation in 65 nm CMOS. *IEEE Journal of Solid-State Circuits,* Vol. 44, No. 6, pp. 1668-1678.

[24] Carls, J., Ellinger, F., Joerges, U. and Krcmar, M., 2009. Highly-efficient CMOS C-band class-F power amplifier for low supply voltages. *Electronic Letters,* Vol. 45, No. 24, pp. 1240-1241.

[25] Post., J.E., 2000. Optimizing the Design of Spiral Inductors on Silicon. *Transactions on Circuits and Systems—II: Analog and Digital Signal Processing,* Vol. 47, No. 1, pp. 15-17.

[26] Talwalkar, N.A., Yue, C.P. and Wong, S.S., 2005. Analysis and Synthesis of On-Chip Spiral Inductors. *IEEE Transactions on Electron Devices,* Vol. 52, No. 2, pp. 176-182.

[27] Sia, C.B., Ong, H., Chan, K.W., Yeo, K.S., Ma, J.G. and Do, M.A., 2005. Physical Layout Design Optimization of Integrated Spiral Inductors for Silicon-Based RFIC Applications. *IEEE Transactions on Electron Devices,* Vol. 52, No. 12, pp. 2559-2567.

[28] Sun, H., Liu, Z., Zhao, J., Wang, L. and Zhu, J., 2008. The Enhancement of Q-Factor of Planar Spiral Inductor With Low-Temperature Annealing. *IEEE Transactions on Electron Devices,* Vol. 55, No. 3, pp. 931-936.

[29] Tu, H.L., Chen, I.S., Yeh, P.C. and Chiou, H.K., 2006. High Performance Spiral Inductor on Deep-Trench-Mesh Silicon Substrate. *IEEE Microwave and Wireless Component Letters,* Vol. 16, No. 12, pp. 654-656.

[30] Lee, T.H., 2004, *The Design of CMOS Radio-Frequency Integrated Circuits*, 2nd edition, Cambridge: Cambridge University Press.

[31] Raab, F.H., Asbeck, P., Cripps, S., Kenington, P.B., Popović, Z.B., Pothecary, N., Sevic, J.F. and Sokal, N.O., 2002. Power Amplifiers and Transmitters for RF and Microwave. *IEEE Transactions on Microwave Theory and Techniques,* Vol. 50, No. 3, pp. 814-826.

[32] Van, J.H., Kim, M.S., Jung, S.C., Park, H.C., Ahn, G., Park, C.S., Kim, B.S. and Yang, Y., 2007. A High Frequency and High Power Quasi-Class-E Amplifier Design Using a Finite Bias Feed Inductor. *Microwave and Optical Technology Letters,* Vol. 49, No. 5, pp. 1114-1118.

[33] Ludwig, R. and Bretchko, P., 2000, *RF Circuit Design: Theory and Applications*, 1st edition, Upper Saddle River: Prentice Hall.

[34] Grebennikov, A., 2008. High-Efficiency Class-FE Tuned Power Amplifiers. *IEEE Transactions on Circuit and Systems — I: Regular Papers,* Vol. 55, No. 10, pp. 3284-3292.

[35] Raab, F.H., 2001. Class-E, Class-C, and Class-F Power Amplifiers Based Upon a Finite Number of Harmonics. *IEEE Transactions on Microwave Theory and Techniques,* Vol. 49, No. 12, pp. 1462-1468.

[36] Aflaki, P., Negra, R. and Ghannouchi, F.M., 2009. Enhanced architecture for microwave currentmode class-D amplifiers applied to the design of an S-band GaN-based power amplifier. *IET Microwave Antennas Propagation,* Vol. 3, No. 6, pp. 997-1006.

[37] Grebennikov, A. and Sokal, N.O., 2007, *Switchmode RF Power Amplifiers*, 1st edition, Burlington: Elsevier.

[38] El-Sabban, A.A.F. and Ragai, H.F., 2008. Design of power-controlled class1 Bluetooth CMOS power amplifier. *International Journal of Electronics,* Vol. 95, No. 3, pp. 265-273.

[39] Sokal, N.A. and Sokal, A.D., 1975. Class E-A New Class of High-Efficiency Tuned Single-Ended Switching Power Amplifiers. *IEEE Journal of Solid-State Circuits,* Vol. 10, No. 3, pp. 168-176.

[40] Colantonio, P., Giannini, F., Leuzzi, G. and Limiti, E., 1999. On the Class-F Power Amplifier Design. *International Journal of RF and Microwave Computer-Aided Engineering,* Vol. 9, No. 2, pp. 129-149.

[41] Lee, D.H., Park, C., Han, J., Kim, Y., Hong, S., Lee, C.H. and Laskar, J., 2008. A Load-Shared CMOS Power Amplifier With Efficiency Boosting at Low Power Mode for Polar Transmitters. *IEEE Transactions on Microwave Theory and Techniques,* Vol. 56, No. 7, pp. 1565-1574.

[42] Gao, S., 2006. High-Efficiency Class F RF/Microwave Power Amplifiers. *IEEE Microwave Magazine,* Vol. 7, No. 1055, pp. 40-48.

[43] Raab, F.H., 2001. Maximum Efficiency and Output of Class-F Power Amplifiers. *IEEE Transactions on Microwave Theory and Techniques,* Vol. 49, No. 6, pp. 1162-1166.

[44] Kang, D., Yu, D., Min, K., Han, K., Choi, J., Kim, D., Jin, B., Jun, M. and Kim, B., 2008. A Highly Efficient and Linear Class-AB/F Power Amplifier for Multimode Operation. *IEEE Transactions on Microwave Theory and Techniques,* Vol. 56, No. 1, pp. 77-87.

[45] Kim, D.G., Choi, D.H., Moon, Y.T., Baek, D.H., Baek, K.H. and Choi, Y.W., 2010. The Design and Realization of a Highly Linear Power Amplifier Module for a WiMAX/WiBro (802.16e) Base Station. *Microwave and Optical Technology Letters,* Vol. 52, No. 5, pp. 1952-1955.

[46] Jiang, H. and Wilford, P.A., 2010. Digital predistortion for power amplifiers using separable functions. *IEEE Transactions on Signal Processing,* Vol. 58, No. 8, pp. 4121-4130.

[47] Walling, J.S. and Allstot, D.J., 2010. Linearizing CMOS Switching Power Amplifiers Using Supply Regulators. *IEEE Transactions on Circuits and Systems II: Express Briefs,* Vol. 57, No. 7, pp. 497-501.

[48] Nam, J. and Kim, B., 2007. The Doherty Power Amplifier With On-Chip Dynamic Bias Control Circuit for Handset Application. *IEEE Transactions on Microwave Theory and Techniques,* Vol. 55, No. 4, pp. 633-642.

[49] Aoki, I., Kee, S., Magoon, R., Aparicio, R., Bohn, F., Zachan, J., Hatcher, J., McClymont, D. and Hajimiri, A., 2008. A Fully-Integrated Quad-Band GSM/GPRS CMOS Power Amplifier. *IEEE Journal of Solid-State Circuits,* Vol. 43, No. 12, pp. 2747-2758.

[50] An, K.H., Lee, O., Kim, H., Lee, D.H., Han, J., Yang, K.S., Kim, Y., Chang, J.J., Woo, W., Lee, C.S., Kim, H. and Laskar, J., 2008. Power-Combining Transformer Techniques for Fully-Integrated CMOS Power Amplifiers. *IEEE Journal of Solid-State Circuits,* Vol. 43, No. 5, pp. 1064-1075.

[51] Donguye, J., Wanrong, Z., Pei, S., Hongyun, X., Yang, W., Wei, Z., Lijian, H., Yongping, S., Jia, L. and Junning G., 2008. Multi-finger power SiGe HBTs for thermal stability enhancement over a wide biasing range. *Solid-state electronics,* Vol. 52, No. 6, pp. 937-940.

[52] Johnson, J.B., Joseph, A.J., Sheridan, D.C., Maladi, R.M., Brandt, P.O., Persson, J., Andersson, J., Bjorneklett, A., Persson, U., Abasi, F. and Tilly, L., 2004. Silicon-Germanium BiCMOS HBT Technology for Wireless Power Amplifier Applications. *IEEE Journal of Solid-State Circuits,* Vol. 39, No. 10, pp. 1605-1614.

[53] Raab, F.H., Asbeck, P., Cripps, S., Kenington, P.B., Popovic, Z.B., Pothecary, N., Sevic, J.S. and Sokal, N.O., 2003. RF and Microwave Power Amplifier and Transmitter Technologies — Part 1. *High Frequency Electronics,* Vol. 2, pp. 24-36.

[54] Lee, D.H., Chen, Y., Lee, K.A. and Hong, S., 2008. A Differential HBT Power Cell and Its Model. *Microwave and Optical Technology Letters,* Vol. 50, No. 9, pp. 2262-2268.

[55] Brama, R., Larcher, L., Mazzanti, A. and Svelto, F., 2008. A 30.5 dBm 48% PAE CMOS Class-E PA With Integrated Balun for RF Applications. *IEEE Journal of Solid-State Circuits,* Vol. 43, No. 8, pp. 1755-1762.

[56] Gaw, C., Arnold, T., Martin, R., Zhang, L. and Zupac, D., 2006. Evaluation of SiGe:C HBT intrinsic reliability using conventional and step stress methodologies. *Microelectronics Reliability,* Vol. 46, No. 8, pp. 1272-1278.

[57] Anon., 2011. RS Online [Online]. Available from: http://za.rs-online.com/web/search/searchBrowseAction.html?method=searchProducts&searchTerm=inductor&x=14&y=13 [Accessed: 24 February 2011].

[58] Uyanik, H. and Tarim, N., 2007. Compact low voltage high-Q CMOS active inductor suitable for RF applications. *Analog Integrated Circuit Signal Processing,* Vol. 51, pp. 191-194.

[59] Ler, C.L., A'ain, A.K.B. and Kordesh, A.V., 2008. CMOS source degenerated differential active inductor. *Electronics Letters,* Vol. 44, No. 3, pp. 196-197.

[60] Bakken, T. and Choma, J., 2003. Gyrator-Based Synthesis of Active On-Chip Inductances. *Analog Integrated Circuit Signal Processing,* Vol. 34, No. 3, pp. 171-191.

[61] De Los Santos, H.J., 2001. MEMS for RF/Microwave Wireless Applications. *Microwave Journal,* Vol. 44, No. 3, pp. 20-24, 28, 32-41.

[62] Lin, J.W., Chen, C.C. and Cheng, Y.T., 2005. A Robust High-Q Micromachined RF Inductor for RFIC Applications. *IEEE Transactions on Electron Devices,* Vol. 52, No. 7, pp. 1489-1496.

[63] Gu, L. and Li, X., 2007. High-Q Solenoid Inductors with a CMOS-Compatible Concave-Suspending MEMS Process. *Journal of Microelectromechanical Systems,* Vol. 16, No. 5, pp. 1162-1172.

[64] Chua, L.C., Fork, D.K., Van Schuylenbergh, K. and Lu, J.P., 2003. Out-of-Plane High-Q Inductors on Low-Resistance Silicon. *Journal of Microelectromechanical Systems,* Vol. 12, No. 6, pp. 989-995.

[65] Foty, D., 2008. Prospects for Nanoscale Electron Devices: Some Little-Recognized Problems*, Proceedings: International Semiconductor Conference (CAS)*, Sinaia, 13-15 October, pp. 1-12.

[66] Murad, S.A.Z., Pokharel, R.K., Kanaya, H., Yoshida, K. and Nizhnik, O., 2010. A 2.4-GHz 0.18-µm CMOS ClassE single-ended switching power amplifier with a self-biased cascode. *International Journal of Electronics and Communication,* Vol. 64, No. 9, pp. 813-818.

[67] Khatri, H., Gudem, P.S. and Larson, L.E., 2008. Integrated RF Interference Suppression Filter Design Using Bond-Wire Inductors. *IEEE Transactions on Microwave Theory and Techniques,* Vol. 56, No. 5, pp. 1024-1034.

[68] Masu, K., Okada, K. and Ito, H., 2006. RF Passive Components Using Metal Line on Si CMOS. *Transactions on Electronics,* Vol. E89–C, No. 6, pp. 681-691.

[69] Vroubel, M., Zhuang, Y., Rejaei, B. and Burghartz, J.N., 2004. Integrated Tunable Magnetic RF Inductor. *IEEE Electronic Device Letters,* Vol. 25, No. 12, pp. 787-789.

[70] Seo, S., Ryu, N., Choi, H. and Jeong, Y., 2007. Novel high-Q inductor using active inductor structure and feedback parallel resonance circuit*, Proceedings: IEEE Radio Frequency Integrated Circuits Symposium*, Honolulu, 3-5 June, pp. 467-470.

