A versatile pulse programmer for magnetic resonance imaging

Article *in* The Review of scientific instruments · May 2011

DOI: 10.1063/1.3587068 · Source: PubMed

3 authors, including:

Rui-peng Ning
East China Normal University
9 PUBLICATIONS **13** CITATIONS

SEE PROFILE

Guang Yang
East China Normal University
80 PUBLICATIONS **587** CITATIONS

SEE PROFILE

A versatile pulse programmer for magnetic resonance imaging

Ruipeng Ning,[1,2] Guang Yang,[2,a)] and Gengying Li[1,b)]
[1]*Shanghai Key Laboratory of Magnetic Resonance, East China Normal University, 3663 North Zhongshan Road, Shanghai 200062, China*
[2]*Department of Physics, East China Normal University, 3663 North Zhongshan Road, Shanghai 200062, China*

(Received 24 March 2011; accepted 14 April 2011; published online 9 May 2011)

Hardware and software solutions for a versatile pulse programmer have been presented. The core of the pulse programmer is an FPGA device that provides flexibility to the design and reduces the number of electronics elements needed. The event of the pulse programmer consists of 16 bits. The main feature of the proposed pulse programmer is that the 16 outputs can be independently delayed. This is important for correcting delays of the RF channels or the gradient channels due to various causes. The proposed pulse programmer is integrated into an MRI scanner, and the correction of the gradient system delay is taken as an example to experimentally demonstrate its performance. © *2011 American Institute of Physics*. [doi:10.1063/1.3587068]

I. INTRODUCTION

Although major manufacturers of magnetic resonance imaging (MRI) instruments have offered commercially available products with high performance, there are still a lot of researchers who focus their interests on the development of custom-built MRI systems.[1] Commercially available instruments are complicated and multifunctional because they are designed to meet various demands from diverse applications,[2] and as a result, they are expensive. However, in general cases, only a subset of the available functions is used.[3] Moreover, standard commercially available instruments cannot always meet the special requirements of certain studies and applications.[4] Thus, various custom-built MRI systems are necessary complements to the commercial instruments[5] for the MRI researchers.

In the design of a home-built MRI system, the pulse programmer used to control an MRI system is a key component. This is because the flexibility and the performance of MRI systems are determined directly by the pulse programmers, and pulse programmers suitable for MRI systems are hard to obtain commercially. In addition, since MRI is continuously undergoing rapid developments, recent sophisticated MRI sequences impose even more requirements on the performance and flexibility of the pulse programmer units.

Over the past decade, the image qualities of both commercial and custom-built MRI systems have been delightfully improved due to advances in imaging methods and instrumentation technologies.[6] Nevertheless, system imperfections have gradually become an important bottleneck in the practical application of certain imaging methods and scanning strategies. Gradient system delay is just a common and stubborn hardware imperfection. It has been shown that the gradient system delay will lead to obvious image degradation and distortion for fast imaging and non-grid scans.[7,8] This delay can be regarded as time shift relative to RF channels. The effects of the delay can be reduced by modifying imaging strategies,[7,8] or eliminated directly by shifting (delaying) the RF channels according to the experimentally measured gradient delays. The latter solution is simpler and more direct and requires that the pulse programmer support a so-called "bit-shift" with respect to each other within the event of the pulse sequence. That is to say, a pulse programmer, which is able to apply different delays to each output bit independently in the events of a pulse sequence, offers a direct and simple solution to the problem of the gradient system delay. From sequence timing aspects, an MRI pulse sequence can be viewed as a group of events with different durations. Each event consists of a number of triggering of RF and gradient waveforms, and acquisition of the MR signal.[9] Many designs of pulse programmers have been suggested.[1–4,9,10] One approach to the MRI pulse programmer is to use a 32-bit microcontroller,[1] in which, the interrupt request from Timer1 of the microcontroller was used for the event processing. This may result in time jitter within the imaging pulse sequence in theory. Moreover, the minimum time delay between two successive events achieved by the microcontroller design is not short enough to trigger soft radio-frequency (RF) pulses in conventional multiple-slice two-dimensional (2D) imaging. Pulse programmers with higher performance implemented using computer programmable logic devices or field programmable gate array (FPGA) device have also been proposed.[2,9] These designs offer the flexibility of modifying or upgrading the architecture of pulse programmer through software programming. However, none of the existing pulse programmers reported in the literature has the independent bit-shift capability that can be directly used to remove the gradient system delay effects.

