# Low Power SPI Design Based on Relative Timing Techniques

Guillermo H. Makar*, Francisco J. Badenas†, Roberto G. Simone†, Alejandro Furfaro†,
Kenneth S. Stevens* and Roberto Suaya†
*University of Utah, Salt Lake City, USA
†Universidad Tecnológica Nacional, Buenos Aires, Argentina
Email: g.makar@utah.edu, fbadenas@est.frba.utn.edu.ar

***Abstract*—The Serial Peripheral Interface (SPI) is a specification for serial synchronous communications. It is widely adopted in industry as a solution for communication between system core and peripherals in embedded systems. This work introduces the design and implementation of a SPI using a novel approach to asynchronous circuits: Relative Timing (RT). RT is a technique that provides a formal representation of timing requirements on circuits and can be integrated into a standard Computer-Aided Design (CAD) flow with minor additions. It has profound impact on design style and global parameters such as power consumption, area and performance. The obtained results are compared to an industry standard synchronous implementation. The proposed RT design shows a benefit of 2.25× in power per transfer for a fully active operation mode that improves with the addition of idle time, and a benefit of 1.35× in silicon area.**

## I. Introduction

The Serial Peripheral Interface is a specification for serial synchronous communications. The devices follow a master-slave architecture and communicate in full-duplex mode. It supports the connection of multiple slaves with a single master module [1]. It employs four signals:

- MOSI (Master Out Slave In)
- MISO (Master In Slave Out)
- SCLK (Serial Clock)
- SS (Slave Select).

The sampling of the data lines (MOSI and MISO) occurs at one of the SCLK edges, rising or falling depending on the definition, to preserve the synchronous characteristic of the communication. A dedicated line (SS) from the master to each slave is used to select which of them is enabled. SPI has no official standard. This allows for customization of the protocol so it can be adapted to a wide range of applications.

Several synchronous implementations for SPI have been designed in the past [2] [3]. These implementations share similar architectures that use synchronous FIFOs, counters and shift registers. Efforts in energy-per-bit minimization and activity factor characterization of communication circuits show the potential a low power SPI solution [4].

Asynchronous circuits are naturally reactive to changes of state in the control signals. This presents the advantage of not having to continuously sample the input lines, which is ideal for low power applications. Asynchronous design techniques, and Relative Timing in particular, have been applied over serial communication protocols showing significant power improvements [5]. This work introduces the design of a complete fabricated Relative Timed SPI interface using an industry standard CAD flow with minor additions and compares the results with a synchronous design.

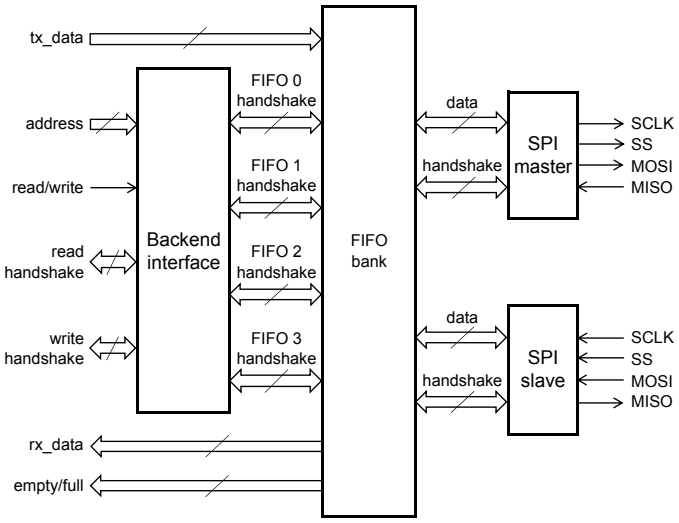


Fig. 1. Block diagram. The backend interface communicates with the system core and decodes the incoming operations to the FIFOs. The four FIFOs manage the SPI modules transactions. The SPI modules perform the serial/parallel translations and handle the SPI signals according to the implemented definition.

