DESAIN *HIGH FREQUENCY* PWM MENGGUNAKAN CPLD DAN PEMANFAATAN SISTEM SEBAGAI KONTROL PADA DC-DC *FLYBACK UP CONVERTER*

SKRIPSI

Sebagai salah satu syarat untuk memperoleh gelar
Sarjana Sains dalam bidang Fisika

oleh :
SIGIT KURNIAWAN
0810930056-93

JURUSAN FISIKA
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
UNIVERSITAS BRAWIJAYA
MALANG
2014

LEMBAR PENGESAHAN SKRIPSI

DESAIN *HIGH FREQUENCY* PWM MENGGUNAKAN CPLD DAN PEMANFAATAN SISTEM SEBAGAI KONTROL PADA DC-DC *FLYBACK UP CONVERTER*

oleh :
SIGIT KURNIAWAN
0810930056-93

Telah dipertahankan di depan Majelis Penguji
pada tanggal ……..……..…
dan dinyatakan memenuhi syarat untuk memperoleh gelar Sarjana Sains dalam bidang Fisika

Pembimbing I	Pembimbing II
Dr.Ing. Setyawan P Sakti, M.Eng	**Hari A Dharmawan, M.Eng.,Ph.D**
NIP. 19650825 199002 1 001	**NIP. 19690920 199410 01**

Mengetahui,
Ketua Jurusan Fisika
Fakultas MIPA Universitas Brawijaya

Adi Susilo, Ph.D
NIP. 19631227 199103 1 002

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini :

Nama	**: Sigit Kurniawan**
NIM	**: 0810930056**
Jurusan	**: Fisika**
Penulis Skripsi berjudul	**:**

DESAIN *HIGH FREQUENCY* PWM MENGGUNAKAN CPLD DAN PEMANFAATAN SISTEM SEBAGAI KONTROL PADA DC-DC *FLYBACK UP CONVERTER*

Dengan ini menyatakan bahwa :

1. **Isi dari skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama termaktub di isi dan tertulis di daftar pustaka dalam Skripsi ini.**
2. **Apabila dikemudian hari ternyata skripsi saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

Demikian pernyataan ini dibuat dengan segala kesadaran.

Malang, …….
Yang menyatakan,

Sigit Kurniawan
NIM. 0810930056

DESAIN *HIGH FREQUENCY* PWM MENGGUNAKAN CPLD DAN PEMANFAATAN SISTEM SEBAGAI KONTROL PADA DC-DC *FLYBACK UP CONVERTER*

ABSTRAK

Pulse Width Modulation (PWM) menjadi bagian yang tak terpisahkan dari kebanyakan sistem kontrol. Salah satu kegunaan dari PWM adalah sebagai pengontrol daya pada rangkaian konverter DC-DC/DC-AC. Penggunaan PWM dengan frekuensi tinggi selalu menjadi alternatif untuk mengurangi besarnya ukuran transfomator pada konverter *flyback*. Frekuensi tinggi PWM dapat dihasilkan jika mengunakan model pencacah dengan kemampuan menghitung cepat. Teknik pencacahan *rising* dan *falling* untuk model pencacah yang dirancang mampu menghasilkan PWM dengan frekuensi lebih tinggi daripada teknik PWM yang dibangun pada mikrokontroller PIC16F87X. Besarnya frekuensi PWM yang dibangun pada komponen CPLD bergantung pada resolusi dan osilator kristal yang digunakan, nilai frekuensi yang mampu dihasilkan untuk resolusi PWM 10 bit dan frekuensi osilator kristal 48 MHz pada arsitektur PWM yang telah dirancang adalah 93.5 kHz. Desain *high frequency* PWM pada komponen CPLD dapat digunakan sebagai kontrol pada konverter *Flyback* untuk menaikkan tegangan DC dari 12V ke 400V.

Kata kunci: *Pulse Width Modulation*, Konverter *Flyback*, CPLD, Teknik pencacahan *Rising* dan *Falling*

DESIGN OF HIGH FREQUENCY PWM USING CPLD AND IMPLEMENTATION SYSTEM FOR DC-DC FLYBACK UP CONVERTER CONTROL

ABSTRACT

Pulse Width Modulation (PWM) becomes the important part for most control systems. One of many functions of PWM is as power control for DC-DC/DC-AC converter circuit. The use of high frequency PWM always be alternative for decreasing size of transfomer in Flyback converter. High frequency PWM can be produced by using counter model with fast counting capability. The rising and falling counter that has been designed can produce PWM frequency that is higher than frequency that has been developed with PWM technique using microcontroller PIC16F87X. The result of PWM frequency using CPLD depends on resolution of N-bit counter and frequency of crystal oscilator that used. Frequency value that can be produced for 10 bit resolution of counter and 48 MHz frequency of crystal oscilator in PWM architecture which has been designed is 93.5 kHz. High frequency PWM using CPLD can be used as control on Flyback converter for step up DC voltage from 12V to 400V.

Keyword: Pulse Width Modulation, Flyback converter, CPLD, The Rising and Falling counting technique

KATA PENGANTAR

Alhamdulillahirabbil alamin, dengan ungkapan rasa syukur pada Allah, sehingga penulis dapat menyelesaikan penyusunan skripsi yang merupakan salah satu syarat untuk memperoleh gelar sarjana sains dalam bidang fisika. Pada kesempatan ini penulis menghaturkan terimakasih kepada:

a. Kedua orang tua, serta keluarga yang selalu memberi dukungan kepada penulis.
b. Dr.Ing. Setyawan P Sakti, M.Eng. dan Hari A Dharmawan, M.Eng.,Ph.D. selaku pembimbing skripsi atas segala bimbingan selama skripsi.
c. Dr. Sunaryo, M.Si. selaku dosen pembimbing akademik, atas semua nasihat dan bimbingan Bapak selama ini
d. Sdr. Hari Murti Widodo. serta rekan-rekan di Laboratorium Instrumentasi dan Pengukuran Jurusan Fisika.
e. Kepada teman-teman Fisika angkatan 2008.
f. Seluruh *civitas academica* Jurusan Fisika atas bantuan dan kerjasamanya.

Semoga Allah membalas kebaikan dan pertolongan yang telah diberikan.

Penulisan skripsi ini adalah upaya optimal dari penulis, namun masih banyak kekurangan yang harus diperbaiki. Oleh karenanya, penulis mengharapkan kritik dan saran yang membangun dari pembaca. Semoga skripsi ini bermanfaat, terutama bagi mahasiswa Fisika, dan masyarakat pada umumnya.

Malang, Desember 2013

Penulis

DAFTAR ISI

DAFTAR GAMBAR

Halaman

DAFTAR TABEL

Halaman

DAFTAR GRAFIK

Halaman

DAFTAR LAMPIRAN

Halaman

Halaman ini sengaja dikosongkan

BAB I
PENDAHULUAN

1.1 Latar Belakang

Tegangan tinggi DC senantiasa dimanfaatkan untuk menghasilkan medan listrik statis, dimana medan ini akan menghasilkan gaya listrik statis yang dapat digunakan untuk menarik partikel-partikel bermuatan. Salah satu instrumen yang banyak dimanfaatkan sebagai pembangkit tegangan tinggi adalah konverter *Flyback*.

Tegangan keluaran dari konverter *Flyback* dapat dikontrol oleh sinyal PWM, jika tegangan keluaran dari konverter *Flyback* diregulasi berdasarkan nilai *duty cycle* PWM maka hubungan input dan output dari konverter *Flyback* ditentukan oleh nilai *duty cycle* PWM dan rasio lilitan pada transfomer *Flyback*. Jika inti dari trafo menggunakan bahan ferit maka bahan material ini mampu bekerja pada frekuensi hingga ratusan kHz dengan sedikit dispiasi daya yang terjadi (Kourtrolis, dkk., 2005).

Ukuran dari transformer selalu menjadi pertimbangan dalam setiap perancangan konverter *Flyback*. Jika nilai induktansi dari masing-masing bagian transformer sebanding dengan ukuran induktornya maka semakin besar frekuensi PWM yang digunakan, ukuran dari transformer *Flyback* akan semakin kecil. Ini berguna untuk setiap perancangan konverter daya yang *portable*.

Pulse Width Modulation (PWM) dapat dibangkitkan melalui teknik analog maupun teknik digital. Dalam teknik analog PWM dibangkitkan melalui sinyal pembawa dan sinyal modulasi, sedangkan dalam teknik digital PWM dapat dibangkitkan dengan teknik *capture* dan *compare*. Teknik digital mempunyai beberapa keunggulan jika dibandingkan dengan teknik analog antara lain tingkat *noise* yang rendah, tidak terpengaruh oleh kenaikan temperatur ataupun kenaikan tegangan, serta tidak terjadi perubahan berkaitan dengan variasi komponen yang digunakan (Rahim N.A. and Islam Z., 2009).

Salah satu komponen digital yang dapat digunakan untuk membangitkan PWM adalah mikrokontroller. Kemampuan mikrokontroller dalam menghasilkan PWM dengan frekuensi tertentu terbatas pada N-bit pencacah dan nilai osilator yang digunakan, semakin tinggi nilai resolusi dari PWM maka frekuensi PWM akan berkurang. Selain itu, penggunaan kristal osilator sebagai pewaktu (*clock*) pada mikrokontroller juga dibatasi pada nilai frekuensi dibawah 20 MHz. Oleh karenanya diperlukan komponen lain yang mempunyai kemampuan untuk menghasilkan PWM dengan jangkauan frekuensi yang lebih tinggi sehingga dapat dimanfaatkan sebagai pengatur tegangan keluaran dari rangkaian konverter *Flyback*.

Dalam penelitian ini ditawarkan bagaimana merancang *high frequency* PWM melalui komponen *Programable Logic Devices* (PLD) jenis CPLD dengan jangkauan frekuensi hingga 93 kHz (untuk nilai osilator 48 MHz), serta resolusi 10 bit. Aplikasi yang ditawarkan dalam penelitian ini berupa penggunaan komponen CPLD sebagai pengatur tegangan keluaran pada konverter DC-DC *Flyback* dimana konverter ini digunakan untuk mengubah tegangan DC dari 12V menjadi 400V tegangan DC.

1.2 Rumusan Masalah

Permasalahan dalam penelitian ini dapat dirumuskan sebagai berikut :

1. Bagaimana merancang pembangkit PWM dengan frekuensi tinggi menggunakan komponen *Complex Programable Logic Device* (CPLD)?
2. Bagaimana perbandingan nilai frekuensi PWM yang dihasilkan antara mikrokontroller dengan komponen *Complex Programable Logic Device* (CPLD)?
3. Bagaimana pemanfaatan PWM dari komponen CPLD sebagai pengantur nilai tegangan keluaran pada rangkaian konverter *Flyback*?

1.3 Batasan Masalah

Dalam perencanaan dan pembuatan alat pada penelitian ini diperlukan pembatasan masalah, pembatasan masalah dalam skipsi ini adalah:

1. Pembangkit PWM dirancang pada komponen CPLD jenis *Complex Programable Logic Device* (CPLD) jenis XC9572.
2. Sistem kontrol dari Konverter *Flyback* dibangun dengan model *open loop* (sistem kontrol terbuka).
3. Tegangan keluaran dari konverter *Flyback* diatur berdasarkan lebar *duty cycle* dari PWM.
4. Jenis mikrokontroller yang digunakan pada penelitian ini adalah PIC 16F877.

1.4 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penelitian ini adalah :

1. Memanfaatkan komponen CPLD sebagai pembangkit PWM dengan frekuensi tinggi.
2. Mengetahui keunggulan dari CPLD dibandingkan mikrontroller sebagai pembangkit PWM.
3. Pemanfaatan PWM sebagai pengatur tegangan keluaran pada rangkaian konverter *Flyback.*

1.5 Manfaat Penelitian

Melalui penelitian ini diharapkan dapat dijadikan acuan untuk merancang pembangkit PWM melalui komponen *Programable Logic Device* (PLD) seperti komponen CPLD atau FPGA. Selain itu melalui penelitian juga diharapkan pemanfaatan komponen tersebut untuk berbagai perancangan sistem digital dapat dikembangkan lebih lanjut.

(halaman ini sengaja dikosongkan)

BAB II
TINJAUAN PUSTAKA

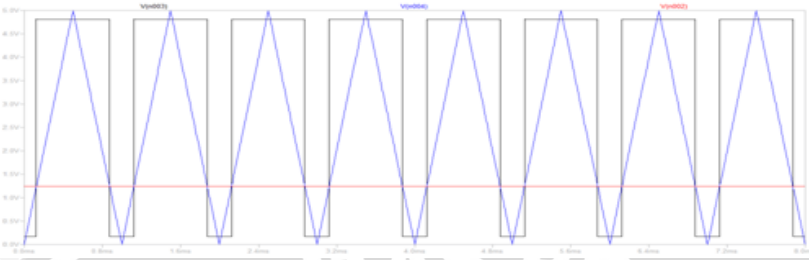
2.1 Teknik PWM

2.1.1 Teknik analog

Pulse WitdhModulation (PWM) dapat dibangkitkan dengan teknik analog dan teknik digital. Pada teknik analog metode yang dipakai menggunakan konsep *carrier modulation*, dimana pada teknik modulasi ini terdapat satu gelombang pembawa berupa gelombang segitiga dengan frekuensi tinggi dan sinyal modulasi dapat berupa sinyal sinus, sinyal DC dan sebagainya, bergantung dari tipe modulasi yang digunakan. Kemudian dua sinyal dibandingkan dengan suatu komparator untuk mendapatkan sinyal keluaran berupa bentuk gelombang PWM. Terdapat empat dasar metode modulasi secara analog yaitu: *(a) Single Pulse Modulation (b)Sinusoidal Pulse Width Modulation (c) Modified Sinusoidal PWM (d) Multiple PWM.* Tetapi yang lebih umum digunakan adalah metode *Sinusoidal Pulse Width Modulation* dan metode *Mutilple* PWM.

2.1.1.1 *Multiple Pulse Width Modulation*

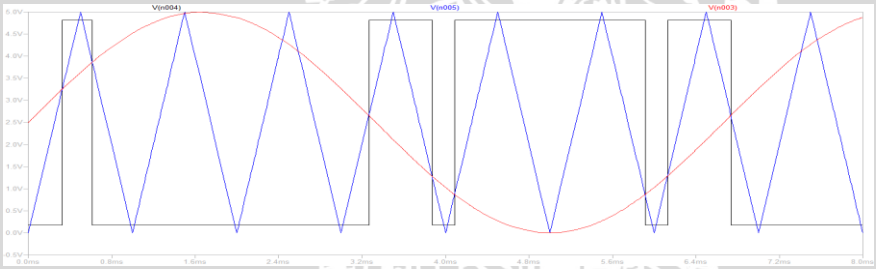
Pada teknik modulasi ini suatu sinyal DC dibandingkan dengan gelombang segitiga untuk mendapatkan hasil berupa PWM (Gambar 2.1). Modulasi ini memberikan kelipatan pulsa untuk mereduksi gejala harmonik. Nilai RMS dari tegangan keluaran dapat diatur oleh variasi lebar pulsa. Perbandingan frekuensi sinyal dari gelombang segitiga (f_c) dan frekuensi dari gelombang output (f_o) disebut sebagai perbandingan frekuensi modulasi sebagai $m_f = f_c/f_o$. Lebar pulsa dapat diubah dengan variasi amplitudo dari tegangan DC dan jumlah pulsa per setengah siklus sebagai $P = m_f/2$.