In this paper, a pulse programmer with independent bit-shift capability is developed. The principle of our pulse generator is similar to that mentioned in a previous article.[11] The core of the pulse programmer is an FPGA device that provides flexibility to the design and reduces the number of electronics elements needed. The output of the pulse programmer consists of 16 independent lines. The state of each output line is identified by a binary bit. The composition of the 16 states of the output lines is defined as an "event." The duration of each event is controlled by a 32-bit counter whose

a)Electronic mail: gyang@phy.ecnu.edu.cn.
b)Electronic mail: gyli@phy.ecnu.edu.cn.

value is proportional to the duration defined in the imaging pulse sequence. Therefore, it is convenient to describe the trigger timing of a pulse sequence using a series of event-duration pairs. In other papers,[7,8,11] the "duration" is also called "delay." In order to avoid confusion with the gradient system delay, "Duration" is used in this paper. The timing resolution of the duration is 20 ns when a clock of 50 MHz frequency is used. In each event, all 16 bits is simultaneously updated, hence RF and gradient channels in the MRI spectrometer can be controlled by the pulse programmer. Based on this simple hardware design, the pulse programmer provides both flexibility and high performance. The removal of the gradient system delay effects is simply achieved by inserting a function where the event and duration are altered to produce delayed trigger capability before the event-duration table is downloaded to the pulse programmer. In the following, detailed description of the pulse programmer is presented. Then the pulse programmer is integrated into an MRI scanner, and the correction of the gradient system delay is taken as an example to experimentally demonstrate its performance.

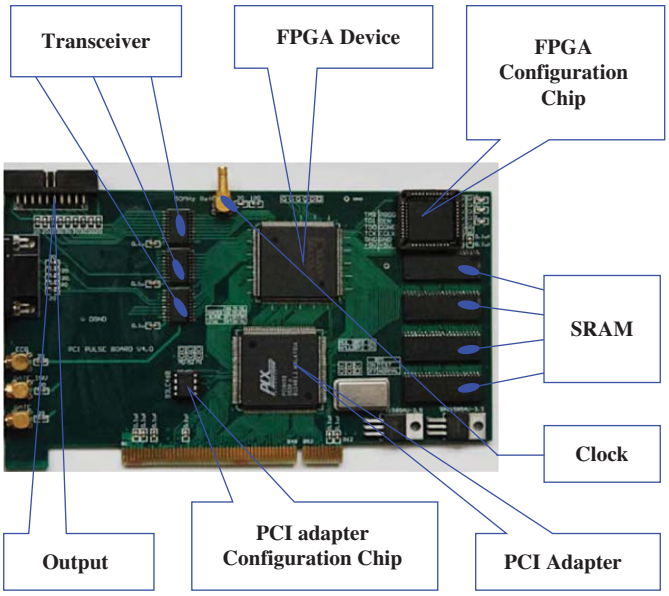

FIG. 2. (Color online) Photo of the pulse programmer.

II. HARDWARE

In our home-built spectrometer,[11] an industrial personal computer with 12 peripheral component interconnect (PCI) slots is chosen as the host computer. Each module in the spectrometer is designed as a PCI based card that can be inserted into the spectrometer system easily. The data of the RF and gradient waveform generators are stored in the corresponding on-board memory. The output of these waveform generators is immediately updated after they received a trigger pulse from the pulse programmer. The pulse programmer in this paper is also implemented as a PCI-based card. As shown in Fig. 1, a PCI adapter (PCI9052, PLX, USA) is utilized to deal with the communication between the pulse programmer and the host computer. The on-board memory consists of four pieces of 256 K × 16 Static RAMs (SRAM, CY7C1041B, Cypress Co. Ltd., USA) that are used to store the event-duration table translated from the imaging pulse sequence. An FPGA device (XC2S200, Xilinx, USA) is used for logic control of the pulse programmer. In addition, the output of the event is buffered by two transceivers (ABT2245, TI, USA), and an external master