## II. Design Architecture

The architecture for both implementations of the SPI is shown in Figure 1. It consists of an 8-bit SPI master module, an 8-bit SPI slave module, four 14-deep First In First Out (FIFO) stacks to store the data that each module transmits and receives, and a backend interface that allows the system core to read and write at a high frequency while the SPI signals operate at a much slower rate. A read/write input signal and a 2-bit input bus for the address are used to select between the FIFOs and forward data correctly. All FIFOs can flag their empty and full status for the system to use as needed. Both read and write handshake channels employ a pair of signals to indicate data validity and the end of an operation.

978-1-7281-0996-1/19/$31.00 ©2019 IEEE

## III. Synchronous SPI Design

For the synchronous SPI, each of the SPI modules contains two shift registers. The transmitting shift register is preloaded with the first available piece of data from the transmitting FIFO before the communication starts. The output of the register holding the least significant bit is connected to the serial output, and following an inactive edge of the SCLK data is shifted to accommodate the next bit to be transmitted. The receiving shift register is filled with the serial input data. At the active edge of the SCLK, the shifting allows a new bit to be captured as the previous ones are moved across to accommodate the complete piece of data. Once this operation concludes, the receiving FIFO is written. A synchronous counter uses the SCLK to compute the number of processed bits and detect the communication start and finish conditions. In the master module only, the SCLK signal is generated as a division of the system clock. The read and write handshake channels are sampled using the system clock and consist of a signal that indicates data validity from the sender (valid) and a signal that indicates the inability to proceed from the receiver (stall).

## IV. Relative Timed SPI Design

Relative Timing (RT) methodology provides a formal representation of timing requirements of circuits. This allows to leverage timing as an additional design variable [6]. RT uses timing constraints for correct circuit operation, instead of solely logic as other asynchronous techniques. RT constraints consist of a common timing reference and a pair of events that are ordered in time. We call the common reference a point-of-divergence, $pod$, and each of the ordered events a point-of-convergence, $poc$. A constraint is expressed as $pod \rightarrow poc_0 + m \prec poc_1$ where $poc_0$ must occur in time before $poc_1$ with a margin $m$ for robustness. This is represented by two related design constraint equations, $set\_max\_delay$ and $set\_min\_delay$, which are used to perform the timing driven synthesis that enforces timing on the logic paths. All blocks of the proposed SPI architecture were redesigned using this approach while keeping the same functionality.

### A. Serial Clock Generator and Bit Counter

An asynchronous ripple counter generates the SCLK signal in the master module. In the first register the output is looped back into the input, which causes the counter to toggle once every rising edge of the system clock dividing the frequency by two. Every additional cascaded register is clocked by the output of the previous stage, which divides the frequency again. A multiplexer selects which division value is fed to the rest of the circuit. The SCLK output is disabled by the SS line before a transaction begins and after a transaction concludes. The advantage of this topology is that each stage is clocked only when a new bit needs be stored in each register. Less than two transitions per state change are needed on average, reducing the energy consumption by more than 50% for a 4-bit counter. Moreover, no additional logic is required to calculate the next state value.

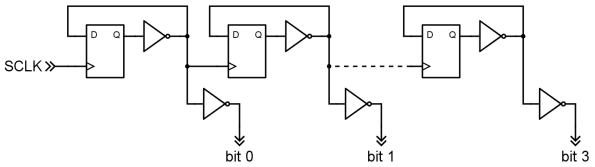


Fig. 2. Bit counter circuit. Uses a ripple counter architecture.

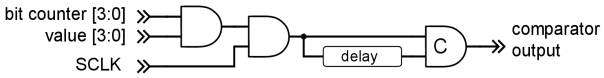


Fig. 3. Comparator circuit. The delay line and C-element structure avoid hazards at the output.

The bit counter circuit is shown in Figure 2. It is an essential piece of this design to keep track of the sequence of bits during transactions in the master and slave devices. Active energy of the storage elements is minimized using a ripple counter architecture, as each stage is triggered to store only when the previous bits reach their maximum count. Inverters are placed at each bit output so the feedback loops are isolated from the logic that follows, which may present different loads depending on the state of the circuit. Transmission and reception registers are clocked using 4-bit combinational comparators that detect specific values in the bit counter state as shown in Figure 3. The output signal of the comparators needs to be hazard free since a glitch can be interpreted as a valid signal transition compromising data validity. For that purpose, 2-input C-element gates [6] with the comparator output at one input, and a delayed copy of it at the other input can filter out any glitch as wide as the delay time. The C-element output only transitions when the two inputs have already transitioned to an identical state. The circuit for the C-element gate is shown in Figure 4. RT constraints allow us to define the delay based on the maximum time that the comparator takes to present a stable output plus a margin added for robustness to variations.

