Functional Basis for Efficient Physical Layer Classical Control in Quantum Processors

Harrison Ball, Trung Nguyen, Philip H. W. Leong, and Michael J. Biercuk

A functional basis for efficient physical-layer classical control in quantum processors

Harrison Ball,[1] Trung Nguyen,[2, 1] Philip H. W. Leong,[2] and Michael J. Biercuk[1, *]

[1]*ARC Centre of Excellence for Engineered Quantum Systems,*
School of Physics, The University of Sydney, NSW 2006 Australia
[2]*School of Electrical and Information Engineering, The University of Sydney, NSW 2006 Australia*
(Dated: November 4, 2016)

The rapid progress seen in the development of quantum coherent devices for information processing has motivated serious consideration of quantum computer architecture and organization. One topic which remains open for investigation and optimization relates to the design of the classical-quantum interface, where control operations on individual qubits are applied according to higher-level algorithms; accommodating competing demands on performance and scalability remains a major outstanding challenge. In this work we present a resource-efficient, scalable framework for the implementation of embedded physical-layer classical controllers for quantum information systems. Design drivers and key functionalities are introduced, leading to the selection of Walsh functions as an effective functional basis for both programming and controller hardware implementation. This approach leverages the simplicity of real-time Walsh-function generation in classical digital hardware, and the fact that a wide variety of physical-layer controls such as dynamic error suppression are known to fall within the Walsh family. We experimentally implement a real-time FPGA-based Walsh controller producing Walsh timing signals and Walsh-synthesized analog waveforms appropriate for critical tasks in error-resistant quantum control and noise characterization. These demonstrations represent the first step towards a unified framework for the realization of physical-layer controls compatible with large-scale quantum information processing.

I. INTRODUCTION

The suppression of error in quantum systems is a key consideration in many studies of quantum computer architecture [1–6], and emerged early in the community [7] as a significant driver for system design. Unsurprisingly, many studies have focused on the extreme demands in quantum device performance and allocation imparted by logical encoding in quantum error correction (QEC), revealing substantial resource demands on the path to fault-tolerance. Simultaneously, open-loop dynamic error suppression (DES) techniques [8–23] operating at the physical layer have widely been identified as an important potential complement to QEC [5, 24–26]. This is in part because of their potential to improve the resource-efficiency of QEC encoding by reducing the physical-qubit error rate. We expect that in general, DES-operations will be implemented at the physical level for effectively all physical qubit manipulation.

DES techniques have now been experimentally demonstrated to be highly effective [27–32], but are themselves complex, and consideration of overhead and resource allocation associated with their application is also necessary. For instance, general DES protocols require classical control hardware appropriate for the generation of precisely timed and temporally modulated sequences of physical operations tailored to the underlying hardware noise model. The performance specifications of these classical controllers (e.g. in timing resolution or latency [23, 33]) become increasingly demanding as net error suppression requirements increase.

The bulk of today's control-hardware solutions employing high-performance but high-cost waveform generators at the benchtop are not scalable. Meanwhile, custom electronics solutions [34, 35] have focused on reproducing conventional control capabilities in streamlined hardware. To date, studies in this space have not addressed the key opportunity to efficiently allocate classical control hardware resources by considering how to leverage the underlying physics of the control operations themselves.

In this manuscript we address this challenge by first proposing a functional basis for physical-layer classical controllers compatible with both conventional digital electronics and a wide variety of quantum control protocols. Our analysis identifies the Walsh functions [36, 37] as providing a critical mathematical link between the physics of quantum control and the requisite physical implementation of control timing, gate implementation, modulation-waveform synthesis, and environmental noise spectroscopy. We build on a growing body of literature focusing on Walsh functions in quantum control [31, 38–46], and consider here how the use of Walsh functions as a functional programming basis dramatically simplifies the design of embedded classical controllers at the physical-layer. In order to demonstrate the benefits of working within the overarching Walsh framework, we build customized real-time electronic modules in a field-programmable gate array (FPGA) and demonstrate critical functionality including: Walsh-function generation, sequencing based on Walsh-timing information, Walsh-synthesis of analog DES modulation waveforms, and Walsh-based noise spectroscopy and signal reconstruction. These functions demonstrate the core capabilities of a high-efficiency classical controller at the physical layer, and are shown to outperform generic signal synthesis in an embedded microprocessor by factors of $> 10^5$ in terms of signal latency and required computational resources. Our demonstrations highlight a path to realize scalable, "lightweight," embedded control solutions providing high-performance quantum control capabilities.

---

* To whom correspondence should be addressed: michael.biercuk@sydney.edu.au

The remainder of this manuscript is organized as follows. We begin by introducing the relevance of Walsh functions for physical-layer control of quantum information systems in Sec. II. We then move on to articulate key *Summary Requirements* for physical-layer controller functionality in the Walsh framework as well as *Design Drivers* constraining potential solutions in Sec. III, followed by an experimental implementation of a resource-efficient real-time embedded controller addressing the desiderata laid out earlier in the manuscript. We present the relevant controller architecture, demonstrations of representative output, and a comparative analysis relative to a simple microcontroller-based implementation before providing a summary and conclusion in Sec. IV.

II. WALSH FUNCTIONS A FUNCTIONAL PROGRAMMING BASIS FOR PHYSICAL-LAYER CONTROLLERS

Physical layer controllers have the role of implementing control operations in a manner invisible to higher level operations, effectively constituting a type of quantum firmware. This firmware is designed to enact manipulations on physical qubits according to sequencing determined in an abstracted algorithm, and in a manner compatible with the scaling demands of quantum error correction. This implies that physical-layer controls must be robust against ambient and control-based noise sources, and even capable of suppressing error below fault-tolerance thresholds. Our objective here is to understand how these functional requirements, combined with system-level constraints, drive the design of classical controllers at the physical level, and how such design considerations in turn affect the choice of control framework.

Contemporary system-level analyses of quantum computer architecture [5] make clear that there will be a need for dedicated classical control hardware, often discussed in a physically distributed hierarchy [34, 35], with dedicated local resources at the physical-qubit layer. In this generalized framework electronics operating at the physical level can be thought of in the context of *embedded* controllers, bringing with them a new class of specific design constraints. Such embedded controllers must operate semi-autonomously within a hierarchical control architecture, applying and managing complex physical-layer protocols on demand from a master controller operating at higher levels of system abstraction.

In the most naive approach, one may consider implementing physical-layer controllers using standard microcontrollers generating and outputting relevant control signals. However, achieving broad flexibility of the controller's functionality through such general-purpose hardware leads to degraded performance; the controller effectively provides unnecessary capability to the system (e.g. not all possible signal output patterns will be necessary to control qubits), and simultaneously introduces shortcomings in terms of latency and the complexity of programming the device relative to application-specific designs.

We ask the question whether it is possible to constrain the space of available controls in such a way as to both provide all necessary physical-layer control and sequencing functionality and simplify the underlying classical hardware. Our analysis answers this question in the affirmative; the identification of an appropriate mathematical framework can provide a critical link between a (truncated) space of allowed quantum control protocols and sequences and the architecture of the associated hardware. Here, we identify the Walsh functions as a simple encoding basis for the the physical layer controller, bridging the hardware-software interface for quantum control. This family of bi-valued piecewise-constant functions, Fig. 1a, has seen a large variety of applications in digital signal processing [36, 37], and has more recently been adopted in the fields of quantum control and quantum information (see Appendix A for a full summary). In our analyses we consider the Walsh functions as a single unifying mathematical and programming basis for physical-layer control in quantum information processors, leveraging the following key strengths:

Use for a diversity of applications in quantum control: This includes dynamical decoupling for quantum memory (Fig. 1b) [40, 46], Walsh-modulated (Fig. 1c) dynamically protected single-qubit gates (including well known composite pulses) [31, 42, 47], dynamically protected multiqubit gates [48, 49], noise spectroscopy and real-time quantum signal reconstruction [43], the construction of quantum algorithms [39] (language therein refers to Hadamard matrices), and the construction of quantum circuits for the implementation of unitary operators [45, 50]. Across various applications, the Walsh family of controls is shown to incorporate previously known sequences and protocols in a single unifying mathematical framework.

Compatibility with classical digital hardware: This arises from their bi-valued representation and composition from segments with minimum duration $\tau = \beta T_c$, defined as integer multiples of an underlying digital clock cycle, T_c. The period τ defines all other periods within a Walsh function, and gives total sequence duration as a power-of-two multiple of τ. This structure is compatible with both minimum-timing [33] and timing-precision [51] constraints in DES, and is also easily realized in hardware (see Sec. III A).

Simple programming: The error suppressing properties of Walsh DES protocols are captured via a small number of critical parameters, such as the Paley-ordered index of the selected Walsh function, and Walsh-synthesized DES waveforms are sparse in the Walsh basis. For instance the red trace appearing in Fig. 1c consists only of a weighted sum of Walsh functions W_0 and W_3 (subscripts denote the Paley-ordered index, see Sec. III A), but still provides first-order noise filtering comparable with more complex modulation techniques [31, 42]. As a result only limited information is required to define a high-performance Walsh-DES protocol, minimizing the classical communication necessary to program the controller as well as the memory required to produce and store a modulation waveform.

