Outline of the Design of a Cascaded H-bridge Medium Voltage STATCOM

R.E. Betz[†*], B.J. Cook[‡], T.J. Summers[†§], R. Fisher[‡], A. Bastiani[‡], S. Shao[‡], P. Stepien[‡], K. Willis[‡]

[†]School of Electrical Engineering and Computer Science
University of Newcastle, Australia, 2308
email:[*]Robert.Betz@newcastle.edu.au;
[§]Terry.Summers@newcastle.edu.au

[‡]ResTech Pty Ltd
University of Newcastle, Australia, 2308
email: [§]info@restech.net.au

***Abstract*—The University of Newcastle and its joint venture company, ResTech Pty Ltd, are developing a cascaded H-bridge based multilevel STATCOM. This tutorial paper outlines the salient design issues for this system. The issues covered in the paper include the choice of the converter topology, the structure of the control system hardware, the software structure and methodology, some details on the control algorithm, and the rationale behind the design decisions.**

***Index Terms*—Static Synchronous Compensator (STATCOM), Multilevel converters, SVC Static Var Compensator, Power Quality**

I. Introduction

Static Compensators (STATCOMs) are currently a topic of active research around the world [1]–[6] because they promise to be a viable and high performance replacement for the traditional thyristor based Static Var Compensator (SVCs). STATCOMs differ from SVCs in that they employ forced commutation techniques and higher switching frequencies compared to the line commutated SVC. The use of higher switching frequency and forced commutation gives the STATCOM several advantages over the SVC, namely:

- STATCOMs can respond faster to changes on the grid as they don't have to wait for 1/6th of the supply cycle as the SVC does to change their output.
- They can be used for traditional displacement power factor control, but in addition can be used as active filters to eliminate harmonics, flicker mitigation etc.
- The higher switching frequency means that their switching harmonics are easily filtered with modest sized filters between the STATCOM and the grid.
- With sophisticated control STATCOMs can rebalance unsymmetrical currents and voltages on the grid, and even provide transient dip support in the case of fault instigated grid disturbances.

In short, STATCOMs, by virtue of their higher switching frequency, are able to be controlled so that virtually any desired control objective can be achieved. SVCs are much more restricted in application because of their low effective switching frequency due to the natural commutation requirement of thyristors.

The topology and hardware of STATCOMs built to date varies depending on the voltage level that they operate at. For example, very high voltage/high power STATCOMs are usually based on variants of GTO thyristor technology connected to the grid via transformers [7], [8]. The switching speeds of the devices is relatively low, with interleaved switching techniques and intricate transformer configurations being used to help eliminate the harmonics.

The development of power switching devices, and in particular IGBTs, has meant that IGBT based STATCOMs have been increasingly used at higher power levels. For example, a large European company uses IGBT technology for HVDC applications in the hundreds of megawatts power range. In general, IGBT systems switch faster than the GTO based topologies. The IGBT based STATCOMs are usually connected to the grid via interposing transformers.

In this paper we shall be considering an IGBT based converter topology, operating at the lowest level substation distribution voltage (in Australia this is 11kV line-to-line). Multilevel converter topologies offer the possibility of eliminating the bulky and lossy transformer from the STATCOM at such voltage levels. For higher voltage levels transformers would probably be used. Under this circumstance the advantage of using the multi-level technology is related to the above-mentioned harmonic performance, and the fact that current levels in the power electronics can be maintained at relatively lower levels.

There are three fundamental topologies of multi-level converters:

- the neutral point (or diode) clamped converter (NPC);
- the flying capacitor converter (FCC);
- and the cascaded H-bridge converter (CHC).

There are more multi-level topologies than this, but they are essentially composed of various novel combinations of these basic ones.

This paper will concentrate on the development of a STATCOM based on the use of cascaded H-bridges for the following reasons:

- The number of components in a multilevel CHC topology scales linearly with the number of output levels [5].
- High output level numbers are achievable with this topology making a medium voltage direct connection (i.e. no interposing transformer) STATCOM feasible.

978-1-4244-1742-1/08/$25.00

- The topology is very modular which makes construction, maintenance and provision of redundancy simple.
- The topology is very suited to applications that do not involve real power. Even though the CHC topology was conceived for medium voltage variable speed drives, the CHC is ideal for the STATCOM application because it does not require real power to be handled. This simplifies the CHC system, since a complex input transformer is not required to supply power to the individual H-bridges in the converter. VAR compensation and active filtering only mostly imaginary power to be handled, with only small amounts of real power that can be supplied from the grid.

