TABLE IV
EXPERIMENTAL RESULTS FOR THE IP ROUTING DATABASE (VIRTEX-6)

Implementation	Avg. Freq. (MHz)	Avg. # of EMBs	Avg. # of LUTs
BTFVSM-EMB	286	131	446
BTFVSM-LUT	170	0	31950
ROM-EMB	112	101	19

sub-FSMs, the number of sub-FSM with less than six states may be significant, resulting in a wastage of memory. Second, the configuration of the depth of EMBs is limited to powers of two with a minimum value of 512 [14]; therefore, part of the address space of some EMBs is wasted when the depth of the ROM to be implemented does not match the available configurations of EMBs (we will refer to this effect as *memory fragmentation* [11]). This fragmentation particularly affects BTFVSM-EMB because each one of the four blocks of S-memory has less depth than the ROM of ROM-EMB and than the S-memory of GFVSM-EMB. In addition, S-memory blocks of BTFVSM-EMB usually have a very large width, so the number of EMBs that suffer memory fragmentation is high. For example, in the BT-FSM with the least number of states, ROM-EMB requires a ROM of 292×15 bits; so, only one half EMB configured as 512×16 is used. The S-memory of GFVSM-EMB is a ROM of 64×90 bits, which requires 5 half EMBs configured as 512×36 (i.e., a total of 2,5 EMBs). Finally, each one of the four S-memory blocks of BTFVSM-EMB is a ROM of 8×192 bits, which requires six EMBs configured as 512×36 (making a total of 24 EMBs).

The relative influence of the memory fragmentation in the EMB usage decreases with depth; this explains that the reduction reached by ROM-EMB respect to BTFVSM-EMB decreases with the number of states (the average reduction is 92% for the cases in the first half of the graph whereas this value is 42% in the cases of the second half). On the other hand, the size of S-memory of GFVSM-EMB is on average a 17% greater than that of BTFVSM-EMB, so it is expected that BTFVSM-EMB uses a less number of EMBs if the number of states is large enough.

With the aim of reducing the effect of the memory fragmentation in BTFSVM-EMB implementations, two S-memory blocks with depth less than 256 words could be mapped into the same EMBs by exploiting the dual-port feature available on EMBs [14]. In this case, the average EMB reduction respect to BTFVSM-EMB could be decreased to 59% in the case of ROM-EMB and to 55% in the case of GFVSM-EMB.

D. Results for the BT-FSMs Obtained from the IP Routing Database

The average values of the post place-and-route results are summarized in Table IV. BTFVSM-EMB is faster than BTFVSM-LUT in all cases. This confirms the trend observed in the results obtained for synthetic test benches. The average speed improvement of BTFVSM-EMB respect to BTFVSM-LUT is 72%. On the other hand, both implementations of the BT-FVSM architecture are faster than ROM-EMB in all cases. The average speed improvement of BTFVSM-EMB and BTFVSM-LUT respect to ROM-EMB are 155% and 53%, respectively.

VI. CONCLUSION

In this paper, the BT-FVSM architecture has been proposed. The goal of this architecture is to achieve high-speed implementations of BT-FSMs. The differences between the proposed architecture and the general FVSM architecture have been described in detail. An experimental study that include BT-FVSM, general FVSM, and conventional implementations has been presented. Both synthetic BT-FSMs and BT-FSMs obtained from an IP routing database have been used.

In synthetic test benches, the average speed improvement of the proposed architecture with respect to the best results of the other approaches (including the general FVSM) is 41% (there are some cases in which the speed is more than double). In the case of IP routing database, the average speed improvement achieves 155%. Therefore, considering all test benches, BTFVSM-LUT is the best option if EMBs must be preserved (FSM-LUT is slightly better only for BT-FSMs with less than 512 states), otherwise BTFVSM-EMB is the best option for BT-FSMs with a large number of states.

As future work, we plan to improve the area and speed results by using a more complex algorithm for joining incomplete sub-FSMs. In addition, we are going to generate BT-FVSM implementations that use the dual-port available on EMBs in order to reduce memory fragmentation.

