OPTIMIZATION OF SPI CONTROLLER FOR HARDWARE EMBEDDED
SYSTEMS USING FINITE STATE MACHINES

Edwar Jacinto G., Fernando Martínez S. and Holman Montiel A.
Universidad Distrital Francisco José de Caldas, Bogotá, Colombia
E-Mail: hmontiela@udistrital.edu.co

ABSTRACT
This article shows the design and implementation of a finite state machine that describes the operation of the SPI
protocol in a standard way, which is a serial protocol used for transfers with peripherals with low / medium bandwidth, all
the technical detail of a specific application, and emphasis is placed on making a description with a low level of abstraction
to reduce the amount of hardware resource used, arriving at a general-purpose IP CORE type solution. The state machine
controls the sequence of the communication that complies with the protocol. By having a design of low complexity, it
achieves to be easily adaptable for a number of plates (shields) that proliferate in a significant number of academic and
commercial applications.

Keywords: finite state machine, intellectual property core, serial peripheral interface, hardware description language.

1. INTRODUCTION
Currently in the technology market there are a lot
of electronic elements supported on serial to peripheral
interface (SPI) technology, it can made direct reference to
elements such as: Touch screens, communication modules
(Wi-Fi, 3G, Ethernet), DAC / ADC converters, among
many others [1-5]. This communication protocol has
acquired a great boom in its implementation due to its
simplicity of interface and its high transmission speed,
characteristics that make it ideal for real-time application
implementations associated with development devices
such as Field Programmable Gate Array (FPGA) or
system on chip processors (SoC) [2][8].

At the moment of making use of SoC processors,
the SPI controller is usually incorporated, allowing the
communication between the processor peripherals and also
with other processors [3]. However, currently the use of
reconfigurable hardware technologies for digital systems
is increasing, this is mainly because the solutions
developed in this technology are based on its versatility,
high integration capabilities, low cost and flexible
architecture [6-12].

The present work focuses on the development of
an optimized core using state machines for SPI protocols,
developed in hardware description language (HDL) so that
it can be implemented in any FPGA in the market. The
aim is to use a finite state machine (FMS) as a control
methodology for the synchronous system of the controller,
thus achieving a functional development that occupies the
minimum amount of resources (Slices) and guarantees a
functional and appropriate response time to the temporal
requirements of current applications.

2. METHODOLOGY
The proposed development consists of three
stages: Functional analysis of the SPI protocol, design of
the state machine and implementation and validation of
the solution in a functional application on a Spartan 3AN
FPGA.

2.1 SPI protocol
SPI is a master-slave synchronous serial bus
protocol used for data transmission in Full-Duplex mode,
it consists of 4 signal lines: SCK (Clock), CS (Chip
Selector), MOSI (Master Output Slave Input) and MISO
(Master Input Slave Output). This type of serial
communication allows the transmission of up to 32 bits
per frame, the connection scheme allows a master device
to connect to one or more slave devices, see Figure-1; the
generation of the synchronism clock is a unique function
of the master device, the start of the transmission is
established when the CS line is set to logical zero, at
which point the movement of information inside and
outside the device is synchronized through the MOSI lines
and MISO, it must be taken into account that bit
transmission is synchronized by edge detection, that is,
it can operate with a rising or falling edge (with delay or
without delay).

Figure-1. SPI core in a multi-slave communication.
2.2 System architecture proposed

It is proposed a system that manages the SPI Bus of a Spartan 3AN Started Board development card. For this specific case, the communication was made with the Linear Tech peripheral LTC2624 Quad DAC, it should be noted that this SPI bus is common for several devices of the development system, among which are an analog to digital ADC converter, a PGA programmable gain amplifier and two Flash memories, see Figure-2; each of these peripherals has a CS enabling bit (Chip enable) that must be controlled in a coordinated manner to avoid short circuit between the different devices connected to the bus.

Since there is a driver written as an IP Core for managing a DAC SPI, a generator was implemented using the DDS (Direct Digital Synthesizer) technique to carry out performance tests, which has a ROM memory where the samples to be generated are stored, together with a counter to index the data to be placed in the SPI driver input. In addition to this, two frequency dividers are required, one that controls the sample frequency of the system’s output signal and another that controls the SPI bus speed, avoiding exceeding the times established by the manufacturer of the integrated circuit.

2.3 State machine

In the design of the SPI driver, a description of an FSM finite state machine that controls the SCK, MOSI and CS signals of the LTC2624 at any moment of time. Figure-3 shows the state diagram that describes the system.

It is identified as a fairly simple linear state machine, which complies with SPI type I, that is, it must keep the CS in a high state in the resting state and transmits the data on the rising edges, then one by one the states as follows:

Init: In this state all variables must be in the idle state, high, the clock of the SCK bus in low state and the Mosi information bit in low, besides that the data to be sent is continuously being captured along with the channel of the converter through which the analog output signal will be output.