ResTech Pty Ltd, a joint venture company between Ampcontrol Pty Ltd and the University of Newcastle, Australia, is developing a CHC direct connect STATCOM designed for 11kV applications. A detailed block diagram of the STATCOM system is shown in Fig. 1. The two main functional blocks are identified by the enclosed dashed lines. The power circuitry block consists almost entirely of the H-bridge hardware, the circuit breakers, the initial bridge charge control, and the associated transient over voltage protection circuitry. The H-bridges are connected together in a Wye configuration to minimise the number of H-bridges required for direct grid connection (although this increases their current rating).

The control hardware section of the system is quite complex, and is based on a multi-processor architecture. A Dual Processor PC is used as the main control computer, and three 32 bit microprocessors are used for the voltage control of each phase leg. This multi-processor architecture creates a high degree of decoupling between different blocks of the control algorithm. The dual processor central computer implements the highly numerical sections of the algorithm, whose output is the desired voltage to be produced by each of the phase legs over the next control cycle. The reference voltages are sent by the dual processor central computer to each of the phase control processors, whose function is to produce these voltages by selecting which H-bridges to use taking into account the voltage balancing issues for the individual H-bridge capacitors. As far as the dual processor computer is concerned the phase legs simply produce a desired voltage, and there is no need for it to be involved in the details of how this is implemented. The algorithms for H-bridge phase balancing of the three phase legs are independent of each other, therefore there is no need for any communication between the phase legs, which simplifies the communications architecture.

A design decision was made at the outset of the project to use as much off-the-shelf electronics hardware as possible so that the development time of the prototype was reduced. The choice of an industrial PC for the central control computer and Altera EPLD NIOSii development boards for the phase controllers has turned out to be an excellent decision in that these devices have successfully fulfilled the roles intended in the control system, and achieved the desired outcome of minimising the development of custom hardware.

The remainder of this paper will look in detail at each of the sub-sections in Fig. 1, and as well present an overview of the control strategies to be implemented in the system.

II. The Power Circuit

The rationale for the choice of the CHC topology was presented in the previous section. In this particular implementation of the topology nine H-bridge modules are connected in series for each of the Wye connected phase legs. This means that each bridge supports approximately 1100 Volts DC, allowing the use of readily available 1700 Volt IGBTs as the power device, with reasonable head room still available to handle grid voltage events, the inevitable ripple on the DC link capacitor, and the voltage headroom required for control purposes. For simplicity reasons the prototype unit will use bond wire IGBT modules, but in the final system a press pack variant be considered because of the advantage that damaged press pack IGBTs tend to become a short circuit (as compared to open circuit for bond wire devices). Furthermore the final system will have redundant H-bridges modules for high reliability.

Fig. 2 shows the basic structure of a phase leg and the constitutent H-bridge modules. One of the key requirements for the H-bridges were that they were completely modular, with the only connections to the units being optical fibres. All the internal power supplies are derived from their own DC bus via internal high-to-low voltage DC-DC converters. Analogue values required for the control algorithms are sampled using on-bridge module A/D converters, and the results are transmitted serially to the controlling computers.

There is a four tiered protection system for the phase legs and associated modules. The control system implements a current controlled voltage source, and the software current limits prevent large currents from being demanded. Therefore if the control system is working correctly over-currents should not occur. The second level of protection are hardware programmed current limits in the EPLD current hardware, which if exceeded trigger a software based over-current trip. Thirdly, to back-up the software over-current limit the individual modules implement hardware instigated V_{CE}^{sat} short-circuit protection, which in turn informs the control system of a failure resulting in an orderly shutdown. If the third level of protection fails, then the fourth and final layer is hardware circuit breaker protection.