These characteristics, and the demonstrations surveyed in Appendix A, provide compelling evidence that Walsh functions form a suitable programming basis for the implementation of the physical layer controller. In summary, if we restrict ourselves to timing patterns and controls readily represented
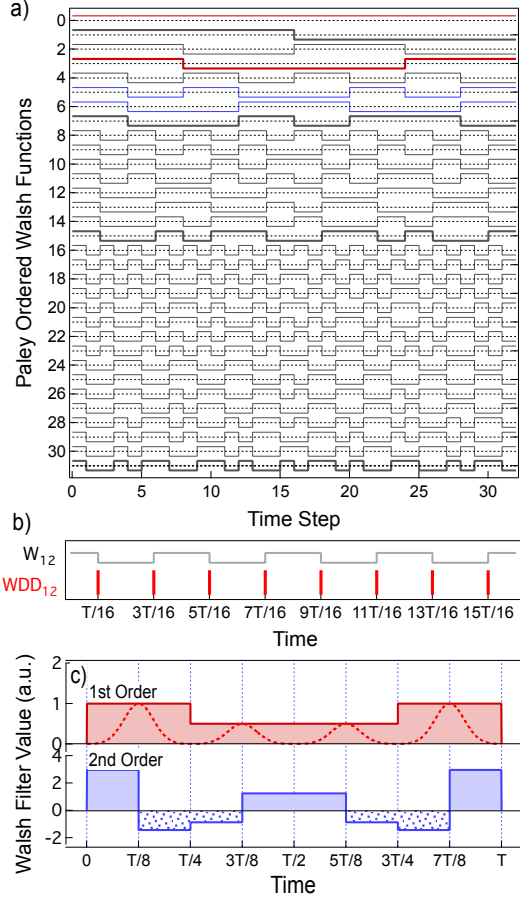
FIG. 1. a) A finite set of the Walsh functions appearing in Paley ordering with normalized total durations. Walsh functions highlighted in bold correspond to the *Thue-Morse* subset, possessing Paley-ordered indices with the highest binary Hamming weight. b) Example trace showing how Paley-order Walsh function with index 12, W_{12} yields the pulse locations for a WDD_{12} sequence via the location of digital transitions. Vertical lines schematically indicate control-pulse locations, normalized to total time, T. c) Example Walsh synthesized filter waveforms providing first and second order noise filtering. Red trace composed of Walsh functions highlighted in red above, blue trace composed of Walsh functions highlighted in both red and blue. Red dashed line indicates that arbitrary shapes for individual pulse segments can be accommodated with the Walsh waveform defining the overall amplitudes of each segment [42].

in mathematical framework of Walsh functions, we will be able to both accomplish a wide range of relevant control tasks and also streamline the architecture of the control hardware. We therefore proceed by considering how the key functionality required of a physical layer controller may be mapped to the design and programming of hardware using the Walsh basis.

III. EXPERIMENTAL IMPLEMENTATION OF AN FPGA WALSH CONTROLLER

In order to demonstrate the value of this design framework for physical layer classical controllers we are motivated to develop a real-time Walsh controller based on a field-programmable gate array (FPGA). Synthesizing the practical demands associated with the application of quantum control operations in the Walsh framework for either algorithmic sequencing at the physical level or DES, we define a set of summary requirements which capture the required output performance and characteristics of the controller. These requirements then inform the specific experimental demonstrations which follow. A broad overview of the use of Walsh functions in quantum control providing justification of these requirements is presented in Appendix A. For full mathematical details describing the Walsh functions in the context of DES see Refs. [40, 42].

Summary Requirements:

(1) Generate time-scaled Walsh functions to establish timing patterns for algorithmic execution or DES application.

(2) Synthesize analog waveforms or digital programming instructions from weighted sums of Walsh functions and use them to modulate a control system in time.

(3) Trigger the generation and output of an analog Walsh-synthesized waveform or other system output synchronously with the Walsh timing system.

(4) Repeat the resulting Walsh-timed sequence of operations a user-defined number of times, meeting needs for long-time memory storage.

(5) Employ Walsh-based signal reconstruction and spectrum analysis using sensor inputs derived from e.g. dedicated sensor qubits.

Even with these functional requirements in hand, there are many approaches to realizing appropriately synthesized outputs in an embedded FPGA controller, and we must therefore consider system-level design drivers that will narrow our choice of implementation.

Design Drivers:

(1) The need for "lightweight" designs minimizing the use of computational hardware in order to maximize scalability.

(2) Constraints on communication bandwidth to the embedded controller, mandating simple programming, especially when deployed in cryogenic environments.

(3) The ability to adjust controller outputs with minimum latency based on new information derived at the physical level or from higher-level algorithmic instructions

Consideration of these design drivers narrows the space of acceptable controller designs by steering implementation towards real-time waveform synthesis instead of precompilation of output control waveforms within the embedded controller. This approach dramatically reduces hardware memory requirements and also minimizes the latency to output of the desired waveform. We proceed by providing an example implementation meeting both the Summary Requirements and the Design Drivers articulated above. The architecture which we employ does not represent a unique solution, but as we will show, achieves the desired performance with significant gains relative to alternate approaches.

In the remainder of this section we describe and experimentally demonstrate the functionality achieved for our Walsh-programmed controller, and provide a detailed description of the hardware implementation. We broadly divide discussion of FPGA performance into the tasks of control-signal generation and system identification. We begin our discussion by introducing the Walsh Generator as the fundamental module used throughout this work, and then describe how various subsystems function in combination to permit a broad range of complex outputs within the Walsh basis.

A. Architecture for controller output-stage

The performance of our FPGA controller architecture is summarized in Figs. 2 and 3, building on modules described in detail in Appendix B. The key element of our approach is a hardware-efficient Walsh Generator (Fig. 2) performing modulo-2 addition of Rademacher functions [52] according to the binary representation of the Paley-ordered Walsh index [53], via combinatorial logic. We construct the Walsh function of Paley order l as

$$\overline{\mathsf{W}}_l(x) = \bigoplus_{j=0}^{m-1} b_j \mathsf{R}_j(x) \tag{1}$$

where $x \in [0, 1]$ defines the continuous domain, $b_j \in \{0, 1\}$ are bit-values in the binary representation of l, written $(b_{m-1}b_{m-2}...b_0)_2$, and $\oplus$ denotes modulo-2 addition. The building blocks in this construction are the Rademacher functions, with the jth order Rademacher defined by

$$\mathsf{R}_j(x) = [1 - \mathrm{sgn}\left[\sin\left(2^{j+1}\pi x\right)\right]]/2, \;\; j \geq 0. \tag{2}$$

This corresponds to a digital "clock" starting in the 0 state, and switching between state values $\{0, 1\}$ with frequency 2^j (period 2^{-j}). Following Eq. 1, target Walsh functions are generated in real time via composition of the Rademacher functions under modulo-2 addition. An example of this construction for Walsh function $\overline{\mathsf{W}}_{12}(x)$ is presented in the lower panel of Fig. 2. All Walsh functions defined by Eq. 1 start with initial value 0, and in particular the constant-valued zeroth order function is $\overline{\mathsf{W}}_0(x) = 0$. For some tasks, however, it is useful to redefine the Walsh basis with a non-zero value for the zeroth-order. For this we define the complement Walsh basis
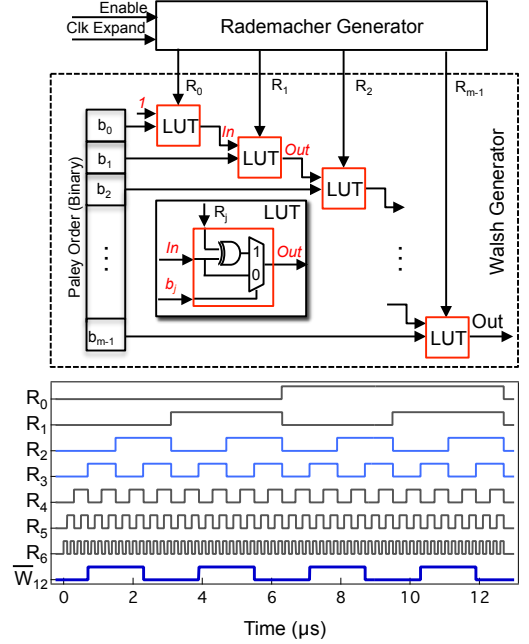

FIG. 2. Schematic of a high-efficiency real-time Walsh-function generator implemented in FPGA. Square-wave Rademacher functions are summed with the binary digits of the desired Walsh function in Paley ordering via a cascade of lookup tables (LUT), each composed of an XOR gate and a multiplexer. Timing of the Rademacher functions is set by the Clk Expand input such that the base time period of the Walsh function is a fixed multiple of the clock period. The Enable input begins the signal generation via an external hardware trigger. Lower panel shows real-time oscilloscope data-capture of Rademacher functions of various orders, with period increasing by a factor of two with each increase in the Rademacher order. The two light blue Rademachers (R_2 and R_3) undergo modular addition via the Walsh Generator to form the output Walsh function with Paley order 12 ($12 = 2^3 + 2^2$, or $12 \rightarrow 1100$, $b_2 = b_3 \neq 0$), shown in dark blue at the bottom of the figure. Additional circuitry has been constructed to route Rademacher functions to digital output pins for this diagnostic measurement. In this demonstration we employ a Xilinx Zynq 7010 as part of a Dual-core ARM Coretex A9 + FPGA system-on-chip packaged by Red Pitaya.

via a bit-inversion of Eq. 1 as

$$\mathsf{W}_l(x) = \overline{\mathsf{W}}_l(x) \oplus 1 \tag{3}$$

which is also generated by the Walsh Generator module in Fig. 2 after a trivial modification.