Gambar 2.1. Hasil simulasi dari LTSpice untuk metode *Multiple* PWM

2.1.1.2 *Sinusoidal Pulse Width Modulation*

Pada teknik modulasi ini sinyal sinus dibandingan dengan gelombang segititiga untuk menghasilkan sinyal PWM (Gambar 2.2). Lebar tiap pulsa bergantung pada amplitudo gelombang segitiga. Nilai RMS dari tegangan keluaran dapat diatur oleh variasi lebar pulsa.



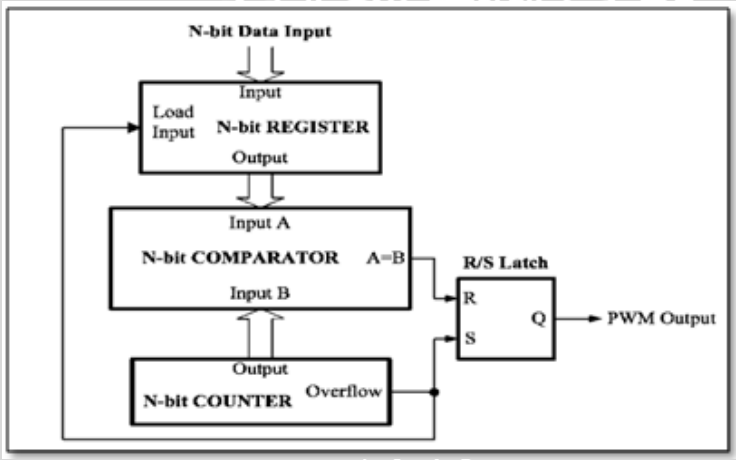
Gambar 2.2. Hasil simulasi dari LTSpice untuk metode *Sinusoidal* PWM

Perbandingan frekuensi sinyal dari gelombang segitiga (f_c) dan frekuensi dari gelombang output (f_c) disebut sebagai perbandingan frekuensi modulasi sebagai $m_f = f_c/f_o$. Perbandingan sinyal amplitudo gelombang segitiga (P_c) dan sinyal amplitudo gelombang sinus (P_r) disebut sebagai indeks modulasi $m_f = P_c/P_r$. Teknik ini mempunyai jangkauan luas untuk digunakan sebagai kontrol tegangan pada inverter (Rahim N.A. dan Islam Z, 2009)

2.1.2 Teknik digital

Teknik digital yang digunakan sebagai pembangkit PWM umumnya dikembangkan dari komponen pencacah dan komparator. Gambar 2.3 merupakan arsitektur yang ditawarkan oleh Koutroulis, dkk (2005).

Input sistem berupa N bit data yang bersesuaian dengan nilai *duty cycle* PWM, dimana dapat dihubungkan dengan komponen lain misalnya mikrokontroller. Nilai dari N bit *register* kemudian dibandingkan dengan data hasil perhitungan suatu pencacah oleh sebuah komparator. Ketika kedua nilai tersebut sama, maka komparator akan me-*reset* R/S flip-flop sehingga dihasilkan pulsa dengan lebar sama dengan N-bit data input. Kemudian R/S flip-flop dalam keadaan *set* ketika pencacah mencapai kondisi *overflow* di akhir periode PWM, keadaan ini dapat dimanfaatkan untuk memasukan nilai N bit data pada *register*. Sehingga *duty cycle* PWM dapat ditingkatkan begitu juga untuk nilai frekuensinya.



Gambar 2.3. Blok diagram dari teknik digital PWM

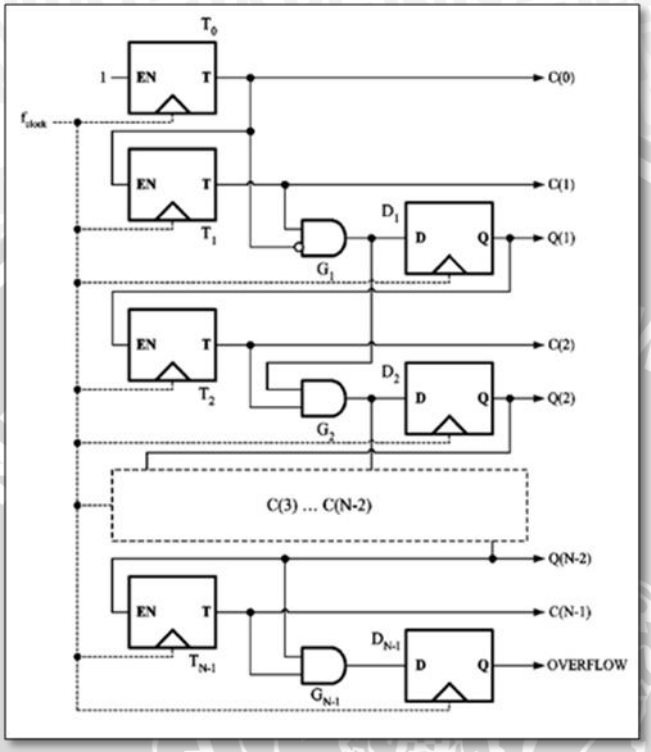
Frekuensi PWM ditentukan oleh frekuensi osilator dan N bit pencacah, sebagaimana diberikan pada persamaan 2.1.

$$f_{pwm} = f_{clock}/2^N \qquad (2.1)$$

Resolusi dari PWM bergantung pada N bit dari pencacah yang digunakan (Persamaan 2.2). Jika digunakan 8 bit pencacah maka nilai resolusinya berkisar $\alpha = \frac{100\%}{256} = 0{,}39\%$, semakin tinggi nilai bit pencacah maka resolusinya semakin besar.

$$\alpha = \frac{100\%}{2^N} \qquad (2.2)$$

Frekuensi tinggi dari PWM dapat dihasilkan jika digunakan mode pencacah dengan kemampuan menghitung cepat. Pencacah dengan mode sinkron mempunyai karakteriktik secara implisit terhadap *state decoding bounds* sesuai dengan maksimum frekuensi pewaktu yang diizinkan. Untuk mengurangi hal tersebut maka pencacah dibuat melingkar seperti yang ditunjukan pada Gambar 2.4, dimana terdiri dari (a) *Toggle* flip-flop (T0...TN-1) yang saling berhubungan dengan keluaran C(0)...C(N-1) dan (b) Kombinasi dari D flip-flop (D1...DN-1) dengan keluaran Q(1)...Q(N-2) serta gerbang AND (G1...GN-1) yang berisikan logika *state decode* antar pencacah. Semua flip-flop dibangkitkan dengan sinyal pewaktu yang sama, f_{clock} dan ketika T flip-flop dalam kondisi di *set*, maka T flip-flop di *toggle* untuk waktu berikutnya.



Gambar 2.4. Blok diagram dari pencacah

Pengurangan waktu akibat *counter state decoding* secara kristis, dikurangi dengan menaikan performa pencacah agar mampu beroperasi pada frekuensi tinggi. Dengan didasarkan pada prediksi tiap keluaran pencacah akan berubah sebelum waktu sebenarnya terjadi yang mana perubahan ini harus dihindari maka tiap D flip-flop (D_i) difungsikan untuk menyiapkan sub sekuensial dari T flip-flop ($Ti+1$). Perubahan keluaran dari *state* diatur sesuai dengan satu siklus input sebelum perubahan ini harus ditempatkan. Demikian juga tiap gerbang AND (G_i) digunakan untuk memprediksi perubahan *state* keluaran dari pencacah C(i + 1) tiap dua siklus pewaktu sebelum siklus pewaktu yang ini berubah. Akhirnya sinyal *overflow* dari pencacah diproduksi oleh flip-flop D_{N-1} pada akhir

periode PWM dan digunakan untuk memasukkan nilai *duty cycle* yang baru ke dalam register input (Kourtrolis, dkk., 2005).

2.2 *Programmable Logic Devices* (PLDs)

Komponen digital dapat dibedakan dalam dua katagori yaitu katagori terikat *(fixed logic)* dan dapat diprogram *(programmable logic)*. Komponen terikat merupakan komponen permanen, dimana fungsi dan kemampuannya telah ditentukan oleh pabrik pembuatnya dan tidak dapat diubah. Sedangkan komponen yang dapat diprogram atau disebut dengan *Programmable Logic Devices* (PLDs) merupakan IC standar yang menawarkan penggunaan dengan jangkauan kapasitas gerbang logika, kecepatan dan karakteristik tegangan serta komponen ini sewaktu-waktu dapat diubah untuk meningkatkan beberapa fungsinya.

Waktu yang dibutuhkan untuk satu jenis komponen *fixed logic*, mulai dari desain, prototipe, dan pembuatan sampai dengan berbulan-bulan hingga lebih dari satu tahun bergantung pada kompleks tidaknya rancangan. Jika komponen yang dibuat tidak bekerja sebagaimana desain yang dirancang maka suatu desain baru harus kembali dibangun, dan ini membutuhkan biaya yang besar mulai *software* yang digunakan hingga teknologi semikonduktor yang dipakai. Dengan *Programmable Logic Devices*, perancang hanya perlu menggunakan *software* yang murah untuk merancang, mensimulasikan, dan menguji rancangannya.

Keuntungan yang lain jika menggunakan PLD adalah bahwa selama merancang pengguna dapat merubah rangkaian yang telah diprogram dalam PLD sampai dengan rancangan beroperasi sebagaimana yang dikehendaki. Hal ini karena suatu PLD dibangun pada *re-writable memory technology*, dimana dengan teknologi ini pengguna dapat menulis-menghapus program yang telah ditanamkan pada komponen hingga ribuan kali (Anonimous[1]).

Berbagai jenis PLD telah yang dikembangkan antara lain:

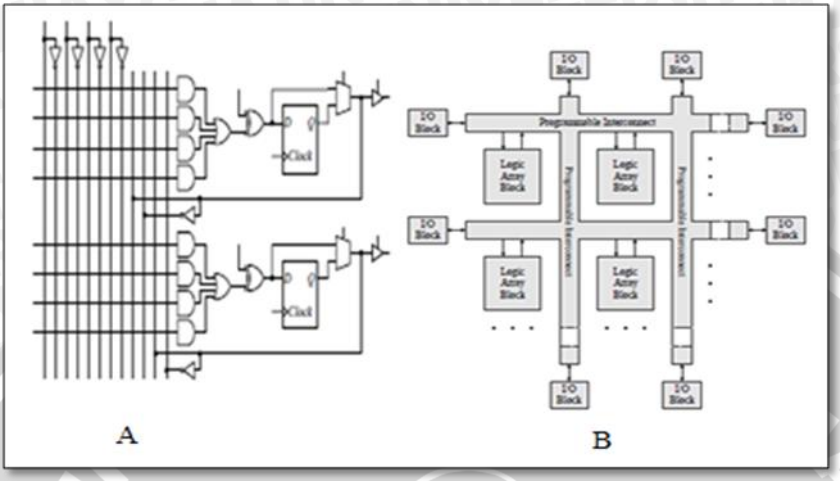
1. *Simple* PLD (SPLD), terdiri susunan gerbang AND dan OR yang memungkinkan untuk dirancang sebagai standar logika kombinasi.

2. *Complex* PLD (CPLD), selain susunan gerbang AND dan OR sebagaimana SPLD. CPLD juga dilengkapi dengan flip-flop yang membentuk suatu *macrocells* yang dapat digunakan sebagai standar logika sekuensial.
3. *Field Programmable Gate Array* (FPGA), susunan FPGA terdiri dari susunan*logiccells* (LC) dan *programmable interconnect matrix* yang mengelilingi *Configure Logic Block* (CLB). Jika dalam CPLD suatu CLB dibangun oleh beberapa *macrocells* namun dalam FPGA suatu CLB dibangun oleh satu atau lebih *Look-UpTables* (LUT) dan *bistables.* Sedangkan sebuah LUT sendiri merupakan sel-sel memori sebagaimana SRAM (Grout, 2008).

2.2.1 *Complex Programable Logic Devices*(CPLD)

CPLD (*Complex Programable Logic Devices*) merupakan salah satu jenis PLD. CPLD terdiri dari *Programable Array Logic* (PAL) sebagai suatu blok yang disebut *macrocells,* seperti yang ditunjukkan Gambar 2.5 (a). Tiap *macrocells* mempunyai susunan dari gerbang AND yang terikat dengan gerbang OR dan dapat diprogram sebagaimana PAL sebagai implementasi logika kombinasi. Kemudian sebuah gerbang XOR dihubungkan dengan keluaran PAL secara terbalik maupun tidak terbalik, serta sebuah flip-flop yang juga dihubungkan agar memberikan kemampuan dalam menerapkan bentuk skuensi dari rangkaian digital. Flip-flop ini di *by-pass* mengunakan *multiplexer* untuk dikombinasikan secara bersama-sama.

Group dari 16 *macrocells* dihubungkan secara bersama-sama dalam *Logic Array Block* (LAB). Semua LAB dihubungkan oleh *programablei nterconect* sebagaimana ditunjukan Gambar 2.5 (b), sedangkan *Global Bus* merupakan jalur yang dapat diprogram dan bisa menghubungkan antara satu sinyal dengan sinyal yang lain untuk tujuan tertentu.



Gambar 2.5(a) LAB dengan 2 *macrocells,* (b) Rangkaian internal dari CPLD yang terdiri dari LAB, *programmable interconect* dan I/O *block*

Blok I/O mengizinkan tiap I/O untuk dikonfigurasi sebagai input, ouput atau sebagai input-output. Semua pin I/O mempunyai *tri-state buffer* yang dapat dikontrol satu per satu. Pin I/O dikonfigurasi sebagai *port* input jika *tri state buffer* tidak dijalankan dan sebaliknya jika *tri-state buffer* dijalankan maka pin I/O difungsikan sebagai output (Hwang, 2005).