[71] Zine-El-Abidine, I. and Okoniewski, M., 2007. High Quality Factor Micromachined Toroid and Solenoid Inductors*, Proceedings: 37th European Microwave Conference*, Munich, 9-12 October, pp. 1149-1152.

[72] Mohan, S.S., del Mar Hershenson, M., Boyd, S.P. and Lee, T.H., 1999. Simple Accurate Expressions for Planar Spiral Inductances. *IEEE Journal of Solid-State Circuits,* Vol. 34, No. 10, pp. 1419-1424.

[73] Ooi, B.L. and Xu, D.X., 2004. A Novel Equivalent Circuit Model for Two-Layered Spiral Inductor with Eddy-Current Effect in the Substrate. *Microwave and Optical Technology Letters,* Vol. 40, No. 6, pp. 484-487.

[74] Hsu, H.M., 2006. Investigation on the layout parameters of on-chip inductor. *Microelectronics Journal,* Vol. 37, No. 8, pp. 800-803.

[75] Koutsoyannopoulos, Y.K. and Papananos, Y., 2000. Systematic Analysis and Modeling of Integrated Inductors and Transformers in RF IC Design. *IEEE Transactions on Circuits and Systems—II: Analog and Digital Signal Processing,* Vol. 47, No. 8, pp. 699-713.

[76] Watson, A.C., Melendy, D., Francis, P., Hwang, K. and Weisshaar, A., 2004. A Comprehensive Compact-Modeling Methodology for Spiral Inductors in Silicon-Based RFICs. *IEEE Transactions on Microwave Theory and Techniques,* Vol. 52, No. 3, pp. 849-857.

[77] Musunuri, S., Chapman, P.L., Zou, J. and Liu, C., 2005. Design Issues for Monolithic DC–DC Converters. *IEEE Transactions on Power Electronics,* Vol. 20, No. 3, pp. 639-649.

[78] Huo, X., Chan, P.C.H., Chen, K.J. and Luong, H.C., 2006. A Physical Model for On-Chip Spiral Inductors with Accurate Substrate Modeling. *IEEE Transactions on Electron Devices,* Vol. 53, No. 12, pp. 2942-2949.

[79] Lee, C.Y., Chen, T.S., Deng, J.D.S. and Kao, C.H., 2005. A Simple Systematic Spiral Inductor Design with Perfected Q Improvement for CMOS RFIC Application. *Transactions on Microwave Theory and Techniques,* Vol. 53, No. 2, pp. 523-528.

[80] Xue, C., Yao, F., Cheng, B. and Wang, Q., 2008. Effect of the silicon substrate structure on chip spiral inductor. *Frontiers of Electrical and Electronic Engineering in China,* Vol. 3, No. 1, pp. 110-115.

[81] Hastings, A., 2006, *The Art of Analog Layout*, 2nd edition, Upper Saddle River: Prentice Hall.

[82] Kroemer, H., 1982. Heterostructure Bipolar Transistors and Integrated Circuits. *Proceedings of the IEEE,* Vol. 70, No. 1, pp. 13-25.

[83] Vitusevich, S.A., Kurakin, A.M., Klein, N., Petrychuk, M.V., Naumov, A.V. and Belyaev, A.E., 2008. AlGaN/GaN High Electron Mobility Transistor Structures: Self-Heating Effect and Performance Degradation. *IEEE Transactions on Device and Materials Reliability,* Vol. 8, No. 3, pp. 543-548.

[84] Nellis, K. and Zampardi, P., 2004. A Comparison of Linear Handset Power Amplifiers in Different Bipolar Technologies. *IEEE Journal of Solid-State Circuits,* Vol. 39, No. 10, pp. 1746-1754.

[85] Zampardi, P., 2007. Performance and Modeling of Si and SiGe for Power Amplifiers, *Proceedings: 2007 Topical Meeting on Silicon Monolithic Integrated Circuits in RF Systems*, Long Beach, 10-12 January, pp. 13-17.

[86] Davies, A.C., 2002. An overview of Bluetooth Wireless Technology™ and some competing LAN standards, *Proceedings: 1st IEEE International Conference on Circuits and Systems for Communications*, St. Petersburg, 26-28 June, pp. 206-211.

[87] Anon., 2009. Tanner EDA Products [Online]. Available from: http://www.tannereda.com/ [Accessed: 24 February 2011].

[88] Anon., 2005. *0.35 µm HBT BiCMOS Process Parameters*, Unterpremstätten: Austriamicrosystems.

[89] Anon., 2005. *0.35 µm HBT BiCMOS RF SPICE Models*, Unterpremstätten: Austriamicrosystems.

[90] Anon., 2004. *0.35 um HBT BiCMOS Design Rules*, Unterpremstätten: Austriamicrosystems.

[91] Anon., 2011. MOSIS Integrated Circuit Fabrication Service [Online]. Available from: http://www.mosis.com/ [Accessed: 24 February 2011].

[92] Anon., 2008. *BiCMOS7WL Design Manual*, Armonk: IBM Corporation.

[93] Anon., 2007. *Virtuoso Schematic Composed User Guide*, San Jose: Cadence Design Systems.

[94] Anon., 2007. *Virtuoso Analog Design Environment*, San Jose: Cadence Design Systems.

[95] Anon., 2006. *Spectre Circuit Simulator User Guide*, San Jose: Cadence Design Systems.

[96] Anon., 2006. *Virtuoso SpectreRF Simulation Option User Guide*, San Jose: Cadence Design Systems.

[97] Anon., 2007. *Virtuoso Visualization and Analysis Tool Tutorial*, San Jose: Cadence Design Systems.

[98] Anon., 2007. *Virtuoso Layout Suite L User Guide*, San Jose: Cadence Design Systems.

[99] Zhu, H., 2000. *Modeling and Simulation of On-Chip Spiral Inductors and Transformers*, San Jose: Cadence Design Systems.

[100] Degachi, L. and Ghannouchi, F.M., 2008. An Augmented Small-Signal HBT Model With Its Analytical Based Parameter Extraction Technique. *IEEE Transactions on Electron Devices,* Vol. 55, No. 4, pp. 968-972.

[101] Anon., 2008. *BiCMOS-7WL Model Reference Guide*, Armonk: IBM Corporation.

[102] Anon., 2010. *R&S®ZVA Vector Network Analyzer Specifications*, Munich: Rohde & Schwarz.

[103] Bowick, C., Ajluni, C. and Blyler, J., 2007, *RF Circuit Design*, 2nd edition, Burlington: Newnes.

[104] Sproull, R., Lyon, R. & Trimberger, S., 1980. *The Caltech Intermediate Form for LSI Layout Description*, Reading: Addison-Wesley Publishing Company, Inc.

[105] Anon., 2006. *Design Data Translator's Reference*, San Jose: Cadence Design Systems.

[106] Božanić, M. and Sinha, S., 2009. Design Methodology for SiGe-Based Class-E Power Amplifier, *Proceedings: South African Conference on Semi and Superconductor Technology*, Stellenbosch, 8-9 April, pp. 31-36.

[107] Božanić, M. and Sinha, S., 2009. Design Flow for CMOS Based Class-E and Class-F Power Amplifiers, *Proceedings: 2009 AFRICON*, Nairobi, 22-24 September, pp. 1-6.

[108] Pozar, D.M., 2005, *Microwave Engineering*, 3rd edition, New Jersey: John Wiley & Sons.

[109] Božanić, M., Sinha, S., Du Plessis, M. and Müller, A., 2009. Design Flow for a SiGe BiCMOS Based Power Amplifier, *Proceedings: International Semiconductor Conference (CAS)*, Sinaia, 12-14 October, pp. 311-314.

[110] Božanić, M., Sinha, S. and Müller, A., 2010. Streamlined Design of SiGe Based Power Amplifiers. *Romanian Journal of Information Science and Technology*, Vol. 13, No. 1, pp 22-32.

[111] Božanić, M. and Sinha, S., 2009. Design Approach to CMOS Based Class-E and Class-F Power Amplifier. *SAIEE Africa Research Journal,* Vol. 100, No. 3, pp. 79-86.

[112] Anon., Unknown. *Very Wideband RF Choke*, New York: Mini-Circuits.

[113] Anon., 2008. Amphenol RF - SMA Connector Series [Online]. Available from: http://www.amphenolrf.com/products/sma.asp?N=0&sid=4C86D2004C3A617F& [Accessed: 24 February 2011].

[114] Foty, D., Sinha, S., Weststrate, M., Coetzee, C., Uys, A.H. and Sibanda, E., 2008. mm-Wave Radio Communications Systems: The Quest Continues, *Proceedings: 3rd International Radio Electronics Forum (IREF) on Applied Radio Electronics: The State and Prospects of Development*, Kharkov, 22-24 October, pp. 1-14.

[115] Božanić, M. and Sinha, S., 2011. Switch-Mode Power Amplifier Design Method, submitted for publication in *Microwave and Optical Technology Letters*.

APPENDIX A MATLAB CODE

A.1 INTRODUCTION

Appendix A lists the MATLAB code for all sub-routines of the program for the complete PA design integration. The sub-routines are listed in order in which they first appear in the MATLAB code, starting with the main PA design program.

A.2 MAIN PA DESIGN PROGRAM

Figure A.1 and Figure A.2 show the MATLAB code for the main subroutine of the PA design program (paprog.m).

```
try
    clear all; %Clear all variables from workspace
    fprintf('Welcome to PA Design Program v2.0.0\n\n');
    choice = 0;
    inductor = zeros(1,10); %declare array size 10
    while (choice ~= 1 && choice ~= 2)
        choice = input('Please enter 1 for Class-E Amplifier design or 2 for Class-F
Amplifier design: ');
        if (choice ~= 1 && choice ~= 2)
            fprintf('Wrong entry. Please try again.\n');
        end
    end
    if (choice == 1)
        classE; %Class-E PA design program
    end
    if (choice == 2)
        classF; %Class-F PA design program
    end
    %Impedance matching
    wo = 2*pi*fo; %Calc. radian frequency
    cmat = input('\nDo you want to perform the impedance matching to 50 ohm antenna
impedance (y/n)? ', 's');
    if cmat == 'y' || cmat == 'Y';
        %impedance
        BW = input('\nPlease enter the matching bandwidth (MHz): '); %Input bandwidth
        BW = BW * 1e6;
        RS = RL; %Source resistance
        RL = 50; %Load resistance
        QL = fo/BW; %Quality factor
        calcMatch;
    end%if
    %cnet - flag - if selected, matching will be performed
    cnet = input('\nDo you want to export the netlist of the PA (y/n)? ', 's');
fprintf('\n');
    if cmat == 'y' || cmat == 'Y';
        %cind - flag - if selected, spiral inductors will be found
cind = input('\nDo you want to attempt to find inductor(s) for the PA (y/n)? ', 's');
    end

    if (cind == 'y' || cind == 'Y') || (cnet == 'y' || cnet == 'Y');
        %Overrides any inductor values if needed
        overrideInductors;
    end
```

Figure A.1. MATLAB code of the main subroutine of the PA design program.

```
    fprintf('\n');
    if cind == 'y' || cind == 'Y';
        %Find inductors
        %Define constants
        b = 1.62e-3;
        a1 = -1.21;
        a2 = -0.147;
        a3 = 2.40;
        a4 = 1.78;
        a5 = -0.030;
        f = fo;
        omega = wo;
        %Setup the inductance search algorithm search paramters
        indSetup;
        if cnet == 'y' || cnet == 'Y';
            %Name of the PA used for netlist extraction
            paName = input('\nPlease enter the PA name (press "s" to skip): ', 's');
            if paName(1) == 's' || paName(1) == 'S';
                %Default PA name
                paName = 'MyPA';
            end%if
            RFC = 1;
            fileName = strcat(paName, '_net.spc');
            fid = fopen(fileName, 'w');
        end
        fprintf('\nINDUCTORS\n');
            %Find the inductors designed by the PA program
            findInductors;
        end
        %Export the netlist
        exportPA;
    catch
        fprintf('MATLAB returns: \n%s\nError occurred while running the program.
    Please try running the program again.\n\n', lasterr);
    end%try
```

Figure A.2. MATLAB code of the main subroutine of the PA design program (continued).