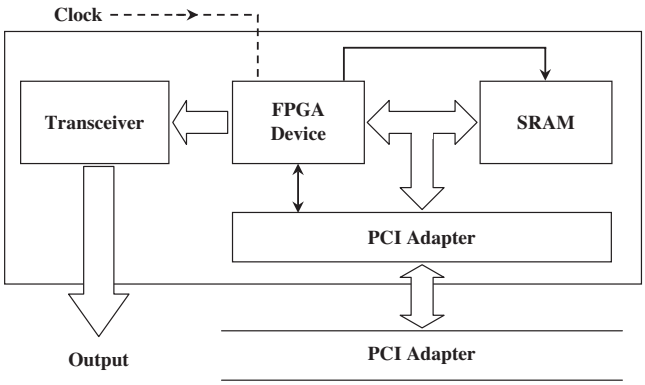

FIG. 1. Block diagram of the pulse programmer.

clock (50MHz) is used. Figure 2 shows a photo of the pulse programmer.

In MRI pulse sequence, RF soft pulses (SINC envelop) are commonly used to achieve selective-excitation. Typically, several hundreds of data points are required to produce a satisfactory excitation profile for a SINC waveform. In order to save the on-board memory space and to speed up data transferring from the host computer, the data in the event-duration table are compressed and encoded in a format as "E1-D1-E2-D2-LN", where E, D and LN are abbreviations of event, duration and loop-number, respectively. It should be noted that the symbol D represents duration measured in number of master-clock cycles inside the FPGA device. Before the running of a pulse sequence, the encoded event-duration table is downloaded from the host computer to the on-board SRAM. Then the first encoded data is loaded into the registers of the FPGA. Now the pulse programmer is ready for operation.

When the pulse programmer receives a "start" command from the host computer, the pulse programmer starts to produce the imaging pulse sequence. During the execution of the pulse sequence, the pulse programmer is under the control of the FPGA. Figure 3 depicts the flow chart of the pulse programmer. The control logic inside the FPGA first decodes the data in the registers from "E1-D1-E2-D2-LN" structure to an array of "event-duration," and updates the output according to the current event. Meanwhile, a 32-bit counter (event-counter) counts down its number that is proportional to the duration of the current event. This event lasts until the event-counter reaches zero.

The principle of decoding is rather simple and direct. There are 2 × LN elements in an encoded data, and each element consists of an event (E1 or E2) and corresponding duration (D1or D2). An event-duration pointer is assigned to the first "event-duration," and the output of the pulse programmer and the content of the event-counter are set to the current event E1 and duration D1, respectively. When