2.2.2 Xilinx XC9500 *Family*

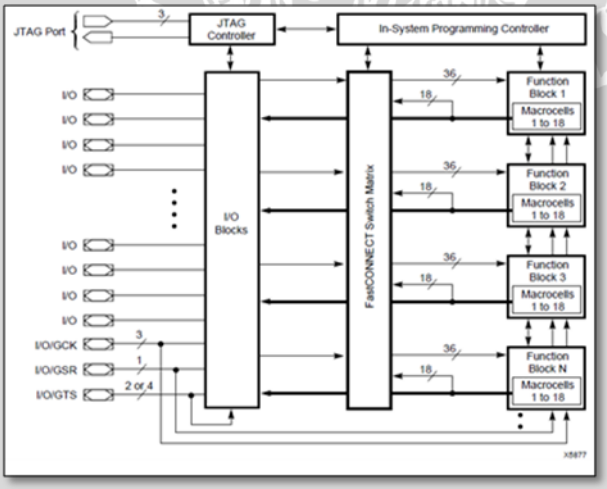
XC9500 merupakan salah satu jenis CPLD yang diproduksi oleh Xilinx.inc. Tiap komponen XC9500 sebagai sub sistem yang terdiri dari beberapa *Function Blocks* (FBs) dan I/O *Blocks* (IOBs) secara penuh dihubungkan dengan *Fast CONNECT switch matrix*. IOBs menyediakan *buffering* untuk komponen input-output. Sebagai mana ditunjukkan oleh Tabel 2.1 Sembilan komponen dari XC9500 *family* mempunyai jangkauan kapasitas 800 sampai 12800 gerbang yang dapat digunakan dengan 36 sampai 576 register secara berurutan. Beberapa jenis kemasan yang dapat dipilih dan kapasitas dari jumlah pin input-ouput.

Tabel 2.1 XC9500 *Family*

	XC9536	XC9572	XC95108	XC95144	XC95180	XC95216	XC95288	XC95432	XC95576
Macrocells	36	72	108	144	180	216	288	432	576
Usable Gates	800	1,600	2,400	3,200	4,000	4,800	6,400	9600	12,800
Registers	36	72	108	144	180	216	288	432	576
t_{PD} (ns)	5	7.5	7.5	7.5	10	10	10	10	12
t_{SU} (ns)	4.5	5.5	5.5	5.5	6.5	6.5	6.5	6.5	9.5
t_{CO} (ns)	4.5	5.5	5.5	5.5	6.5	6.5	6.5	6.5	9.5
f_{CNT} (MHz)	125	125	125	125	111	111	111	111	100
f_{SYSTEM} (MHz)	100	83	83	83	67	67	67	67	67
Preliminary									

Note: f_{CNT} = Operating frequency for 16-bit counters
f_{SYSTEM} = Internal operating frequency for general purpose system designs spanning multiple FBs.

Tiap FBs menyediakan kemampuan pemograman logika dengan kapasitas 36 input dan 18 output. *Fast CONNECT* menghubungkan semua FBs output dan input sinyal pada FBs input. Untuk tiap FBs, mempunyai 12 sampai 18 output dan disatukan oleh *enable signals drive* secara langsung pada IOBs seperti yang ditunjukkan pada Gambar 2.6 (Anonimous[2]).



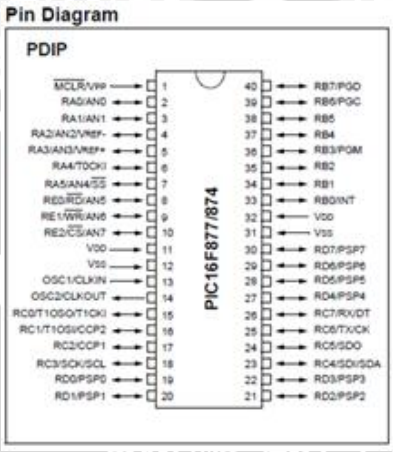
Gambar 2.6. Arsitektur dari XC9500 *family*.

Keuntungan dari CPLD jenis ini adalah bahwasanya level tegangan yang digunakan sebesar 5V dan mampu beroperasi hingga frekuensi eksternal 125 Mhz, serta model dari kemasan komponen cukup beragam.

Gambar 2.7. CPLD XC9536 -5PC44

2.3 Mikrokontroller

Mikrokontroller merupakan salah satu tipe dari mikroprosessor yang telah ditambahkan komponen-komponen pendukung seperti memori dan *port* komunikasi (misal: *Universal Asynchronous Reciever Transmitter*) disamping CPU yang tertanam dalam satu chip (Grout, 2008). Karena beberapa fungsional yang dibutuhkan telah terintegrasi di dalam mikrokontroller maka komponen ini lebih banyak digunakan dalam berbagai perancangan sistem instrumentasi. Mikrokontroller PIC16F877 pada Gambar 2.8 merupakan salah satu mikrokontroller yang diproduksi oleh Microchip Technology Inc. Salah satu keunggulan PIC dibanding mikrokontroller yang lain adalah arsitektur prosessor dari mikrokontroller termasuk dalam kelompok RISC (*Reducing Instructur System Computer*) sehingga mempunyai kecepatan tinggi dalam teknik pemrosesan data.



Gambar 2.8. Pin Diagram dari PIC16F877/874

Beberapa fitur yang terdapat pada mikrokontroller PIC16F877 antara lain:

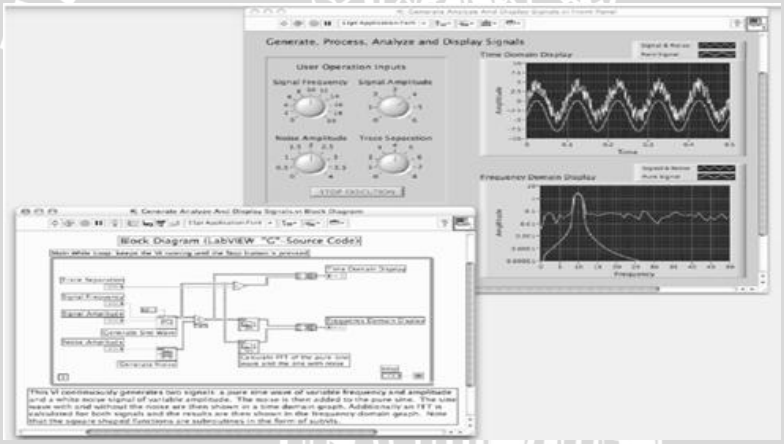
a. Timer0: 8-bit *timer/counter* dengan 8-bit *prescaler*
b. Timer1: 16-bit *timer/counter* dengan *prescaler*, dapat ditingkatkan selama mode *SLEEP* melalui *clock* eksternal
c. Timer2: 8-bit *timer/counter* dengan periode register 8-bit, *prescaler* dan *postscaler*

- *Two Capture*, *Compare*, PWM modul
  - 16-bit *Capture*, maksimal resolusi 12.5 ns
  - 16-bit C*ompare*, maksimal resolusi 200 ns
  - PWM dengan resolusi 10-bit
- 10-bit *multi-channel Analog-to-Digital Converter* (ADC)
- *Universal Synchronous Asynchronous Receiver Transmitter* (USART/SCI) dengan 9-bit alamat (Anonimous[3]).

2.4 LabVIEW

Laboratory Virtual Instruments Engineering Workbench dikenal sebagai LabVIEW merupakan bahasa pemograman secara grafik yang menggunakan ikon sebagai pengganti teks untuk

membuat aplikasi pemograman. Perbedaannya, bahasa pemograman dalam bentuk teks eksekusi program ditentukan oleh perintah-perintah yang ditawarkan dan seringkali terjadi kesalahan dalam membuat program akibat kesalahan peletakan teks maupun kode program yang dituliskan. Pada LabVIEW metode yang digunakan berupa alur program dari data, dimana aliran data ini melalui suatu rangkaian yang dibuat pada blok diagram ditawarkan pada *Virtual Instrumentasi* (Vls). *Virtual Instrumentasi* (Vls) merupakan program LabVIEW yang berupa instrumen fisis buatan, Gambar 2.9 merupakan contoh tampilan dari program LabVIEW yang terdiri dari diagram blok dan panel muka, diagram blok digunakan sebagai media perancangan dan panel muka digunakan sebagai penampil program yang telah dibuat pada diagram blok.

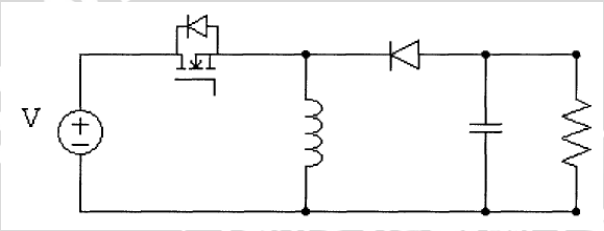


Gambar 2.9. Blok diagram dan *front panel* program LabVIEW

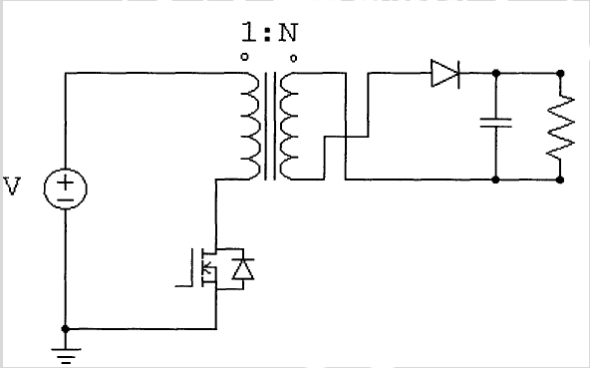
Dalam LabVIEW, sebuah *user Interface* bisa dibangun menggunakan seperangkat alat dan objek. *User interface* ini dikenal sebagai panel muka. Kemudian kode yang digambarkan secara grafik fungsi dibuat terpisah untuk dapat mengontrol objek pada panel muka. Kode-kode secara grafik ini diketahui juga sebagai kode G atau kode blok diagram menyerupai alur data atau *flowchar*t (Sumathi dan Surekha, 2007).

2.5 Tinjauan Konsep Konverter*Flyback*

Konverter *Flyback* merupakan salah satu tipe konverter *switching* atau dikenal sebagai *indirect converter*, salah satu jenis konverter daya tingkat menengah. Konverter ini dikembangkan berdasarkan pada konverter *Buck-Boost* seperti yang ditunjukkan pada Gambar 2.10, jika diperhatikan induktor tunggal pada Gambar 2.10 dapat diganti dengan induktor berpasangan atau sebuah tranformator tanpa mengubah cara kerja rangkaian maka sebuah konverter *Flyback* dapat dibuat (ditunjukan pada Gambar 2.11).



Gambar 2.10. Konverter *Buck-Boost*



Gambar 2.11. Konverter *Flyback*

Mode operasi konverter *Flyback* mirip dengan mode operasi dari konverter *Buck-Boost* kecuali perbedaan relatif untuk penggunaan Transformer menggantikan induktor sebagai penyetor daya tingkat

menengah. Sebagai contoh, *Flyback* bisa mengatur tegangan keluaran baik positif maupun negatif dengan bergantung pada perubahan fase antara lilitan primer dan sekunder. Sedangkan pada konverter *Buck-Boost* tegangan keluaran harus berlawanan dengan polaritas tegangan input, serta perbandingan lilitan trafo menjadi alternatif mudah untuk menaikkan atau menurunkan tegangan pada konverter *Flyback.*

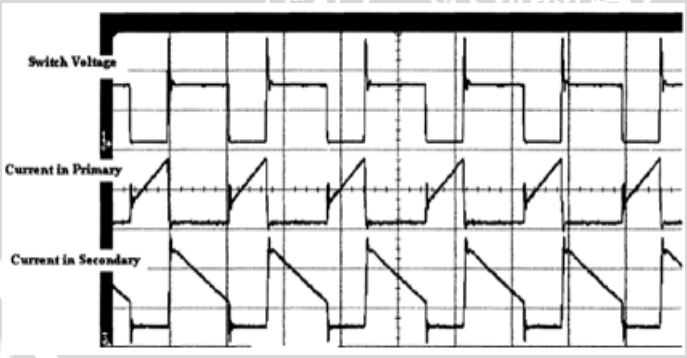
$$\frac{Vout}{Vin} = -\frac{D}{(1-D)} \qquad \frac{Vout}{Vin} = N \times \frac{D}{(1-D)}$$

Buck-Boost Flyback

Gambar 2.12. Hubungan input-ouput untuk konverter *Buck-Boost* dan Konverter *Flyback*

2.5.1 *Continous Conduction Mode* (CCM)

Mode operasi *Continous Conduction Mode* (CCM) merupakan tipe konveter yang dapat diimplementasikan pada frekuensi sinyal tetap. Arus lilitan sekunder tidak kembali menjadi nol ketika Transformer memindahkan energi yang berasal dari lilitan primer, jumlah energi yang dipindahkan untuk setiap kenaikan beban sesuai dengan persamaan $\frac{1}{2}LI^2$ (Kenia, 2004).



Gambar 2.13. Bentuk gelombang dari arus pada lilitan Transformer

Induksi magnetik dari *Flyback* pada mode operasi CCM dimulai pada kondisi dimana nilai arus tidak nol ketika saklar dalam kondisi *on* dan membutuhkan siklus per siklus energi untuk disimpan pada transformator (Gambar 2.13). Ketika saklar dalam posisi on maka arus pada lilitan primer akan mengalami kenaikan dengan *gradien* sebanding dengan besarnya harga induktansi dari lilitan primer, perubahan arus ini kemudian menyebabkan terjadi medan magnet. Sedangkan perubahan medan magnet yang ditimbulkan oleh arus pada lilitan primer menyebabkan induksi magnet pada lilitan sekunder dan menyebabkan timbulnya arus akibat induksi magnetik, besarnya arus yang mengalir bergantung pada banyaknya lilitan dari lilitan sekunder.

Besarnya nilai induktansi pada lilitan primer bergantung pada tegangan input, arus input dan waktu *on* atau waktu yang dibutuhkan agar terjadi induksi pada lilitan primer (lihat Lampiran 5), persamaan induktansi untuk lilitan primer diberikan pada persamaan dibawah ini :

$$L_{primer} = \frac{V_{in} t_{on}}{I_{inpk}} \qquad (3.3)$$

Semakin besar frekuensi *switching* yang digunakan maka semakin kecil nilai induktansi yang dibutuhkan, hal ini berguna dalam setiap perancangan *Flyback* jika mempertimbangkan ukuran dari Transformer yang digunakan. Disamping hal tersebut, untuk setiap perancangan konverter *step up,* jumlah lilitan primer senantiasa berpengaruh terhadap jumlah lilitan sekunder yang ditentukan berdasarkan rasio tegangan input dengan tegangan outputnya.

(halaman ini sengaja dikosongkan)

BAB III
METODE PENELITIAN

3.1 Waktu dan Tempat Penelitian

Penelitian ini dilaksanakan mulai bulan Januari 2013 sampai dengan bulan November 2013 di Laboratorium Instrumentasi Jurusan Fisika Universitas Brawijaya Malang.

3.2 Tahapan Pengerjaan

Tahapan dalam pengerjaan penelitian ini adalah sebagai berikut :

1. Studi literatur,
2. Perancangan sistem,
3. Pembuatan Alat,
4. Pengujian dan Pengolahan data (BAB IV),
5. Analisis hasil.