### B. Data Reception Circuit

The reception circuit topology is shown in Figure 5. The incoming data stream is connected to every register input. Each bit is captured at the correct instant of time and position of the byte by clocking the registers with the corresponding comparator output. Only one register transition per transmitted

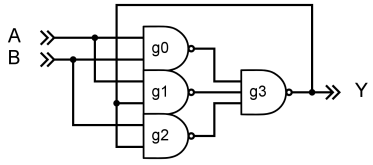


Fig. 4. 2-input C-element circuit. Defined by the equation $Y_n = A.B + (A + B).Y_{n-1}$

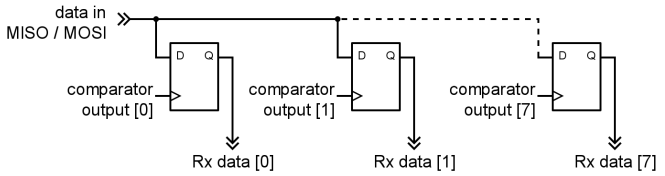


Fig. 5. Reception circuit. The comparators are used for clocking the registers one a time.

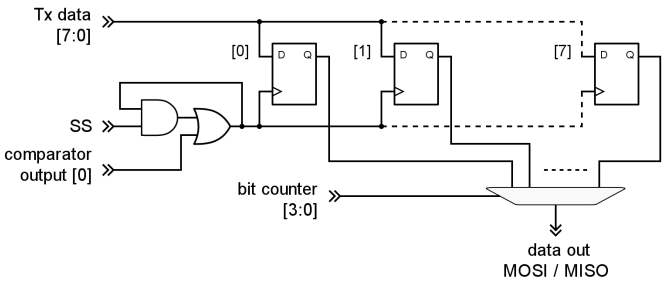


Fig. 6. Transmission circuit. The registers are only clocked once at the beginning of an operation. The fault tolerant multiplexer selects which bit to output.

bit is required to obtain the complete piece of data once all registers are clocked.

### C. Data Transmission Circuit

The transmission circuit topology is shown in Figure 6. The eight registers that store the transmit data are only clocked at the first count of the bit counter and remain latched for the rest of the communication time. Bits are selected for serial transmission using the bit counter state to control a fault tolerant multiplexer that uses the same C-element and delay topology previously described. To prevent unwanted transitions to be forwarded to the SPI MISO or MOSI lines, the delay is sized to handle the worst time that the multiplexer takes to switch from one input to another plus a margin added for robustness to variations.

### D. Backend Circuit

The handshake communications follow a 4-phase bundled data protocol where data validity and consumption are managed by a pair of control signals, namely request (req) and acknowledge (ack) [6]. The timing diagram for this signals is shown in Figure 7. RT is used to define the necessary timing assumptions for data to be stable before the handshake signals arrive at every stage. Asynchronous FIFOs use latches instead of flip-flops as storage elements and replace the global clock network with local controllers for each pipeline stage as shown in Figure 8 for a simple linear FIFO. The designed FIFOs have a 1-2-4-4-2-1 tree architecture, that has proven to be the most energy efficient [7]. This structure requires additional toggle and merge components to handle the data branching following the same handshake protocol. These modules are designed as finite state machines following the templates proposed in [7] and require RT constraints to guarantee fundamental mode

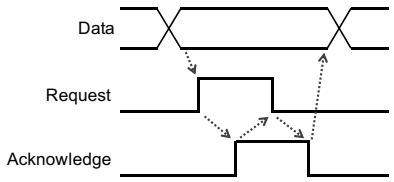


Fig. 7. 4-phase handshaking protocol time diagram.