III. Control Hardware Architecture

Fig. 3 is a detailed block diagram of the control hardware for the system. As mentioned previously it is designed as a multiprocessor architecture. This architecture was chosen because of the complexity of the control task for this system. Each phase leg of the system has to be controlled so that the capacitor voltage of each individual H-bridge is such that the total phase voltage is evenly distributed between the H-bridges, and at the same time

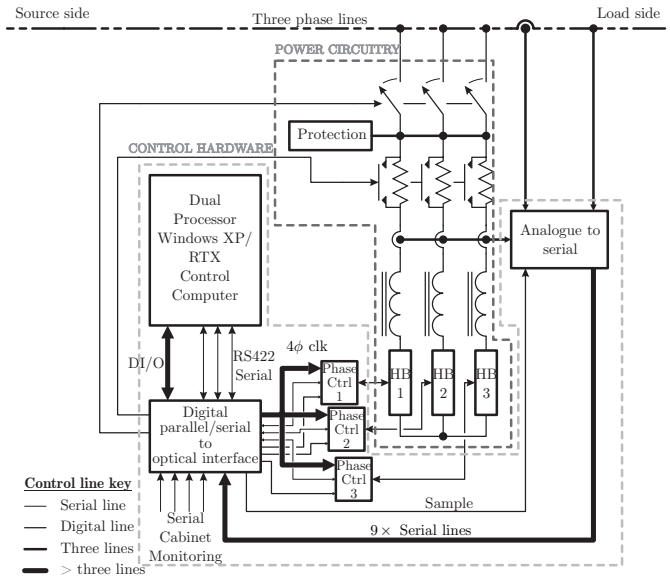

Fig. 1. Detailed block diagram of the proposed cascaded H-bridge multilevel STATCOM.

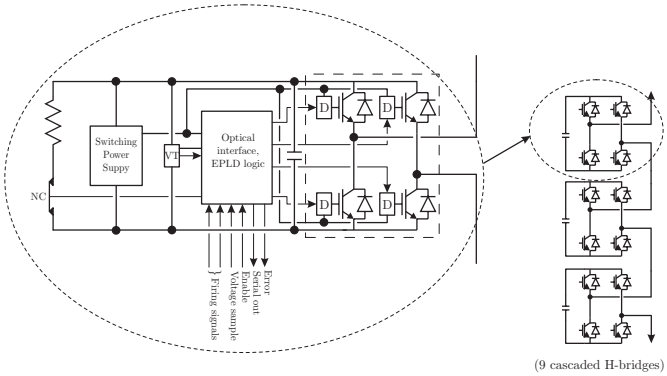

Fig. 2. Block diagram of the H-bridges used in the phase legs.

the overall desired output phase voltage is produced. The production of the desired output voltage, in this particular system, involves using PWM switching for one of the H-bridges in the H-bridge chain. Since the information to carry out these tasks only becomes available towards the end of the control cycle, considerable processing power has to be employed to achieve the this in the time allowed. This is the task of the phase controllers. As mentioned above, the calculation of the desired voltage sent to each of the phase controllers is undertaken in the central control PC, and it does not need to know the details of how this voltage is produced by the phase controller.

The distributed computing architecture requires efficient communications between the key processing units. High speed optical serial communications (3Mbits/sec) is used for sending measurements and commands between the processing units.

The phase leg processors are implemented using Altera NIOSii Development Boards which incorporate the Altera Cyclone EPLD. This approach was chosen because a full 32 bit microprocessor could be embedded into the EPLD along with the custom high speed serial communications channels required for communicating with the central PC controller and the individual H-bridges. Furthermore the use of the custom UARTs allowed non-standard length data to be efficiently sent via the serial communications (e.g. the A/D converters generate 12 bit data, and this data can be sent using a single data packet.). See Section V for information on the control algorithms implemented in the phase leg controller.

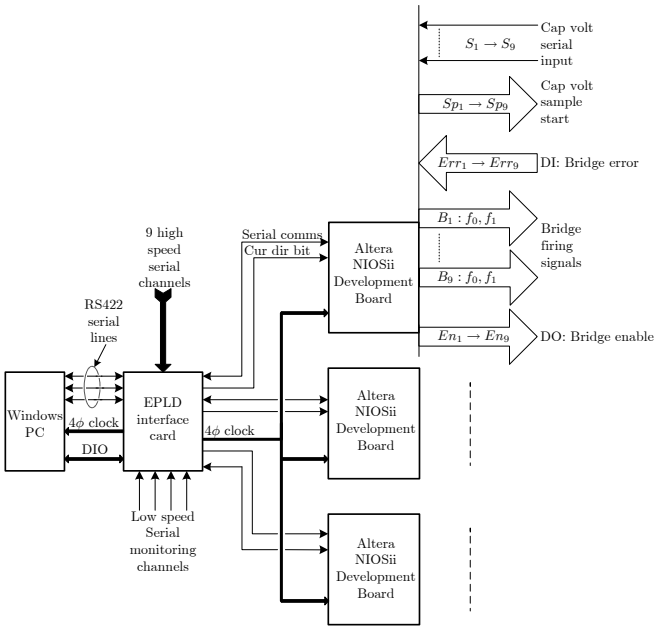

Fig. 3. Block diagram of the control hardware.