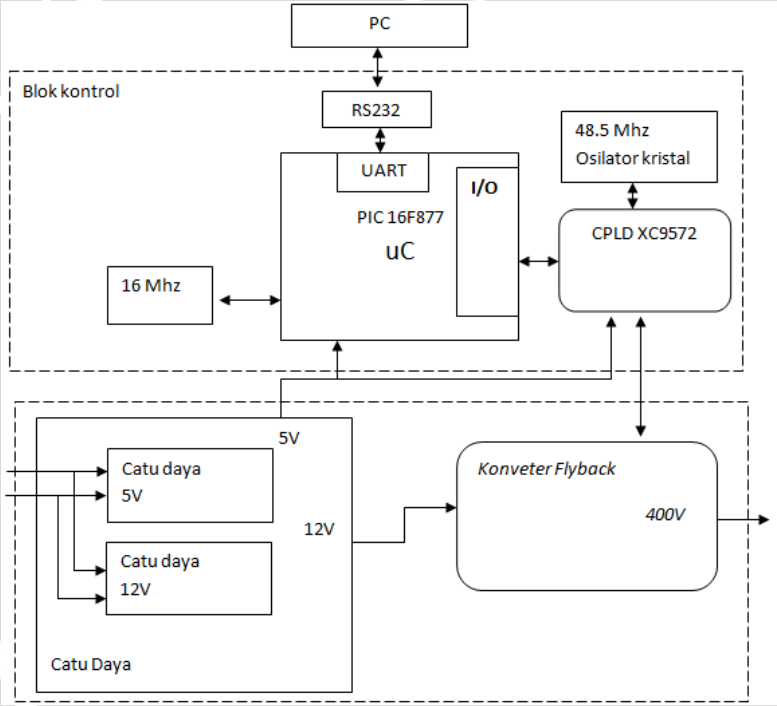
3.3 Perancangan Sistem.

Perancangan sistem intrumentasi (Gambar 3.1) secara umum dikelompokan ke dalam tiga sub bagian, yaitu : sub bagian perancangan teknik digital PWM pada komponen CPLD, sub bagian perancangan sistem pengiriman data untuk menentukan *duty cycle* PWM dan sub bagian perancangan sistem elektronik untuk konverter *Flyback.*

Prinsip kerja sistem ini meliputi pembangkitan PWM pada komponen CPLD. Pulsa digital dibangkitan melalui osilator eksternal dari komponen CLPD, kemudian pulsa digital tersebut dihitung melalui komponen pencacah yang telah dirancang didalam komponen CPLD untuk dibandingkan dengan nilai maksimum berdasarkan bit resolusi dari PWM. *Duty cycle* dari PWM ditentukan secara eksternal melalui program LabVIEW pada PC, nilai ini kemudian dikirim melalui komunikasi UART (*Universal Asynchronous Reciever Transmitter*) dengan mikrokontroller. Selanjutnya mikrokontroller mengirimkan data secara paralel melalui I/O, mikrokontroller diperlukan sebagai media komunikasi dengan

PC dikarenakan didalam komponen CPLD tidak dirancang untuk dapat berkomunikasi secara UART.

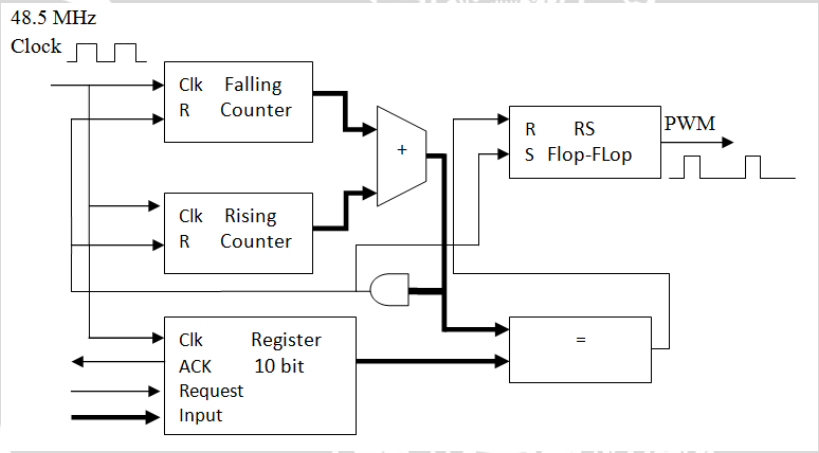
Hasil keluaran PWM dari komponen CLPD digunakan untuk memodulasililitan primer trafo dari rangkaian konverter *Flyback* agar menghasilkan arus induksi, sedangkan nilai dari *duty cycle* menentukan besarnya arus induksi tersebut. Semakin besar arus induksi pada lilitan primer maka semakin besar pula medan magnet yang dihasilkan, menurut teori *Faraday* yang menyatakan bahwa perubahan medan magnet akan menyebabkan terjadinya gerak gaya listrik (ggl) dan adanya ggl ini menyebabkan munculnya arus pada lilitan sekunder dari trafo.



Gambar 3.1. Sistem instrumentasi untuk pengaturan level tegangan DC pada konverter *Flyback.*

3.3.1 Perancangan teknik digital PWM pada komponen CPLD

Teknik digital yang digunakan dalam membangkitkan PWM pada komponen CPLD digambarkan pada blok diagram pada Gambar 3.2. *External clock* yang berasal dari osilator kristal, digunakan untuk membangkitkan pulsa digital kedalam komponen CPLD. Untuk mendapatkan satu periode PWM maka nilai dari deretan pulsa digital dibagi melalui komponen pencacah (*counter*) dimana besarnya nilai pembagian ditentukan oleh bit dari komponen pencacah. Selama proses pembagian, hasil dari perhitungan dibandingkan dengan nilai dari *duty cycle* yang berasal dari register lain yang telah dirancang, hasil dari komponen pembanding akan memberikan nilai *set* atau *reset* pada SR flip-flop. Hasil dari keluaran komponen SR flip-flop ini sebagai bentuk dari sinyal PWM.



Gambar 3.2. Blok diagram untuk arsitektur PWM yang dirancang

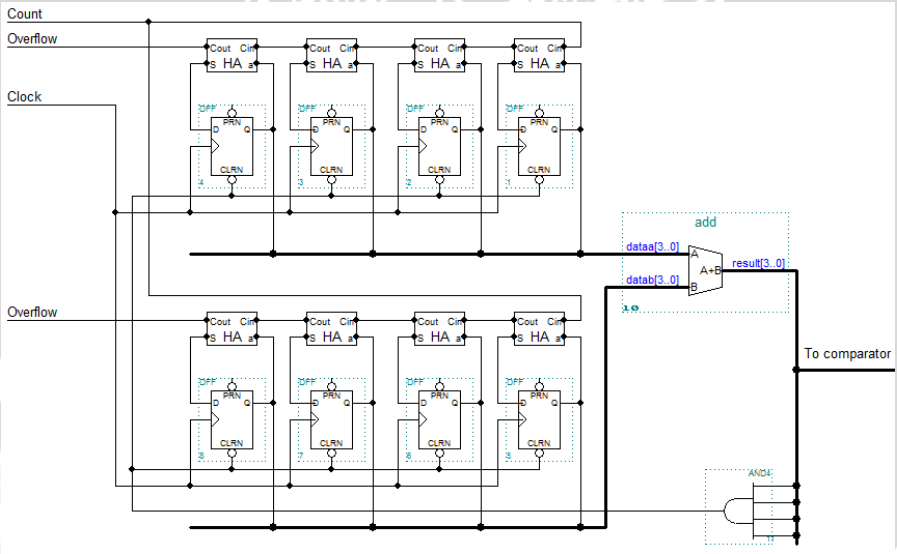
Frekuensi PWM yang dihasilkan oleh komponen CPLD bergantung pada nilai frekuensi dari *external clock* dan bit dari komponen pencacah (*counter*) dirumuskan kedalam persamaan:

$$F_{pwm} = \frac{F_{clock}}{2^n - 1} \qquad (3.1)$$

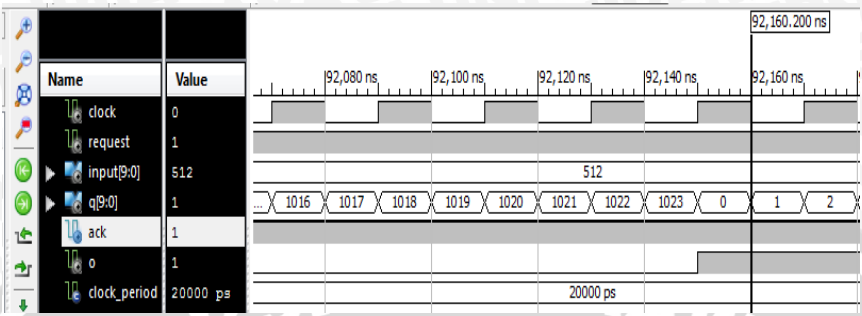
Jika tiap pewaktu dari kristal, baik saat *falling* maupun *rising* masing-masing di baca oleh pencacah maka persamaan 3.1 dapat ditulis kembali menjadi persamaan 3.2.

$$F_{pwm} = \frac{F_{clock}}{2^n - 1} x2 \qquad (3.2)$$

sehingga frekuensi dari PWM dapat ditingkatkan menjadi dua kali dari frekuensi sebelumnya. Teknik ini mempunyai keuntungan jika dibandingkan dengan menambah nilai dari osilator kristal, penggunaan pewaktu dengan frekuensi tinggi menyebabkan disipasi daya yang berlebihan pada komponen semikonduktor serta terjadinya *clock gating* yaitu terjadinya *delay* propragansi dari sinyal jika melewati deretan gerbang logika. Detail struktur pencacah dari pembangkit PWM yang dirancang ditunjukkan pada Gambar 3.3 sedangkan hasil simulasi dari program VHDL dari pembangkit PWM ditunjukan pada Gambar 3.4.

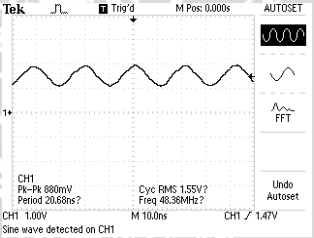


Gambar 3.3. Blok diagram dari pencacah yang dirancang



Gambar 3.4. Hasil simulasi VHDL untuk pencacah yang dirancang

Salah satu ketentuan yang harus dipenuhi dalam membangun model ini adalah jenis pewaktu atau kristal osilator yang digunakan haruslah memiliki tingkat presisi yang tinggi, sehingga tingkat kesalahan dari perbedaan periode masing-masing perhitungan dapat diperkecil. Jenis kristal osilator yang digunakan sebagai pewaktu dalam penelitian serta hasil keluaran dari kristal tersebut ini ditunjukkan pada Gambar 3.5.



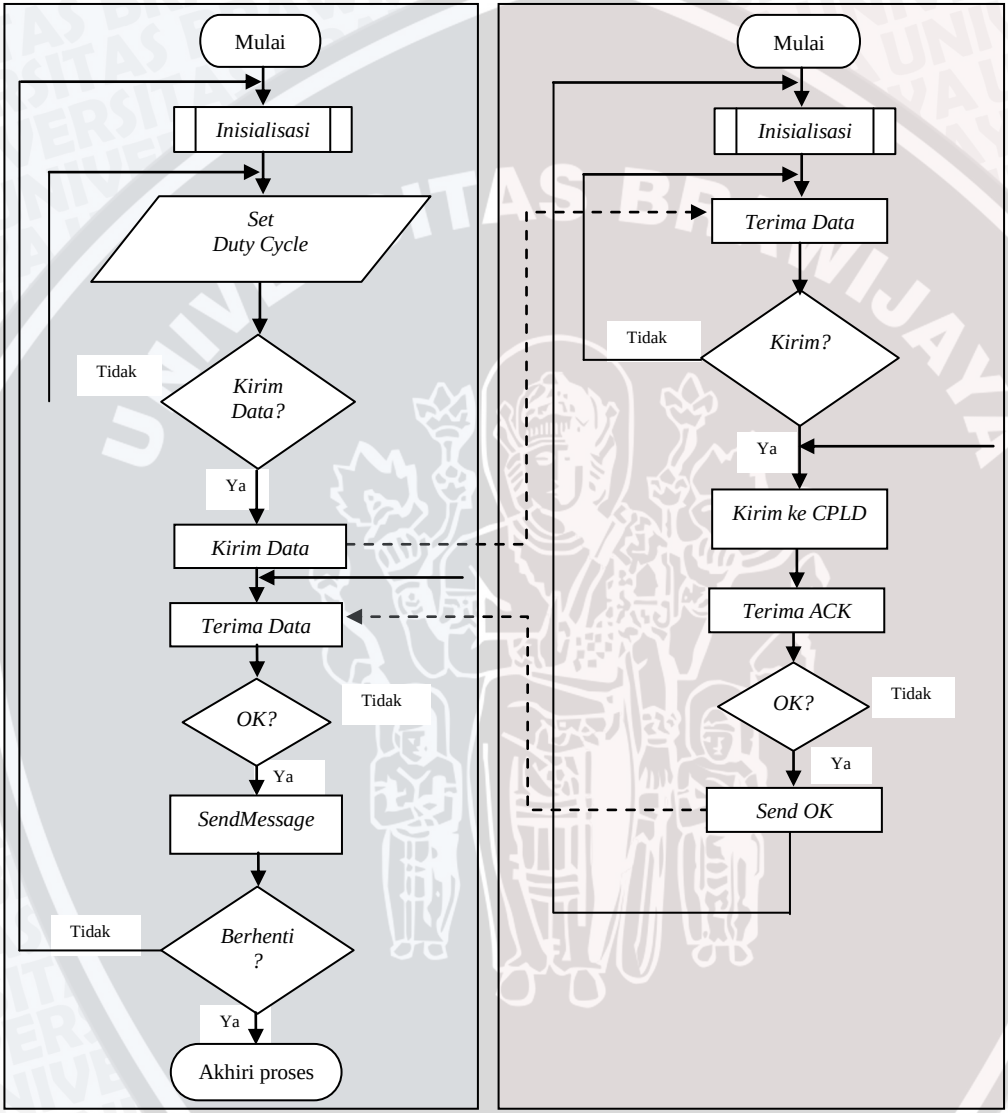
(a) (b)

Gambar 3.5.(a) TCXO *Crystal Oscilator*. (b) Hasil tes pada Osiloskop

3.3.2 Perancangan sistem pengiriman data untuk menentukan *Duty Cycle* PWM

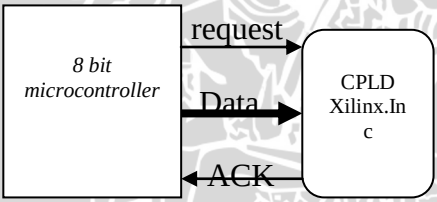
Komponen CPLD hanya memiliki sistem I/O secara digital, sehingga untuk menghubungkan komponen tersebut dengan komponen lain melalui sistem I/O yang ada harus diubah ke dalam format data digital. Input data digital yang mempresentasikan nilai dari *duty cycle* PWM dapat dilakukan dengan berbagai cara diantaranya adalah memanfaatkan komunikasi paralel antar I/O antara CPLD dan mikrokontroller untuk kemudian data yang dikirimkan dapat berupa digital maupun analog, hal ini dilakukan untuk memudahkan pengolahan data secara digital mengingat kapasitas dari CPLD yang terbatas. Kapasitas CPLD sendiri dibatasi pada jumlah *macrocell* yang dapat dijadikan *register* atau *counter* dan gerbang logika yang mempunyai fungsi matematis sebagaimana *product of terms*.