A.3 CLASS-E DESIGN SUBROUTINE

Figure A.3 and Figure A.4 show the MATLAB code for the Class-E design subroutine (ClassE.m).

```
%Enter PA design parameters
fo = input('Please enter the frequency of operation, fo (MHz): ');
if (fo <= 0)
    fo = 1e9; %Frequency
else
    fo = fo * 1e6;
end
w = 2*pi*fo; %Radian frequency
QL = input('Please enter the loaded quality factor, QL: ');
if (QL < 1)
    QL = 10; %Loaded quality factor
end
Po = input('Please enter the required output power, Po (mW): ');
if (Po <= 0)
    Po = 0.01; %Output power
else
    Po = Po/1000;
end
Vcc = input('Please enter the value of the regulated power supply (Vcc/Vdd): ');
if (Vcc <= 0)
    Vcc = 1; %Power supply
end
VCEsat = input('Please enter VCEsat/VDSsat (-1 to skip): ');
if (VCEsat <= 0)
    VCEsat = 0; %Saturation VCE voltage
end

Lmax = input('Please enter the highest allowed inductance (nH) (-1 to skip): ');
if (Lmax <= 0)
    Lmax = 0; %Maximum inductance
else
    Lmax = Lmax * 1e-9;
end
Lmin = input('Please enter the lowest allowed inductance (nH) (-1 to skip): ');
if (Lmin <= 0)
    Lmin = 0; %Minimum inductance
else
    Lmin = Lmin * 1e-9;
end

BVCE = input('Please enter collector-emiiter breakdown voltage (V) (-1 to skip): ');
if (BVCE <= 0)
    BVCE = 0; %Breakdown voltage
end

%Calculate
vcp = 3.56 * Vcc;
if (vcp > BVCE && BVCE > 0)
    NVcc = BVCE / 3.56;
    fprintf('Warning: Vcc should not exceed %.2f or the transistor will go into
breakdown!\n', NVcc);
end
RL = 0.577*(Vcc - VCEsat)^2/Po; %Load resistance
L2 = QL*RL/w; %Series inductance
C1 = 1/(w*RL*5.447); %Parallel capacitance
C2 = C1*(5.447/QL)*(1+1.42/(QL - 2.08)); %Series Capacitance

Idc = Vcc / (1.734*RL); %DC current consumption
isp = 2.86 * Idc; %Maximum collector current dissipation

%Q-factor adjustment
if (Lmax ~= 0)
    if (L2 > Lmax)
```

Figure A.3. MATLAB code for the Class-E design subroutine.

```
        QLB = w*Lmax/RL;
        if (QLB > 1)
        L2 = Lmax;
            C2 = C1*(5.447/QLB)*(1+1.42/(QLB - 2.08));
            fprintf('New QL = %.2f \n', floor(100 * QLB + 0.5) / 100);
        else
            fprintf('Cannot find QL greater than 1 to keep L2 <= Lmax\n');
        end
    end
end

if (Lmin ~= 0)
    if (L2 < Lmin)
        QLB = w*Lmin/RL;
        L2 = Lmin;
        C2 = C1*(5.447/QLB)*(1+1.42/(QLB - 2.08));
        fprintf('New QL = %.2f \n', floor(100 * QLB + 0.5) / 100);
    end
end

%Display calculated values
fprintf('\nRESULTS:\n');
fprintf('RL = %.2f Ohm \n', floor(100 * RL + 0.5) / 100);
fprintf('L2 = %.2f nH \n', floor(1e11 * L2 + 0.5) / 100);
fprintf('C2 = %.3f pF \n', floor(1e14 * C2 + 0.5) / 100);
fprintf('C1 = %.3f pF \n', floor(1e14 * C1 + 0.5) / 100);
fprintf('Idc = %.2f mA \n', floor(1e5 * Idc + 0.5) /100);
fprintf('vcp = %.2f V \n', floor(1e2 * vcp + 0.5) / 100);
fprintf('isp = %.2f mA \n', floor(1e5 * isp + 0.5) / 100);

Lfeed = 0;
```

Figure A.4. MATLAB code for the Class-E design subroutine (continued).

A.4 CLASS-F DESIGN SUBROUTINE

Figure A.5 and Figure A.6 show the MATLAB code for the Class-F design subroutine (ClassF.m).

```
%Enter PA design parameters
harmonics = 0;
%Choose between the 3 and 5 harmonic designs
while (harmonics ~= 1 && harmonics ~= 2)
    harmonics = input('Please enter 1 for 3-harmonic design or 2 for 5-harmonic design:
');
    if (harmonics ~= 1 && harmonics ~= 2)
       fprintf('Wrong entry. Please try again.\n');
    end
end
fo = input('Please enter the frequency of operation, fo (MHz): ');
if (fo <= 0)
    fo = 1e9; %Frequency of operation
else
    fo = fo * 1e6;
end
w = 2*pi*fo;
Po = input('Please enter the required output power, Po (mW): ');
if (Po <= 0)
    Po = 0.01; %Output power
else
    Po = Po/1000;
end
Vcc = input('Please enter the value of the regulated power supply (Vcc/Vdd): ');
if (Vcc <= 0)
    Vcc = 1; %Voltage supply
end
BVCE = input('Please enter collector-emiiter breakdown voltage (V) (-1 to skip): ');
if (BVCE <= 0)
    BVCE = 0; %Collector-emitter breakdown voltage
end
L0 = input('Please enter the inductance for the base filter (nH): ');
if (L0 <= 0)
    L0 = 1e-9; %Base filter inductance
else
    L0 = L0 * 1e-9;
end
L3 = input('Please enter the inductance for the third harmonic filter (nH): ');
if (L3 <= 0)
    L3 = 1e-9; %Third harmonic filter inductance
else
    L3 = L3 * 1e-9;
end
if harmonics == 2;
    L5 = input('Please enter the inductance for the fifth harmonic filter (nH): ');
    if (L5 <= 0)
        L5 = 1e-9; %Fifth harminic filter inductance
    else
        L5 = L5 * 1e-9;
    end
else
    L5 = 0;
end

if harmonics == 1
    %third harmonic peaking circuit coefficients
    dV = 2;
    gV = 1.1547;
    dI = 2.91;
    gI = 1.4142;
else
    %5 resonators circuit coefficients
    dV = 2;
```

Figure A.5. MATLAB code for the Class-F design subroutine.

```
    gV = 1.2071;
    dI = 3;
gI = 1.5;
end

%Calculate optimum resistance and waveforms
RL = gV^2*Vcc^2/(2*Po);
Vom = gV*Vcc;
vCm = dV*Vcc;
Idc = gV*Vcc/(gI*RL);
iCm = dI*Idc;
A = 2*acos(Idc/(Idc-iCm));

%Calculate Filters
%base
C0 = 1/((2*pi*fo)^2*L0);
%3rd harmonic
C3 = 1/((2*pi*3*fo)^2*L3);
if harmonics == 2;
    C5 = 1/((2*pi*5*fo)^2*L3);
end
%Display calculated values
fprintf('\nRESULTS:\n');
fprintf('RL = %.2f Ohm \n', floor(100 * RL + 0.5) / 100);
fprintf('L0 = %.2f nH \n', floor(1e11 * L0 + 0.5) / 100);
fprintf('C0 = %.3f pF \n', floor(1e14 * C0 + 0.5) / 100);
fprintf('L3 = %.2f nH \n', floor(1e11 * L3 + 0.5) / 100);
fprintf('C3 = %.3f pF \n', floor(1e14 * C3 + 0.5) / 100);
if harmonics == 2;
    fprintf('L5 = %.2f nH \n', floor(1e11 * L5 + 0.5) / 100);
    fprintf('C5 = %.3f pF \n', floor(1e14 * C5 + 0.5) / 100);
end
fprintf('Idc = %.2f mA \n', floor(1e5 * Idc + 0.5) /100);
fprintf('vCm = %.2f V \n', floor(1e2 * vCm + 0.5) / 100);
fprintf('Vom = %.2f V \n', floor(1e2 * Vom + 0.5) / 100);
fprintf('iCm = %.2f mA \n', floor(1e5 * iCm + 0.5) / 100);

if (vCm > BVCE && BVCE > 0)
    NVcc = BVCE / dV;
    fprintf('Warning: Vcc should not exceed %.2f or the transistor will go into
breakdown!\n', NVcc);
end
```

Figure A.6. MATLAB code for the Class-F design subroutine (continued).

A.5 IMPEDANCE MATCHING

Figure A.7 and Figure A.8 show the MATLAB code for the impedance matching subroutine (calcMatch.m).

```
%L network
LM = 1/wo * (RL^2*RS / (RL - RS))^(1/2);
CM = 1/wo * (RL^2 + (wo*LM)^2) / (RL^2 * wo * LM);

%Results for L network
fprintf('\nL network: \n');
fprintf('LM = %.3f nH\n', floor(1e11 * LM + 0.5) / 100);
fprintf('CM = %.3f pF\n', floor(1e14 * CM + 0.5) / 100);
```

Figure A.7. MATLAB code for the impedance matching subroutine.

```
%T networks
Rsmall = RS;
if RL < Rsmall
    Rsmall = RL;
end
R = Rsmall*(QL^2 + 1);
XS1 = QL*RS;
XP1 = R/QL;

Q2 = (R/RL - 1)^(1/2);
XP2 = R/Q2;
XS2 = Q2*RL;

CT1 = XP1*XP2/(XP1+XP2);
CT2 = XP1*XP2/abs(XP1-XP2);

LM1 = XS1/wo;
LM2 = XS2/wo;
LM3 = CT1/wo;

CM1 = 1/(CT1*wo);
CM2 = 1/(XS1*wo);
CM3 = 1/(XS2*wo);

%Results for T network
fprintf('\nind-cap-ind T network: \n');
fprintf('L1 = %.3f nH\n', floor(1e12 * XS1/wo + 0.5) / 1000);
fprintf('C1 = %.3f pF\n', floor(1e15 * 1/(CT1*wo) + 0.5) / 1000);
fprintf('L2 = %.3f nH\n', floor(1e12 * XS2/wo + 0.5) / 1000);

fprintf('cap-ind-cap T network: \n');
fprintf('C1 = %.3f pF\n', floor(1e15 * 1/(XS1*wo) + 0.5) / 1000);
fprintf('L1 = %.3f nH\n', floor(1e12 * CT1/wo + 0.5) / 1000);
fprintf('C2 = %.3f pF\n', floor(1e15 * 1/(XS2*wo) + 0.5) / 1000);

%Pi network
RH = RS;
if RL > RH
    RH = RL;
end
R = RH/(QL^2 + 1);
XS2 = QL*R;
XP2 = RL/QL;

Qi = (RS/R - 1)^(1/2);
XP1 = RS/Q2;
XS1 = Qi*R;

CT1 = XS1 + XS2;
CT2 = abs(XS1-XS2);

LM4 = XP1/wo;
LM5 = XP2/wo;
LM6 = CT1/wo;

CM4 = 1/(CT1*wo);
CM5 = 1/(XP1*wo);
CM6 = 1/(XP2*wo);
%Results for Pi network
fprintf('\nind-cap-ind Pi network: \n');
fprintf('L1 = %.3f nH\n', floor(1e12 * XP1/wo + 0.5) / 1000);
fprintf('C1 = %.3f pF\n', floor(1e15 * 1/(CT1*wo) + 0.5) / 1000);
fprintf('L2 = %.3f nH\n', floor(1e12 * XP2/wo + 0.5) / 1000);

fprintf('cap-ind-cap Pi network: \n');
fprintf('C1 = %.3f pF\n', floor(1e15 * 1/(XP1*wo) + 0.5) / 1000);
fprintf('L1 = %.3f nH\n', floor(1e12 * CT1/wo + 0.5) / 1000);
fprintf('C2 = %.3f pF\n', floor(1e15 * 1/(XP2*wo) + 0.5) / 1000);
```

Figure A.8. MATLAB code for the impedance matching subroutine (continued).