The fact that all of the communications is optical means that there is a total of 61 fibres from the phase controller, consisting of 54 fibres to the phase leg, and a further 7 fibres to the EPLD interface card to the control PC. System wide there is a total of 183 fibres involving the phase controllers. The number of fibres is this large because of a decision to use plastic fibre. This fibre has limited bandwidth, and was not fast enough to allow all of the information to be transmitted to the phase controllers and the central control PC using a single serial communications channel in the times required. Recent improvements in the bandwidth of plastic fibre my make it possible to significantly reduce the number of optical connections in the next version of the hardware.

The EPLD interface card shown in Fig. 3 has several functions. Firstly it converts the optical serial communications from the phase controllers to RS422 signals required for the high speed serial communications boards in the PC. It also generates the timing for the whole system in the form of a 4 phase clock that is distributed to all the control system components [1]. It also provides, via several custom programmed EPLDs, high speed serial channels to interface data from the voltage and current sensors in the system. As mentioned previously, custom serial receivers are used because the word length and data format is different than that normally supported by conventional UARTs.

The EPLD interface card can be interrogated by the central PC, which is the processing unit where the main control algorithms are executed. The central PC is an industrial PC with dual Pentium Xeon processors, a high speed RS422 serial card, and several other parallel digital I/O cards, as well as the normal networking interfaces available in a PC. The dual processor architecture was used so that the main control algorithm could run under a real-time operating system on one processor, and the other processor could attend to non-real-time tasks such as operator touch screen user interface, data logging, network interface, web server access to the system, and so on. It was felt that the Pentium processor would offer sufficient power to execute the control at the required rates. The use of a conventional processor, as compared to a DSP or embedded controller, allows the code to execute faster when written in 'C' because the more general purpose instruction set of these processor is able to produce more efficient code. In addition to this, the Pentium processor clock speeds are far higher than DSPs, and one is able to use floating point in all calculations without any degradation in speed. This makes writing the software simpler and easier to maintain.

Another significant advantage of using a standard Windows XP® PC for the control computer is that the software development tools are mature, reliable and easy to use.

IV. Software Architecture

The software and control hardware architectures are closely related. The phase controllers have a very specific and well specified task. Therefore the approach to the software design for this section is very simple, consisting of a small looping executive which implements a phase leg capacitor voltage balancing algorithm, as well as

[1] Equal length optical fibres are used so prevent any significant clock skew.

desired output voltage generation using PWM. The code for the phase controllers is written in 'C' to facilitate fast coding and easy maintenance.

There were several approaches available with respect to the software running on the central PC. One could again use a simple looping or interrupt driven executive, or a propriety real-time operating system, or even employ public domain operating systems such as real-time Linux. It was decided to use the proprietory Citrix/Ardence® real-time extension (RTX), which allows the execution of *true* real-time code whilst at the same time running Windows XP®. Windows XP provides powerful resources, such as networking access, a nice GUI environment, access to a disk system, and a solid and well tested software development environment.

The Ardence RTX real-time extension is not termed to be a real-time operating system, but is a Hardware Abstraction Layer/kernel extension to the standard Windows XP system that allows XP to operate as a hard real-time system. The RTX system handles all of the interrupts, and offers very fine process and thread priority control. The system comes with a set of libraries that contain routines to implement most of the normal functions of a real-time kernel – semaphores, locks, priority inheritance and disinheritance, and so on. In addition it has 'C' library emulation modules that allow real-time performance to be obtained when they are used. The RTX system is designed to integrate with the Microsoft Visual C++ .NET development environment. Code written for the system operates at Ring 0 of the processor, and therefore allows unfettered access to the hardware resources of the PC.

The Ardence RTX allows a mode of operation for multiprocessor architectures where the RTX code runs on one processor under the real-time kernel, and the non-real-time Windows code runs on the other processor under Windows XP. This approach allows a closer approximation to completely deterministic real-time performance, as compared to running the real-time software on the same processor as Windows XP. Testing of the RTX system indicated that its maximum interrupt latency is extremely low (of the order of 1μsec). Given that the control period of the STATCOM is of the order of 400μsec this is more than satisfactory.