Leveraging the ability to generate Walsh functions of arbitrary order in real time we develop complex modules for Walsh-timing signal generation and Walsh-based waveform synthesis, in order to deliver on the Summary Requirements outlined above. All submodules are synchronized to the same 100 MHz system clock supplied by the FPGA hardware. First, as represented in Fig. 3 the Walsh Timing Sequencer determines the timing of application of physical-layer control operations such as WDD, with control operations applied on rising and falling edges of the Walsh functions scaled temporally by
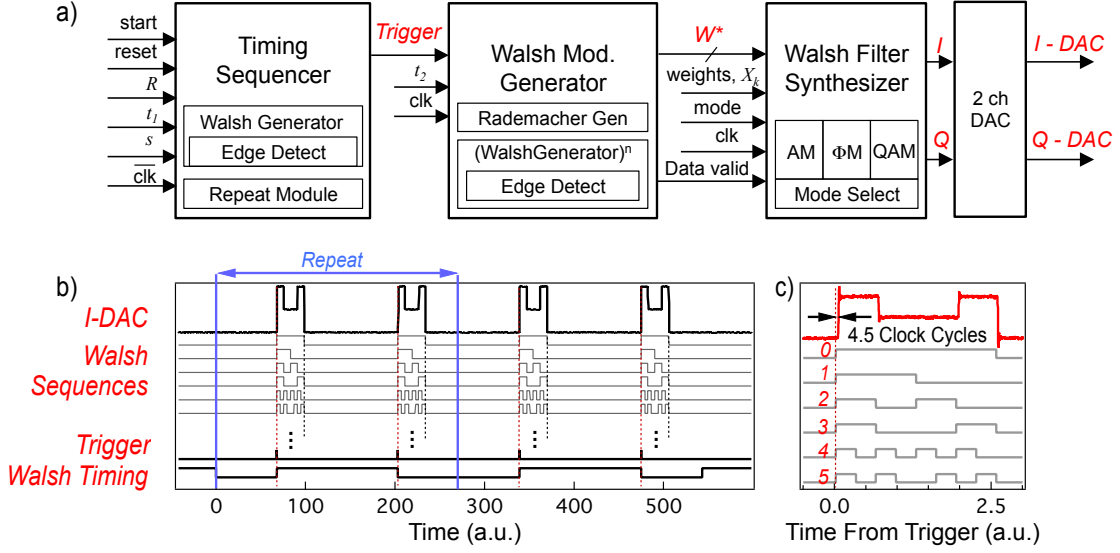
FIG. 3. Schematic of a high-efficiency real-time Walsh-based controller implemented in FPGA. a) Key subsystems used to fulfil the *Summary Requirements* listed in the main text. Inputs to the Timing Sequencer include "start" which triggers the output, "reset" which resets the state of the system, "repeat" which determines, R, the number of sequence repeats , "t_1" which sets the clock expansion on signal "clk" for the primary Walsh timing signal and "order" which selects the Paley ordered Walsh function to define the timing signal. The Walsh Modulation Generator takes as input the "Trigger" signal output by the Timing Sequencer as well as another clock and clock-expansion parameter. The Walsh Filter Synthesizer takes input Walsh sequences, denoted W^* as well as "weights" which define the analog Walsh synthesis coefficients, "mode" which selects the modulation protocol, and a clock. All clock signals are generated by a phase-lock-loop. We employ two clocks with equivalent frequencies but differing by a π phase shift for the Timing Sequencer and the two subsequent modules respectively; this results in a half-clock-cycle delay as observed in panel (c). b) Representative example outputs of the relevant main subsystems using our FPGA implementation and captured using an oscilloscope. Red vertical dotted lines indicate trigger locations associated with Walsh timing, here $\overline{W}_3$ repeated twice. In this example I-DAC corresponds to an AM-only Walsh noise filtering protocol highlighted in c) The modulation waveform for each pulse provides a first-order Walsh Amplitude Modulated Filter via a weighted sum of W_0 and W_3. The first six Walsh functions generated by the Walsh Modulation Generator on a trigger from the Timing Sequencer are shown. Again, in order to access signals not typically routed directly to FPGA outputs (e.g. the synthesized Walsh functions), additional FPGA circuitry has been implemented for diagnostics only.

parameter t_1. For this task we use Walsh functions of the form $\overline{W}_l(x)$ where the first rising edge occurs after $x = 0$, so it remains useful as a timing trigger in real devices with (small) finite physical delay times. Another critical task of the FPGA is to generate, in real time and on a designated trigger, a set of Walsh functions which are then used to form an analog waveform as a superposition via standard techniques in Walsh synthesis. Outputs of the Walsh Timing Sequencer generated by an Edge Detect submodule trigger a Walsh Modulation Generator module, which uses n copies of the Walsh Generator to produce, in parallel, the first n Walsh functions in Paley order. For this task we use Walsh functions of the form $W_l(x)$ to enable waveform synthesis with a non-zero constant-offset component ($W_0(x) = 1$).

The resultant Walsh functions, scaled by timing information t_2, are fed into the Walsh Filter Synthesizer where they are combined in real time with user-defined weights, X_k. The outputs of this module are streaming digital programming instructions representing a modulation waveform for a control operation within the Walsh family. We customize solutions for Walsh filter synthesis for three key modulation techniques in use: amplitude modulation (AM) of a control signal im-

plementing a Pauli-x operation for suppression of dephasing noise; phase modulation (ΦM) for suppression of amplitude damping noise; and Quadrature Amplitude Modulation (QAM) employed for simultaneous suppression of noise in both quadratures. Through this approach we can realize a wide variety of DES modulation protocols such as Walsh amplitude modulated filters [42].

Our controller architecture segregates pulsed-carrier synthesis (resonant with the qubit transition) from baseband modulation generation, providing a range of modulation implementation options. For instance, we know that Walsh-synthesized modulation waveforms may be used either to directly define the shape of a square pulse, or to define the maximum-amplitude scaling of pulse segments with arbitrary shape (e.g. Gaussian), as shown in Fig. 1c. Hence the downstream use of signals I and Q may be tailored to the needs of a specific hardware system (see Fig. 5 in Appendix B for details). In our example implementation these signals are constructed to directly program a digital-to-analog converter (DAC). The DAC outputs voltages that may be used to drive the in-phase (I-DAC) and quadrature (Q-DAC) channels of an IQ mixer which directly modulates the near-resonant carrier.

Example digital and analog output signals captured by a mixed-signal oscilloscope are demonstrated in the lower panel of Fig. 3. Here the output is a first-order Walsh-Amplitude-Modulated Filter triggered synchronously with a repeated $\overline{W}_3$ timing pattern. This particular gate has been experimentally demonstrated to provide efficient suppression of errors induced by non-commuting dephasing noise [31]. Timing, given by a repeated Walsh sequence, could be used for WDD application in error suppression or system identification as described in the next section. Total system latency from trigger to output of the analog waveform is only 4.5 clock cycles. A detailed performance comparison relative to programming of a conventional microcontroller is provided in Sec. III C.

B. Real-time Walsh signal reconstruction

Another set of tasks in implementing the embedded controller relates to the application of sensing and signal-reconstruction techniques using embedded sensors at the hardware level. We proceed by defining an approach consistent with Walsh control and leveraging concepts developed in [43] for real-time signal reconstruction using Walsh-modulated quantum sensors. The implementation of a real-time Walsh-signal-reconstruction protocol outlined in [43] requires functionality similar to that described above. This primarily includes the ability to sequence WDD on multiple parallel sensors and then process the resulting projective measurements in order to determine a decomposition of the underlying noise process in the Walsh basis.

Given an ensemble of qubits, each subjected to a time-varying dephasing field $b(t)$, and modulated by a Walsh dynamic decoupling sequence of different orders defined by $W_k(t/T)$, each will accumulate a net dynamic phase $\phi_k(T)$. This phase is linked to a measurable quantity P_k defining the weight of each Walsh function in a decomposition as

$$X_k(T) = \frac{\sin^{-1}[2P_k - 1]}{\gamma T}. \tag{4}$$

With the resulting Walsh spectrum over order k in hand one may reconstruct an approximation of the underlying noise signal as

$$\hat{b}(t) = \sum_{k=0}^{N-1} X_k W_k(t/T). \tag{5}$$

See Appendix C for full details of the measurement protocol.