REFERENCES

[1] M. P. Desai, H. Narayanan, and S. B. Patkar, “The realization of finite state machines by decomposition and the principal lattice of partitions of a submodular function,” *Discrete Appl. Math.*, vol. 131, no. 2, pp. 299–310, Sep. 2003.

[2] M. Desai, R. Gupta, A. Karandikar, K. Saxena, and V. Samant, “Reconfigurable finite-state machine based IP lookup engine for high-speed router,” *IEEE J. Sel. Areas Commun.*, vol. 21, no. 4, pp. 501–512, May 2003.

[3] J. Li, Y. Chen, C. Ho, and Z. Lu, “Binary-tree-based high speed packet classification system on FPGA,” in *Proc. Int. Conf. Inf. Netw.*, Jan. 2013, pp. 517–522.

[4] S. Beak *et al.*, “Novel binary tree Huffman decoding algorithm and field programmable gate array implementation for terrestrial-digital multimedia broadcasting mobile handheld,” *IET Sci. Meas. Technol.*, vol. 6, no. 6, pp. 527–532, Nov. 2012.

[5] A. Oliveri, A. Oliveri, T. Poggi, and M. Storace, “Circuit implementation of piecewise-affine functions based on a binary search tree,” in *Proc. Eur. Conf. Circuit Theory Design*, Antalya, Turkey, Aug. 2009, pp. 145–148.

[6] P. Brox *et al.*, “A programmable and configurable ASIC to generate piecewise-affine functions defined over general partitions,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 12, pp. 3182–3194, Dec. 2013.

[7] S. S. Roy, F. Vercauteren, and I. Verbauwhede, *High Precision Discrete Gaussian Sampling on FPGAs*. Heidelberg, Germany: Springer, 2014, pp. 383–401.

[8] S. L. Aritz, “Analysis of the FSMs implementation with mini-microprocessors in FPGAs,” in *Proc. IEEE Int. Symp. Ind. Electron.*, Vigo, Spain, Jun. 2007, pp. 2290–2294.

[9] X. Zhang *et al.*, “Ztcore: Zero-waste tiny core for real-time control within FPGA,” in *Proc. IEEE Int. Symp. Ind. Electron.*, Seoul, South Korea, Jul. 2009, pp. 359–363.

[10] R. Senhadji-Navarro and I. Garcia-Vargas, “Finite virtual state machines,” *IEICE Trans.*, vol. 95-D, no. 10, pp. 2544–2547, 2012.

[11] R. Senhadji-Navarro, I. Garcia-Vargas, and J. L. Guisado, “Performance evaluation of RAM-based implementation of finite state machines in FPGAs,” in *Proc. IEEE Int. Conf. Electron. Circuits Syst.*, Seville, Spain, 2012, pp. 225–228.

[12] *XST User Guide 13.1*, Xilinx, San Jose, CA, USA, 2011.

[13] I. Garcia-Vargas and R. Senhadji-Navarro, “Finite state machines with input multiplexing: A performance study,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 5, pp. 867–871, May 2015.

[14] *Spartan-6 FPGA Block RAM Resources User Guide*, Xilinx, San Jose, CA, USA, 2011.

[15] A. Barkalov, L. Titarenko, M. Kolopienczyk, K. Mielcarek, and G. Bazydlo, *Design of EMB-Based Mealy FSMs*. Cham, Switzerland: Springer, 2016, pp. 193–237.

[16] *Cyclone V Device Overview*, Altera, San Jose, CA, USA, 2016.

[17] *Spartan-6 FPGA Configurable Logic Block User Guide*, Xilinx, San Jose, CA, USA, 2010.

[18] R. Senhadji-Navarro, I. Garcia-Vargas, G. Jimenez-Moreno, and A. Civit-Balcells, “FPGA-based implementation of RAM with asymmetric port widths for run-time reconfiguration,” in *Proc. IEEE Int. Conf. Electron. Circuits Syst.*, Marrakesh, Morocco, Dec. 2007, pp. 178–181.

[19] R. Senhadji-Navarro and I. Garcia-Vargas, “Minimum maximum reconfiguration cost problem,” *Optim. Lett.*, vol. 10, no. 3, pp. 605–617, 2016.

[20] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Series of Books in the Mathematical Sciences). New York, NY, USA: W. H. Freeman, Jan. 1979.