Fig. 4 is a block diagram of the software structure for the central PC. One can see that it is designed so that all the software can operate either on a single physical dual processor machine, or if need be in client/server mode with the GUI code being executed on a separate machine connect via a local area network or via the Internet. This approach to the software architecture not only offers more options in the way the system can be physically configured, but it also decouples one section of the system from another making software development less complex.

In order to speed up the development process it was decided to write the non-real-time parts of the system in the interpretative high level language Python. This language has excellent support for multi-threaded programming, a standard GUI library based on wxWidgets, and excellent support for networking. In the applications where it is being used its interpretative nature does not cause execution speed problems. Furthermore it is reported to generate code of approximately 1/5th of the number of lines of other languages such as Java or C++ for the same functionality, and therefore allows rapid development.

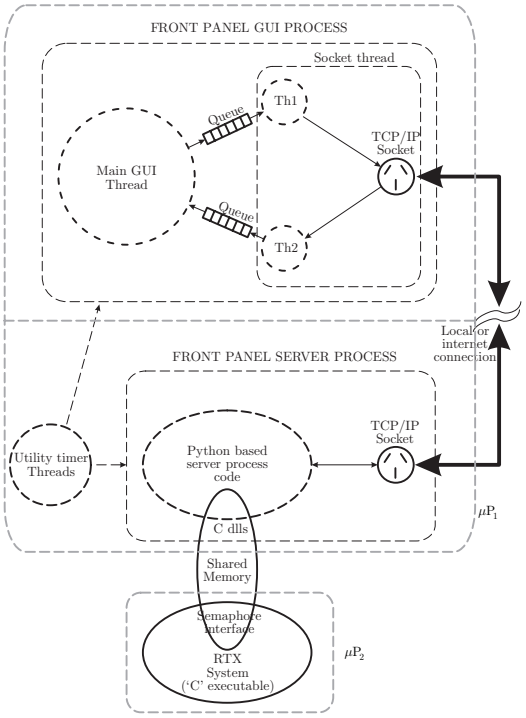

Fig. 4. Block diagram of the central computer software structure.

The real-time code, as mentioned previously, is executed on a separate *processor*. However it also needs to communicate with the non-real-time Windows side of the system. This is implemented using a shared memory to pass data structures between the Python code and the real-time code. Particular attention was paid to ensure that under no circumstances will the real-time code be held up by this shared memory interaction. The synchronization between the two systems is implemented using RTX semaphore primitives.

Internally the Python processes have been implemented with multiple threads of execution using thread safe queue based message passing to prevent any critical section issues. There are no common shared data structures between the Python threads – the only interaction between threads is via message passing. This also means that the software threads can be developed and debugged with a large degree of independence. This approach also makes the addition of new features simple.

Fig. 5 is a conceptual diagram of the software for the front panel server process. Each circle is a separate thread in the server, and the arrowed lines indicate the

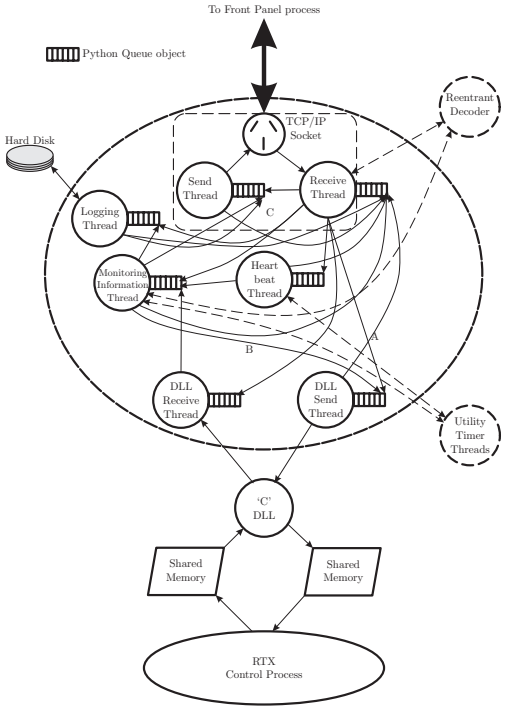

Fig. 5. Block diagram of the front panel server software.