A.6 INDUCTANCE VALUE OVERRIDE

Figure A.9 and Figure A.10 show the MATLAB code for the inductance values override subroutine (overrideInductors.m). Any inductance value calculated by the PA design program can be overridden by this subroutine manually if needed.

```
%Overrides any inductance values calculated by PA design program
cov = input('Do you want to override any of the inductance values (y/n)? ', 's');
if cov == 'y' || cov == 'Y';
    if choice == 1;
        fprintf('Old L2 = %.2f nH. ', L2*1e9);
        iL2 = input ('New L2 (-1 to skip): ');
        if iL2 > 0;
            L2 = iL2/1e9;
        end
        fprintf('Old Lfeed = %.2f nH. ', Lfeed*1e9);
        iLfeed = input ('New Lfeed (-1 to skip): ');
        if iLfeed > 0;
            Lfeed = iLfeed/1e9;
        end
    elseif choice == 2;
        fprintf('Old Lfeed = 100.00 nH. ');
        iLfeed = input ('New Lfeed (-1 to skip): ');
        if iLfeed > 0;
            Lfeed = iLfeed/1e9;
        else
            Lfeed = 1e-7;
        end
        fprintf('Old L0 = %.2f nH. ', L0*1e9);
        iL0 = input ('New L0 (-1 to skip): ');
        if iL0 > 0;
            L0 = iL0/1e9;
        end
        fprintf('Old L3 = %.2f nH. ', L3*1e9);
        iL3 = input ('New L3 (-1 to skip): ');
        if iL3 > 0;
            L3 = iL3/1e9;
        end
        if harmonics == 2;
           fprintf('Old L5 = %.2f nH. ', L5*1e9);
           iL5 = input ('New L5 (-1 to skip): ');
            if iL5 > 0;
                L5 = iL5/1e9;
            end
        end
    end
    if cmn == 1;
        fprintf('Old LM = %.2f nH. ', LM*1e9);
        iLM = input ('New LM (-1 to skip): ');
        if iLM > 0;
            LM = iLM/1e9;
        end
    elseif cmn == 2;
        fprintf('Old LM1 = %.2f nH. ', LM1*1e9);
        iLM1 = input ('New LM1 (-1 to skip): ');
        if iLM1 > 0;
            LM1 = iLM1/1e9;
        end
        fprintf('Old LM2 = %.2f nH. ', LM2*1e9);
        iLM2 = input ('New LM2 (-1 to skip): ');
        if iLM2 > 0;
            LM2 = iLM2/1e9;
        end
   elseif cmn == 3;
        fprintf('Old LM3 = %.2f nH. ', LM3*1e9);
        iLM3 = input ('New LM3 (-1 to skip): ');
        if iLM3 > 0;
```

Figure A.9. MATLAB code for the inductance values override subroutine.

```
            LM3 = iLM3/1e9;
        end
    elseif cmn == 4;
        fprintf('Old LM4 = %.2f nH. ', LM4*1e9);
        iLM4 = input ('New LM4 (-1 to skip): ');
        if iLM4 > 0;
            LM4 = iLM4/1e9;
        end
        fprintf('Old LM5 = %.2f nH. ', LM5*1e9);
        iLM5 = input ('New LM5 (-1 to skip): ');
        if iLM5 > 0;
            LM5 = iLM5/1e9;
        end
    elseif cmn == 5;
        fprintf('Old LM6 = %.2f nH. ', LM6*1e9);
        iLM6 = input ('New LM6 (-1 to skip): ');
        if iLM6 > 0;
            LM6 = iLM6/1e9;
        end
    end
else
    %fix feed for Class-F amplifier
    if choice == 2;
        Lfeed = 1e-7;
    end
end
```

Figure A.10. MATLAB code for the inductance values override subroutine (continued).

A.7 SETUP OF INDUCTANCE SEARCH ALGORITHM PARAMETERS

Figure A.11 shows the MATLAB code for the setup of inductance search algorithm parameters (indSetup.m).

```
%Setup the inductance search algorithm search parameters
tolerance = input('Please enter the tolerance for the inductance value (%) (-1 to
skip): ');
if (tolerance <= 0)
    tolerance = 0.01;
else
    tolerance = tolerance/100;
end
resolution = input('Please enter the search grid resolution (um) (-1 to skip): ');
if (resolution <= 0)
    resolution = 1;
end
%Setup the geometry parameters
cminpar = input('Do you want to change any of the default geometry parameters (y/n)? ',
's');
if cminpar == 'y' || cminpar == 'Y'
  dinmin = input('\nPlease enter the minimum value for din (um) (-1 to skip): ');
  if (dinmin < 30)
    dinmin = 30;
    fprintf('Using the default value of %.0f um for minimum din.\n', dinmin);
  end%if
  doutmax = input('Please enter the maximum value for dout (um) (-1 to skip): ');
  if (doutmax <= 0)
    doutmax = 500;
    fprintf('Using the default value of %.0f um for maximum dout.\n', doutmax);
  end%if
  smin = input('Please enter the minimum value for turn spacing s (um) (-1 to skip):
');
  if (smin <= 0)
    smin = 2;
    fprintf('Using the default value of %.0f um for s.\n', smin);
  end%if
  wmin = input('Please enter the minimum turn width w (um) (-1 to skip): ');
  if (wmin <= 0)
    wmin = 2;
    fprintf('Using the default value of %.0f um for w.\n', wmin);
  end%if
else
    %Default parameters
  dinmin = 30;
  doutmax = 500;
  smin = 2;
  wmin = 2;
  fprintf('Using the default values of %.1f, %.1f, %.1f and %.1f for dinmin, doutmax, s
and wmin respectively.\n', dinmin, doutmax, smin, wmin);
end%if
```

Figure A.11. MATLAB code for the setup of inductance search algorithm parameters.

A.8 SEARCH FOR REQUIRED INDUCTORS

Figure A.12 and Figure A.13 show the MATLAB code for the subroutine that searches for inductor geometries of all inductors needed by PA design program (findInductors.m).

```
%Enter inductor process parameters
entProcParam;
if choice == 1;
    Ls = L2*1e9;
    fprintf('\nInductor value: %.3f nH',Ls);
    indSearch; %Inductance search algorithm
    if cnet == 'y' || cnet == 'Y';
        inductorName = 'L2';
        exportSubckt; %Exports the spiral inductor subcircuit
    end
    if Lfeed ~= 0;
        Ls = Lfeed*1e9;
        fprintf('\nInductor value: %.3f nH',Ls);
        indSearch; %Inductance search algorithm
        if cnet == 'y' || cnet == 'Y';
            inductorName = 'Lfeed';
            exportSubckt; %Exports the spiral inductor subcircuit
        end
    end
elseif choice == 2;
    Ls = L0*1e9;
    fprintf('\nInductor value: %.3f nH',Ls);
    indSearch; %Inductance search algorithm
    if cnet == 'y' || cnet == 'Y';
        inductorName = 'L0';
        exportSubckt; %Exports the spiral inductor subcircuit
    end
    if L3 ~= L0;
        Ls = L3*1e9;
        fprintf('\nInductor value: %.3f nH',Ls);
        indSearch; %Inductance search algorithm
    end
    if cnet == 'y' || cnet == 'Y';
        inductorName = 'L3';
        exportSubckt; %Exports the spiral inductor subcircuit
    end
    if harmonics == 2;
        if L5 ~= L3;
            Ls = L5*1e9;
            fprintf('\nInductor value: %.3f nH',Ls);
            indSearch; %Inductance search algorithm
        end
        if cnet == 'y' || cnet == 'Y';
            inductorName = 'L5';
            exportSubckt; %Exports the spiral inductor subcircuit
        end
    end
end
if cmn == 1;
    Ls = LM*1e9;
    fprintf('\nInductor value: %.3f nH',Ls);
    indSearch; %Inductance search algorithm
    if cnet == 'y' || cnet == 'Y';
        inductorName = 'LM';
        exportSubckt; %Exports the spiral inductor subcircuit
    end
end
if cmn == 2;
    Ls = LM1*1e9;
    fprintf('\nInductor value: %.3f nH',Ls);
    indSearch; %Inductance search algorithm
    if cnet == 'y' || cnet == 'Y';
        inductorName = 'LM1';
        exportSubckt; %Exports the spiral inductor subcircuit
    end
```

Figure A.12. MATLAB code for search for geometries of all needed inductors.

```
    Ls = LM2*1e9;
    fprintf('\nInductor value: %.3f nH',Ls);
    indSearch; %Inductance search algorithm
    if cnet == 'y' || cnet == 'Y';
        inductorName = 'LM2';
        exportSubckt; %Exports the spiral inductor subcircuit
    end
end
if cmn == 3;
    Ls = LM3*1e9;
    fprintf('\nInductor value: %.3f nH',Ls);
    indSearch; %Inductance search algorithm
    if cnet == 'y' || cnet == 'Y';
        inductorName = 'LM3';
        exportSubckt; %Exports the spiral inductor subcircuit
    end
end
if cmn == 4;
    Ls = LM4*1e9;
    fprintf('\nInductor value: %.3f nH',Ls);
    indSearch; %Inductance search algorithm
    if cnet == 'y' || cnet == 'Y';
        inductorName = 'LM4';
        exportSubckt; %Exports the spiral inductor subcircuit
    end
    Ls = LM5*1e9;
    fprintf('\nInductor value: %.3f nH',Ls);
    indSearch; %Inductance search algorithm
    if cnet == 'y' || cnet == 'Y';
        inductorName = 'LM5';
        exportSubckt; %Exports the spiral inductor subcircuit
    end
end
if cmn == 5;
    Ls = LM6*1e9;
    fprintf('\nInductor value: %.3f nH',Ls);
    indSearch; %Inductance search algorithm
    if cnet == 'y' || cnet == 'Y';
        inductorName = 'LM6';
        exportSubckt; %Exports the spiral inductor subcircuit
    end
end
```

Figure A.13. MATLAB code for search for geometries of all needed inductors (continued).

A.9 PA NETLIST EXPORT

Figure A.14 through to Figure A.16 show the MATLAB code for the subroutine that exports the netlists of designed Class-E or Class-F PA (exportPA.m).

```
%This procedure exports the netlist of Class-E or Class-F PA
if (cnet == 'y' || cnet == 'Y')
    cModel = 0;
    %Input the transistor choice
    while cModel ~= 1 && cModel ~= 2 && cModel ~= 3;
        fprintf('\n');
        cModel = input('Please enter 1 for HBT, 2 RF NMOS or 3 for any other
transistor: ');
    end
    if cModel == 1;
       modelName = input('Please enter the model name (e.g. npn254h5): ', 's');
       if modelName(1) == 's' || modelName(1) == 'S';
            modelName = 'npn254';
       end
        wEmitters = input('Please enter the total emitter width (um) (-1 to skip): ');
        if wEmitters < 0;
            wEmitters = 1;
        end
        transistor = strcat(['X',modelName,'_1 N_1 bias Gnd Gnd ',modelName,' area=',
num2str(wEmitters)]);
    elseif cModel == 2;
        lnmos = input('Please enter the NMOS length (um): ');
        wnmos = input('Please enter the NMOS width (um): ');
        ng = input('Please enter the number of NMOS fingers: ');
        addparam = input('Please type in any additional SPICE parameters in format
PARAM=param ("s" to skip): ', 's');
        if lnmos < 0;
            lnmos = 1;
        end
        if wnmos < 0;
            wnmos = 100;
        end
        if ng < 0;
            ng = 10;
        end
        if addparam(1) == 's'
            addparam = ' ';
        end
        transistor = strcat (['Xnmosrf_1 N_1 bias Gnd Gnd nmosrf l=',num2str(lnmos),'u
wtot=',num2str(wnmos),'u ng=',num2str(ng),' ',addparam]);
    else
        transistor = input('Please type the SPICE command for your transistor: ','s');
    end
end
%Export the netlist
if (cnet == 'y' || cnet == 'Y') && (cind ~= 'y' && cind ~= 'Y') && choice == 1;
        paName = input('\nPlease enter the PA name (press "s" to skip): ', 's');
        if paName(1) == 's' || paName(1) == 'S';
            paName = 'MyPA';
        end%if
        if Lfeed == 0;
            RFC = 100;
        else
            RFC = Lfeed*1e9;
        end
        fileName = strcat(paName, '_net.spc');
        fid = fopen(fileName, 'w');
        fprintf(fid, '.SUBCKT %s bias supply Gnd\n', paName);
        fprintf(fid, 'LRFC_2 supply N_1 %.2fn\n', RFC);
        if cmn == 0;
          fprintf(fid, 'LInductor_2 N_5 N_3 %.2fn\n', floor(1e11 * L2 + 0.5) / 100);
          fprintf(fid, 'RResistor_1 N_3 Gnd %.2f TC=0.0, 0.0\n', floor(100 * RL + 0.5)
/ 100);
```