Pengaturan nilai *duty cycle* PWM pada penelitian ini dilakukan dengan bantuan *software* LabVIEW pada PC kemudian data dikirimkan ke CPLD dengan memanfaatkan *interface* mikrokontroller dengan PC secara UART, selanjutnya data dikirim secara paralel dari mikrokontroller ke CLPD. PWM yang dirancang pada CPLD mempunyai resolusi 10 bit sehingga data *duty cycle* maksimum yang dapat dikirim adalah 10 bit, mengingat kapasitas pengiriman data pada modul UART yang ada adalah 8 bit makadata 10 bit dikirimkan dalam dua kali pengiriman berupa 8 bit *low* dan 8 bit *high* (dimana hanya diambil 2 bit sebagai bit ke-9 dan ke-10 nya). *Flowchart* untuk pengiriman data antara PC dan mikrokontroller ditunjukkan pada gambar dibawah ini :



Gambar 3.6. *Flowchart* pengiriman data *duty* PC-mikrokontroller

Kapasitas masing-masing *register* pada mikrokontroller adalah 8 bit sehingga diperlukan sebuah teknik untuk mengirimkan 10 bit data secara paralel dari mikrokontroller ke CPLD, disamping itu kedua komponen dibangkitkan dengan pewaktu (*clock*) yang berbeda oleh karenanya dalam teknik pengiriman data sebaiknya dilakukan secara asinkron. Nilai masing-masing siklus kerja pada mikrokontroller adalah sekitar satu *microsecond* (μs), selama perintah *clear* atau perintah untuk mengganti data pada masing-masing register belum dilakukan maka nilai pada register tersebut adalah tetap sehingga dalam waktu yang hampir bersamaan perintah untuk mengaktifkan *port* keluaran secara bersamaan dapat dilakukan. Saat data siap dikirimkan maka fungsi *trigger* diaktifkan untuk memberi perintah kepada komponen CPLD untuk siap menerima data, sebagai sinkronisasi antara komponen CPLD dengan mikrokontroller maka komponen CPLD akan memberikan sinyal ACK kepada mikrokontroller untuk melakukan siklus kerja selanjutnya.



Gambar 3.7. Komunikasi antara komponen uC dengan CPLD

Psedocode bahasa asembli

```
Movflo,w
Movwf  PORTB   ; move data low
PORTB
Movfhi,w
Movwf PORTD     ;move data high
PORTD
Bsf       PORTD,5 ; selectrequest
Btfss      PORTD,4 ;skip if ACK='1'
Goto     $-1          ;goto return if ACK not
```

3.3.3 Perancangan sistem elektronik untuk konverter *Flyback.*

Dalam setiap perancangan konverter *Flyback*, hal utama yang perlu dilakukan adalah menentukan kapasitas transformator yang digunakan. Spesifikasi dari transformator ditentukan berdasarkan tegangan input, tegangan output, arus input, dan periode saat *switch* dalam keadaan *on*. Dari persamaan untuk perbandingan input dan output dari konverter *Flyback* (lihat Lampiran 5) diketahui bahwa:

$$\frac{V_{out}}{V_{in}} = \left(\frac{N2}{N1}\right) x \left(\frac{t_2}{t_1}\right) \qquad (3.3)$$

Jika konverter *Flyback* diharapkan mampu mengkonversi tegangan DC 12V menjadi 400V tegangan DC, maka untuk besarnya *duty-cycle* 0.5 perbandingan jumlah lilitan antara primer dan sekunder adalah

$$\frac{400}{12} = \left(\frac{N2}{1}\right) \qquad (3.4)$$

$$N2 = 33 \qquad (3.5)$$

Selanjutnya besarnya nilai minimum induktansi dari lilitan primer agar dapat mengkonversi seluruh daya input yang tersimpan ditentukan sebagai:

$$L_{min} = \left(\frac{V_{min}}{I_{inpk}}\right) x t_{on} \qquad (3.6)$$

Tegangan input V_{min} besarnya adalah 12V sedangkan

$$t_{on} = DxT = 0.5x10us = 5 \qquad (3.7)$$

dan arus puncak dari input *Flyback* ditentukan berdasarkan:

$$I_{inave} = \frac{P_{in}}{V_{in}} = \frac{12\ Watt}{12V} = 1A \qquad (3.8)$$

Sesuai dengan Gambar 2.13, untuk menentukan besarnya arus puncak pada lilitan primer ditentukan berdasarkan luasan daerah segitiga, dimana :

$$I_{inave} = 0.5xDxI_{inpk} \quad (3.9)$$

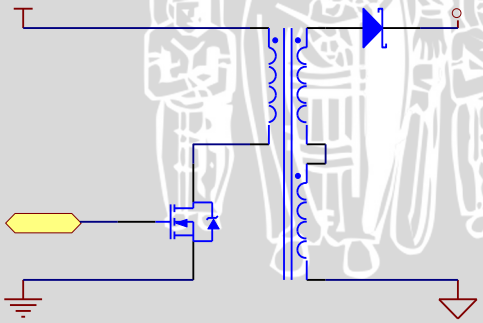
Maka arus puncak dari lilitan primer dapat dituliskan sebagai:

$$I_{inpk} = \frac{I_{inave}}{0.5x0.5} = \frac{1}{0.25} = 4A \quad (3.10)$$

Sehingga L_{min} adalah

$$L_{min} = \left(\frac{12V}{4A}\right)x5us = 15uH \quad (3.11)$$

Skematik dari konverter *Flyback* yang dirancang ditunjukan pada Gambar 3.8 di bawah ini.

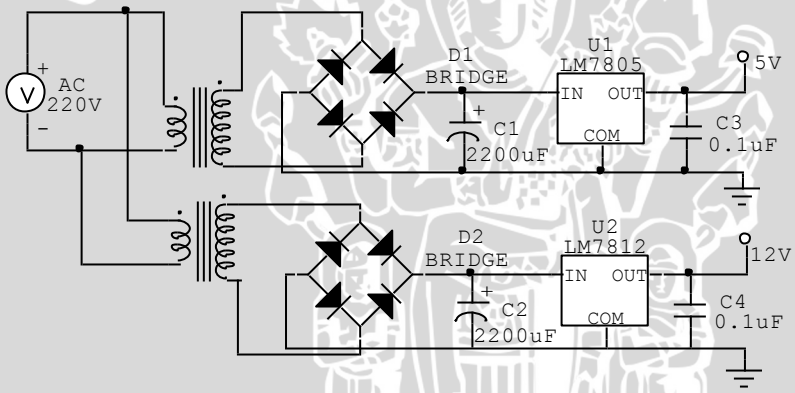


Gambar 3.8. Skematik rangkaian konverter *Flyback* yang dirancang

Pada skematik diatas terlihat bahwa antara komponen input dan output dari konverter *Flyback* terisolasi (tidak dihubungkan), ini berguna untuk mengurangi *loss* daya ketika terjadi kenaikan beban secara tiba-tiba pada sisi sekunder. Tegangan riak hasil induksi lilitan sekunderdisearahkan dengan dioda *schottky*, dioda ini mempunyai kemampuan *fast recovery* sehingga memungkinkan jika digunakan pada frekuensi tinggi.

3.3.4 Catu Daya

Catu daya merupakan seperangkat komponen yang dirancang untuk mensuplai daya bagi berbagai perangkat diantaranya sistem minimum dari mikrokontroller-CPLD dan Tegangan sumber untuk *Flyback*. Skematik dari catu daya yang dirancang ditunjukkan pada Gambar 3.9.



Gambar 3.9. Skematik dari catu daya yang di rancang

Catu daya yang dirancang terdiri dari trafo CT, dioda *bridge* sebagai penyearah tegangan AC dan komponen IC regulator LM78XX. Masing-masing perangkat baik untuk sistem minimum mikrokontroller maupun konverter *Flyback* diberi daya secara terpisah, hal ini untuk menghindari penurunan daya akibat dari kenaikan beban pada konverter *Flyback*.

3.4 Pembuatan Alat

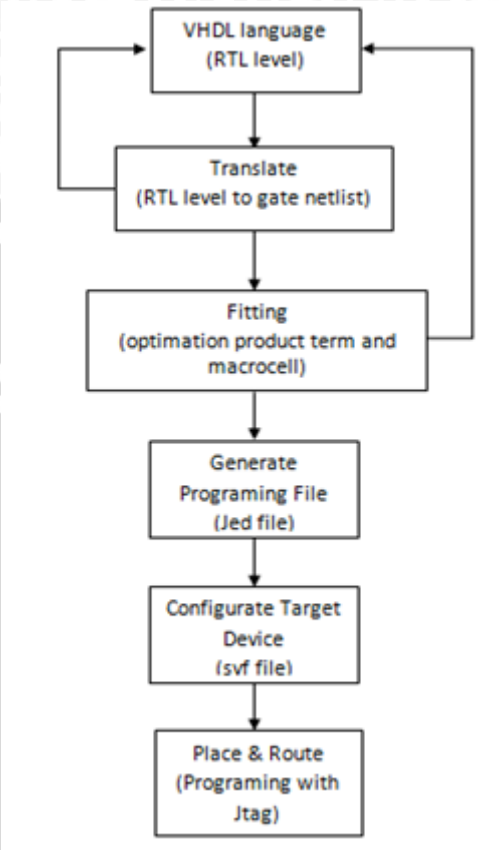
3.4.1 Implementasi desain kedalam komponen CPLD.

Proses perancangan pembangkit PWM pada komponen CPLD dilakukan beberapa tahap sebagaimana ditunjukkan pada Gambar 3.10 serta beberapa perangkat yang dibutuhkan dalam perancangan diberikan pada Tabel 3.1.

Tabel 3.1. Macam alat yang dibutuhkan dalam perancangan CPLD.

Tool	**Requirement**
VHDL synthesizer	Xilinx Ise Design Suite 13.4
Place &Route	Ise IMPACT-Adept Digilent
Simulation	Xilinx Ise Design Suite 13.4 (*simulation*)

Tahap awal berupa perancangan desain secara *behavioral* atau *struktural* dengan bahasa tingkat tinggi VHDL, proses ini disebut sebagai proses sintesis. Tahap berikutnya dikenal sebagi proses *translate* dimana bahasa VHDL diterjemahkan menjadi rancangan komponen-komponen digital yang dibangun dari gerbang-gerbang logika. Setelah tahap *translate*, proses selanjutnya adalah proses *fitting*. Sebagaimana kita tahu bahwa struktur CPLD terdiri dari *macrocell-macrocell* yang merupakan gabungan dari PAL dan T flip-flop sehingga perlu dilakukan semacam optimasi desain sesuai pada kapasitas *macrocell* CPLD.

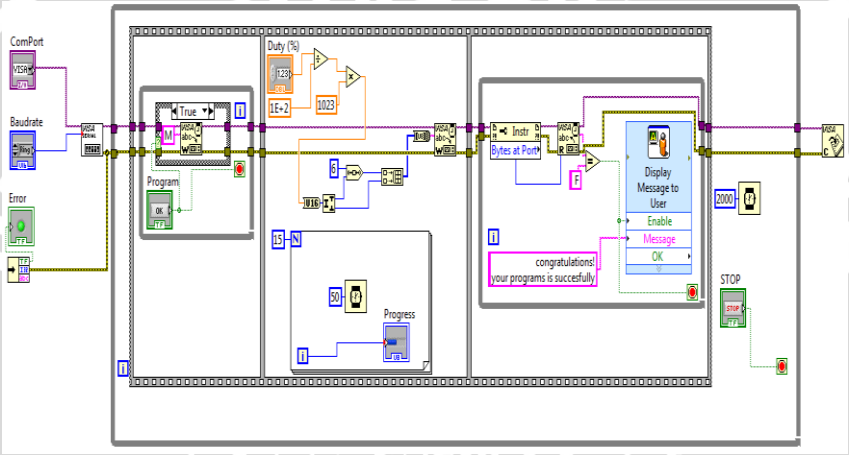


Gambar 3.10. Alur perancangan CPLD melalui bahasa VHDL (disadur dari *Circuit Design with VHDL* penulis Predoni,2004)

Semua proses diatas dilakukan dengan *software Xilinx Ise Design Suite* versi *13.4* keluaran dari perusahaan Xilinc.Inc. Sedangkan proses *place and route* dari komponen CLPD dilakukan dengan perangkat Jtag dan *software* Adept-Digilent, suatu produk yang dikembangkan oleh Digilent.Inc

3.4.2 Pembuatan sistem komunikasi data dari PC ke CPLD melalui Mikrokontroller.

Sistem komunikasi data untuk PC dibuat pada program LabVIEW dengan memanfaatkan komponen Instrument I/O, tampilan *block diagram* dari program tersebut ditunjukkan pada Gambar 3.11.

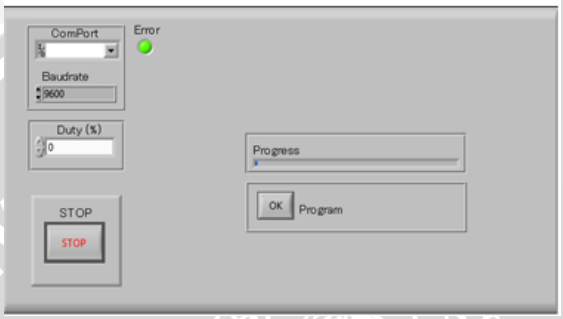


Gambar 3.11. Block diagram LabVIEW

Alur kerja dari program yang telah dibuat dalam *block diagram* diatasadalah sebagai berikut :

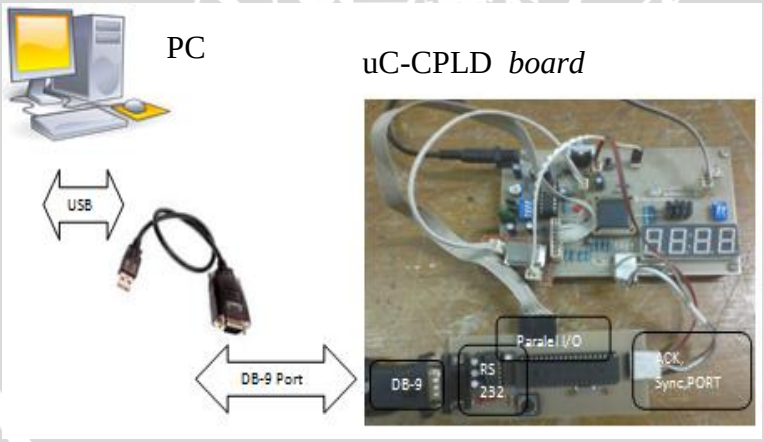
1. Data dalam format desimal sebagai representasi dari nilai *duty cycle* dikonversikan kedalam format data *unsigned16-bit integer* agar dapat diubah ke dalam format data 8 bit.
2. Data u*nsigned16-bit integer* diubah ke dalam 8-bit data berupa 8-bit data *high* dan 8-bit data *low*.
3. 8-bit data *high* dan 8-bit data *low* digabung dan diubah dalam format data *array* untuk kemudian dikonversikan ke dalam bentuk string, format data string ini diperlukan dalam setiap pengiriman data secara serial.