Tx: In this state the CS signal is lowered to low, whereupon the communication is started, at the same time the SCK clock is enabled and the vector is started to be indexed to send the data in the MOSI, a counter is placed to verify when each and every data has been sent, in this way a repetitive cycle is carried out that guarantees the sending of all the data.
End: It is the final state of the communication, where the bus is disabled by placing the CS in high state and placing the vectors and counters in initial conditions, in order to wait for the initial state, in addition to returning to the initial state all the variables are placed the Done bit in a high state to signal the end of the communication.

3. RESULTS
The design of a digital system that manages an analog digital converter of 5 million samples per second was carried out, where the management block of the SPI protocol, time that can be seen in Table-1 already has a maximum propagation delay of 0.979 ns (max Delay), which makes the total system operate at a maximum speed of 1.021 Gigahertz which makes this communication block can be adapted to a converter with a higher sampling rate. In addition to this, it was possible to realize a complete system of basic wave generation to test the performance of both the designed driver and the capacity of the FPGA, making a description of the hardware so small that only occupies 1% of the capacity of the FPGA, evidenced in Table-2, in the box corresponding to the number of flip-flops and slice used, which allows to perform multiple tasks with a low-cost FPGA and which is currently considered small in comparison with the devices available in recent years in the market.

Table-1. Response times associated with operating blocks.

<table>
<thead>
<tr>
<th>Clock net</th>
<th>Routed</th>
<th>Resource</th>
<th>Locked</th>
<th>Fanout</th>
<th>Net skew (ns)</th>
<th>Max delay (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk_BUFGP</td>
<td>ROUTED</td>
<td>BUFGMUX_X2Y11</td>
<td>No</td>
<td>3</td>
<td>0.014000</td>
<td>0.936000</td>
</tr>
<tr>
<td>XLXI_4/clk_int</td>
<td>ROUTED</td>
<td>BUFGMUX_X1Y10</td>
<td>No</td>
<td>35</td>
<td>0.078000</td>
<td>0.979000</td>
</tr>
<tr>
<td>XLXI_5/clk_int</td>
<td>ROUTED</td>
<td>Local</td>
<td>4</td>
<td>0</td>
<td>0.070000</td>
<td>0.661000</td>
</tr>
</tbody>
</table>

Table-2 shows a complete report of the available resources of an FPGA. In the first 3 rows, a count is made of the Flip-flops and LUT (Look up tables) of the system, which are counted as the Slices (BLC basic logics cells) that the system has, then a count of the possible logic to be used in the system is made, finally the pins, currents and capacity of the inputs and outputs of the system are listed.

A major innovation brought by this development, is to deliver a driver SPI totally standard, easy to use and understand by a possible designer, which makes it adaptable to future designs that handle analog-digital converters through the SPI protocol, what’s more, it could be modified quickly for any type of peripheral or communication module that uses this protocol and by only using 1% of the available hardware resource it is possible to implement multiple tasks in parallel; providing in this way a robust solution and practice adaptable to various electronic applications.

Table-2. Summary of resources used in the FPGA.

<table>
<thead>
<tr>
<th>Logic Utilization</th>
<th>Used</th>
<th>Available</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Slice Flip Flops</td>
<td>53</td>
<td>11,776</td>
<td>1%</td>
</tr>
<tr>
<td>Number of 4 input LUTs</td>
<td>63</td>
<td>11,776</td>
<td>1%</td>
</tr>
<tr>
<td>Number of occupied Slices</td>
<td>47</td>
<td>5,888</td>
<td>1%</td>
</tr>
<tr>
<td>Number of Slices containing only related logic</td>
<td>47</td>
<td>47</td>
<td>100%</td>
</tr>
<tr>
<td>Number of Slices containing unrelated logic</td>
<td>0</td>
<td>47</td>
<td>0%</td>
</tr>
<tr>
<td>Total Number of 4 input LUTs</td>
<td>67</td>
<td>11,776</td>
<td>1%</td>
</tr>
<tr>
<td>Number used as logic</td>
<td>63</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number used as route-thru</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of bonded OIBs</td>
<td>11</td>
<td>372</td>
<td>2%</td>
</tr>
<tr>
<td>Number of BUFGMUXs</td>
<td>2</td>
<td>24</td>
<td>8%</td>
</tr>
<tr>
<td>Average Fanout of Non-Clock Nets</td>
<td>4.44</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4. CONCLUSIONS
A totally modular design was developed, with the description of a driver for handling digital converters to analog and digital to analog by SPI, high speed and low use of hardware resources, being able to handle multiple modules of this type in a simple and effective way, which makes it easily usable and reconfigurable in future applications.

On the other hand, with the help of the Top-Down methodology a finite state machine was designed that simply describes the SPI protocol, in only 3 states, something that complies with the protocol requirements and makes it easily understandable, modifiable and scalable. This description was entirely made in the VHDL language that is standard worldwide and makes the design
able to be integrated with tools and modern hardware description languages, with higher performance devices.

ACKNOWLEDGEMENT
This work was supported by the Universidad Distrital Francisco José de Caldas - Technological Faculty. The views expressed in this paper are not necessarily endorsed by the University. The authors thank the research group ARMOS for the evaluation carried out on prototypes of ideas and strategies.

REFERENCES