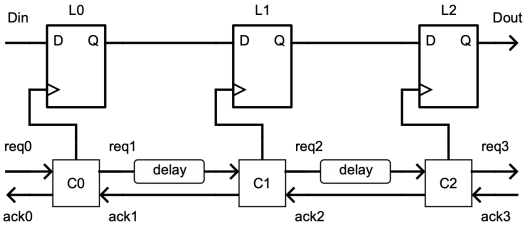


Fig. 8. Bundled data linear FIFO design. Delay sized by RT constraints as $req_i \rightarrow L_{i+1}/D + margin \prec L_{i+1}/CLK$. Each $req_i$ handshake on controller $C_i$ indicates new data is presented to pin $D$ of $L_i$.

operation. Each FIFO has empty and full outputs that can be used as interrupts to manage data flow.

### E. CAD Flow Integration

RT facilitates the integration of the timing requirements of the circuits into the design flow. Beginning with a design specification, the circuits are synthesized either by hand or using asynchronous synthesis tools like petrify [8] or 3D [9] and then technology mapped to gates of the standard cell library. The initial state of the circuit is set by adding a reset signal that controls the output value when needed. The timing graph of the circuit must be represented as a directed acyclic graph to be handled by a standard CAD flow, so timing cuts are generated for the design as $set\_disable\_timing$ constraints where necessary. In order to prevent the tools to modify the structure of the circuits but allowing the drive strength to be optimized for power and delay, a set of $set\_size\_only$ constraints is employed. The RT constraints are generated either by hand with the assistance of a timing analysis tool or using a formal verification engine [10]. Finally, the constraints are mapped onto the module instances in the design to allow the use of commercial tools to perform the timing driven synthesis.

## V. Implementation, Simulation and Testing

The synchronous and RT versions of the SPI interface are written in Verilog and Synopsys tools are used for synthesis, physical place and route, parasitic extraction and post-layout timing and performance analysis. Clock gating is introduced in the synchronous design. Both circuits are implemented using vendor provided cells in a standard IBM 8RF 130nm CMOS process. Only the RT design was fabricated. It was successfully tested by filling the master module transmitting FIFO, monitoring the SPI interface signals and reading back the collected data from the slave module receiving FIFO. The

TABLE I
AREA COMPARISON

| | Cell area ($\mu$m sq.) | | | |
|---|---|---|---|---|
| Design | Backend | SPI Master | SPI Slave | Total |
| RT | 21244 | 2349 | 1637 | 25282 |
| Synchronous | 31596 | 1401 | 989 | 34086 |
| Benefit | 1.49× | 0.60× | 0.60× | 1.35× |

TABLE II
POWER COMPARISON

| | Avg. power per transfer ($\mu$W) | | |
|---|---|---|---|
| Design | 0% idle | 50% idle | 90% idle |
| RT | 2.189 | 1.795 | 1.457 |
| Synchronous | 4.935 | 5.008 | 4.966 |
| Benefit | 2.25× | 2.79× | 3.41× |

presented results correspond to simulation values obtained as follows. The functional validation is performed using the exact same testbench and random data packages for the two designed solutions, with Modelsim simulator and back-annotated timing values loaded as a standard delay format (SDF) file. A value change dump (VCD) file containing the switching information of each component is generated for all simulation runs and the parasitic delays from the physical design are extracted as a standard parasitic exchange format (SPEF) file. Using these files, the timing analysis and power numbers calculation are performed within PrimeTime tool. For testing purposes only, the SPI signals of the master and slave modules are looped to each other.

## VI. RESULTS

Table 1 shows the hierarchical area comparison for the designed blocks. The silicon area required for the circuits present a 1.35× benefit for the RT version over the synchronous one. In the RT backend design a considerable area reduction is obtained since the handshake protocol rules the data flow across stages in the FIFOs and makes the logic for controlling read and write positions unnecessary. Table 2 shows the power numbers comparison for different values of idle time, which represent the activity factor of the circuit. The power benefit of the RT design is 2.25× for a fully active mode (0% idle time), where the maximum transfer rate of the protocol is used. This benefit increases to 2.79× for a 50% idle time operation mode and up to 3.41× for a 90% idle time operation mode. The main contributor to the power consumption of this circuits is the cell internal power, which is reduced in the RT design by lowering the activity factor of the devices with the proposed topologies and data management techniques. Figure 9 shows a performance comparison of the designs in terms of energy per transfer over varying idle time percentages ranging from 0% to 90%. As idle time increases, the synchronous design pays a larger energy penalty for a single data transfer. The reactive characteristic of RT design considerably reduces such penalty keeping the energy values closer across the operation modes range. The energy per transfer benefit of the RT design for the corner cases is 2.84× for 0% idle time and 4.3× for 90% idle time.