Selanjutnya tampilan *user interface* yang dibangun dalam *front diagram* LabVIEW ditunjukkan pada Gambar 3.12 di bawah ini, setiap nilai *duty cycle* PWM dapat dikirimkan secara *real-time* melalui *user interface* ini.



Gambar 3.12. Front diagram LabVIEW

Perangkat keras yang dibuat dalam membangun komunikasi ini ditunjukkan pada gambar 3.13.



Gambar 3.13. Pembuatan sistem komunikasi data PC-CPLD

Komunikasi antara mikrokontroller dengan PC dihubungkan dengan perangkat *USB to Serial converter* melalui *port* RX-TX pada mikrokontoller, sedangkan komunikasi antara CPLD dan mikrokontroller dilakukan lewat *paralel port*. Pemilihan *port* I/O pada komponen CPLD harus selektif dilakukan mengingat setiap desain dari CPLD harus melalui proses *fitting* atau optimasi, jika *port* I/O dalam setiap FBs dari CPLD terlalu banyak yang dipakai maka besar kemungkinan proses optimasi tidak berhasil dan disesuaikan juga dengan tingkat kesulitan dari arsitektur perancangan.

3.4.3 Pembuatan konverter *Flyback*

Salah satu proses yang paling penting dalam pembuatan konverter *Flyback* adalah pembuatan Transformer, proses ini penting karena transformator difungsikan sebagai alat transfer daya. Pembuatan komponen Transformer dilakukan dengan mempertimbangkan besarnya induksi minimum yang diperlukan, perbandingan lilitan primer dan sekunder serta nilai *leakage inductance.*

Nilai induktansi yang diperolah dari persamaan 11 adalah 15 μH, sedangkan dari proses pembuatan komponen Transformer diperoleh nilai induktansi sebesar 20 μH sehingga nilai tersebut memenuhi nilai induktansi minimum dari konverter *Flyback*. Pengukuran nilai induktansi ini dilakukan pada alat LCR meter seperti yang ditunjukkan pada gambar dibawah ini:

Gambar 3.14. Pengukuran nilai induktansi Trafo dengan LCR meter

Untuk menghasilkan nilai induktansi sebesar 20 µH diperlukan lilitan primer sebanyak 12 lilitan, berdasarkan rasio antara lilitan primer dan sekunder (persamaan 3.5) diketahui bahwa banyaknya lilitan sekunder yang diperlukan adalah :

$$\frac{N2}{N1} = \frac{22}{1} = \frac{N2}{12} \qquad (3.12)$$

sehingga banyaknya lilitan sekunder untuk trafo:

$$N2 = 12x22 = 264 \text{ lilitan} \qquad (3.13)$$

Untuk mengurangi kehilangan daya pada saat proses konversi daya dilakukan maka dalam setiap pembuatan trafo perlu dilakukan pengukuran terhadap nilai *leakage inductance*, nilai *leakage inductance* menunjukkan kemampuan dari trafo untuk saling menginduksi. Teknik pengukuran yang dilakukan hampir serupa dengan pengukuran nilai induktansi primer dari trafo hanya saja pada lilitan sekunder perlu dihubungkan singkat (*short*) sehingga hasil pengukuran pada LCR meter merupakan nilai dari *leakage inductance*. Besarnya nilai efektif dari induktansi dapat diperoleh melalui persamaan:

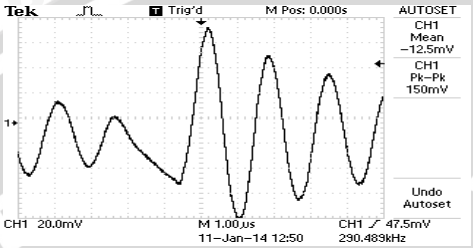
$$K = \sqrt{1 - \frac{Lk}{Lp}} \qquad (3.14)$$

dari pengukuran pada LCR meter diperoleh $L_k = 2{,}06\ \mu H$ dan $L_p = 20\mu H$ sehingga nilai dari nilai efektif induktansinya:

$$K = \sqrt{1 - \frac{2\mu H}{20\mu H}} = 0.89$$

Untuk mengurangi *leakage inductance* ini maka diperlukan induktor tambahan dengan nilai induktansi sama dengan nilai *leakage inductance* yang telah diukur. Sedangkan untuk mengurangi *over damping* akibat dari pengaturan *duty cycle* digunakan kapasitor

snubber dengan nilai yang disesuaikan dengan frekuensi dari *over damping*, frekuensi ini ditentukan oleh hasil pengukuran tegangan *drain-source* dari komponen MOSFET.



Gambar 3.15. Hasil pengukuran tegangan *drain-source* dari MOSFET

Hasil pengukuran pengukuran menunjukkan bahwa frekuensi *over damping* sekitar 780 kHz, sehingga nilai kapasitor *snubber* yang diperlukan dapat diperoleh dari :

$$f = \frac{1}{2\pi\sqrt{LkC}} \qquad (3.15)$$

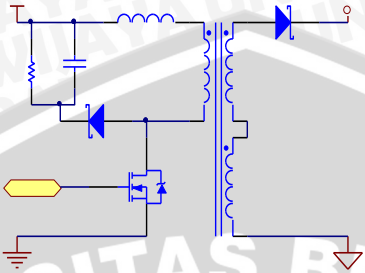
atau dapat ditulis menjadi

$$C = \frac{1}{4\pi^2 f^2 Lk} \qquad (3.16)$$

sehingga nilai kapasitor *snubber* diperoleh :

$$C = \frac{1}{4(3.14)^2(290x10^3)^2 2x10^{-6}} = 150nF$$

skematik yang dibuat ditunjukkan pada Gambar 3.16 dibawah ini:



Gambar 3.16. Skematik rangkaian *Flyback* dengan kapasitor *snubber*

BAB IV
HASIL DAN PEMBAHASAN

4.1 Pengujian CPLD Sebagai Pembangkit PWM

4.1.1 Hasil sintesis untuk teknik digital pembangkit PWM pada CPLD

Tabel 4.1 dan Tabel 4.2 mendeskripsikan jumlah gerbang dan *macrocell* yang digunakan dalam membangun pembangkit PWM pada komponen CLPD. Data Tabel 4.1 dan Tabel 4.2 diatas merupakan hasil dari proses sintesis dan *fitting* dari perancangan pembangkit PWM pada komponen CPLD dengan bahasa VDHL. Hasil *fitting* diatas mengindikasikan bahwa arsitektur pembangkit PWM ini dapat dirancang pada komponen CPLD dengan kapasitas minimum 55 *macrocell*, serta jumlah *product of term* yang tersedia adalah 280 *term*.

Tabel 4.1. *Macro-statistics* dari perancangan arsitektur PWM

Komponen	Total
10-bit adder carry out	1
10-bit up counter	2
11-bit comparator equal	1
10-bit register	1

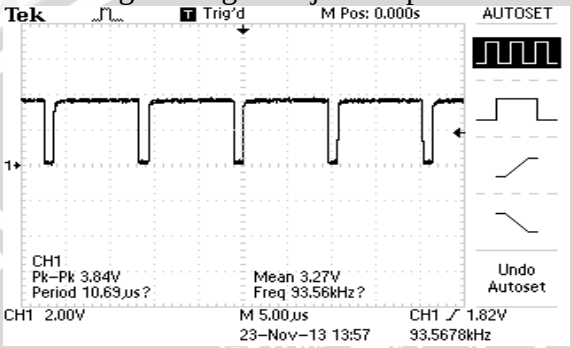
Tabel 4.2. *Resource summary* dari perancangan arsitektur PWM

Macrocells Used	Pterms Used	Registers Used	Pins Used	Function Block Inputs Used
55/72 (77%)	280/360 (78%)	30/72 (42%)	14/34 (42%)	132/144 (92%)

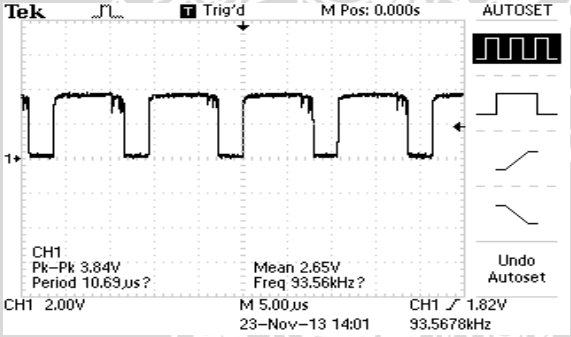
Besarnya jumlah *macrocell* yang digunakan umumnya berkaitan dengan nilai bit yang digunakan oleh masing-masing komponen, sedangkan kapasitas penggunaan *product of term* berkaitan erat dengan tingkat kesulitan dari sebuah algoritma pemograman dari VHDL.

4.1.2 Pengujian keluaran PWM dari komponen CPLD

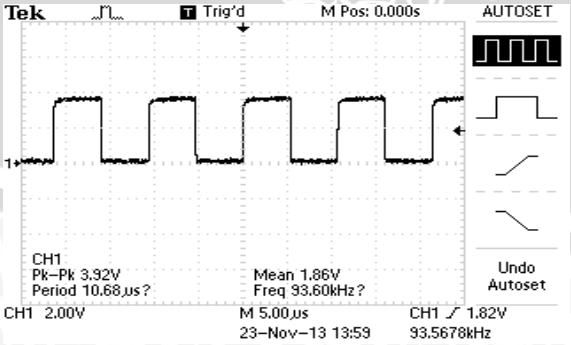
Pengujian hasil PWM dari komponen CLPD dilakukan pada alat ukur osiloskop dengan beberapa nilai *duty cycle* yang diberikan, hasil pengukuran ini masing-masing ditunjukkan pada Gambar 4.1-4.5.



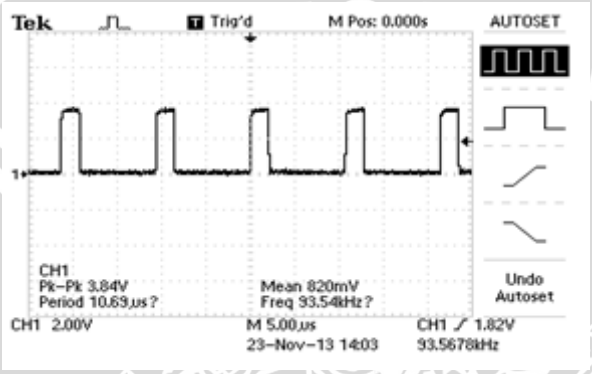
Gambar 4.1. Hasil pengukuran untuk *duty cycle* 90%



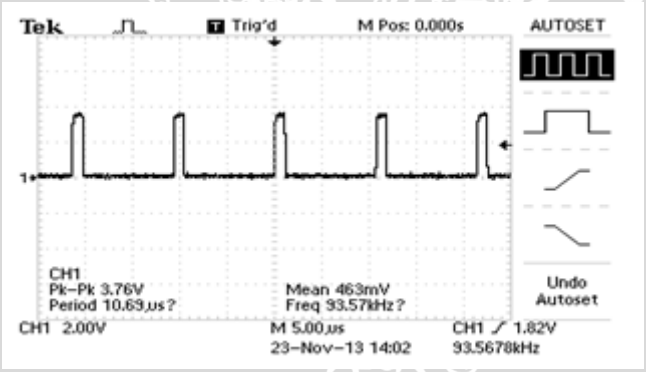
Gambar 4.2. Hasil pengukuran untuk *duty cycle* 75%



Gambar 4.3. Hasil pengukuran untuk *duty cycle* 50%



Gambar 4.4. Hasil pengukuran untuk *duty cycle* 20%



Gambar 4.5. Hasil pengukuran untuk *duty cycle* 10%

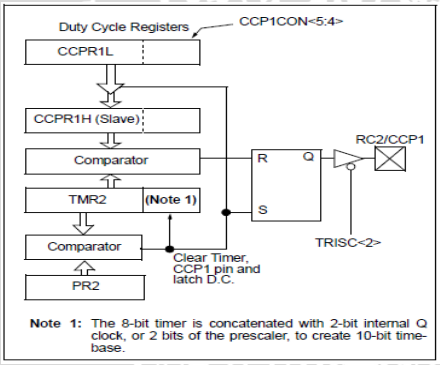
Hasil pengukuran diatas menunjukan bahwa metode deteksi *rising* dan *falling* untuk model pencacah yang dirancang dapat diimplementasikan sebagai komponen pencacah dari pembangkit PWM. Disamping itu, nilai *duty cycle* PWM yang dirancang dapat

berupa bilangan bulat atau pecahan bergantung pada tingkat resolusi dari PWM.

4.2 Perbandingan nilai frekuensi PWM antara mikrokontroller dan CPLD

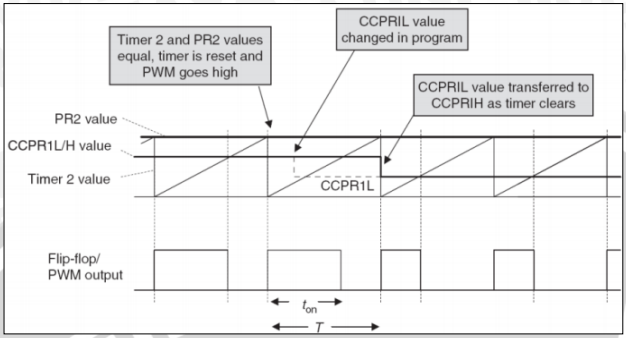
Pulse width modulation (PWM) pada mikrokontoller PIC16F87X dihasilkan oleh modul CCP (*Capture/Compare/PWM*), walaupun fokus penelitian ini hanya pada modul PWM. Untuk membangkitkan PWM modul CCP pada mikrokontroller PIC 16F87X, dapat dilakukan dengan tiga mode operasi yaitu : *Capture* dan *Compare* menggunakan *register* Timer1, serta mode PWM melalui *register* Timer2. Masing-masing mode operasi dapat dirancang pada modul CCP1 maupun CCP2 dengan beberapa pengecualian.