With this framework in mind, we design a Walsh SID module to calculate Walsh weights, and produce real-time reconstructions $\hat{b}(t)$ from a given set of fidelity values. The input information for this module is the vector of fidelity values, $\boldsymbol{P} = (P_0, P_1, ..., P_{N-1})$ generically describing the measurement data of the sensor array, though here generated from numerical simulations (Appendix C). These values are converted into estimates of the corresponding Walsh weights using Eq. 4, and subsequently incorporated in a Walsh waveform synthesizer module to implement a real-time reconstruction $\hat{b}(t)$
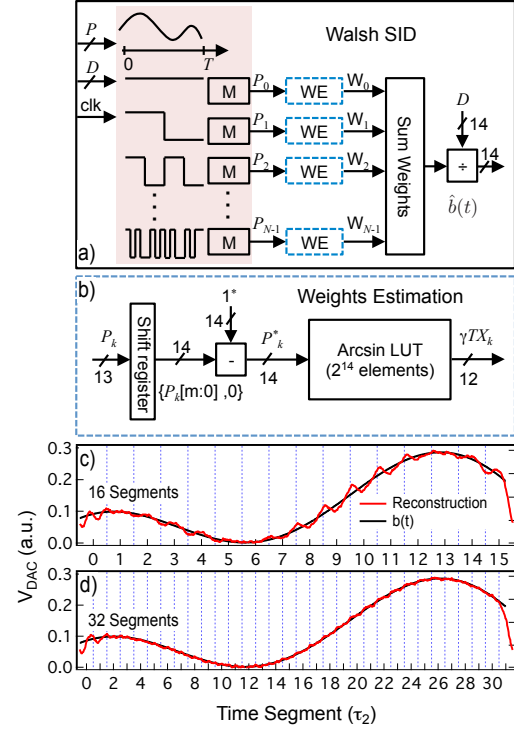

FIG. 4. Schematic of Walsh-based systems identification (SID) module implemented in FPGA. a) Here the red shaded box indicates a protocol performed either at the qubit level, or as here, in simulation, returning fidelities P_k from measurements M and feeding them to Walsh Estimation modules (b). The input parameter D is responsible for scaling the the arcsin function, taking the value $D = \gamma T$. The output of weight summation module is the reconstructed signal $\hat{b}(t)$ for the magnetic field, represented digitally and here used to program a DAC for signal monitoring. b) Detailed schematic of Weights Estimation. The 13-bit binary presentation of Fidelity (P_k) doubles its value by using a shift-left 1-bit register and then subtracts to 14-bit binary presentation of "1" to derive $\arcsin(\gamma T X_k)$ as shown in Eq. C4. The arcsin LUT is a 2^{14}-element look-up table which returns TX_k. A divider is used to scale the output of arcsin LUT with divisor T and output the final result Walsh weights X_k. The selected value of the bit-depth for the arcsin function is determined by the specific hardware implementation employed here. c)-d) Analog waveforms output from the DAC overlaid with applied noise signal given numerically calculated values for P_k, using 16 or 32 Walsh functions (hence, individual fidelity measurements).

on an FPGA. The central capability of the Walsh SID module is captured in the Weights Estimation module shown in Fig. 4, with reconstruction of an analog waveform subsequently produced by the Sum Weights module relying on the same architecture as the Walsh Filter Synthesizer described in Sec. B 3.

C. Comparative performance of the Walsh controller

The key strengths of this approach are the ability to produce outputs in real time with limited hardware resources dedicated to the controller. For instance on the Xilinx ZC7Z010 FPGA

device, one of the smallest in the System-on-Chip family, implementation of the scheme in Fig. 3 for the first 8 Paley orders, occupies less than 5% of the available 17,600 lookup tables (LUTs). Moreover, devices such as the XC7Z100 have 277,400 LUTs, this being a factor of 15 larger. The direct mapping of Walsh functions to simple digital logic constructions allows for a highly autonomous system with minimal requirements on communication bandwidth to the controller, requiring only 345 bits for full programming. This efficiency arises because our control solutions are sparse in the Walsh basis.

A key performance metric for this system is hardware latency, which is optimized through use of a hardware-timing solution. In our implementation, from trigger input to the start of signal generation requires only 4.5 clock cycles and from changes in input variables to ready-to-output status requires only two clock cycles. The system may be reset at any time and returns to ready-to-output status in three clock cycles. Key contributions to the output delay are as follows:

For comparison we built a system with similar functionality using an embedded microcontroller (Arduino MEGA 2560). In this implementation, because the output is compiled rather than generated by hardware in real time, the entire output waveform must be calculated and stored in memory before the system is ready-to-output. This process requires nearly 1.3 M clock cycles for the waveforms demonstrated above, and must be performed every time a variable is changed or reset signal received. Similarly, Walsh-sequence calculation is 8,000 times slower on the microprocessor. Other challenges with timing resolution, synchronization, and repeatability due to the microcontroller interpreting instructions to produce output signals make this implementation far inferior relative to the FPGA. For instance, the sequential architecture of the microcontroller requires that instructions be completed before the system responds to a change in state; this results in delays from trigger to output of an AM envelope of ~ 125 clock cycles, contrasted to just 4.5 in the FPGA. Similarly, any update to variables such as the Walsh order, Walsh weighting coefficients, or timing information requires full calculation of the waveform, or ~ 1.3 M clock cycles. Importantly, these delays would persist in the microcontroller solution regardless of the selected functional basis for waveform creation.

Finally, because many high-performance modulation patterns are sparse in the Walsh basis (e.g. the example in Fig. 1c consisting only of a weighted sum of W_0 and W_3) they may be generated in real time using simple digital logic on the FPGA, in contrast to generating a similar modulation waveform using e.g. Fourier synthesis in either a microcontroller or an alternate FPGA-based synthesizer. For instance, assuming a base period τ and clock cycle, T_c, generating the same waveform using Fourier synthesis would require at least τ/T_c sine functions to be combined to achieve the same timing resolution. Such an approach would lead to a significant expansion of required memory allocation, programming complexity, and combinatorial logic.

The Walsh SID module has similar performance characteristics of reset-to-ready delay time or changes to ready-to-output latency, with trigger-to-output requiring 5 clock cycles

to produce a valid output. This performance metric plays an important role when we try to optimize the latency of each design module to achieve a real-time response. Low output latency requires significant hardware resource allocation, with particular focus on the arcsin LUT, which consumes $\sim 5\%$ of Block Ram (total 60 units) for a single instance. Other modules remain resource efficient, maintaining hardware requirements at less than 1% for FF (Flip Flops) or 6% of LUTs (total 35,200 FFs and 17,600 LUTs). The trigger-to-output latency distribution are listed as follows: The Weights Estimation module generates the parameter $\gamma T X_k$ in two clock cycles. The output of this module will enable the Sum Weights module (Fig. 4) to begin reconstruction of the digital representation of $\hat{b}(t)$ after two clock cycles. The final divider module scales the output amplitude within one clock cycle.

IV. CONCLUSION

In this work we have explored the functional drivers and system-level constraints associated with implementing physical-layer control, and used them to identify the Walsh functions as a useful functional basis for the programming and design of efficient embedded classical controllers. We have considered the requirements of dynamic error suppression and physical-layer operation sequencing in order to develop a notional design for a Walsh-based controller. Following this we have implemented this architecture on an FPGA and demonstrated considerable performance advantages relative to approaches leveraging general-purpose signal synthesis via microcontrollers. Alternative FPGA architectures may be considered, but we emphasize that our approach of real-time Walsh-signal synthesis and SID provides substantial benefits in terms of FPGA resources and output latency – both key drivers in the context of designing embedded physical layer controllers in quantum information.

Overall we believe that the requirements of implementing physical-layer control, and in particular physical-layer error evasion, have largely been overlooked in past architectural studies in favor of mapping physical qubit layouts to QEC-code connectivity. This is understandable given the extraordinary demands arising from consideration of QEC, but is in our view insufficient to consider realistic quantum information architectures at scales where logical qubit error rates will need to be exceedingly small (comparable to the inverse problem size [54]). By showing how existing analyses may be augmented through consideration of the design of embedded physical-layer controllers, this work advances broad architectural studies in a manner consistent with our current understanding of projected physical-layer error-rates and control requirements. In addition, this work addresses the general question of how to efficiently sequence and apply physical-qubit manipulations that are required even without use of DES. Future work will consider the integration of FPGA Walsh controllers in quantum information experiments, and detailed consideration of the interface between the classical controller and logical layers [5].

ACKNOWLEDGMENTS

[1] R. V. Meter and M. Oskin, J. Emerg. Technol. Comput. Syst. **2**, 31 (2006).

[2] R. Van Meter, W. J. Munro, and K. Nemoto, in *Theory of Quantum Computation, Communication, and Cryptography* (Springer Berlin Heidelberg, Berlin, Heidelberg, 2008) pp. 105–114.

[3] D. P. DiVincenzo, Physica Scripta **2009**, 014020 (2009).

[4] S. J. Devitt., A. G. Fowler, T. Tilma, W. J. Munro, and K. Nemoto, International Journal of Quantum Information **08**, 121 (2010).

[5] N. C. Jones, R. Van Meter, A. G. Fowler, P. L. McMahon, J. Kim, T. D. Ladd, and Y. Yamamoto, Phys. Rev. X **2**, 031007 (2012).

[6] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Phys. Rev. A **86**, 032324 (2012).

[7] M. Oskin, F. Chong, and I. Chuang, Computer **35**, 79 (2002).

[8] L. Viola and S. Lloyd, Phys. Rev. A **58**, 2733 (1998).

[9] L. Viola, S. Lloyd, and E. Knill, Phys. Rev. Lett. **83**, 4888 (1999).

[10] P. Zanardi, Phys. Lett. A **258**, 77 (1999).

[11] D. Vitali and P. Tombesi, Phys. Rev. A **59**, 4178 (1999).

[12] L. Viola and E. Knill, Phys. Rev. Lett. **90**, 037901 (2003).

[13] M. S. Byrd and D. A. Lidar, Phys. Rev. A **67**, 012324 (2003).

[14] A. G. Kofman and G. Kurizki, Phys. Rev. Lett. **93**, 130406 (2004).