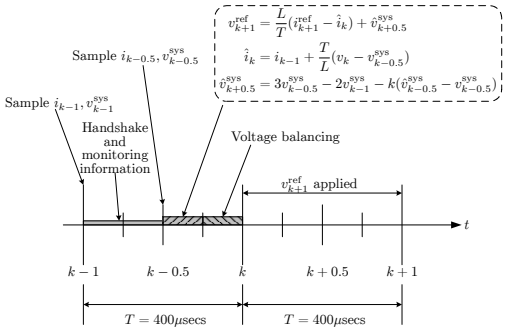

Fig. 6. Control timing for the central control computer.

various communications paths throughout the thrreads. As mentioned above the communications uses thread safe Queue objects which are natively supported in Python. All messaging passing has been organised using a "mailbox" and address approach, so that communications can be simply established between the various threads using the thread address that the message has to be sent to. The message format also contains a return address so that acknowledge protocols can be naturally handled. The receive thread uses a re-entrant table driven decoder to decode the address associated with a received message so that it can be sent from the front panel GUI process to the appropriate thread for action. The same decoder is also used b all the threads in the front panel server process.

The data sent via the Queue objects is serialised Python objects (encoded in ASCII). This allows simple communications of complex data structures between the server threads and between the server and front panel processes. All of the internal communications between the threads are subject to timeouts. A "heart beat" thread generates regular messages between the front panel server process and the front panel GUI process so that both processes are sure that the TCP/IP communications is working correctly and that the other process in the system is working. The user interface upon receipt of the heart beat message rotates an object on the screen so that the user has positive confirmation that the communications between the two front panel server and GUI processes is continuing to function. If a timeout occurs on the heart beat, or if a timeout is violated between thread interactions in any of the threads of the system, the STATCOM is safely shutdown and a diagnostic message is passed via the TCP/IP channel (if available) to the front panel GUI process.

The optical serial communications channels also have error checking to make sure that data received is valid. There is a comprehensive error and warning system built into the software so that the errors are dealt with in a consistent way, and the system response is tailored to the nature of the error. There is also a comprehensive error logging system to complement this so that error data can be analysed for diagnostic purposes.

The Python front panel server process also provides an interface to the real-time code. This code is written in 'C' and physically runs on another separate process in the dual processor control computer. Windows XP cannot see the processor used for this process (this is hidden from Windows XP by the Ardence RTX), and therefore has no direct control over it. The communications between the processors is implemented using semaphore protected shared memory that is visible to both processors.

Python can interface directly to 'C' dynamic link libraries via several different techniques, and this allows data structures to be passed to and from the real-time system. It is intended in the final system that these data structures will be passed using XML so that precise alignment of the Python data structure with a 'C' data structure is not required (as would be the case if 'C' type structures are passed between the two systems). These shared memory interactions are implemented in such a way that the real-time system can never be blocked on the semaphore, ensuring that its real-time deadlines are always met.

Figure 6 shows the basic timing of the system. Samples of the currents and voltages of the system are taken precisely at the beginning and middle of the control cycles. There is a delay as the samples are serially transmitted to the control computer. The control cycle time is $T = 400\mu$sec, which is divided into four phases, each of 100μsec duration. The real-time code is written in 'C' and implemented as an event driven routine. It is suspended pending an interrupt from one of the four

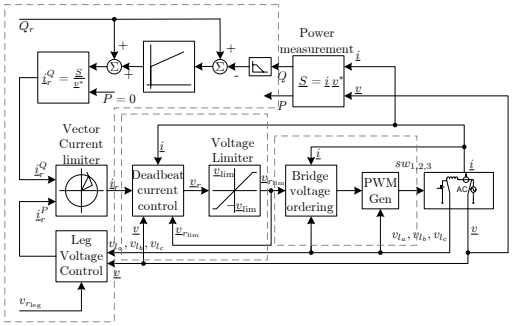

Fig. 7. Block diagram of the STATCOM control system.

phase clock edges. Upon receipt of an interrupt a port is read to check the control phase that the system is in and the appropriate code is executed based on this. The first two phases are concerned with receiving the sampled currents from the current transducers, as well as receiving monitoring information from the phase controllers. The actual current control algorithm is executed in the third phase of the control period (since it has to wait for the mid control interval voltage sample to compute the control). At the end of the third control period, the desired voltages for the phase legs are transmitted to the phase control processors. Therefore the phase control processors have a little less than 100μsec to execute the voltage balancing and PWM algorithms so that the firing times can be latched into the hardware timers that fire the power devices before the end of the control period.