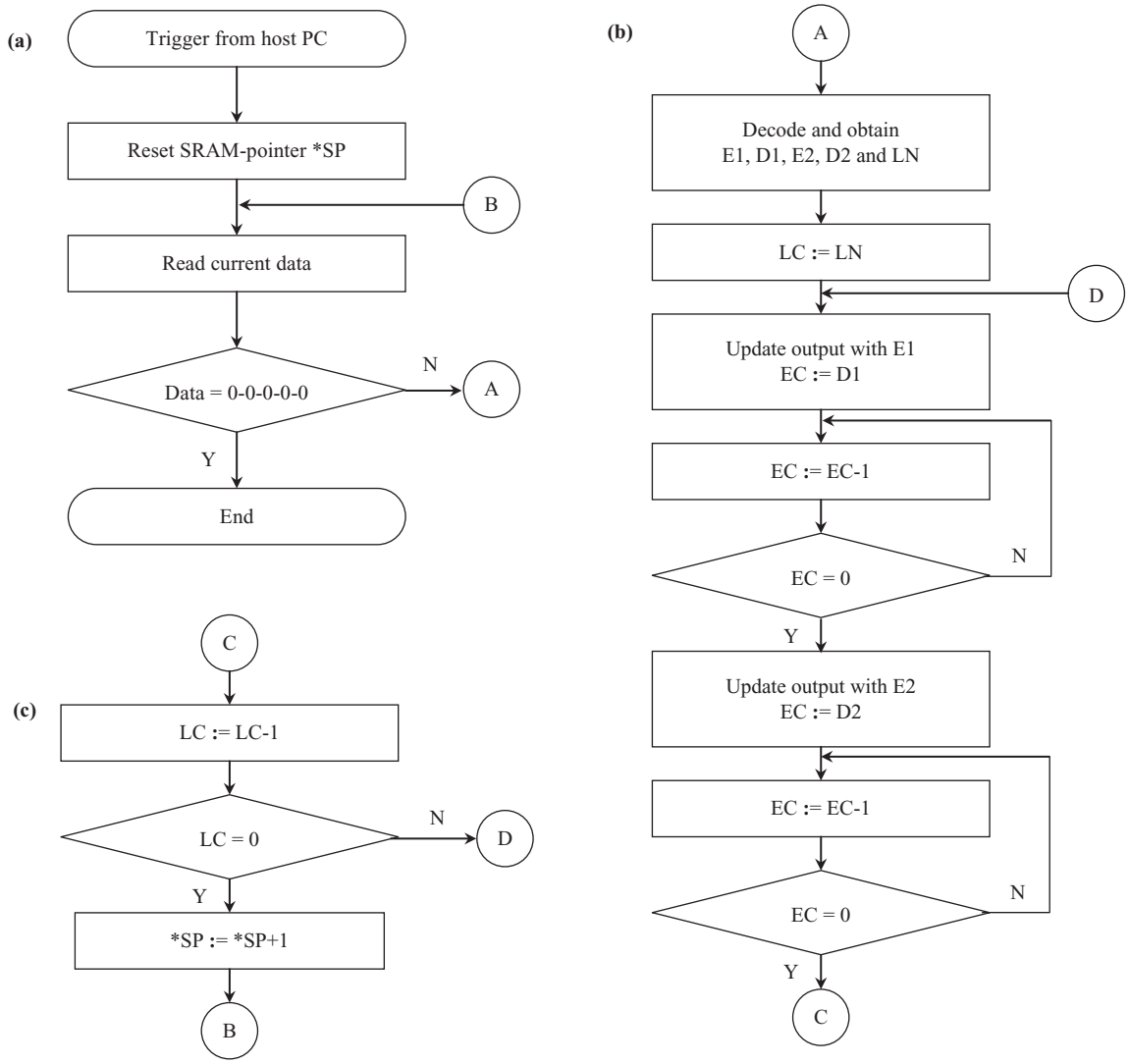

FIG. 3. Flow chart for updating trigger signal.

the event-counter reaches zero, the output of the pulse programmer and the content of the event-counter are set to the next event E2 and duration D2. The event E2 lasts until the event-counter reaches zero. After the above steps have been repeated LN times, the next entry of encoded event table is loaded to the register and processed. This process is repeated until the FPGA reaches the ending flag of "0–0-0–0-0."

III. SOFTWARE

A pulse sequence compiler with graphical user interface is employed to translate imaging pulse sequences to the event-duration table and acquisition data. An encoding procedure is further utilized to convert the event-duration table to the encoded structure before the data is downloaded to the pulse programmer. After the preparation of the pulse programmer, the pulse sequence parameters are transferred to RF transmitter board, gradient boards, and receiver boards, respectively. A full description about the pulse sequence compiler can be found elsewhere.[12]

The pulse programmer will synchronously trigger RF and gradient channels in the MRI spectrometer since all 16 output lines of are simultaneously updated using the event-duration table generated by the pulse sequence compiler. Therefore, in order to update each bit of the event word independently, the entire event-duration table is required to be modified before it is downloaded into the pulse programmer. In this paper, a procedure named "independent-delay" is written to further process the event-duration table originally generated by the pulse sequence compiler. As shown in Fig. 4, the "independent-delay" procedure consists of five steps, they are as follows: parameter-management, event-decomposition, time-gridding, bit-shift, and event- reconstruction.