Diagram blok PWM untuk mode operasi PWM dari mikrokontroller PIC 16F87X ditunjukkan pada Gambar 4.6



Gambar 4.6. Diagram blok mode operasi PWM PIC 16F87X

Pada gambar diatas terlihat bahwa diagram blok yang telah dirancang pada mikrokontroller hampir sama dengan diagram blok yang telah dirancang pada penelitian ini (Gambar 3.2), hanya saja kapasitas dari mikrokontroller untuk membangkitkan PWM terbatas model pencacah yang digunakan (Timer2) dan resolusi PWM yang ditentukan berdasarkan nilai dari PR2.



Gambar 4.7. Teknik pembangkitan PWM untuk mode PWM

Pada Gambar 4.6 dan Gambar 4.7 terlihat bahwa nilai *duty cycle* PWM diatur berdasarkan nilai dari register CCP1H dan CCP1L (16 bit *compare register*), dimana untuk resolusi 10 bit PWM di ambil 2 bit nilai tertinggi dan 8 bit nilai terendahnya. Kedua *register* pada modul CCP dikontrol oleh CCP1CON untuk modul CCP1 dan CCP2CON untuk modul CCP2.

Besarnya frekuensi PWM yang dihasilkan oleh mikrokontroller PIC16F87X ditentukan oleh persamaan:

$$f_{pwm} = \frac{1}{T} = \frac{1}{(PR2+1)x(Timer2\ input\ period)} \quad (4.1)$$

atau dapat dituliskan sebagai

$$f_{pwm} = \frac{1}{T} = \frac{1}{(PR2+1)x(T_{osc}x\ 2^2\ x(prescale))} \quad (4.2)$$

dimana T_{osc} merupakan periode dari osilator kristal yang digunakan dan 2^2 adalah nilai 2 bit Q *internal clock* dari Timer2 dan PR2 sebagai *register* pembanding untuk Timer2, nilainya antara 0 sampai dengan 255. Untuk besarnya nilai kristal osilator yang sama dengan nilai osilator yang digunakan pada komponen CPLD yaitu 48 MHz, frekuensi PWM yang mampu dihasilkan oleh mikrokontroller PIC 16F87X adalah:

$$f_{pwm} = \frac{1}{T} = \frac{1}{(255+1)x(2x10^{-8}x\ 2^2\ x(1))} = \frac{1}{2048x10^{-8}} = 48\ kHz$$

dimana nilai perbandingan frekuensi untuk besarnya resolusi PWM dan kristal osilator yang sama dapat diketahui melalui persamaan dibawah ini:

$$\frac{f_{pwm-CPLD}}{f_{pwm-uC}} = \frac{93.5\ kHz}{48\ kHz} = 1{,}94 \qquad (4.3)$$

dari persamaan diatas diketahui bahwa nilai frekuensi PWM yang mampu dihasilkan oleh komponen CPLD hampir dua kali lebih tinggi dibandingkan nilai frekuensi PWM yang mampu dihasilkan oleh mikrokontroller.

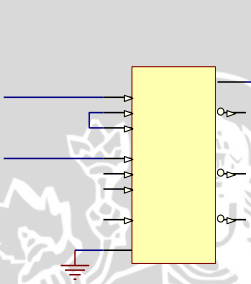
Beberapa nilai frekuensi PWM yang mampu dihasilkan oleh mikrokontroller PIC 16F87X berdasarkan resolusi dan kristal osilator yang digunakan ditabelkan seperti pada Tabel 4.3, dimana nilai frekuensi maksimum dari kristal osilator yang diizinkan adalah sebesar 20 MHz.

Tabel 4.3 Frekuensi dan Resolusi PWM berdasarkan nilai osilatornya

PWM Resolution	20 MHz		10 MHz		2 MHz		Units
	Min	Max	Min	Max	Min	Max	
10-bit	1.22	19.53	0.613	9.77	0.123	1.96	kHz
9-bit	1.22	39.06	0.613	9.77	0.123	3.92	kHz
10-bit	1.22	19.53	0.613	9.77	0.123	1.96	kHz
9-bit	1.22	39.06	0.613	9.77	0.123	3.92	kHz
10-bit	1.22	19.53	0.613	9.77	0.123	1.96	kHz
9-bit	1.22	39.06	0.613	9.77	0.123	3.92	kHz
10-bit	1.22	19.53	0.613	9.77	0.123	1.96	kHz
9-bit	1.22	39.06	0.613	9.77	0.123	3.92	kHz

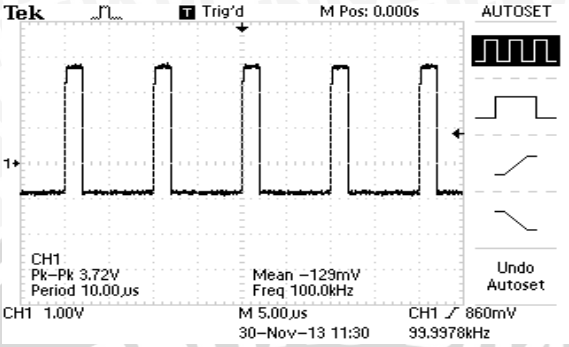
4.3 Pembuatan dan Pengujian Rangkaian Penguat Tegangan PWM

Sinyal PWM yang berasal dari CPLD berada dalam level tegangan 3V sampai dengan 3.3V, agar mampu men-*drive* komponen MOSFET pada rangkaian konverter *Flyback* tegangan tersebut perlu dinaikkan hingga level tegangan 5V sehingga diperlukan rangkaian penguat tegangan. Rangkain penguat tegangan yang dirancang dalam penelitian ini memanfaatkan komponen IC74HC04 *hex inverter*, sebagaimana ditunjukkan pada Gambar 4.8.

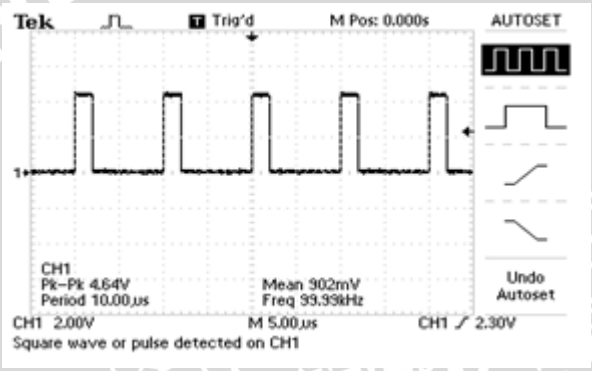


Gambar 4.8. Rangkain *hex inverter* IC 74HC04.

Tiap sinyal yang masuk pada komponen *hex inverter* akan dibalikkan fasenya, untuk itu diperlukan pembalikan dua kali agar sinyal masuk sama fasenya dengan dengan sinyal keluaran dari komponen *hex inverter*. Sesuai dengan karakteristik IC CMOS, tegangan yang input minimum yang mampu di *treshold* adalah berkisar antara 2,5V-3V oleh karenanya komponen ini dapat dijadikan sebagai penguat tegangan keluaran dari komponen CPLD. Hasil pengujian yang dilakukan untuk rangkaian penguat tegangan ditunjukkan pada Gambar 4.7 dan Gambar 4.8.



Gambar 4.9. Sinyal input untuk pengujian rangkaian *hex inverter*



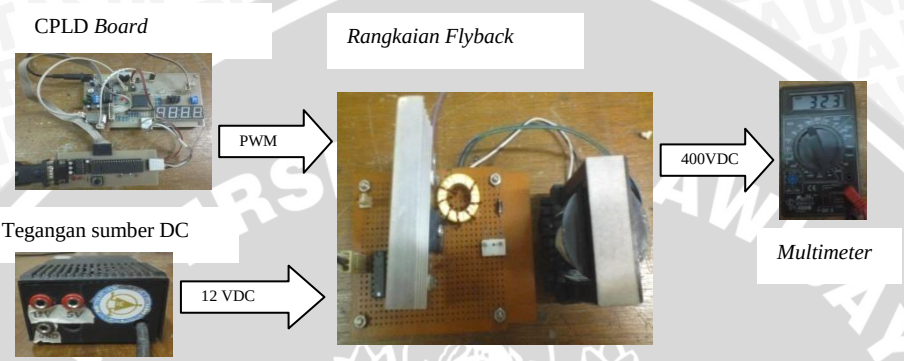
Gambar 4.10. Sinyal keluaran hasil pengujian rangkaian *hex inverter*

Dari hasil pengujian diatas menunjukkan bahwa komponen *hex inverter* tidak hanya berfungsi sebagai penguat level tegangan keluaran dari komponen CPLD tetapi juga dapat dimanfaatkan untuk memperbaiki kualitas dari PWM.

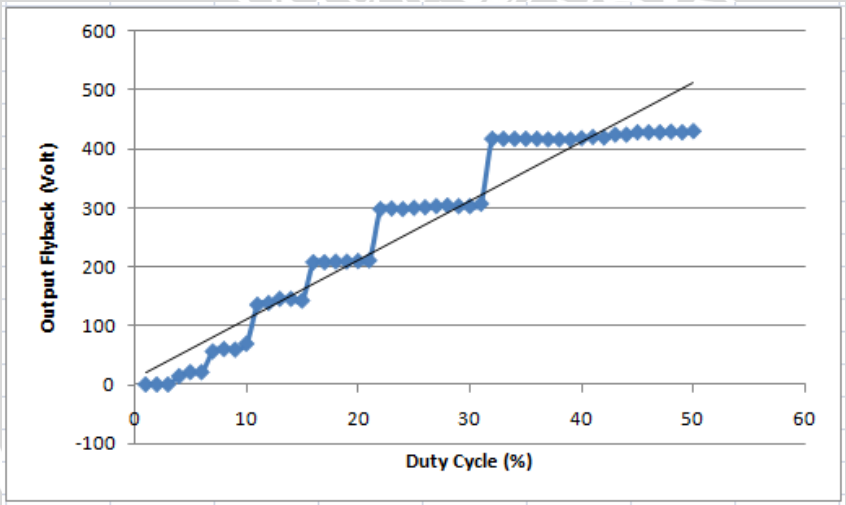
4.4 Pengujian CPLD sebagai kontrol rangkaian *Flyback* untuk konverter DC-DC

Pengujian rangkaian *Flyback* dilakukan dengan memberikan input tegangan DC sebagai tegangan sumber dan sinyal PWM

dengan nilai *duty cycle* tertentu yang berasal dari komponen CPLD. Pengujian ini dilakukan untuk mengetahui besarnya tegangan output dari rangkain *Flyback* untuk beberapa *duty cycle* yang diberikan.



Gambar 4.11. Pengujian CPLD sebagai kontrol rangkaian *Flyback*

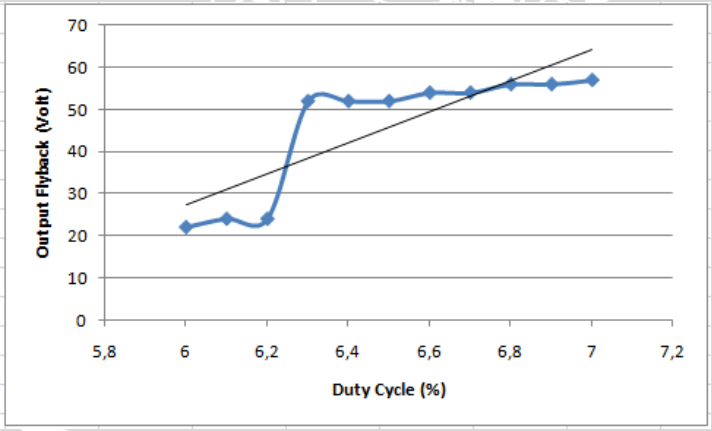


Grafik 4.1 Grafik Tegangan keluaran *Flyback* sebagai fungsi *Duty Cycle* PWM

Grafik 4.1 merupakan representasi dari hasil pengukuran tegangan keluaran dari konverter *Flyback* untuk nilai *duty cycle* nol sampai dengan *duty* maksimum yang diizinkan. Dari Grafik 4.1 dapat dijelaskan bahwa lebar *duty cycle* PWM minimum yang diperlukan agar sinyal PWM mampu men-*drive* komponen MOSFET adalah sekitar empat persen, jika dihitung berdasarkan tegangan yang keluaran dari komponen *hex-inverter* yaitu sebesar 5.5 V maka tegangan yang dibutuhkan agar komponen MOSFET dalam kondisi aktif adalah:

$$V_{Gate} = \frac{4}{100} x 5.5\ Volt = 0.22\ Volt \qquad (4.4)$$

Sedangkan nilai *duty* maksimum dari sinyal PWM yang dibutuhkan agar konverter *Flyback* mampu menghasilkan tegangan keluaran sebesar 400 V hampir sama dengan nilai yang diharapkan dari teori yang telah ditentukan pada persamaan 3.3, tegangan keluaran yang sama bisa jadi dapat diperoleh untuk nilai *duty* lebih dari *duty* maksimum namun secara efisiensi tentu akan berkurang karena dapat menyebabkan peningkatan temperatur pada komponen MOSFET hingga terjadi disipasi daya pada komponen tersebut.



Grafik 4.2 Grafik Tegangan keluaran *Flyback* untuk beberapa tingkat presisi dari *Duty Cycle* PWM

Dengan resolusi PWM yang dibangun mampu meningkatkan nilai presisi dari *duty cycle* PWM sehingga dapat memberikan nilai yang bervariasi pada keluaran konverter *Flyback*. Sebagaimana yang ditunjukkan pada Grafik 4.2, bahwasanya tingkat presisi *duty cycle* PWM yang mampu dicapai hingga nilai 0,1 persen, oleh karenanya pengaturan tegangan keluaran pada konverter *Flyback* dapat dilakukan melalui nilai *Duty Cycle* dalam bentuk bilangan bulat ataupun dalam bentuk bilangan pecahan.

Untuk mengurangi disipasi daya pada MOSFET akibat dari pemasangan tunggal komponen tersebut pada konverter *Flyback* maka digunakan media pendingin dengan ukuran yang lebih besar. Disamping itu, salah satu kelemaham dari rangkaian *Flyback* adalah penggunaan MOSFET tunggal sebagai kontrol rangkaian sehingga digunakan jenis MOSFET yang mempunyai tegangan V_{ds} relatif besar.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan dari pengujian alat, pengambilan data dan pengolahan data, maka dapat disimpulkan bahwa *high frequency* PWM telah berhasil dirancang secara digital pada komponen CPLD. Sinyal PWM yang dihasilkan dengan teknik pencacahan *rising* dan *falling* pada komponen CPLD mampu menghasilkan frekuensi lebih tinggi daripada teknik PWM pada mikrokontroller PIC16F87X dan konverter tegangan DC dari 12V ke 400V dengan rangkaian *Flyback* yang dikontrol oleh CPLD telah berhasil dibuat.

5.2 Saran

Untuk penelitian selanjutnya perlu dilakukan penyempurnaan terkait dengan desain dan implementasi PWM pada komponen CPLD maupun implementasinya sebagai pengontrol rangkaian konverter DC-DC/DC-AC.