V. Control Algorithm

The control algorithm for the STATCOM is hierarchical in nature, following the structure of the hardware and software that was designed to execute it. A block diagram of the control algorithm is shown in Fig. 7. It should be noted that the entire control algorithm operates in a stationary reference frame.

The inner part of the control algorithm is associated with the control of the individual bridges and phase legs. At the lowest level this involves the control of the H-bridge capacitor voltages. These are controlled at the control rate – i.e. at 2.5kHz. In each control interval the current direction is used in conjunction with the state of charge of the H-bridge capacitors and the desired output voltage, to determine which H-bridges will be used to generate the desired output. This approach gives very accurate balancing of the capacitor voltages, and minimises the ripple current stress on the DC link capacitors. A converter model is employed to remove the effects of the diode and bulk resistance drops of the power devices. The output voltage is made more accurate by employing symmetrical PWM on one of the H-bridges being used to form the output voltage.

The next level in the control algorithm is the determination of desired voltage for the H-bridge legs. The approach taken in this STATCOM is that it operates as a voltage controlled current source. This has the advantage that the control provides current limiting under fault conditions, and it that only instantaneous values of voltages and currents are required for the control computations, and does not require pre-computation of switching angles. The current control is a dead-beat (sometimes called predictive) current controller which is more commonly used in variable speed drive (VSD) applications [9], [10]. These algorithms have very fast transient response and are computationally very simple. In this implementation of the algorithm the "back-emf" does not have to be estimated (as is the case in the VSDs), since it is the grid system voltage and can be measured. There is however an issue that the grid voltage must be estimated one control interval ahead of where the control is evaluated. This can be seen in the control equations (see Fig. 6) where the $\hat{v}^{\text{sys}}_{k+0.5}$ value is used in the development of the reference voltage v^{ref}_{k+1}. If the $\hat{v}^{\text{sys}}_{k+0.5}$ value is not known with reasonable accuracy then the current will have significant errors. A special digital phase locked loop is used to generate these estimates, and it has the added advantage that noise on the grid supply voltage is minimised. An additional advantage of this algorithm is that precise knowledge of parameters such as the connection inductance are not required, and the control is dependent only on instantaneous samples and does not rely on precise knowledge of grid system phase. Futhermore, because of the 2.5kHz control rate the harmonics produced are small and the filtering requirements are minimial.

The outer control loops of the system look after the power control. This can be divided into the real power control, which is very important as the real power control manages the overall voltages on the individual phase legs. Because of issues such as lack of symmetry of the grid supply, and the inevitable variations in the values of the bridge components, the real power calculations are carried out on a per phase leg basis. The net result a desired two phase dq real power current that forms one component of the two phase current reference for the current controller.

The other current component is associated with the desired imaginary power that the overall control strategy desires. Using the instantaneous imaginary power expression (from PQ theory) a reference imaginary power current is generated. The combined current references are passed to a vector current limiter, which ensures that the desired current does not exceed the STATCOM current limit. This limit is carried out in such a way that the real component of the current is preserved in both magnitude and phase (if possible), but always with respect to phase. The imaginary power current component is limited before the real power current, since real power control is essential for the control of the H-bridge leg voltages. The limited value is then passed to the dead-beat current control algorithm to generate the reference output voltages.

VI. Current status

At the time of writing this paper a low voltage prototype of the above-mentioned system has been constructed and is currently being tested. The power hardware used for this low voltage prototype is based on MOSFETs. The unit is rated to operate from a 415VAC line-to-line

Fig. 8. Photograph of the low voltage prototype STATCOM.

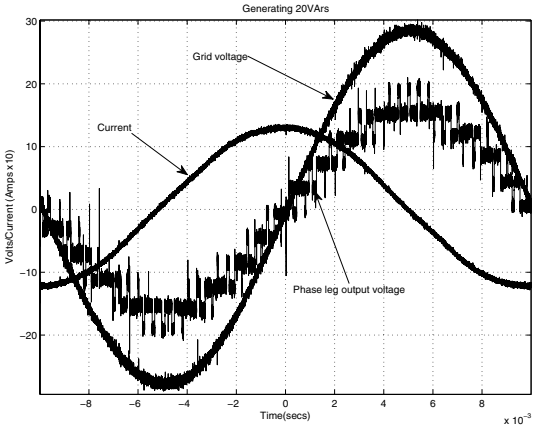

Fig. 9. Experimental result showing phase output waveforms for the low voltage prototype system.