[15] K. Khodjasteh and D. A. Lidar, Phys. Rev. Lett. **95**, 180501 (2005).

[16] W. Yao, R. B. Liu, and L. J. Sham, Phys. Rev. Lett. **98**, 077602 (2007).

[17] G. S. Uhrig, Phys. Rev. Lett. **98**, 100504 (2007).

[18] G. Gordon, G. Kurizki, and D. A. Lidar, Phys. Rev. Lett. **101**, 010403 (2008).

[19] K. Khodjasteh and L. Viola, Phys. Rev. Lett. **102**, 080501 (2009).

[20] K. Khodjasteh and L. Viola, Phys. Rev. A **80**, 032314 (2009).

[21] K. Khodjasteh, D. A. Lidar, and L. Viola, Phys. Rev. Lett. **104**, 090501 (2010).

[22] W. Yang, Z.-Y. Wang, and R.-B. Liu, Frontiers Phys. **6**, 1 (2010).

[23] M. J. Biercuk, A. C. Doherty, and H. Uys, J. Phys. B **44**, 154002 (2011).

[24] H. K. Ng, D. A. Lidar, and J. Preskill, Phys. Rev. A **84**, 012305 (2011).

[25] K. Khodjasteh and D. A. Lidar, Phys. Rev. Lett. **95**, 180501/1 (2005).

[26] K. Khodjasteh and D. A. Lidar, Physical Review A **78**, 012355 (2008).

[27] M. J. Biercuk, H. Uys, A. P. VanDevender, N. Shiga, W. M. Itano, and J. J. Bollinger, Nature **458**, 996 (2009).

[28] J. Du, X. Rong, N. Zhao, Y. Wang, J. Yang, and R. B. Liu, Nature **461**, 1265 (2009).

[29] H. Bluhm, S. Foletti, I. Neder, M. Rudner, D. Mahalu, V. Umansky, and A. Yacoby, Nature Phys. **7**, 109 (2011).

[30] S. Gustavsson, J. Bylander, F. Yan, P. Forn-Díaz, V. Bolkhovsky, D. Braje, G. Fitch, K. Harrabi, D. Lennon, J. Miloshi, P. Murphy, R. Slattery, S. Spector, B. Turek, T. Weir, P. B. Welander, F. Yoshihara, D. G. Cory, Y. Nakamura, T. P. Orlando, and W. D. Oliver, Phys. Rev. Lett. **108**, 170503 (2012).

[31] A. Soare, H. Ball, D. Hayes, J. Sastrawan, M. C. Jarratt, J. J. McLoughlin, X. Zhen, T. J. Green, and M. J. Biercuk, Nat. Phys. **10** (2014).

[32] F. K. Malinowski, F. Martins, P. D. Nissen, E. Barnes, L. Cywinski, M. S. Rudner, S. Fallahi, G. C. Gardner, M. J. Manfra, C. M. Marcus, and F. Kuemmeth, Nature Nanotechnology (2016).

[33] T. E. Hodgson, L. Viola, and I. D'Amico, Phys. Rev. A **81**, 062321 (2010).

[34] I. C. Lamb, J. Colless, J. Hornibrook, S. Pauka, S. Waddy, M. Frechtling, and D. Reilly, Review of Scientific Instruments **87**, 014701 (2016).

[35] J. Hornibrook, J. Colless, I. C. Lamb, S. Pauka, H. Lu, A. Gossard, J. Watson, G. Gardner, S. Fallahi, M. Manfra, and D. Reilly, Physical Review Applied **3**, 024010 (2015).

[36] S. G. Tzafestas, ed., *Walsh functions in signal and systems analysis and design*, Benchmark papers in electrical engineering and computer science, Vol. 31 (Van Nostrand Reinhold Company, New York, 1985).

[37] K. G. Beauchamp, *Walsh Functions and their Applications* (Academic Press, London, 1975).

[38] D. W. Leung, I. L. Chuang, F. Yamaguchi, and Y. Yamamoto, Phys. Rev. A **61**, 042310 (1999).

[39] D. Leung, J. Modern Optics **49**, 1199 (2002).

[40] D. Hayes, K. Khodjasteh, L. Viola, and M. J. Biercuk, Phys. Rev. A **84**, 062323 (2011).

[41] D. Hayes, S. M. Clark, S. Debnath, D. Hucul, I. V. Inlek, K. W. Lee, Q. Quraishi, and C. Monroe, Phys. Rev. Lett. **109**, 020503 (2012).

[42] H. Ball and M. Biercuk, EPJ Quantum Technology **2**, 11 (2015), 10.1140/epjqt/s40507-015-0022-4.

[43] A. Cooper, E. Magesan, H. N. Yum, and P. Cappellaro, Nat. Commun. **5** (2014).

[44] T. J. Green and M. J. Biercuk, Phys. Rev. Lett. **114**, 120502 (2015).

[45] J. Welch, D. Greenbaum, S. Mostame, and A. Aspuru-Guzik, New Journal of Physics **16**, 033040 (2014).

[46] K. Khodjasteh, J. Sastrawan, D. Hayes, T. J. Green, M. J. Biercuk, and L. Viola, Nat. Commun. **4**, 2045 (2013).

[47] J. T. Merrill and K. R. Brown, "Progress in compensating pulse sequences for quantum computation," in *Quantum Information and Computation for Chemistry* (John Wiley & Sons, Inc., 2014) pp. 241–294.

[48] D. Hayes, S. M. Clark, S. Debnath, D. Hucul, I. V. Inlek, K. W. Lee, Q. Quraishi, and C. Monroe, Phys. Rev. Lett. **109**, 020503 (2012).

[49] T. J. Green and M. J. Biercuk, Phys. Rev. Lett. **114**, 120502

[50] J. Welch, A. Bocharov, and K. M. Svore, Quantum Information & Computation **16**, 87 (2016).

[51] M. J. Biercuk, A. C. Doherty, and H. Uys, J. Phys. B **44**, 154002 (2011).

[52] H. Rademacher, Math. Ann. **87**, 112 (1922).

[53] R. Paley, Proceedings of the London Mathematical Society **2**, 241 (1932).

[54] M. Suchara, A. Faruque, C.-Y. Lai, G. Paz, F. T. Chong, and J. Kubiatowicz, arXiv:1312.2316 (2015).

[55] T. J. Green, J. Sastrawan, H. Uys, and M. J. Biercuk, New J. Phys. **15**, 095004 (2013).

[56] K. Khodjasteh, H. Bluhm, and L. Viola, Phys. Rev. A **86**, 042329/1 (2012).

[57] K. R. Brown, A. W. Harrow, and I. L. Chuang, Phys. Rev. A **70**, 052318 (2004).

[58] D. Leibfried, B. DeMarco, V. Meyer, D. Lucas, M. Barrett, J. Britton, W. M. Itano, B. Jelenkovic, C. Langer, T. Rosenband, and D. J. Wineland, Nature **422**, 412 (2003).

[59] J. Majer, J. M. Chow, J. M. Gambetta, J. Koch, B. R. Johnson, J. A. Schreier, L. Frunzio, D. I. Schuster, A. A. Houck, A. Wallraff, A. Blais, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf, Nature **449**, 443 (2007).

[60] R. F. Stengel, *Optimal Control and Estimation* (Dover, Mineola, New York, 1994).

[61] G. A. Álvarez and D. Suter, Phys. Rev. Lett. **107**, 230501 (2011).

[62] J. S. Bylander, S. Gustavsson, F. Yan, F. Yoshihara, K. Harrabi, G. Fitch, D. G. Cory, Y. Nakamura, J.-S. Tsai, and W. D. Oliver, Nature Phys. **7**, 565 (2011).

[63] L. M. Norris, G. A. Paz-Silva, and L. Viola, Phys. Rev. Lett. **116**, 150503 (2016).

[64] S. Kotler, N. Akerman, Y. Glickman, and R. Ozeri, Phys. Rev. Lett. **110**, 110503 (2013).

Appendix A: Physical layer controls in the Walsh basis

In the following subsections we provide greater detail on the specific application of Walsh functions in the implementation of physical-layer quantum control.

1. Walsh timing in quantum control

Walsh sequences can be used to provide timing information relevant to the sequencing of control operations on qubits at the physical level, with rising and falling edges employed to trigger the application of a desired operation. Generically this can be employed for logical or physical-layer algorithmic implementation or unitary gate construction [39, 45, 50], and of course it may also be advantageous to apply all physical-layer operations timed via Walsh sequences due to their discrete timing structure. We therefore refer generically to Walsh *timing* patterns.

The simplest application of Walsh functions in the algorithmic application of DES specifically comes through association of the Walsh function itself with the control propagator of a dynamical decoupling (DD) sequence for implementation of the identity (memory) [40]. Each digital transition in a Walsh function corresponds to the location of a control pulse in the resulting WDD sequence, Fig. 1b. Incredibly, a number of familiar DD sequences can be identified as special instances of WDD sequences: Using the Paley-ordering convention as an index WDD$_0$ (a WDD sequence with pulses prescribed at the transitions in a W_0 Walsh function) is free evolution and WDD$_1$ is the spin echo sequence. Similarly, special instances of Walsh dynamical decoupling correspond to Periodic Dynamical Decoupling (PDD), CPMG, and Concatenated Dynamical Decoupling [40]. Changing the value of τ, the base period of the Walsh function, changes the frequency response of the WDD sequence [55]. Importantly, the order of error suppression realized in a WDD sequence is determined entirely by a single number, r, the Hamming weight (number of ones) of the binary representation of the Walsh function's Paley-ordered index.