Figure A.14. MATLAB code for the subroutine used for netlist export.

```
        else
          fprintf(fid, 'LInductor_2 N_5 N_M1 %.2fn\n', floor(1e11 * L2 + 0.5) / 100);
          fprintf(fid, 'RResistor_1 N_M2 Gnd %.2f TC=0.0, 0.0\n', 50);
        end
        fprintf(fid, transistor);
        fprintf(fid, '\n');
        if Lfeed == 0;
          fprintf(fid, 'CCapacitor_1 N_1 Gnd %.2fp\n', floor(1e14 * C1 + 0.5) / 100);
        else
          fprintf(fid, 'CCapacitor_1 N_1 Gnd %.2fp\n', floor(1e14 * Cp + 0.5) / 100);
        end
        fprintf(fid, 'CCapacitor_2 N_1 N_5 %.2fp\n', floor(1e14 * C2 + 0.5) / 100);
        matchIdeal; %Export matching inductors
        fprintf(fid, '.ENDS\n');
        fclose(fid);
        fprintf('File %s created successfully.\n\n', fileName);
    end

    if (cnet == 'y' || cnet == 'Y') && (cind == 'y' || cind == 'Y') && choice == 1;
        fprintf(fid, '.SUBCKT %s bias supply Gnd\n', paName);
        if Lfeed == 0;
          RFC = 100;
          fprintf(fid, 'LRFC_2 supply N_1 %.2fn\n', RFC);
        else
          fprintf(fid, 'XLfeed_1 supply N_1 Gnd Lfeed\n');
        end
        if cmn == 0;
          fprintf(fid, 'XL2_1 N_5 N_3 Gnd L2\n');
          fprintf(fid, 'RResistor_1 N_3 Gnd %.2f TC=0.0, 0.0\n', floor(100 * RL + 0.5)
/ 100);
        else
          fprintf(fid, 'XL2_1 N_5 N_M1 Gnd L2\n');
          fprintf(fid, 'RResistor_1 N_M2 Gnd %.2f TC=0.0, 0.0\n', 50);
        end
        fprintf(fid, transistor);
        fprintf(fid, '\n');
        fprintf(fid, 'CCapacitor_1 N_1 Gnd %.2fp\n', floor(1e14 * C1 + 0.5) / 100);
        fprintf(fid, 'CCapacitor_2 N_1 N_5 %.2fp\n', floor(1e14 * C2 + 0.5) / 100);
        matchSpiral; %Export matching inductors
        fprintf(fid, '.ENDS\n');
        fclose(fid);
        fprintf('File %s created successfully.\n\n', fileName);
    end

  if (cnet == 'y' || cnet == 'Y') && (cind ~= 'y' && cind ~= 'Y') && choice == 2;
        paName = input('\nPlease enter the PA name (press "s" to skip): ', 's');
        if paName(1) == 's' || paName(1) == 'S';
          paName = 'MyPA';
        end%if
        fileName = strcat(paName, '_net.spc');
        fid = fopen(fileName, 'w');
        RFC = Lfeed*1e9;
        fprintf(fid, '.SUBCKT %s bias supply Gnd\n', paName);
        fprintf(fid, 'LRFC_2 supply N_1 %.2fn\n', RFC);
        fprintf(fid, 'CCapacitor_3 N_1 N_4 %.2fp\n', 1);
        if cmn == 0;
          fprintf(fid, 'LInductor_2 N_2 Gnd %.2fn\n', floor(1e11 * L0 + 0.5) / 100);
          fprintf(fid, 'CCapacitor_1 N_2 Gnd %.2fp\n', floor(1e14 * C0 + 0.5) / 100);
          if harmonics == 1;
            fprintf(fid, 'LInductor_1 N_4 N_2 %.2fn\n', floor(1e11 * L3 + 0.5) / 100);
            fprintf(fid, 'CCapacitor_2 N_4 N_2 %.2fp\n', floor(1e14 * C3 + 0.5) / 100);
          end
          if harmonics == 2;
            fprintf(fid, 'LInductor_1 N_4 N_3 %.2fn\n', floor(1e11 * L3 + 0.5) / 100);
            fprintf(fid, 'CCapacitor_2 N_4 N_3 %.2fp\n', floor(1e14 * C3 + 0.5) / 100);
            fprintf(fid, 'LInductor_3 N_3 N_2 %.2fn\n', floor(1e11 * L5 + 0.5) / 100);
            fprintf(fid, 'CCapacitor_4 N_3 N_2 %.2fp\n', floor(1e14 * C5 + 0.5) / 100);
          end
          fprintf(fid, 'RResistor_1 N_2 Gnd %.2f TC=0.0, 0.0\n', floor(100 * RL + 0.5)
/ 100);
```

Figure A.15. MATLAB code for the subroutine used for netlist export (continued).

```
        else
          fprintf(fid, 'LInductor_2 N_M1 Gnd %.2fn\n', floor(1e11 * L0 + 0.5) / 100);
          fprintf(fid, 'CCapacitor_1 N_M1 Gnd %.2fp\n', floor(1e14 * C0 + 0.5) / 100);
          if harmonics == 1;
            fprintf(fid, 'LInductor_1 N_4 N_M1 %.2fn\n', floor(1e11 * L3 + 0.5) / 100);
            fprintf(fid, 'CCapacitor_2 N_4 N_M1 %.2fp\n', floor(1e14 * C3 + 0.5) /
100);
          end
          if harmonics == 2;
            fprintf(fid, 'LInductor_1 N_4 N_3 %.2fn\n', floor(1e11 * L3 + 0.5) / 100);
            fprintf(fid, 'CCapacitor_2 N_4 N_3 %.2fp\n', floor(1e14 * C3 + 0.5) / 100);
            fprintf(fid, 'LInductor_3 N_3 N_M1 %.2fn\n', floor(1e11 * L5 + 0.5) / 100);
            fprintf(fid, 'CCapacitor_4 N_3 N_M1 %.2fp\n', floor(1e14 * C5 + 0.5) /
100);
          end
          fprintf(fid, 'RResistor_1 N_M2 Gnd %.2f TC=0.0, 0.0\n', 50);
        end

        fprintf(fid, transistor);
        fprintf(fid, '\n');
        matchIdeal; %Export matching inductors
        fprintf(fid, '.ENDS\n');
        fclose(fid);
        fprintf('File %s created successfully.\n\n', fileName);
    end

    if (cnet == 'y' || cnet == 'Y') && (cind == 'y' || cind == 'Y') && choice == 2;
        fprintf(fid, '.SUBCKT %s bias supply Gnd\n', paName);
        RFC = Lfeed*1e9;
        fprintf(fid, 'LRFC_2 supply N_1 %.2fn\n', RFC);
        fprintf(fid, 'CCapacitor_3 N_1 N_4 %.2fp\n', 1);
        if cmn == 0;
          fprintf(fid, 'XL0_1 N_2 Gnd Gnd L0\n');
          fprintf(fid, 'CCapacitor_1 N_2 Gnd\n');
          if harmonics == 1;
            fprintf(fid, 'XL3_1 N_4 N_2 Gnd L3\n');
            fprintf(fid, 'CCapacitor_2 N_4 N_2 %.2fp\n', floor(1e14 * C3 + 0.5) / 100);
          end
          if harmonics == 2;
            fprintf(fid, 'XL3_1 N_4 N_3 Gnd L3\n');
            fprintf(fid, 'CCapacitor_2 N_4 N_3 %.2fp\n', floor(1e14 * C3 + 0.5) / 100);
            fprintf(fid, 'XL5_1 N_3 N_2 Gnd L5\n');
            fprintf(fid, 'CCapacitor_4 N_3 N_2 %.2fp\n', floor(1e14 * C5 + 0.5) / 100);
          end
          fprintf(fid, 'RResistor_1 N_2 Gnd %.2f TC=0.0, 0.0\n', floor(100 * RL + 0.5)
/ 100);
        else
          fprintf(fid, 'XL0_1 N_M1 Gnd Gnd L0\n');
          fprintf(fid, 'CCapacitor_1 N_M1 Gnd %.2fp\n', floor(1e14 * C0 + 0.5) / 100);
          if harmonics == 1;
            fprintf(fid, 'XL3_1 N_4 N_M1 Gnd L3\n');
            fprintf(fid, 'CCapacitor_2 N_4 N_M1 %.2fp\n', floor(1e14 * C3 + 0.5) /
100);
          end
          if harmonics == 2;
            fprintf(fid, 'XL3_1 N_4 N_3 Gnd L3\n');
            fprintf(fid, 'CCapacitor_2 N_4 N_3 %.2fp\n', floor(1e14 * C3 + 0.5) / 100);
            fprintf(fid, 'XL5_1 N_3 N_M1 Gnd L5\n');
            fprintf(fid, 'CCapacitor_4 N_3 N_M1 %.2fp\n', floor(1e14 * C5 + 0.5) /
100);
          end
          fprintf(fid, 'RResistor_1 N_M2 Gnd %.2f TC=0.0, 0.0\n', 50);
        end
        fprintf(fid, transistor);
        fprintf(fid, '\n');
        matchSpiral; %Export matching inductors
        fprintf(fid, '.ENDS\n');
        fclose(fid);
        fprintf('File %s created successfully.\n\n', fileName);
    end
```

Figure A.16. MATLAB code for the subroutine used for netlist export (continued).

A.10 ENTERING PROCESS PARAMETERS

Figure A.17 and Figure A.18 show the MATLAB code for the subroutine that allows for entering process parameters (entProcParam.m).

```
%Enter process parameters
%Change default process parameters - any process can be used
%Enter default process parameters here
TM3default = 1000;
rhodefault = 2.82e-8;
MAldefault = 1.257e-6;
ToxM2M3default = 1000;
E = 8.85e-12;
EroxM2M3default = 4;
ToxM3Sdefault = 5000;
EroxM3Sdefault = 4;
TSidefault = 1000;
ErSidefault = 11.7;
rhoSidefault = 0.2;

ctech = input('Do you want to change any of the default process parameters (y/n)? ',
's');
if ctech == 'y' || ctech == 'Y'
    fprintf('\nPlease enter the thickness of the top metal (nm) (-1 to skip): \nOld
value: %.0f  -->  ', TM3default);
    TM3 = input('New value: ');
    if (TM3 <= 0)
      TM3 = TM3default * 1e-9;
    else
      TM3 = TM3 * 1e-9;
    end%if
    fprintf('Please enter rho (ohm.m) (-1 to skip): \nOld value: %.2e  -->  ',
rhodefault);
    rho = input('New value: ');
    if (rho <=0)
      rho = rhodefault;
    end%if
    fprintf('Please enter MAl (H/m) (-1 to skip): \nOld value: %.2e -->  ',
MAldefault);
    MAl = input('New value: ');
    if (MAl <= 0)
      MAl = MAldefault;
    end%if
    delta = sqrt(rho / (pi * MAl * f));
    teff = delta*(1-exp(-TM3/delta));
    fprintf('Please enter thickness of the oxide between top two metals (nm) (-1 to
skip): \nOld value: %.0f  -->  ', ToxM2M3default);
    ToxM2M3 = input('New value: ');
    if (ToxM2M3 <= 0)
      ToxM2M3 = ToxM2M3default * 1e-9;
    else
      ToxM2M3 = ToxM2M3 * 1e-9;
    end%if
    fprintf('Please enter Er of the oxide between top two metals (-1 to skip): \nOld
value: %.2f  -->  ', EroxM2M3default);
    EroxM2M3 = input('New value: ');
    if (EroxM2M3 <= 0)
      EroxM2M3 = EroxM2M3default;
    end%if
    EoxM2M3 = E * EroxM2M3;
    fprintf('Please enter thickness of the oxide between the substrate and the top
metal (nm) (-1 to skip): \nOld value: %.0f  -->  ', ToxM3Sdefault);
    ToxM3S = input('New value: ');
    if (ToxM3S <= 0)
      ToxM3S = ToxM3Sdefault * 1e-9;
    else
      ToxM3S = ToxM3S * 1e-9;
```

Figure A.17. MATLAB code for the subroutine used for entering process parameters.

```
    end%if
    fprintf('Please enter Er of the oxide between the substrate and the top metal (-1
to skip): \nOld value: %.2f  -->  ', EroxM3Sdefault);
    EroxM3S = input('New value: ');
    if (EroxM3S <= 0)
      EroxM3S = EroxM3Sdefault;
    end%if
    fprintf('Please enter TSi (um) (-1 to skip): \nOld value: %.0f  -->  ',TSidefault);
    TSi = input('New value: ');
    if (TSi <= 0)
      TSi = TSidefault * 1e-6;
    else
      TSi = TSi*1e-6;
    end%if
    fprintf('Please enter ErSi (-1 to skip): \nOld value: %.2f  -->  ', ErSidefault);
    ErSi = input('New value: ');
    if (ErSi <= 0)
      ErSi = ErSidefault;
    end%if
    %Rsi
    fprintf('Please enter rhoSi (ohm.m) (-1 to skip): \nOld value: %.2f  -->  ',
rhoSidefault);
    rhoSi = input('New value: ');
    if (rhoSi <= 0)
      rhoSi = rhoSidefault;
    end%if
else
    %If the parameters are not changed, use default parameters
    TM3 = TM3default * 1e-9;
    ToxM3S = ToxM3Sdefault * 1e-9;
    rho = rhodefault;
    MAl = MAldefault;
    delta = sqrt(rho / (pi * MAl * f));
    teff = delta*(1-exp(-TM3/delta));
    ToxM2M3 = ToxM2M3default * 1e-9;
    EroxM2M3 = EroxM2M3default;
    EoxM2M3 = E * EroxM2M3;
    EroxM3S = EroxM3Sdefault;
    TSi = TSidefault * 1e-6;
    ErSi = ErSidefault;
    rhoSi = rhoSidefault;
end%if
fprintf('\n');
```

Figure A.18. MATLAB code for the subroutine used for entering process parameters (continued).