supply. A basic VAr compensation control algorithm has been implemented on this prototype. The control system hardware is the identical hardware for the 11kV prototype system – i.e. as far as the control hardware and algorithms are concerned they are sending communications to the 11kV H-bridges. Fig. 8 is a photograph of the low voltage prototype system. The cubicle on the left contains the phase leg processors, the MOSFET based H-bridge hardware, and the central control computer. The three phase legs and their associate phase leg control boxes are clearly visible, and the box at the top of the cubicle is the control computer. The cubicle on the right contains the three phase load, a California Instruments® programmable three phase power supply, and associated contactors and circuit breakers.

Fig. 9 shows a preliminary oscilloscope trace of the output waveforms for one phase of the prototype unit. One can clearly see the multi-level phase voltage output, and the current which is 90° out of phase with this voltage. The phase leg output waveform is in phase with the system (grid) voltage, but of less magnitude. Therefore as far as the grid is concerned the STATCOM looks like an inductor in this case. At the time of writing this paper further testing is being carried out, and the control strategies are being refined as a result of this testing. More comprehensive results will be presented at the conference.

VII. Contributions

The main contributions of this paper are:

- presentation of the overall structure and design of a 19 level cascaded H-bridge STATCOM.
- discussion of the rationale behind the design decisions for the hardware, software and control algorithms.
- a more detailed description of the key functional hardware and software components of the system.
- presentation of some very preliminary output results from the prototype STATCOM.

Acknowledgment

The authors would like to acknowledge the financial support of Australian Research Council Linkage Program for this project.

References

[1] B. Blazic and I. Papic, "Improved d-statcom control for operation with unbalanced currents and voltages," *IEEE Transactions on Power Delivery*, vol. 21, no. 1, pp. 225–233, Jan. 2006.

[2] Z. Du, L. M. Tolbert, J. N. Chiasson, and B. Ozpineci, "A cascade multilevel inverter using a single DC source," in *Applied Power Electronics Conference and Exposition, 2006. APEC '06. Twenty-First Annual IEEE*, Mar. 19–23, 2006.

[3] H. Masdi, N. Mariun, S. M. Bashi, A. Mohamed, and S. Yusuf, "Design of a prototype d-statcom using DSP controller for voltage sag mitigation," in *Power Electronics and Drives Systems, 2005. PEDS 2005. International Conference on*, vol. 1, Jan. 2006, pp. 569–574.

[4] C. Hochgraf and R. H. Lasseter, "Statcom controls for operation with unbalanced voltages," *IEEE Transactions on Power Delivery*, vol. 13, no. 2, pp. 538–544, Apr. 1998.

[5] J. Rodriguez, J. S. Lai, and F. Z. Peng, "Multilevel inverters: A survey of topologies, controls, and applications," *IEEE Transactions on Industrial Electronics*, vol. 49, no. 4, pp. 724–738, August 2002.

[6] C. A. C. Cavaliere, E. H. Watanabe, and M. Aredes, "Multipulse STATCOM operation under unbalanced voltages," in *Power Engineering Society Winter Meeting, 2002. IEEE*, vol. 1, Jan. 2002, pp. 567–572.

[7] C. Schauder, M. Gernhardt, E. Stacey, T. Lemak, L. Gyugyi, T. W. Cease, and A. Edris, "Operation of ±100 MVAr TVA STATCON," *IEEE Trans. Power Del.*, vol. 12, no. 4, pp. 1805–1811, Oct. 1997.

[8] C. Schauder, "STATCOM for compensation of large electric arc furnace installations," in *Power Engineering Society Summer Meeting, 1999. IEEE*, vol. 2, Edmonton, Alta., Jul. 1999, pp. 1109–1112.

[9] R. Betz, B. Cook, and S. Henriksen, "A digital current controller for three phase voltage source inverters," *Conference Record of the IEEE-IAS Annual Meeting*, pp. 722–729, New Orleans, Oct. 1997.

[10] R. E. Betz, G. Mirzaeva, and D. Pulle, "Frame alignment stability issues in natural field orientation," School of Electrical Engineering and Computer Science, University of Newcastle, Australia, Tech. Rep. V1.84, May 2007. [Online]. Available: http://eecsbobb.newcastle.edu.au/rebetz/Reports/NFO_stability.pdf