For generalized storage of quantum states we previously identified that it is possible to reconcile the oppositional demands for high-fidelity state storage and low access latency to stored states protected by DD via periodic repetition of a base, high-order DD cycle [46]. This insight led to demonstrations that repetition of WDD sequences could result in conditions by which a target memory fidelity could be guaranteed at long storage times, but with access latency capped at the duration of the base WDD cycle. Moreover repetition is a natural mathematical operation within the Walsh family, meaning that repeated sequences generally also fall within the Walsh family.

2. Walsh-synthesized modulation for noise-suppressing control operations

The implementation of DES single and two-qubit operations requires the application of temporally modulated control fields (e.g. frequency, phase, or amplitude of a near-resonant carrier) which enact the qubit manipulations with added robustness against environmental or control fluctuations. Following the general approach of composite pulsing we consider an arbitrary operation decomposed into discrete time-segments; these modulated pulses would then be applied according to a precise timing pattern in order to implement a desired higher-level algorithm or protocol.

The time-domain modulation waveform defining the envelope for the phase or amplitude of a near resonant carrier may be crafted using Walsh synthesis, known to facilitate construction of arbitrary waveforms via superpositions of Walsh functions. The mathematical properties of the Walsh functions establish key analytic design rules simplifying the construction of high-performance Walsh modulation protocols for arbitrary single-qubit gates [31, 42]. These protocols also bring the benefit that high-performance noise-suppressing solutions are sparse in the Walsh basis, requiring superposition of only a few Walsh functions (Fig. 1c). In addition, a variety of well known dynamically corrected gates [20, 21, 56] and NMR-inspired composite pulses [47, 57] are included in the Walsh synthesis framework (Ref. [42]).

Two-qubit gates implemented via an intermediary bosonic mode – such as geometric phase gates in trapped ions [58] and superconducting qubits [59] – can be modulated using

Walsh functions in order to improve robustness against fluctuations in the parameters of the mediating driving field (e.g. amplitude and frequency). This includes phase-modulation applied to gates mediated by a single bosonic mode [48] with error suppressing properties described as in WDD. In addition, application of a phase-modulation pattern described by the Thue-Morse subset of the Walsh functions (i.e. those with Paley order $k = 2^n - 1$ for n an integer), provides a means to decouple multiple bosonic modes from the qubit state [49]. Here, repetition of the relevant Walsh modulation pattern provides a means to improve the robustness of the gate to time-dependent fluctuations. Remarkably, even the phase values applied during each segment in a Walsh-timed modulation pattern may be represented using the properties of Walsh functions.

3. Walsh-based SID

In any control setting it is required that we have both a model of the underlying physical system to be controlled (including its response to external stimuli) and up-to-date information about any disturbances that may need to be suppressed or compensated. In control theoretic language the determination of this information is broadly referred to as "System Identification" [60], overlapping the quantum physics concept of Hamiltonian Estimation when we are addressing the characteristics of the noise processes to be suppressed via physical-layer control. Information gained through such protocols may then be employed to adjust or even optimize the selected physical layer controls.

Techniques for system identification in noisy quantum systems already exist and are widely employed in the community at the physical layer. For instance, dynamical-decoupling-based spectral reconstruction generally involves execution of a set of repeated DD sequences followed by projective measurements [61–63]. The relevant DD sequences fall within the WDD family, and are restricted to those subsequences which have a narrow spectral response in the Fourier domain [51, 64]. By appropriately modulating the temporal evolution of the system-environment interaction with DD, measurement outcomes on the system become maximally sensitive to the corresponding spectral content of the noise. These measurements are followed by processing and reconstruction of data to determine the underlying noise power spectrum.

In addition to ensemble-averaged statistical information about extrinsic noise processes, real-time estimation of the the dominant noise processes may be derived following protocols described by e.g. [43]. Here parallel qubit sensors are subjected to synchronized noisy evolution, while being modulated by WDD sequences of different orders, returning multiple projective-measurement outcomes. From these fidelity measurements the underlying time-varying signal can be reconstructed in each discrete timestep, τ, using concepts from Walsh synthesis. In the limit that these measurement results are not performed synchronously the same information may be used for spectral reconstruction.

Appendix B: Detailed description of FPGA modules

In this Appendix we provide technical details of the architectural approach to realize the FPGA controller modules outlined in the main text. Here we focus on the Walsh Timing Sequencer and the Walsh Synthersizer.

1. Walsh Generator

Following Eq. 1, target Walsh functions are generated in real time via composition of the Rademacher functions under modulo-2 addition. In our framework this is performed by the Walsh Generator module shown in Fig. 2 which stores the binary representation of a given Paley order l in a dedicated register. Each bit is routed to a lookup table where it is combined with the relevant Rademacher, together with the additional input signal from the preceeding-order Rademacher. The lookup table is comprised of an XOR gate and a one-bit multiplexer which returns In XOR R_j if the relevant bit in l is high, and returns In if the bit is low. The cascaded lookup tables in the Walsh Generator combine the Rademacher Generator output with the desired l to produce the target Walsh function. Signal propagation along this cascade forms the critical path of the design, demonstrating delays of less than one clock cycle. An example of this construction for Walsh function $\overline{\mathsf{W}}_{12}(x)$ is presented in the lower panel of Fig. 2. Walsh functions of the complement form given in Eq. 3 are similarly generated after a trivial modification.

In hardware, Rademacher functions $\mathsf{R}_j(x)$ are generated using the Rademacher Generator module shown in Fig. 2. This employs a counter to synchronize the change of state at integer multiples of the underlying clock's base period, T_c. The change-of-state is performed in real time using a comparator after the required number of clock cycles have elapsed. The output is therefore effectively a square wave with period given by "expanding" that of the underlying clock.

2. Walsh Timing Sequencer

The Walsh Timing Sequencer module shown in Fig. 3 outputs timing information in the form a Walsh function generated by an embedded Walsh Generator module (*Summary Requirement 2*). We denote this "sequencing" Walsh function by $\overline{\mathsf{W}}_s^{(1)}(x)$ where s is the user-defined Paley order, which in our implementation can be any positive 8-bit integer $s \leq 255$. The subscript (1) indicates that this Walsh function serves the dedicated purpose of synchronization to distinguish from the application of Walsh functions for waveform synthesis described in the next section.

Temporal scaling of the Walsh function is programmed by the input scaling factor t_1. In the standard Walsh basis with normalized time domain $[0, 1]$ the Walsh function of Payley order s may be divided into $2^{m(s)}$ piecewise-constant segments of equal duration, where $m(s)$ is the bit width of s. Consequently $2^{-m(s)}$ is the minimum (piecewise-constant)

segment duration. In our implementation, the time domain is rescaled such that the minimum segment duration becomes

$$\tau_1 \equiv t_1 T_c \tag{B1}$$

where t_1 is an 8-bit integer and $T_c = 10$ ns is the clock cycle time set by the FPGA hardware. The time domain of the rescaled Walsh basis is therefore given by $[0, t_1 T_c 2^{m(s)}]$ with the minimum timing resolution set by the clock cycle time T_c. Moving from continuous to discrete time, we drop the argument in our notation in Eq. 1, and the digital representation takes the form

$$\overline{\mathsf{W}}_s^{(1)} = \left[\ \vec{\boldsymbol{\sigma}}_{s1}, \vec{\boldsymbol{\sigma}}_{s2}, \dots, \vec{\boldsymbol{\sigma}}_{s,2^{m(s)}}\ \right] \tag{B2}$$

where the ith piecewise-constant segment of duration τ_1 is represented by the subsequence

$$\vec{\boldsymbol{\sigma}}_{si} \equiv \overset{\substack{1 \quad 2 \quad \dots \quad t_1}}{\left[\ \sigma_{si},\ \sigma_{si}\ \dots\ \sigma_{si}\ \right]}, \quad \sigma_{si} \in \{0, 1\}, \tag{B3}$$

constructed from t_1 repetitions of the binary digit σ_{si} occurring once per clock cycle, implemented over the time interval $(i-1, i]t_1 T_c$. The binary Walsh sequence described by Eq. B2 is generated in real time as a stream of binary data by the embedded Walsh Generator module.

We also include a Repeat Module (Fig. 3a) to extend the timing signal by sequential repetition of the basic timing sequence $\overline{\mathsf{W}}_s^{(1)}$ an integer number of times R, yielding the concatenated signal denoted $[\overline{\mathsf{W}}_s^{(1)}]_R$. In our framework R is specified by a 4-bit integer, supporting up to 15 repetitions, however this be extended as required. The number of repeats is monitored by a counter and comparator, and controlled by an enable signal which is high when the counter value is less than or equal to R and low for values beyond R. For $R = 0$, the system output is disabled.

The output of this module serves as a trigger signal to synchronize control operations with the transitioning edges (bit-flip events) in the concatenated signal $[\overline{\mathsf{W}}_s^{(1)}]_R$. This is achieved using the Edge Detect module (Fig. 3), which uses a D-type flip flop (DFF) configured to register bit-flip events in $[\overline{\mathsf{W}}_s^{(1)}]_R$ coinciding with the falling edge of the system clock.