A.11 INDUCTANCE SEARCH ALGORITHM

Figure A.19 and Figure A.20 show the MATLAB code for the inductance search algorithm (indSearch.m).

```
%This procedure searches for the inductance geometry with the highest
%quality factor given the inductance
%Initialize all storage variables to zero
  Qstored = 0;
  fostored = 0;
  Lcstored =0;
  Rsstored = 0;
  RSistored = 0;
  CSistored = 0;
  Coxstored = 0;
  Csstored = 0;
  wstored = 0;
  sstored = 0;
  dinstored = 0;
  doutstored = 0;
  nstored = 0;
  fprintf('\nLooking for the geometry with highest quality factor Q...\n\n');
  %Initialize geometry parameters to default minimum/maximum values
  Lc = 0;
  dout = 0;
  s = smin;
  din = dinmin;
  w = wmin;
  n = 2;
  %Inductance search algorithm
  while (din < 2*doutmax/3)
      s = smin;
          w = wmin;
          while (w <= doutmax/10)
              n = 2;
              dout = 0;
              while (dout < doutmax)
                  dout = din + 2*n*w + 2*(n-1)*s;
                  if (dout > doutmax)
                    break
                  end%if
                  davg = (din + dout) / 2;
                  Lc = b * dout^a1 * w^a2 * davg^a3 * n^a4 * s^a5;
                  calcParasitics; %Procedure to calculate parasitics
                  Lcc = Lc/1e9;
                  Lzz = Lz*1e9;
                  if (Lzz > Ls)
                      if (Lzz < (1 + tolerance) * Ls)
                          %Calculate Q-factor
                          Rp = 1/(omega^2*Cox^2*RSi) + RSi*(Cox + CSi)^2/Cox^2;
                          Cp = Cox*(1 + omega^2*(Cox + CSi)*CSi*RSi^2)/(1 +
omega^2*(Cox + CSi)^2*RSi^2);
                          Q = omega*Lcc/Rs*Rp/(Rp + ((omega*Lcc/Rs)^2 + 1)*Rs)*(1 - (Cp
+ Cs)*(omega^2*Lcc + Rs^2/Lcc));
                          fo = 1/(2*pi)*sqrt(1/(Lcc*(Cp + Cs)) - (Rs/Lcc)^2);
                          if (Q > Qstored)
                            Qstored = Q;
                            fostored = fo;
                            Lclfstored = Lc;
                            Lcstored = Lzz;
                            Rsstored = Rs;
                            RSistored = RSi;
                            CSistored = CSi;
                            Coxstored = Cox;
                            Csstored = Cs;
                            wstored = w;
                            sstored = s;
```

Figure A.19. MATLAB code for the inductance search algorithm.

```
                            dinstored = din;
                            doutstored = dout;
                            nstored = n;
                        end%if
                    end%if
                    Lc = 0;
                    n = 1;
                    break
                end%if
                n = n + 1;
            end%while
            w = w + resolution;
        end%while
    %end%while
    din = din + resolution;
  end%while
  %==== OUTPUT PARAMS ====%
  if Qstored < 1
      fprintf('Could not find a geometry for %.2f nH\nlimited to dinmin and doutmax
with Q greater than 1 at %.2f MHz.\n', Ls, f/1e6)
  end%if
  if Qstored >=1
    fprintf('Ls = %.2f nH \n', Lcstored);
    fprintf('Lslf = %.2f nH \n', Lclfstored);
    fprintf('Q = %.2f \n', floor(100 * Qstored + 0.5) / 100);
    fprintf('fo = %.2f GHz\n', floor(fostored/1e7 + 0.5) / 100);
    fprintf('w = %.2f um\n', floor(100*wstored + 0.5) / 100);
    fprintf('s = %.2f um\n', floor(100*sstored + 0.5) / 100);
    fprintf('din = %.2f um\n', floor(100*dinstored + 0.5) / 100);
    fprintf('dout = %.2f um\n', floor(100*doutstored + 0.5) / 100);
    fprintf('n = %d\n', floor(100*nstored + 0.5) / 100);
  end%if
```

Figure A.20. MATLAB code for the inductance search algorithm (continued).

A.12 EXPORT OF THE SPIRAL INDUCTOR SUBCIRCUIT

Figure A.21 shows the MATLAB code for the export of the netlist of the spiral inductor as a part of the complete PA netlist (exportSubckt.m).

```
%Exports the spiral inductor subcircuit
try
   fprintf(fid, '.SUBCKT %s L1 L2 GND\n', inductorName);
    fprintf(fid, 'CS L1 L2 %.2ffF\n', floor(1e17 * Csstored + 0.5) / 100);
    fprintf(fid, 'LS N4 L1 %.2fn\n', floor(100 * Lclfstored + 0.5) / 100);
    fprintf(fid, 'RS N4 L2 %.2f\n', floor(100 * Rsstored + 0.5) / 100);
    fprintf(fid, 'Csi1 N2 GND %.2ffF\n', floor(1e17 * CSistored + 0.5) / 100);
    fprintf(fid, 'Csi2 N3 GND %.2ffF\n', floor(1e17 * CSistored + 0.5) / 100);
    fprintf(fid, 'Cox1 L1 N2 %.2ffF\n', floor(1e17 * Coxstored + 0.5) / 100);
    fprintf(fid, 'Cox2 L2 N3 %.2ffF\n', floor(1e17 * Coxstored + 0.5) / 100);
    fprintf(fid, 'Rsi1 N2 GND %.2f\n', floor(100 * RSistored + 0.5) / 100);
    fprintf(fid, 'Rsi2 N3 GND %.2f\n', floor(100 * RSistored + 0.5) / 100);
    fprintf(fid, '.ENDS\n\n');
catch
    fprintf('MATLAB reports:\n%s\n', lasterr);
    fprintf('Netlist export failed.\n');
end%try
```

Figure A.21. MATLAB code for the export of the spiral inductor subcircuit.

A.13 EXPORT OF THE PART OF NETLIST INVOLVING MATCHING

Figure A.22 shows the MATLAB code for the export of the part of netlist that involves matching when ideal inductors are used (matchIdeal.m). Figure A.23 shows the MATLAB

code for the export of the part of netlist that involves matching when spiral inductors are used (matchSpiral.m).

```
if cmn == 1;
    fprintf(fid, 'LInductor_M Gnd N_M2 %.2fn\n', floor(1e11 * LM + 0.5) / 100);
    fprintf(fid, 'CCapacitor_M N_M1 N_M2 %.2fp\n', floor(1e14 * CM + 0.5) / 100);
elseif cmn == 2;
    fprintf(fid, 'LInductor_M1 N_M1 N_M %.2fn\n', floor(1e12 * LM1 + 0.5) / 1000);
    fprintf(fid, 'LInductor_M2 N_M N_M2 %.2fn\n', floor(1e12 * LM2 + 0.5) / 1000);
    fprintf(fid, 'CCapacitor_M1 Gnd N_M %.2fp\n', floor(1e15 * CM1 + 0.5) / 1000);
elseif cmn == 3
    fprintf(fid, 'CCapacitor_M1 N_M1 N_M %.2fp\n', floor(1e15 * CM2 + 0.5) / 1000);
    fprintf(fid, 'CCapacitor_M2 N_M N_M2 %.2fp\n', floor(1e15 * CM3 + 0.5) / 1000);
    fprintf(fid, 'LInductor_M1 Gnd N_M %.2fn\n', floor(1e12 * LM3 + 0.5) / 1000);
elseif cmn == 4;
    fprintf(fid, 'LInductor_M1 Gnd N_M1 %.2fn\n', floor(1e12 * LM4 + 0.5) / 1000);
    fprintf(fid, 'LInductor_M2 Gnd N_M2 %.2fn\n', floor(1e12 * LM5 + 0.5) / 1000);
    fprintf(fid, 'CCapacitor_M1 N_M2 N_M1 %.2fp\n', floor(1e15 * CM4 + 0.5) / 1000);
elseif cmn == 5;
    fprintf(fid, 'CCapacitor_M1 Gnd N_M1 %.2fp\n', floor(1e15 * CM5 + 0.5) / 1000);
    fprintf(fid, 'CCapacitor_M2 Gnd N_M2 %.2fp\n', floor(1e15 * CM6 + 0.5) / 1000);
    fprintf(fid, 'LInductor_M1 N_M2 N_M1 %.2fn\n', floor(1e12 * LM6 + 0.5) / 1000);
end
```

Figure A.22. MATLAB code for the export of the part of the netlist that involves matching when ideal inductors are used.

```
if cmn == 1;
    fprintf(fid, 'XLM_1 Gnd N_M2 Gnd LM\n');
    fprintf(fid, 'CCapacitor_M N_M1 N_M2 %.2fp\n', floor(1e14 * CM + 0.5) / 100);
elseif cmn == 2;
    fprintf(fid, 'XLM1_1 N_M1 N_M Gnd LM1\n');
    fprintf(fid, 'XLM2_1 N_M N_M2 Gnd LM2\n');
    fprintf(fid, 'CCapacitor_M1 Gnd N_M %.2fp\n', floor(1e15 * CM1 + 0.5) / 1000);
elseif cmn == 3
    fprintf(fid, 'CCapacitor_M1 N_M1 N_M %.2fp\n', floor(1e15 * CM2 + 0.5) / 1000);
    fprintf(fid, 'CCapacitor_M2 N_M N_M2 %.2fp\n', floor(1e15 * CM3 + 0.5) / 1000);
    fprintf(fid, 'XLM3_1 Gnd N_M Gnd LM3\n');
elseif cmn == 4;
    fprintf(fid, 'XLM4_1 Gnd N_M1 Gnd LM4\n');
    fprintf(fid, 'XLM5_1 Gnd N_M2 Gnd LM5\n');
    fprintf(fid, 'CCapacitor_M1 N_M1 N_M2 %.2fp\n', floor(1e15 * CM4 + 0.5) / 1000);
elseif cmn == 5;
    fprintf(fid, 'CCapacitor_M1 Gnd N_M1 %.2fp\n', floor(1e15 * CM5 + 0.5) / 1000);
    fprintf(fid, 'CCapacitor_M2 Gnd N_M2 %.2fp\n', floor(1e15 * CM6 + 0.5) / 1000);
    fprintf(fid, 'XLM6_1 N_M1 N_M2 Gnd LM6\n');
end
```

Figure A.23. MATLAB code for the export of the part of the netlist that involves matching when spiral inductors are used.

A.14 CALCULATING PARASITICS OF ANY DESIGNED SPIRAL INDUCTOR

Figure A.24 shows the MATLAB code for the subroutine that calculates the parasitics of any designed spiral inductor (calcParasitics.m).

```
%This procedure calculates the parasitics of spiral inductors
ell = 4*n*(din+w) + (2*n)*(2*n-1) * (w+s);
Rs = rho * ell / (w * teff);
ws = w / 1e6;
ells = ell / 1e6;
Cs = n * ws * ws * EoxM2M3 / ToxM2M3;
if (ToxM3S <= ws)
   ToxM1Seff = ws * (ws/ToxM3S + 2.42 - 0.44*ToxM3S/ws + (1 - ToxM3S/ws)^6)^(-1);
else
   ToxM1Seff = ws/(2*pi) * log(8*ToxM3S/ws + 4*ws/ToxM3S);
end%if
EoxM1Seff = E *((EroxM3S + 1)/2 + (EroxM3S - 1)/2*(1 + 10*ToxM3S/ws)^(-1/2));
Cox = ells * ws * EoxM1Seff / ToxM1Seff / 2;
TSieffC = ws/(2*pi) * log(8*TSi/ws + 4*ws/TSi);
ESieff = E *((ErSi + 1)/2 + (ErSi - 1)/2*(1 + 10*TSi/ws)^(-1/2));
CSi = ells * ws * ESieff / TSieffC / 2;
sigmaeff = 1/rhoSi * (1/2 + 1/2*(1 + 10*TSi/ws)^(-1/2));
RSi = 2 * TSieffC / (ells * ws * sigmaeff);

%Effective Ls
%Complex number calculations
% i = sqrt(-1)
ZSi = 1/(i*omega*CSi);
Zox = 1/(i*omega*Cox);
Zs = 1/(i*omega*Cs);
Lss = Lc / 1e9;
ZLs = i*omega*Lss;
ZA = Zox + ZSi * RSi / (ZSi + RSi);
ZB = Zs * ( ZLs + Rs) / (Zs + ZLs + Rs);
Z = ZA * ZB / (ZA + ZB);
R = real(Z);
Lz = imag(Z) / omega;
```

Figure A.24. MATLAB code for the subroutine that calculates the parasitics of any designed spiral inductor.