In the first step, delay information (such as gradient delays in x, y, and z directions) is obtained from configure file or software interface, and converted into binary data. Assuming the delay of ith channel is d_i, then d_{mx} is the maximum in $(d_0 \ldots d_{15})$. In the second step, the event table generated by pulse sequence compiler is decomposed into 16 state tables. In each state table, data is structured by "S_i-D_i," where "$S_i[j]$" represents the ith bit of the jth "event" in event table

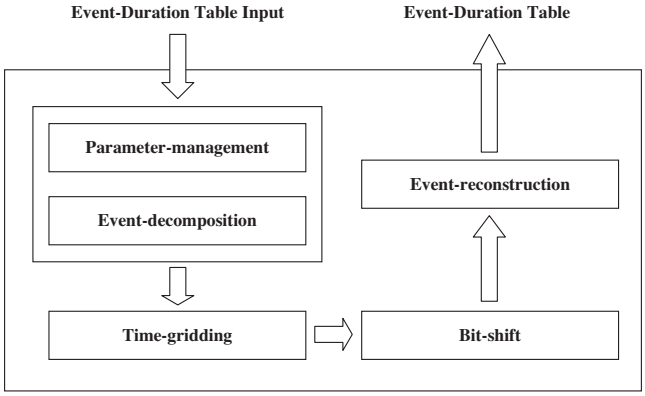

FIG. 4. Structure frame of the "Independent-delay" procedure.

and "$D_i[j]$" is the corresponding "duration" ($i = 0 \ldots 15$; $j = 1 \ldots \text{Num}$).

When the first two steps are executed, the "time-gridding" function can be called to round $D_i[j]$ and d_i to the nearest integral multiple of minD. Herein, the minD is equal to the allowed minimum pulse width of the pulse programmer.

After "time-gridding," the "bit-shift" is called. In this function, an entry "0-HD_i" is inserted in the front of the i^{th} state table, and another entry "0-TD_i" is inserted before the end mark. The "HD_i" is equal to ($d_{mx} - d_i$), and the "TD_i" is equal to d_i. Then 16 new state tables are obtained.

After "bit-shift" function, the 16 new state tables are assembled to a new event table by the "event-reconstruction" function. It should be noted that the projection of "$D_i[j]$" on

```
Step_1: { reset p_c and "p_0"…"p_15" }

Step_2: // check the state table
        for i:=0 to 15 do
            D^a[p_c] := min(D_i[p_i]);
        end.
        if (D^a[p_c] = 0) then
            if (p_i < num) then
                for i:=0 to 15 do
                    p_i := p_i+1;
                end;
                goto Step_2;
            else
                goto Step_5;
            end;
        else
            goto Step_4;
        end.

Step_3: // realign duration and generate D^a
        for i:=0 to 15 do
            D^a[p_c] := min(D_i[p_i]);
        end.
        if (D^a[p_c] = 0) then
            E^a[p_c] = 0;
            goto Step_5
        else
            goto Step_4;
        end.

Step_4: { reset E^a[p_c] }
        // generate E^a
        for i:=0 to 15 do
            E^a[p_c] := E^a[p_c] + S_i[p_i]×2^i;
        end.
        p_c := p_c+1.
        // split D_i
        for i:=0 to 15 do
            D_i[p_i] := D_i[p_i] - D^a[p_c];
            if (D_i[p_i] = 0) then
                p_i := p_i+1;
            end;
            if (p_i > num) then
                p_i := num;
            end;
        end.
        goto Step_3.

Step_5: // combine the adjacent like-terms
        j :=0;
        for i := 1 to p_c do
            if (E^a[j] = E^a[i]) then
                D^a[j] := D^a[j] + D^a[i];
            else
                j := j+1;
                E^a[j] := E^a[i];
                D^a[j] := D^a[i];
            end;
        end.
        num^a := j+1;
        { output num^a, E^a and D^a }
```

FIG. 5. Flow chart of the "Event-reconstruction" function.

time-axis is shifted by the "HD_i." Consequently, "$S_m[j]$" and "$S_n[j]$" may arise at different time points, even if they have the same subscript "j." So the key point of this function is to comb the 16 state tables according to the projection of "$D_i[j]$" on time-axis. After a new event table is generated from 16 state tables with different delays applied, it is downloaded to the pulse programmer.

The delayed event-duration table can be generated through a process just like "flush left." The detail of the process is shown in Fig. 5. As for special variables, "E^a" and "D^a" represent event and duration in the reconstructed event table, "num^a" is the element number of "E^a," and "p_c" is the pointer pointing to current event; "S_i" and "D_i" represent state and duration in the ith state table, "num" is the element number of "S_0", and "p_0"…"p_{15}" are defined as points pointing to current state of 0^{th}…15^{th} state tables, respectively.