3. Walsh synthesis for control modulation waveforms

Another critical task of the FPGA is to generate, in real time and on a designated trigger, a set of Walsh functions which are then used to form an analog waveform as a superposition via standard techniques in Walsh synthesis. Through this approach we can realize a wide variety of DES modulation protocols [42], where for efficiency we have implicitly segregated the task of carrier synthesis (resonant with the qubit transition) from baseband modulation generation.

Using the same general architecture outlined above, the Walsh Modulation Generator module (Fig. 3) addresses this task by generating the first n Paley-ordered Walsh functions

in parallel to serve as a truncated basis for Walsh synthesis. Our implementation accommodates a basis consisting of Paley orders specified by 8-bit integers, namely $n \leq 255$. This basis is scaled in time such that the minimum segment duration is given by $\tau_2 \equiv t_2 T_c$ where the input scaling factor t_2 is a 4-bit integer. The time-domain therefore spans the duration $[0, t_2 T_c 2^{m(n)}]$ where $m(n)$ is the bit width of n. All basis functions are generated simultaneously using the same Rademacher Generator and n independent instances of the Walsh Generator module, each streaming binary sequences in real time over n parallel channels. All n Walsh Generators are triggered simultaneously (after a delay of one clock cycle), using a DFF. The completion time for all n Walsh sequences is tracked by an auxiliary bit in the counter of the associated Rademacher Generator module.

The kth Walsh function output of this module is represented by the digital sequence

$$\mathsf{W}_k^{(2)} = \left[\ \vec{\boldsymbol{\sigma}}_{k1}, \vec{\boldsymbol{\sigma}}_{k2}, \dots, \vec{\boldsymbol{\sigma}}_{k,2^{m(n)}}\ \right] \tag{B4}$$

where the jth piecewise-constant segment of duration τ_2 is represented by the subsequence

$$\vec{\boldsymbol{\sigma}}_{kj} \equiv \overset{\substack{1 \quad 2 \quad \dots \quad t_2}}{\left[\ \sigma_{kj},\ \sigma_{kj}\ \dots\ \sigma_{kj}\ \right]}, \quad \sigma_{kj} \in \{0, 1\}, \tag{B5}$$

constructed from t_2 repetitions of the binary digit σ_{kj} occurring once per clock cycle, implemented over the time interval $(i-1, i]t_2 T_c$. The superscript (2) indicates that these basis functions serve the purpose of modulation-waveform synthesis.

These Walsh functions must now be combined in real time in the Walsh Filter Synthesizer with user-defined weights to produce waveforms appropriate for enacting a modulation protocol on the near-resonant carrier. This superposition is represented as

$$\Sigma_W \equiv \sum_{k=0}^{n-1} [X_k \times \mathsf{W}_k^{(2)}], \quad \mathsf{W}_k^{(2)} \in \{\pm 1\} \tag{B6}$$

where X_k is weight of the kth Walsh sequence, and the quantity Σ_W is implicitly defined to be time-dependent and piecewise constant. The bit width of these weights is 14 bits, selected for compatibility with the underlying hardware employed for this demonstration, and includes a single bit to encode the sign of X_k and the remainder for encoding the amplitude.

This superposition is executed in the Sum Weights submodule illustrated in Fig. 5, in which each weight, X_k is converted to a signed value $\pm X_k$ depending on the real-time state of $\mathsf{W}_k^{(2)}$ using a multiplexer. The output bit width is $\lceil 14 + \log_2(n) \rceil$, where n is highest Paley-order used in the Walsh synthesis protocol. These digital outputs are then added together, checked against an overflow value, and sent to other subsystems.

We customize solutions for Walsh filter synthesis for the three key modulation techniques in use: amplitude modulation (AM) of a control signal implementing a Pauli-x op-
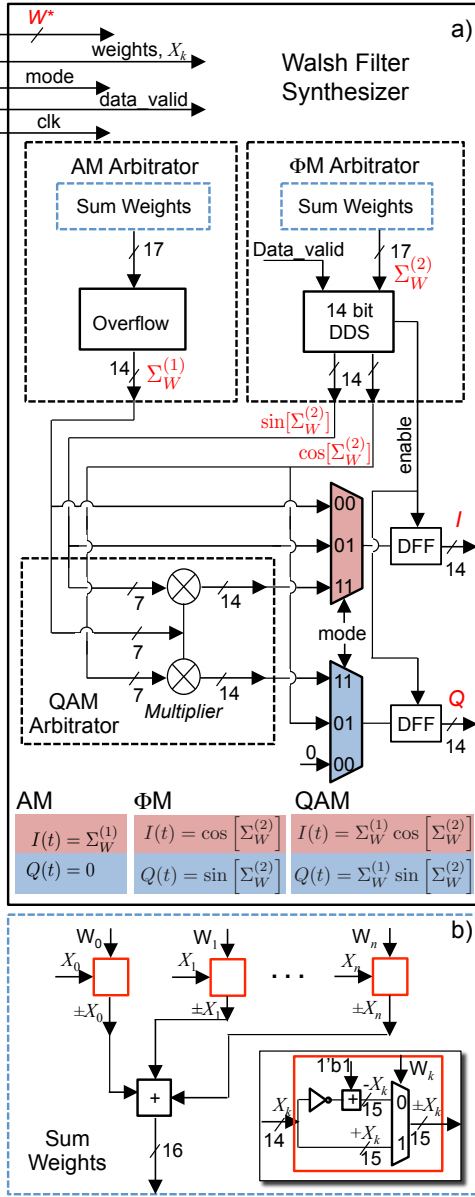
FIG. 5. Schematic of the Walsh Filter Synthesizer employed in the Walsh controller. bit widths of relevant signals are indicated. AM, ΦM, or QAM modulation are selected by the mode input, activating the relevant subunits. We distinguish the outputs of the AM and ΦM arbitrators using superscripts on $\Sigma_W^{(1,2)}$. The ΦM subsystems relies on in-built DDS blocks in the FPGA to directly rotate outputs to the appropriate basis. The QAM Arbitrator takes 14-bit inputs and uses only the seven most-significant bits prior to multiplication. This DDS is enabled by the "data valid" input generated by the Walsh Modulation Generator (Fig. 3). Relationships of outputs under various modulation protocols, indicated by the mode input, are indicated at the bottom of the panel. b) Within the AM and ΦM modules we construct the Sum Weights submodule (blue dashed box). Red boxes are highlighted in the inset and demonstrate the bitwise selection of $\pm X_k$ depending on the value of $W_k^{(2)}$.

eration for suppression of dephasing noise; phase modulation (ΦM) for suppression of amplitude damping noise; and Quadrature Amplitude Modulation (QAM) employed for simultaneous suppression of noise in both quadratures. A 2-bit signal is used to indicate the mode of modulation (Fig. 5).

In AM mode, Walsh functions corresponding to negative weights X_k are inverted. The summation of all converted weights are linked to the input of the mode selector before being registered by DFFs to feed downstream hardware.

In ΦM mode, the modulated control signal may be conveniently represented in the IQ basis through the following form:

$$m_{\Phi M}(t) = \cos\left[\Sigma_W\right]\cos(\omega t) - \sin\left[\Sigma_W\right]\sin(\omega t) \quad \text{(B7)}$$
$$= I(t)\cos(\omega t) - Q(t)\sin(\omega t) \quad \text{(B8)}$$

where Σ_W represents the required phase as a function of time, and ω is the frequency of the carrier signal. A dedicated arbitrator calculates the total phase shift Σ_W, and we then convert to the IQ basis. This process is accomplished in hardware by exploiting the trigonometric form of the in-phase and quadrature modulation coefficients, using an in-built Direct Digital Synthesis (DDS) module to function as a look-up table with 13(14)-bit input (output) (one-bit for sign, 13 for amplitude). Here, a "data valid" signal is employed as a trigger derived from the composite architecture as detailed in the following section. The DDS also outputs an "enable" signal controlling DFFs to prevent glitches due to a different clock period between the DDS and FPGA core. From trigger to output requires one clock cycle in the DDS.

The AM and ΦM arbitrators are used synchronously for QAM, with outputs multiplied within a dedicated QAM arbitrator. DFFs are added to account for a one-clock-cycle delay between the AM and ΦM arbitrator outputs, and an output data valid signal from DDS is used to enable the latching of output to suppress glitches. The use of multipliers in the QAM arbitrator adds another two-clock-cycle delay to the output when this mode is selected.

4. Clocking

To demonstrate how we may reduce latency between the output of the Timing Sequencer and the Walsh Modulation Generator our architecture employs two distinct clock signals. These are generated with minimal skew by a phase-locked loop. As indicated in Fig 6, the Timing Sequencer relies on a clock of the form designated by "clk", to drive an internal "Trigger" signal with one clock cycle delay after the start signal. The subsequent modules use clock signals of the complementary form $\overline{\text{clk}}$, phase shifted by π relative to "clk". This results in a delay of 1.5 and 2 clock cycles for Walsh Modulation Generator and Walsh Filter Synthesizer respectively. In total, latency from receiving a start signal to output is 4.5 clock cycles.