A.15 DESIGN OF SPIRAL INDUCTORS AS A MAIN ROUTINE

Figure A.25 shows the MATLAB code for the routine for Design of Spiral Inductors (indcalc2.m).

```
%This program finds a geometry for given inductance or calculates the inductance for
%given geometry. GDS and CIF file extraction is included as well as netlist
%extraction.
try
    clear all; %Clear all variables from the workspace
    %Define constants
    b = 1.62e-3;
    a1 = -1.21;
    a2 = -0.147;
    a3 = 2.40;
    a4 = 1.78;
    a5 = -0.030;
    fprintf('\nWelcome to Inductor Calculator v2.2 for Matlab. Please follow the
prompts.\n');
    choice = input('\nPlease enter 1 to specify geometry or 2 to specify inductance:
');
    if choice == 1 || choice == 2
      if choice == 2 %If choice is 2, program finds a geometry for given inductance
          Ls = input('Please enter the wanted inductance value (Ls) (nH): '); %Wanted
inductance
          indSetup;
          fprintf('\n');
          f = input('Please enter the operating frequency (MHz): ')*1e6; %Frequency
          omega = (2*pi*f); %Radian frequency
          entProcParam; %Call the procedure for entering process parameters
          indSearch; %Inductance search algorithm
      end%if
      if choice == 1
          indGeom;
      end%if
      %Export the netlist
      cnet = input('\nDo you want to export the netlist of this inductor into a SPC
file (y/n)? ', 's');
      if cnet == 'y' || cnet == 'Y';
        exportNetlist;
      end%if
      %Export the layout
      cwcif = input('\nDo you want to export the layout of this inductor into a CIF or
GDS file (c - CIF, g - GDS, n - none)? ', 's');
      if cwcif == 'c' || cwcif == 'C';
        writecif; %Export layout into a text (CIF) file
      elseif cwcif == 'g' || cwcif == 'G';
        convertGDS; %Export layout into a stream (GDS) file
      end
      fprintf('\n');
    else
      fprintf('Wrong choice. Please run the program again.\n\n');
    end%if
catch
    fprintf('MATLAB returns: \n%s\nError occurred while running the program. Please try
running the program again.\n\n', lasterr);
end%try
```

Figure A.25. MATLAB code for the routine for design of spiral inductors.

A.16 DESIGN OF INDUCTORS WITH KNOWN GEOMETRY

Figure A.26 shows the MATLAB code for the subroutine that allows for design of spiral inductors with known geometry (indGeom.m).

```
%This procedure calculates the inductance and quality factor values for any
%given square inductor geometry
%Input geometry parameters
dout = input('Please enter the dout diameter (dout) (um): ');
din = input('Please enter the din diameter (din) (um): ');
w = input('Please enter the turn width (w) (um): ');
n = input('Please enter the number of turns: ');
ratio = floor(100*(din / dout) + 0.5)/100;
%Perform basic checks
nmin = (dout/2 - din/2 + 1)/(w + 1);
while (nmin < 2) || (n > nmin) || (n < 2)
    if (nmin <2)
        fprintf('Given geometry results in n smaller than 2. Please re-enter the
geometry: \n');
    end%if
    if (n > nmin)
        fprintf('The number of turns (n) is too small for the chosen geometry. Please
re-enter the geometry: \n');
    end%if
    if (n < 2)
        fprintf('Cannot use less than two turns. Please choose n as 2 or greater.
Please re-enter the geometry: \n');
    end%if
    dout = input('Please enter the dout diameter (dout) (um): ');
    din = input('Please enter the din diameter (din) (um): ');
    w = input('Please enter the turn width (w) (um): ');
    n = input('Please enter the number of turns: ');
    ratio = floor(100*(din / dout) + 0.5)/100;
    nmin = (dout/2 - din/2 + 1)/(w + 1);
end%while
%Enter operating frequency
f = input('Please enter the operating frequency (MHz): ')*1e6;
omega = (2*pi*f); %Radian frequency
s = (dout / 2 - din / 2 - n * w) / (n - 1); %Pitch between the turns of the spiral
davg = (din + dout) / 2; %Average diameter
Lc = b * dout^a1 * w^a2 * davg^a3 * n^a4 * s^a5; %Inductance
entProcParam; %Call the procedure for entering process parameters
calcParasitics; %Call the procedure that calculates parasitics and Q factor
Lcc = Lc/1e9;
Lzz = Lz*1e9;
%Calculate Q-factor
Rp = 1/(omega^2*Cox^2*RSi) + RSi*(Cox + CSi)^2/Cox^2;
Cp = Cox*(1 + omega^2*(Cox + CSi)*CSi*RSi^2)/(1 + omega^2*(Cox + CSi)^2*RSi^2);
Q = omega*Lcc/Rs*Rp/(Rp + ((omega*Lcc/Rs)^2 + 1)*Rs)*(1 - (Cp + Cs)*(omega^2*Lcc +
Rs^2/Lcc));
fo = 1/(2*pi)*sqrt(1/(Lcc*(Cp + Cs)) - (Rs/Lcc)^2);
%Outputs
fprintf('Ls = %.2f nH \n', Lzz);
fprintf('Lslf = %.2f nH \n', Lc);
fprintf('Q = %.2f \n', floor(100 * Q + 0.5) / 100);
fprintf('fo = %.2f GHz\n', floor(fo/1e7 + 0.5) / 100);
fprintf('w = %.2f um\n', floor(100*w + 0.5) / 100);
fprintf('s = %.2f um\n', floor(100*s + 0.5) / 100);
fprintf('din = %.2f um\n', floor(100*din + 0.5) / 100);
fprintf('dout = %.2f um\n', floor(100*dout + 0.5) / 100);
fprintf('n = %d\n', floor(100*n + 0.5) / 100);
%Store the parasitic parameters for netlist extraction
Lcstored = Lzz;
Lclfstored = Lc;
Rsstored = Rs;
RSistored = RSi;
CSistored = CSi;
Coxstored = Cox;
Csstored = Cs;
nstored = n;
doutstored = dout;
wstored = w;
sstored = s;
```

Figure A.26. MATLAB code subroutine for design of spiral inductors with known geometry.

A.17 EXPORT OF THE STAND-ALONE NETLIST OF A SPIRAL INDUCTOR

Figure A.27 shows the MATLAB code for the subroutine that exports the stand-alone netlist of a designed spiral inductor (exportNetlist.m).

```
%This procedure exports the netlist of a spiral inductor
try
    inductorName = input('Please enter the inductor name (press "s" to skip): ', 's');
    if inductorName(1) == 's' || inductorName(1) == 'S';
        inductorName = 'MyInductor';
    end%if
    fileName = strcat(inductorName, '_net.spc');
    fid = fopen(fileName, 'w')    ;
    fprintf(fid, '*SUBCKT gen. by Inductor Calculator v2.2\n');
    fprintf(fid, '.SUBCKT %s L1 L2 GND\n', inductorName);
    fprintf(fid, 'CS L1 L2 %.2ffF\n', floor(1e17 * Csstored + 0.5) / 100);
    fprintf(fid, 'LS N4 L1 %.2fn\n', floor(100 * Lclfstored + 0.5) / 100);
    fprintf(fid, 'RS N4 L2 %.2f\n', floor(100 * Rsstored + 0.5) / 100);
    fprintf(fid, 'Csi1 N2 GND %.2ffF\n', floor(1e17 * CSistored + 0.5) / 100);
    fprintf(fid, 'Csi2 N3 GND %.2ffF\n', floor(1e17 * CSistored + 0.5) / 100);
    fprintf(fid, 'Cox1 L1 N2 %.2ffF\n', floor(1e17 * Coxstored + 0.5) / 100);
    fprintf(fid, 'Cox2 L2 N3 %.2ffF\n', floor(1e17 * Coxstored + 0.5) / 100);
    fprintf(fid, 'Rsi1 N2 GND %.2f\n', floor(100 * RSistored + 0.5) / 100);
    fprintf(fid, 'Rsi2 N3 GND %.2f\n', floor(100 * RSistored + 0.5) / 100);
    fprintf(fid, '.ENDS\n\n');
    fclose(fid);
    fprintf('File %s created successfully.\n', fileName);
catch
    fprintf('MATLAB reports:\n%s\n', lasterr);
    fprintf('Netlist export failed.\n');
end%try
```

Figure A.27. MATLAB code for the subroutine that exports the stand-alone netlist of a spiral inductor.

A.18 EXTRACTION OF INDUCTOR LAYOUT INTO A CIF FILE

Figure A.28 though Figure A.30 show the MATLAB code for the subroutine for extraction of the layout of a designed inductor into a CIF file (writecif.m).

```
%This procedure exports the layout of an inductor into a text based (CIF)
%file
try
    ccif = input('Do you want to change any CIF paramaters (y/n)? ', 's');
    if ccif == 'y' || ccif == 'Y'
        %Change default CIF parameters
        scaling = input('Please enter CIF scaling factor (-1 to skip): ');
        if scaling <= 0;
            scaling = 2000;
        end%if
        viaSpacing = input('Please enter via spacing (-1 to skip): ');
        if viaSpacing <= 0;
            viaSpacing = 1;
        end%if
        viaWidth = input('Please enter via width (-1 to skip): ');
        if viaWidth <= 0;
            viaWidth = 0.5;
```

Figure A.28. MATLAB code for the subroutine for inductor extraction into a CIF file.

```
        end%if
    else
       %Default CIF parameters
        scaling = 2000;
        viaSpacing = 1;
        viaWidth = 0.5;
    end%if
    %Calculate usable parameters
    dout2 = round(doutstored*scaling);
    zero = 0;
    w2 = round(wstored * scaling);
    s2 = round(sstored * scaling);
    n = nstored;
    spacing = viaSpacing * scaling;
    width = viaWidth * scaling;
    %Specify the inductor name and open the file
    cellName = input('Please enter the cell (inductor) name (press "s" to skip): ',
's');
    if cellName(1) == 's' || cellName(1) == 'S';
        cellName = 'mycell';
    end%if
    fileName = strcat(cellName, '_lay.cif');
    fid = fopen (fileName, 'w');
    %Print into the file
    %PRINT THE HEADER
    fprintf(fid,'(CIF written by the Inductance Calculator MATLAB version);\n');
    fprintf(fid,'(Version: 2.10);\n');
    fprintf(fid,'(DATE: %s);\n',date);
    fprintf(fid,'(FABCELL: %s  %d x %d Microns);\n',cellName, dout, dout);
    fprintf(fid,'(L-Edit Layer NTUB = CIF Layer NTUB);\n');
    fprintf(fid,'(L-Edit Layer DIFF = CIF Layer DIFF);\n');
    fprintf(fid,'(L-Edit Layer PPLUS = CIF Layer PPLUS);\n');
    fprintf(fid,'(L-Edit Layer CONT = CIF Layer CONT);\n');
    fprintf(fid,'(L-Edit Layer MET1 = CIF Layer MET1);\n');
    fprintf(fid,'(L-Edit Layer MET2 = CIF Layer MET2);\n');
    fprintf(fid,'(L-Edit Layer VIA2 = CIF Layer VIA2);\n');
    fprintf(fid,'(L-Edit Layer MET3 = CIF Layer MET3);\n');
    fprintf(fid,'(L-Edit Layer VIA3 = CIF Layer VIA3);\n');
    fprintf(fid,'(L-Edit Layer MET4 = CIF Layer MET4);\n');
    fprintf(fid,'(L-Edit Layer FIMP = CIF Layer FIMP);\n');
    fprintf(fid,'(L-Edit Layer TEXT = CIF Layer TEXT);\n');
    fprintf(fid,'(L-Edit Layer SFCDEF = CIF Layer SFCDEF);\n');
    fprintf(fid,'(SCALING: 1 CIF Unit = 1/%d Microns);\n',scaling);
    fprintf(fid,'DS 1 2 40;\n');
    fprintf(fid,'9 %s;\n', cellName);
    fprintf(fid,'94 P1 %d,0 TEXT;\n', w2/2);
    fprintf(fid,'94 P2 %d,0 TEXT;\n', dout2 - (n-0.5)*w2 - (n-1)*s2);
    fprintf(fid,'94 %s %d,%d SFCDEF;\n', cellName, dout2/2, dout2/2);

    %PRINT LOWER METAL
    fprintf(fid,'L MET2;\n');
    fprintf(fid,'B %d %d %d,%d;\n',w2, w2*(n + 1) + s2*n, dout2 - n*w2 - (n - 1)*s2 +
w2/2, (w2*(n + 1) + s2*n)/2);

    %PRINT VIAS
    fprintf(fid,'L VIA2;\n');
    x = floor(w2 / (spacing + width));
```