IV. EXPERIMENTS

A home-built MRI spectrometer,[11–13] which consists of a pulse programmer, an RF source, four digital receivers, three gradient waveform generators, and a system clock distribution board, is employed in experiments. As shown in Fig. 6, all the units are inserted into PCI slots of an industrial computer (host computer) running Windows XP (Microsoft, USA). In order to demonstrate the performance of the proposed pulse programmer, the original pulse programmer in the spectrometer is replaced. After suitable modifications, the home-built spectrometer is used to replace the original one in a whole body MRI scanner (OPM 35I, Shanghai Colorful Magnetic Resonance Technology Co., Ltd, Shanghai, China). A C-type permanent magnet with a magnetic field of 0.35 Tesla is used in this scanner. Three sets of bi-planar gradient coils for X, Y, and Z directions are driven by a 3-channel current amplifier (model 231P, Copley Controls Corp., USA). One pre-emphasis unit with 4 time-constants is included in each gradient channel.

Since the minimum pulse width (time between two successive events) of the pulse programmer is affected by the decoding procedure inside the FPGA, the increase in the minimum pulse width is tested using a digital oscilloscope (TDS2024, Tektronix, USA). The measurement result is about 400 ns.

There are some measures to determine the gradient system delay.[7,8,14] In this paper, we use a simple method similar to that proposed in article.[8] In the measurement, a standard

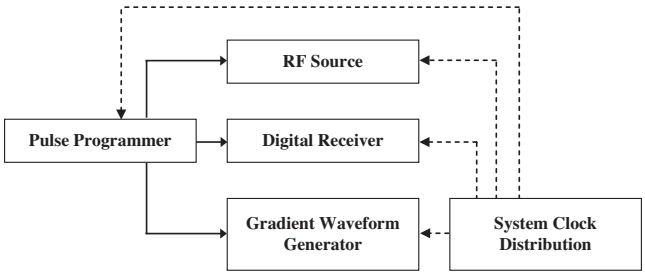

FIG. 6. Block diagram of the spectrometer.

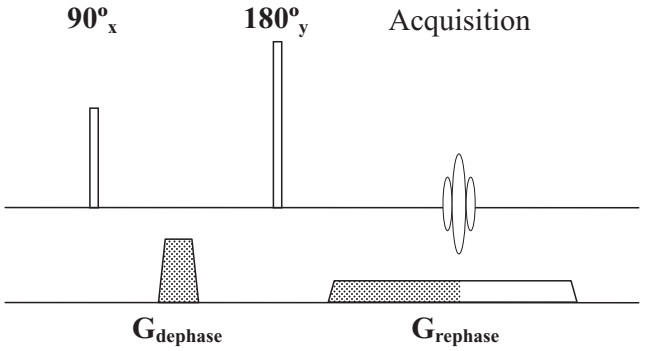

FIG. 7. Pulse sequence timing diagram for testing gradient delay.

receive-only head coil is used to detect the echo signal generator by a cubic phantom. During the execution of a 1D spin echo sequence (shown in Fig. 7), the interval between the desired echo-peak and the measured one is obtained and the gradient system delay of the corresponding channel is calculated accordingly. The gradient system delays of the three gradient channels (x, y, z) are 155 μs, 150 μs, and 190 μs, respectively.

2D fast spin echo (FSE) imaging experiments are executed on a cubic phantom filled with copper sulphate (~1%)