Appendix C: Walsh SID

The experimental protocol referenced in [43] is summarised as follows. At $t = 0$ the qubit sensor is prepared in its initial state $|\psi(0)\rangle \equiv |0\rangle$. A resonant $\pi/2$ pulse is applied about the $\hat{\sigma}_x$ axis, rotating the Bloch vector to the equatorial plane, coinciding an equal superposition of eigenstates, $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. During the subsequent time window $t \in [0, T]$ the qubit evolution is governed by a Hamiltonian term $\gamma b(t)\hat{\sigma}_z$ describing an interaction with a dephasing field $b(t)$ (e.g. fluctuating magnetic field), with γ an experimentally-determined coupling strength. During this time window the noisy evolution is modulated by a control field implementing a sequence of π-pulses about $\hat{\sigma}_x$. These control-modulation pulses are synchronized with the switching times of a target Walsh function $\mathsf{W}_k(t/T)$, where inclusion of the argument of the Walsh function here indicates continuous time. At the end of the acquisition time, $t = T$, a final $\pi/2$ rotation is implemented about the $-\hat{\sigma}_y$ axis, resulting in the final state $|\psi(T)\rangle$.

Evolution due strictly to the Walsh-modulated control (i.e. ignoring the interaction with the dephasing field) results in the final state $|\psi(T)\rangle = |\psi(0)\rangle = |0\rangle$. In this case final state readout (i.e. projective measurement along the initial state direction) yields unit fidelity, $P_k = |\langle\psi(T)|\psi(0)\rangle|^2 = 1$. The dephasing field interaction introduces dephasing errors due to the accumulation of a dynamical phase $\phi_k(T)$. The reduction in fidelity due to these errors results in the general condition $P_k = |\langle\psi(T)|\psi(0)\rangle|^2 \in [0, 1]$ where, for sufficiently weak noise, the characteristic value taken by the observable P_k is related to the dynamical phase by

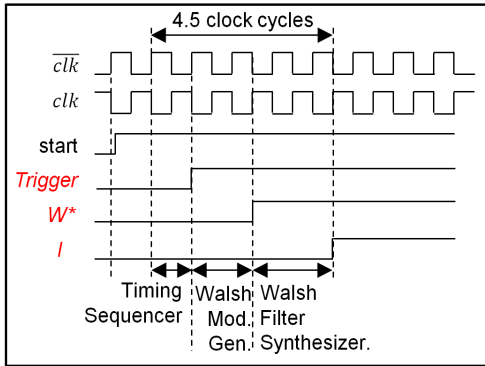$$P_k = \frac{1 + \sin[\phi_k(T)]}{2}. \qquad (C1)$$

FIG. 6. Waveforms illustrating timing and latency distribution for generating output signals via the FPGA modules. Clock signals (first two waveforms) are generated by a phase lock loop to minimise clock skew. These generated clocks take one of two forms: "clk" (used for the Timing Sequencer) or the complement form "$\overline{\text{clk}}$" (used for the Walsh Mod. Generator & Walsh Filter Synthesizer). Latency distribution of internal signals "start", "trigger", "W^*" and output signal "I" are also indicated in terms of the number of clock cycles (see Fig. 3).

Under the digital modulation imparted by $\mathsf{W}_k(t/T)$, the phase takes the form

$$\phi_k(T) = \gamma \int_0^T \mathsf{W}_k(t/T)b(t)dt = T\gamma X_k(T) \qquad (C2)$$

where we define

$$X_k(T) \equiv \frac{1}{T} \int_0^T \mathsf{W}_k(t/T)b(t)dt \equiv \langle \mathsf{W}_k(t/T), b(t)\rangle, \quad (C3)$$

expressing the coefficient of $\mathsf{W}_k(t/T)$ in a Walsh decomposition of $b(t)$ over the interval $t \in [0, T]$. Thus, from Eqs. C1, C2 & C3, the Walsh coefficients are derived from the measured probability according to

$$X_n(T) = \frac{\sin^{-1}[2P_n - 1]}{\gamma T}. \qquad (C4)$$

Note that for $k = 0$ there is no control modulation, since $\mathsf{W}_0(t/T) = 1$ does not switch its value. In this case we recover the well-known Ramsey experiment, for which the accumulated quantum phase is $\phi_0(T) = \gamma \int_0^T b(t)dt$, for Ramsey time T.

The resulting Walsh spectrum $\boldsymbol{X} = (X_0, X_1, ..., X_{N-1})$, obtained over the single experimental window $[0, T]$, may then be used to produce a time-resolved reconstruction

$$\hat{b}(t) = \sum_{k=0}^{N-1} X_k \mathsf{W}_k(t/T) \qquad (C5)$$

of the stochastic signal $b(t)$. Considering the discretization of time due to the use of a fixed maximum Walsh order (corresponding to a maximum number of parallel qubit sensors), the reconstruction in Eq. C5 is more correctly expressed in the digital representation as

$$\hat{b} = \sum_{k=0}^{N-1} [X_k \times \mathsf{W}_k^{(2)}] \qquad (C6)$$

where the Walsh sequence $\mathsf{W}_k^{(2)}$ is defined in Eq. B4, and $\hat{b}$ (argument omitted) denotes the reconstructed data stream, with consecutively streamed bits occuring once per clock cycle.

Using this prescription, we compute the fidelities P_k by numerically solving the qubit time evolution. The control protocol employed in these simulations is conveniently represented in the form of a control matrix listing all relevant control operations in sequence. As an example, returning to the digital representation used in the main text, consider the Walsh sequence $\mathsf{W}_3 = (11000011)$ expressed as a binary-valued 8-bit data stream executed over a time window $[0, T]$. The minimum segment duration (time interval between consecutively streamed bits) is therefore $\tau = T/8$, with two bit-flip events occurring at $t = 2\tau, 6\tau$. The corresponding control matrix

takes the form

$$
\Gamma_3 =
\begin{bmatrix}
\frac{\pi}{\tau_\pi} & \frac{\tau_\pi}{2} & \hat{\sigma}_x \\
0 & \tau - \frac{\tau_\pi}{2} & \hat{\mathbb{I}} \\
0 & \tau & \hat{\mathbb{I}} \\
\frac{\pi}{\tau_\pi} & \tau_\pi & \hat{\sigma}_x \\
0 & \tau - \tau_\pi & \hat{\mathbb{I}} \\
0 & \tau & \hat{\mathbb{I}} \\
0 & \tau & \hat{\mathbb{I}} \\
0 & \tau & \hat{\mathbb{I}} \\
\frac{\pi}{\tau_\pi} & \tau_\pi & \hat{\sigma}_x \\
0 & \tau - \tau_\pi & \hat{\mathbb{I}} \\
0 & \tau & \hat{\mathbb{I}} \\
\frac{\pi}{\tau_\pi} & \frac{\tau_\pi}{2} & -\hat{\sigma}_y
\end{bmatrix}
\tag{C7}
$$

Each row specifies the control variables associated with a unitary rotation of the Bloch vector. The first column specifies the Rabi rate of each sequential unitary, with the value 0 corresponding to a zero rotation (no control applied). The second column specifies the durations of each rotation. The third column specifies the Pauli operator associated with the axis or rotation in each unitary, with the identity $\hat{\mathbb{I}}$ corresponding to no control applied. The variable τ_π denotes the π-time of the system (time required to rotate the Bloch vector through an angle π). In the limit $\tau_\pi \to 0$ the protocol reduces to the regime of instantaneous qubit operations assumed in Ref. [43]. The π pulses (shown in blue) occur at times $t = 2\tau$ and $t = 6\tau$, corresponding to the bit-flip events in W_3.

This Walsh-modulated control protocol is enacted in the presence of the fluctuating dephasing field, corresponding to the stochastic Hamiltonian term $\gamma b(t)\hat{\sigma}_z$. The total Hamiltonian associated with this system is then substituted into the Schrodinger equation which is numerically integrated using a PDE solver package. The resulting qubit state evolution is thereby derived, enabling the computation of state fidelities P_k at the end of the control protocol. These fidelity values are then treated as inputs for the Walsh SID module described in the main text to demonstrate its functionality.

The Weights Estimation module employed in the Walsh SID system computes the values $X_k(T)$ by implementing Eq. 4. The measured probabilities, derived from simulation as above, are first converted to digital using a 13-bit sub-register of the 14-bit DAC. Each value P_k is therefore resolution-limited to an integer multiple (from 0 to $2^{13} - 1$) of the minimum increment $\Delta P = 1/(2^{13} - 1) \approx 1.2 \times 10^{-8}$. The 13-bit probability values are doubled using a shift register, which shifts the binary string to the left, adding a '0' to the least significant bit. Implementing this on the FPGA does not cost any additional logic or create any delay. Input parameters need to be sampled by registers in the FPGA with the use of the system clock to synchronize the whole structure. This sampling register doubles as a shift register in the actual implementation. The shift operation, however, requires at least an α-bit register to transform an $(\alpha - 1)$-bit number (hence the choice to represent the P_k in 13-bits, given a 14-bit DAC). Following the bit shift, we subtract 1 and pass the resulting 14-bit number as the input for an arcsine function on the FPGA, taking the form of a 2^{14}-element LUT embedded in the Verilog code. This operation generally consumes vast memory on the FPGA, however this method provides the highest speed solution for the arcsine function, since it consumes only one clock cycle to complete the operation. The result is a value for the numerator in Eq. 4. The output of the arcsine function is fed into the divider to adjust the amplitude due to the change of coupling strength coefficient (γ) and acquisition time frame (T). The divider will adjust the amplitude of dividend $\gamma T X_k$ with the divisor γT and can handle up to 14 bits of both dividend and divisor. The quotient is in form of 14-bit output integer. This output integer is thereafter the input data for the DAC.