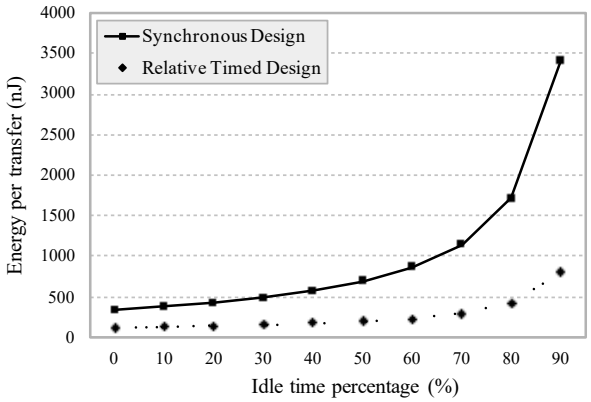


Fig. 9. Energy comparison.

## VII. CONCLUSION

This work presents the design of a Relative Timed SPI. Results are compared to an industry standard synchronous implementation. The application of RT design techniques results in significant improvements in terms of area and energy. The energy benefits are greater when the idle time in the communication increases. These results encourage the implementation of RT techniques in low power applications where the peripherals remain inactive for long periods of time.

## REFERENCES


[1] F. Leens, "An Introduction to I2C and SPI Protocols," *IEEE Inst. & Meas. Magazine,* pp. 8-13, Feb. 2009.

[2] "SPI Block Guide" V04.01, Motorola Inc, Jul. 2004. Available online: https://www.nxp.com/files-static/microcontrollers/doc/ref_manual/S12SPIV4.pdf (accessed on 12 July 2019).

[3] "OPB Serial Peripheral Interface (SPI)" V1.00e, Xilinx Logicore, DS464 Jul. 2006. Available online: https://www.xilinx.com/support/documentation/ip_documentation/opb_spi.pdf (accessed on 12 July 2019).

[4] Lukas, Christopher. (2015). "A 0.38 pj/bit 1.24 nW Chip-to-chip Serial Link for Ultra-low Power Systems," *ISCAS 2015,* pp. 2860-2863.

[5] D. Bhadra, V. S. Vij and K. S. Stevens, "A low power UART design based on asynchronous techniques," *2013 IEEE 56th Int. Midwest Symposium on Circuits and Systems (MWSCAS),* Columbus, OH, 2013, pp. 21-24.

[6] K. S. Stevens, R. Ginosar, and S. Rotem, "Relative Timing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 1, no. 11, pp. 129-140, Feb. 2003.

[7] H. Han and K. S. Stevens, "Clocked and asynchronous FIFO characterization and comparison," *2009 17th IFIP Int. Conf. on Very Large Scale Integration (VLSI-SoC),* Florianopolis, 2009, pp. 101-108.

[8] J. Cortadella, M. Kishinevsky, A. Kondratyev, L. Lavagno, and A. Yakovlev, "Petrify: a tool for manipulating concurrent specifications and synthesis of asynchronous controllers," *IEICE Transactions on Information and Systems,* vol. E80-D, no. 3, pp. 315-325, 1997.

[9] K. Y. Yun and D. L. Dill, "Automatic synthesis of extended burst-mode circuits. I. (Specification and hazard-free implementations)," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 18, no. 2, pp. 101-117, Feb 1999.

[10] Y. Xu and K. S. Stevens, "Automatic synthesis of computation interference constraints for relative timing verification," *2009 IEEE Int. Conf. on Computer Design,* Lake Tahoe, CA, 2009, pp. 16-22.