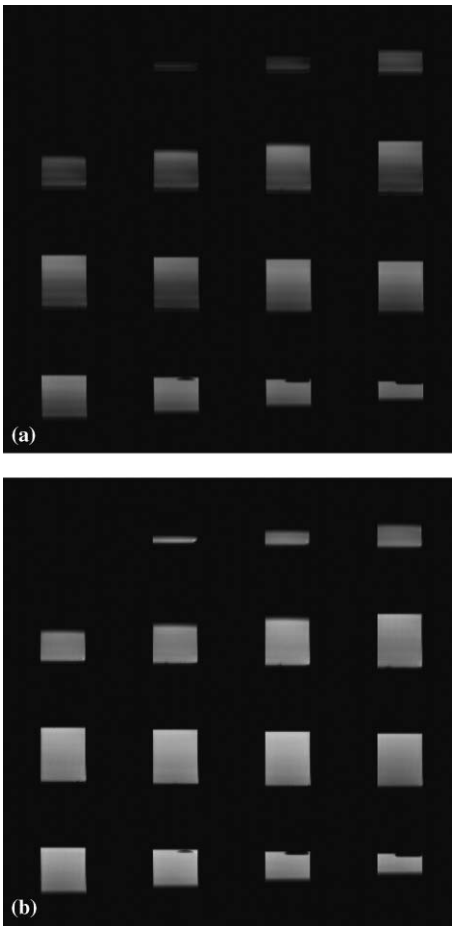

FIG. 8. MRI phantom images in oblique plane acquired using 2D-FSE scan.

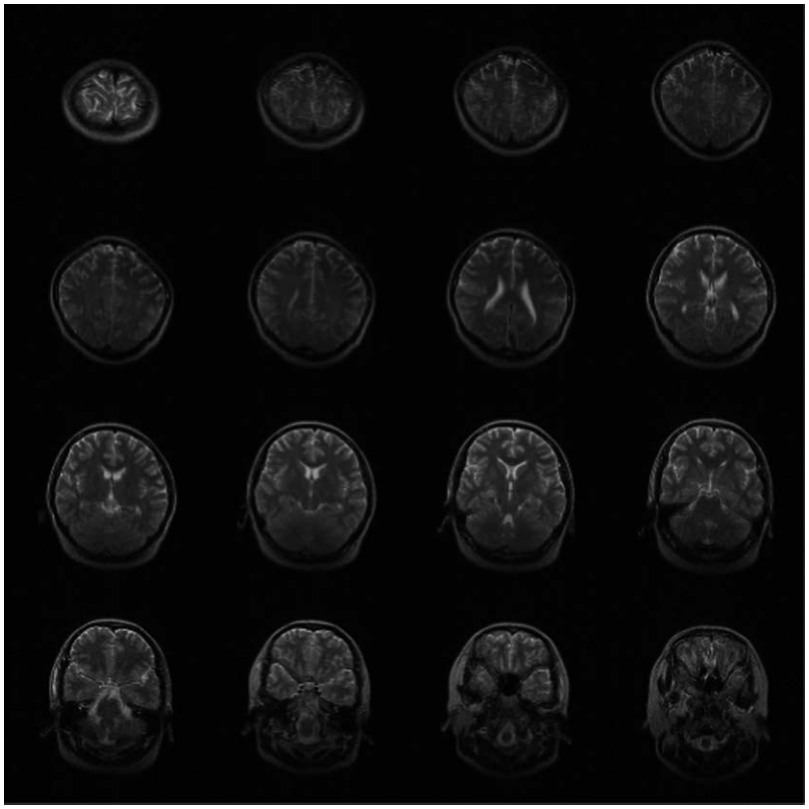

FIG. 9. MRI brain images in oblique plane acquired using 2D-FSE scan.

and sodium chloride (~1%) solution to verify the design of the proposed pulse programmer. FSE imaging is sensitive to gradient delay, so generally a special kind of pre-scan is performed to eliminate band-like artifacts due to the gradient delay. Two scans are carried out in the phantom experiment. In both scans, the pre-scan is omitted. In the first scan, system parameters Dx, Dy, and Dz for the gradient delay compensation are set to 0, and then the FSE imaging is executed with following scanning parameters: 16 slices, 6-mm thickness, FOV = 240, TR = 4000 ms, $TE_{effective}$ = 150 ms, ETL = 9, and NEX = 4. A set of oblique plane are selected. These planes are parallel with each other and the angle between them and the transverse plane is about 30°. In the second scan, the bit-shift is employed to correct the gradient delay and Dx, Dy, and Dz are set to the gradient delays measured previously, i.e., 155 μs, 150 μs, and 190 μs, respectively. The identical planes are scanned with the same sequence and scanning parameters. The acquired images are shown in Fig. 8. It is apparent that images obtained without gradient delay compensation (Fig. 8(a)) suffer from significant band-like artifacts due to the gradient delay. In contrast, the band-like artifacts are invisible in images obtained with bit-shift correction (Fig. 8(b)).