Figure A.29. MATLAB code for the subroutine for inductor extraction into a CIF file (continued).

```
    y = floor((2*w2 + s2) / (spacing + width));
    for i=0:x-1;
        for j=0:y-1;
            fprintf(fid,'B %d %d %d,%d; \n', width, width, dout2 - n*w2 - (n - 1)*s2 +
(spacing + width)*(i + 0.5), (n-1)*w2 + (n-1)*s2 + (spacing + width)*(j + 0.5) );
        end%for
    end%for

    %PRINT HIGHER METAL
    fprintf(fid,'L MET3;\n');
    fprintf(fid,'P ');
    %inner points
    %first turn
    fprintf(fid,'%d,%d\n', w2, zero);
    fprintf(fid,'%d,%d\n', w2, dout2 - w2);
    fprintf(fid,'%d,%d\n', dout2 - w2, dout2 - w2);
    fprintf(fid,'%d,%d\n', dout2 - w2, w2);
    %inner turns
    for turn = 2:n-1;
        fprintf(fid,'%d,%d\n', turn*w2 + (turn-1)*s2, (turn-1)*w2 + (turn-2)*s2);
        fprintf(fid,'%d,%d\n', turn*w2 + (turn-1)*s2, dout2 - (turn-1)*s2 - turn*w2);
        fprintf(fid,'%d,%d\n', dout2 - (turn-1)*s2 - turn*w2, dout2 - (turn-1)*s2 -
turn*w2);
        fprintf(fid,'%d,%d\n', dout2 - (turn-1)*s2 - turn*w2, turn*w2 + (turn-1)*s2);
    end%for
    %last turn
    turn = n;
    fprintf(fid,'%d,%d\n', turn*w2 + (turn-1)*s2, (turn-1)*w2 + (turn-2)*s2);
    fprintf(fid,'%d,%d\n', turn*w2 + (turn-1)*s2, dout2 - (turn-1)*s2 - turn*w2);
    fprintf(fid,'%d,%d\n', dout2 - (turn-1)*s2 - turn*w2, dout2 - (turn-1)*s2 -
turn*w2);
    fprintf(fid,'%d,%d\n', dout2 - (turn-1)*s2 - turn*w2, (turn-1)*w2 + (turn-1)*s2);
    fprintf(fid,'%d,%d\n', dout2 - (turn-1)*s2 - (turn-1)*w2, (turn-1)*w2 + (turn-
1)*s2);
    fprintf(fid,'%d,%d\n', dout2 - (turn-1)*s2 - (turn-1)*w2, dout2 - (turn-1)*w2 -
(turn-1)*s2);
    fprintf(fid,'%d,%d\n', (turn-1)*w2 + (turn-1)*s2, dout2 - (turn-1)*w2 - (turn-
1)*s2);
    fprintf(fid,'%d,%d\n', (turn-1)*w2 + (turn-1)*s2, (turn-2)*w2 + (turn-2)*s2);
    %inner turns
    for turn = n-1:-1:2;
        fprintf(fid,'%d,%d\n', dout2 - (turn-1)*s2 - (turn-1)*w2, (turn-1)*w2 + (turn-
1)*s2);
        fprintf(fid,'%d,%d\n', dout2 - (turn-1)*s2 - (turn-1)*w2, dout2 - (turn-1)*w2 -
(turn-1)*s2);
        fprintf(fid,'%d,%d\n', (turn-1)*s2 + (turn-1)*w2, dout2 - (turn-1)*w2 - (turn-
1)*s2);
        fprintf(fid,'%d,%d\n', (turn-1)*s2 + (turn-1)*w2, (turn-2)*w2 + (turn-2)*s2);
    end%for
    %first turn
    fprintf(fid,'%d,%d\n', dout2, zero);
    fprintf(fid,'%d,%d\n', dout2, dout2);
    fprintf(fid,'%d,%d\n', zero, dout2);
    fprintf(fid,'%d,%d;\n', zero, zero);

    %PRINT FOOTER
    fprintf(fid,'DF;\n');
    fprintf(fid,'C 1;\n');
    fprintf(fid,'E');

    %Close the file
    fclose(fid);
    fprintf('File %s created successfully.\n', fileName');
catch
    fprintf('MATLAB returns:\n%s\n',lasterr);
    fprintf('Could not export to the file.\n');
end%try
```

Figure A.30. MATLAB code for the subroutine for inductor extraction into a CIF file (continued).

A.19 EXTRACTION OF INDUCTOR LAYOUT INTO A GDSII FILE

Extraction of inductor layout into stream (GDSII file) required tedious number format and number-to-string conversions and string manipulation. The routine (convertGDS.m) and its subroutines (str2hex.m, coordinatesSpiral.m, coordinatesUnderpass.m, coordinatesVia.m, coordinatesP1.m, coordinatesP2.m and coordinatesName.m) total to about 500 lines making it impractically long to be included in this appendix. The complete MATLAB code can be purchased from Business Enterprises at the University of Pretoria. For details, please see Appendix C.

A.20 MATCHING AS A MAIN ROUTINE

Figure A.31 shows the MATLAB code for the routine that performs matching between any two real impedances (match.m).

```
%Calculates the match between real source and load given the frequency of
%operation and bandwidth (for narrowband networks)
clear all;

RS = input('Please enter the value for the source resistance RS (ohm): ');
RL = input('Please enter the value for the load resistance RL (ohm): ');
fo = input('Please enter the operating frequency (MHz): ');
BW = input('Please enter the bandwidth (MHz): ');

fo = fo * 1e6;
BW = BW * 1e6;
wo = 2*pi*fo;
QL = fo/BW;

calcMatch;
```

Figure A.31. MATLAB code for the routine that performs matching between any two real impedances.

APPENDIX B PROCESS FILES FOR EM SIMULATIONS

B.1 INTRODUCTION

Appendix B gives the process files required for running the EM simulations of the designed square spiral inductors as stipulated in Section 5.2.

B.2 PROCESS FILE FOR 3M AMS S35 TECHNOLOGY

Figure B.1 shows the process file for 3M variation of the AMS S35 technology. Layer thicknesses are not disclosed to the reader due to the NDA between the author (via University of Pretoria) and AMS.

```
processType CMOS
numLayers 9

layer 0
        name tab
        type ground
        thickness 0.1e-6
        epsr 1
        rho 1.7857e-8
layer 1
        name substrate
        type dielectric
        thickness 725e-6
        epsr 11.7
        rho 0.19
layer 2
        name sio2
        type dielectric
        thickness 1e-6
        epsr 4.0
        rho 1e10
layer 3
        name metal1
        type metal
        thickness 1e-6
        epsr 4
        rho 1e10
        rhom 2.82e-8
layer 4
        name sio2
        type dielectric
        thickness 1e-6
        epsr 4
        rho 1e10
layer 5
        name metal2
        type metal
        thickness 1e-6
        epsr 4
        rho 1e10
        rhom 2.82e-8
layer 6
        name sio2
        type dielectric
        thickness 1e-6
        epsr 4
        rho 1e10
layer 7
        name metal3
        type metal
        thickness 1e-6
        epsr 4
        rho 1e10
        rhom 2.82e-8
layer 8
        name sio2
        type dielectric
        thickness 1e-6
        epsr 4
        rho 1e10
```

Figure B.1. Process file for 3M variation of the AMS S35 technology.

B.3 PROCESS FILE FOR TM AMS S35 TECHNOLOGY

Figure B.2 shows the process file for TM variation of the AMS S35 technology. Layer thicknesses are not disclosed to the reader due to the NDA between the author and AMS.

```
processType CMOS
numLayers 11

layer 0
        name tab
        type ground
        thickness 0.1e-6
        epsr 1
        rho 1.7857e-8
layer 1
        name substrate
        type dielectric
        thickness 725e-6
        epsr 11.7
        rho 0.19
layer 2
        name sio2
        type dielectric
        thickness 1e-6
        epsr 4.0
        rho 1e10
layer 3
        name metal1
        type metal
        thickness 1e-6
        epsr 4
        rho 1e10
        rhom 2.82e-8
layer 4
        name sio2
        type dielectric
        thickness 1e-6
        epsr 4
        rho 1e10
layer 5
        name metal2
        type metal
        thickness 1e-6
        epsr 4
        rho 1e10
        rhom 2.82e-8
layer 6
        name sio2
        type dielectric
        thickness 1e-6
        epsr 4
        rho 1e10
layer 7
        name metal3
        type metal
        thickness 1e-6
        epsr 4
        rho 1e10
        rhom 2.82e-8
layer 8
        name sio2
        type dielectric
        thickness 1e-6
        epsr 4
        rho 1e10
layer 9
        name metal4
        type metal
        thickness 1e-6
        epsr 4
        rho 1e10
        rhom 2.82e-8
layer 10
        name sio2
        type dielectric
        thickness 1e-6
        epsr 3.9
        rho 1e10
```

Figure B.2. Process file for TM variation of the AMS S35 technology.

APPENDIX C DESIGN METHOD – BE AT UP (PTY) LTD

The resulting design method was implemented as a MATLAB script, and extended to the national and international scientific community via Business Enterprises at the University of Pretoria (BE at UP (Pty) ltd). The extension is offered via an online page: http://www.be.up.co.za/pa-inductance-calculator/. A screen capture (accessed: 24 February 2011) of this page can be seen in Figure C.1.

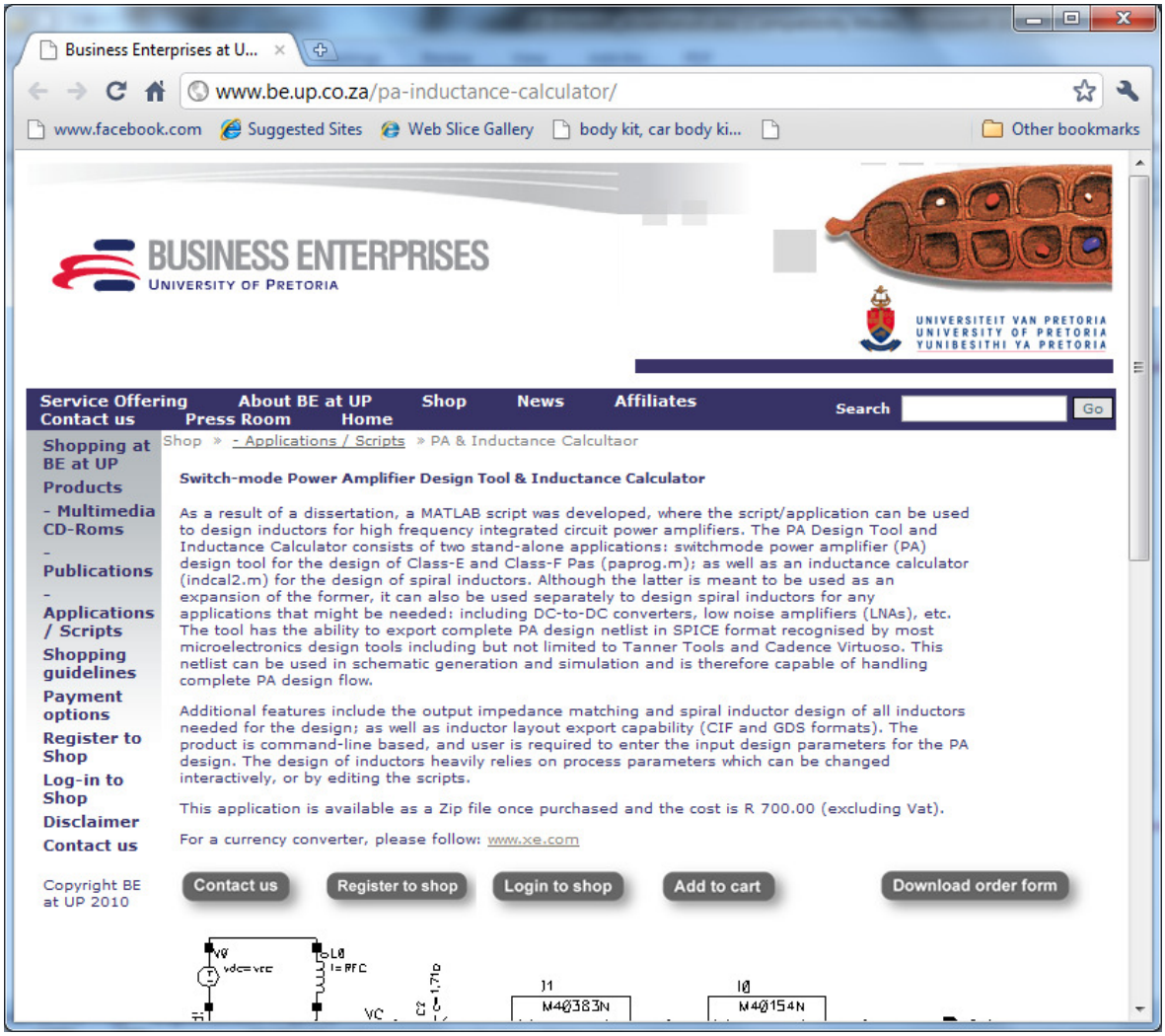

Figure C.1. An offering to the scientific community: Switch-mode Power Amplifier Design Tool & Inductance Calculator.