Finally, the 2D FSE imaging experiment are executed on a healthy volunteer with the proposed bit-shift correction. Scanning parameters are as follows: 16 slices, 6-mm thickness, FOV = 240, TR = 4000 ms, $TE_{effective}$ = 120 ms, ETL = 9, and NEX = 4. The scan planes are parallel with each other and the angle between them and the transverse plane is about 30°. The obtained images are shown in Fig. 9.

V. DISCUSSION

So far, an MRI pulse programmer with independent bit-shift capability is implemented and its flexibility and performance is demonstrated by experimental results. It has been shown that gradient delay can be corrected easily and effectively by using independent shift of the RF transmitter and the gradient channels. Of course, the independent bit-shift capability may be achieved directly by using several independent pulse programmers to separately control the RF transmitter and three gradient channels. However, this solution may complicate the design of the MRI spectrometer and consume more hardware resources, even all of the independent pulse programmers can be implemented on a single FPGA chip. Through rebuild of the event-duration table by software process, the pulse programmer design suggested in this article can be used for delayed trigger of each output lines and provides a direct solution to the gradient system delay problem without imposing any special requirements on the pulse sequence design.

The drawback of the proposed pulse programmer is a slight loss of the time resolution in imaging pulse sequences. As mentioned above, in the function "time-gridding," "duration" is rounded to the nearest integral multiple of minimum pulse width of the pulse programmer. In the "event-reconstruction" function, "duration" may be divided so many times that some segments are smaller than minimum pulse width. In order to avoid this trouble, all "duration" are rounded. As a result, the timing resolution is equal to the

minimum pulse width in our pulse programmer (400 ns). However, the influence of this degradation in timing resolution can be negligible in normal imaging.[7,9]

ACKNOWLEDGMENTS

This study is financially supported by Shanghai Science and Technology funds #08DZ1900702 and Natural Science Foundation of China (NSFC) #81000610.

[1]S. Handa, T. Domalain, and K. Kose, Rev. Sci. Instrum. **78**, 084705 (2007).
[2]K. Takeda, Rev. Sci. Instrum. **78**, 033103 (2007).
[3]X. Wu, D. A. Patterson, and L. G. Butler, Rev. Sci. Instrum. **64**, 1235 (1993).
[4]R. W. Quine, J. R. Harbridge, S. S. Eaton, and G. R. Eaton, Rev. Sci. Instrum. **70**, 4422 (1999).
[5]J. R. Porter, S. M. Wright, and N. Famili, Magn. Reson. Med. **32**, 499 (1994).
[6]S. Zhang, K. T. Block, and J. Frahm, J. Magn. Reson. Imaging **31**, 101 (2010).
[7]C. Barmet, N. De Zanche, and K. P. Pruessmann, Magn. Reson. Med. **60**, 187 (2008).
[8]I. C. Atkinson, A. Lu, and K. R. Thulborn, Magn. Reson. Med. **62**, 532 (2009).
[9]C. C. Odebrecht, D. Tagle, F. B. Zuriaga, and C. A. Steren, Concepts Magn. Reson. **30A**, 127 (2007).
[10]T. Toyoda, H. Yoshida, O. Oishi, and S. Miyajima, Rev. Sci. Instrum. **68**, 3140 (1997).
[11]J. Shen, Q. Xu, Y. Liu, and G. Y. Li, Rev. Sci. Instrum. **76**, 105101 (2005).
[12]J. Shen, Y. Liu, J. Q. Li, and G. Y. Li, Magn. Reson. Mater. Phys., Biol., Med. **18**, 332 (2005).
[13]R. P. Ning, Y. D. Dai, G. Yang, and G. Y. Li, Magn. Reson. Mater. Phys., Biol., Med. **22**, 333 (2009).
[14]H. Han, A. V. Ouriadov, E. Fordham, and B. J. Balcom, Concepts Magn. Reson. **36A**, 349 (2010).