DAFTAR PUSTAKA

Anonimous[1]. *What is Programmable Logic?*. Dalam http://xilinx.com/company/about/programmable diakses tanggal 19 Desember 2012

Anonimous[2]. *PIC 16F87X Data Sheet*. Dalam http://microchip.com/downloads/en/DeviceDoc/30292C.pdf diakses tanggal 24 Juli 2010

Anonimous[3]. *The Programmable data book*. Dalam http://noel.feld.cvut.cz/semi/xilinx/xilinx96.pdf diakses tanggal 19 Desember 2012

Anonimous[4]. AN594 *Using the CCP moduls*. Dalam http://microchip.com/downloads/en/DeviceDoc/31014a.pdf diakses tanggal 30 September 2013

Anonimous[5]. *Analysis and Design Flyback : How to Analysis Flyback Converter*. Dalam //www.youtube.com/watch?v=9YlNMFT3sz8. diakses tanggal 25 Januari 2014

Grout, Ion. 2008. *Digital systems design with FPGAs and CPLDs*. Elsevier:Oxford,UK

Hwang, Enoch O. 2004. *Digital Logic and Microprocessor Design with VHDL*. La seira University:Riverside, USA

Kourtrolis, Eftichios.,dkk. 2005. "*High-frequency pulse width modulation implementation using FPGA and CPLD Ics".* Journal of Systems Architecture , Vol.52 (2006): pp. 332–344

Kenia, Mayur V. 2004. "*Development of a isolated flyback converter Employing Boundary-Mode Operation and Magnetic Flux sensing feedback"*. Departement of Engineering. MIT

Predoni, Volnei A. 2004. *Circuit design with VHDL.* MIT Press: Massachusetts

Rahim, N.A.dan Islam Z. "*Field Programmable Gate Array-Based Pulse- Width Modulation for Single Phase Active Power Filter"*. American Journal of Applied Sciences, Vol.6 (2009): pp1742-1747

Sumathi, S dan Surekha, P. 2007. *LabVIEW based Advanced Instrumentation Systems.* Springer Berlin Heidelberg:New York

```
--------------------------------------------------------------------------------
library IEEE;
USE ieee.STD_LOGIC_1164.all;
USE ieee.STD_LOGIC_UNSIGNED.all;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE work.my_components.all;

entity components is PORT(clock: IN STD_LOGIC;
            request: IN STD_LOGIC;
            Input : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
            Q : OUT STD_LOGIC_VECTOR(9 DOWNTO 0);
            ACK   : OUT STD_LOGIC;
            O : OUT STD_LOGIC);
end components;

architecture Behavioral of components is
signal flag: STD_LOGIC;
signal flow: STD_LOGIC;
signal Q1: INTEGER RANGE 0 TO 1023;
signal Q2: INTEGER RANGE 0 TO 1023;
signal C: INTEGER RANGE 0 TO  1024;
signal duty: INTEGER RANGE 0 TO 1023;

begin
  flow<='1' WHEN C=1024 ELSE '0';
  u1:cnt1   PORT MAP (clock, flow, Q1);
  u2:cnt2   PORT MAP (clock, flow, Q2);
  u3:alu    PORT MAP (Q1,Q2,C);
  u4:data   PORT MAP (clock, request,Input, ACK, duty);
  flag<='1' WHEN C=duty ELSE '0';
  u5:SR_ff  PORT MAP (flow,flag,O);
  Q<=Conv_std_logic_vector(C,10);
end Behavioral;
--------------------------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL; -- need this to add STD_LOGIC_VECTORs

ENTITY cnt1 IS PORT (
            Clock: IN STD_LOGIC;
            flow : IN STD_LOGIC;
            Q   : OUT Integer range 0 to 1023);
END cnt1;

ARCHITECTURE Behavioral OF cnt1 IS
SIGNAL valueA: Integer range 0 to 1023;
BEGIN
```

```
PROCESS (flow, Clock)
        BEGIN
                IF (flow='1') Then
                        valueA <=0;
                ELSIF (clock'event AND clock='1') THEN
                        valueA <= valueA + 1;
                END IF;
END PROCESS;
Q<=valueA;
end Behavioral;
----------------------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL; -- need this to add STD_LOGIC_VECTORs
ENTITY cnt2 IS PORT (
        Clock: IN STD_LOGIC;
        flow : IN STD_LOGIC;
        Q   : OUT Integer range 0 to 1023);
END cnt2;

ARCHITECTURE Behavioral OF cnt2 IS
SIGNAL valueB: Integer range 0 to 1023;

BEGIN
PROCESS (flow, Clock)
    BEGIN
        IF (flow='1') Then
            valueB <=0;
        ELSIF (clock'event AND clock='0') THEN
             valueB <= valueB + 1;
        END IF;
END PROCESS;
Q<=valueB;
end Behavioral;
----------------------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

entity alu is port (
A: IN INTEGER RANGE 0 TO 1023;
B: IN INTEGER RANGE 0 TO 1023;
C: OUT INTEGER RANGE 0 TO 1024);
end alu;

architecture Behavioral of alu is
begin
    C<=(A+B);
end Behavioral;
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity data is port (Clock: in STD_LOGIC;
                     input: in STD_LOGIC_VECTOR(9 DOWNTO 0);
                      request: in STD_LOGIC;
                     ACK    : out STD_LOGIC;
                     output : out INTEGER RANGE 0 TO 1023);
end data;

architecture Behavioral of data is
SIGNAL duty:STD_LOGIC_VECTOR(9 DOWNTO 0);

begin
PROCESS (Clock, request)
  BEGIN
   IF rising_edge(clock) THEN
      IF (request='1') THEN
        duty<=input;
      END IF;
    END IF;
END PROCESS;
output<=conv_integer(duty);
ACK<='1' WHEN request='1' ELSE '0';

end Behavioral;
----------------------------------------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE work.my_components.all;

ENTITY SR_FF IS PORT ( S: IN STD_LOGIC;
               R: IN STD_LOGIC;
               O: OUT STD_LOGIC);
END SR_FF;

ARCHITECTURE structural OF SR_FF IS
SIGNAL q_not:STD_LOGIC;
SIGNAL q:STD_LOGIC;
BEGIN
U1: nor_1 PORT MAP (S, q, q_not);
U2: nor_2 PORT MAP (q_not, R, q);
O<=q;
end structural ;
-------------------------------------------------------------------------------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY nor_1 IS PORT (
               A: IN STD_LOGIC;
               B: IN STD_LOGIC;
               C: OUT STD_LOGIC);
END nor_1;

ARCHITECTURE Behavioral OF nor_1 IS
```

```
BEGIN
C<=A nor B;
end Behavioral;
```

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY nor_2 IS PORT (
            X: IN STD_LOGIC;
            Y: IN STD_LOGIC;
            Z: OUT STD_LOGIC);
END nor_2;

ARCHITECTURE Behavioral OF nor_2 IS
BEGIN
Z<=X nor Y;

end Behavioral;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

package my_components is
component cnt1
PORT (clock: IN STD_LOGIC;
        flow : IN STD_LOGIC;
        Q    : OUT Integer range 0 to 1023);
end component;




component cnt2
PORT (clock: IN STD_LOGIC;
        flow : IN STD_LOGIC;
        Q    : OUT Integer range 0 to 1023);
end component;

component alu
PORT (A: IN INTEGER RANGE 0 TO 1023;
         B: IN INTEGER RANGE 0 TO 1023;
         C: OUT INTEGER RANGE 0 TO 1024);
end component;

component data
PORT (clock:IN STD_LOGIC;
          request: IN STD_LOGIC;
          Input  : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
          ACK    : OUT STD_LOGIC;
         Output : OUT INTEGER RANGE 0 TO 1023);
end component;

component SR_ff
PORT (S: IN STD_LOGIC;
```

```
                R: IN STD_LOGIC;
                O: OUT STD_LOGIC);
end component;

component nor_1
PORT (A: IN STD_LOGIC;
                B: IN STD_LOGIC;
                C: OUT STD_LOGIC);
end component;

component nor_2
PORT (X: IN STD_LOGIC;
                Y: IN STD_LOGIC;
                Z: OUT STD_LOGIC);
end component;

end my_components;
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY test_bench IS
END test_bench;

ARCHITECTURE behavior OF test_bench IS
   COMPONENT components
   PORT(clock : IN  std_logic;
            request : IN  std_logic;
            Input : IN  std_logic_vector(9 downto 0);
           Q : OUT  std_logic_vector(9 downto 0);
          ACK : OUT  std_logic;
          O : OUT  std_logic);
   END COMPONENT;
   --Inputs
  signal clock : std_logic ;
  signal request : std_logic ;
  signal Input : std_logic_vector(9 downto 0);
  --Outputs
  signal Q : std_logic_vector(9 downto 0);
  signal ACK : std_logic;
  signal O : std_logic;
  -- Clock period definitions
  constant clock_period : time := 20 ns;
  BEGIN
  uut: components PORT MAP ( clock => clock,
                                        request => request,
                                        Input => Input,
                                        Q => Q,
                                        ACK => ACK,
                                        O => O );
  clock_process :process
  begin
                clock <= '0';
```

```
            wait for clock_period/2;
            clock <= '1';
            wait for clock_period/2;
   end process;
   begin
     wait for 100 ns;
          request<='1';
         input<="0111110110";
     wait for 10us;
        request<='0';
       input<="1000000000";
      wait for 50us;
      request<='1';
       input<="1000000000";
     wait;
   end process;
END;
```

Lampiran 2

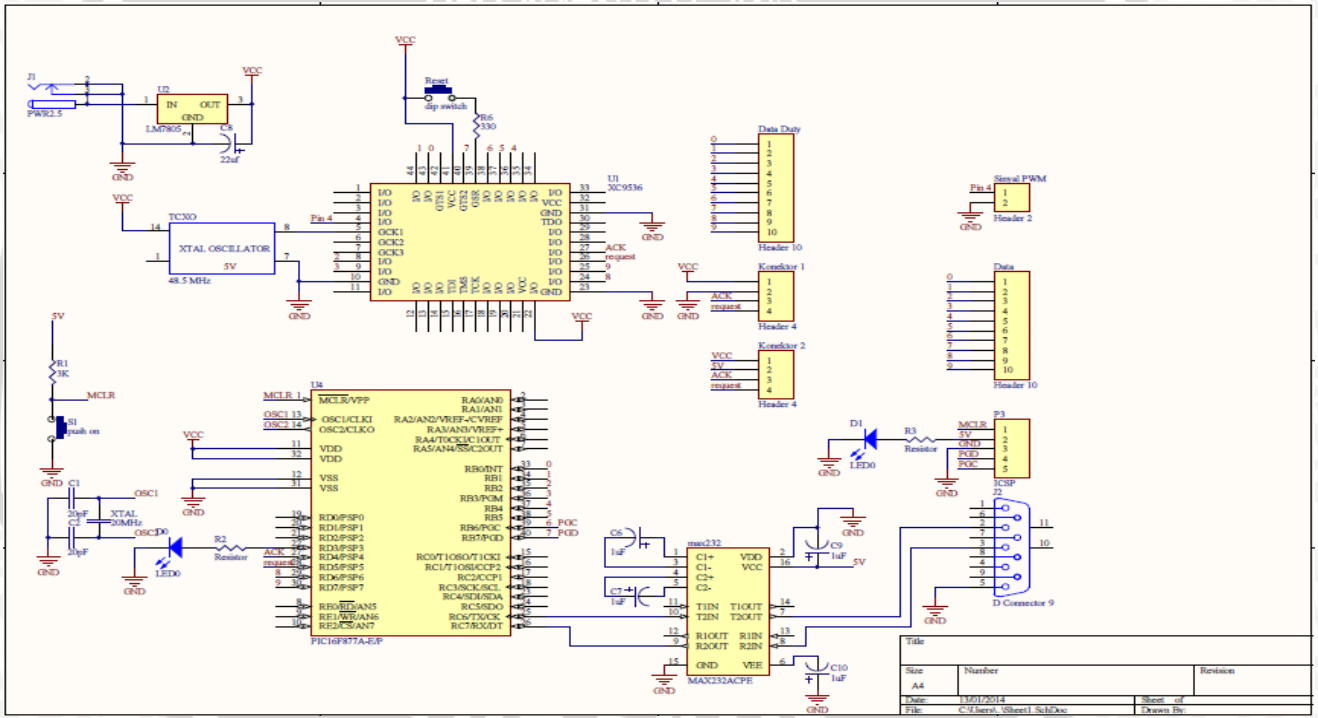
Program Mikrokontroller Untuk pengiriman Data

```
  list P = 16F877
  #include <P16f877.INC>
    ERRORLEVEL   -302
__CONFIG _HS_OSC & _WDT_OFF & _LVP_OFF
CBLOCK 0x20 ; RAM AREA for USE at address 20h
CONT1
CONT2
hi
lo
ENDC

org 0x0000
GOTO START
org 0x0005
;-------------------------------------------------
START:
 loop call init
     call sinkron
                call process
                goto loop
;-------------------------------------------------
init:
                bcf STATUS,RP1
                bsf STATUS,RP0
                movlw   0x24                    ;transmit enabled, high speed
                movwf   TXSTA
                movlw   D'103'
                movwf   SPBRG
                movlw   b'00010111'
                movwf   TRISD
                clrf    TRISB
                clrf    PIE1
```

```
        bcf STATUS,RP0
        movlw   0x08            ;init timer1, pres 1:1, clock source=internal (Fosc/4), osc enable
        movwf   T1CON
        movlw   0x90                    ;rx-tx port enabled, continous recieve
        movwf   RCSTA
        clrf    PIR1
        clrf    hi
        clrf    lo
        return

;-----------------------------------------------------
sinkron:
    bcf STATUS,RP0
    btfss PIR1,RCIF
          goto  $-1
          movfw RCREG
          bcf   STATUS,Z
          xorlw 'M'
          btfss STATUS,Z
          goto  START
          return
;-----------------------------------------------------
process:
        call recieve
        call mode
        call shift
   return

recieve:
        bcf STATUS,RP0
        movlw 0x30
         movwf FSR
continue btfss PIR1,RCIF
         goto  $-1
         movfw RCREG
         movwf INDF
         incf  FSR,F
         btfss FSR,1
         goto  continue
        return
;-------------------------------------
mode:
         bcf STATUS,RP0
         movlw 0x30
         movwf FSR
         movfw INDF
         movwf hi
         incf  FSR,F
         movfw INDF
         movwf lo
         return
```

```
shift:
        bcf STATUS,RP0
        movfw  lo
        movwf PORTB
        movfw  hi
        movwf PORTD
        bsf   PORTD,5
        call  ms_1
        btfss PORTD,4
        goto  $-1
        bcf   PORTD,5
        movlw 'F'
        call  send
        bsf   PORTD,3
        call  s_1
        bcf   PORTD,3
        clrf  PORTD
        clrf  PORTB
        return

send:bcf STATUS,RP0
        btfss PIR1,TXIF
        goto  $-1
        movwf TXREG
        return
;-------------------------------------------------------------
s_1:movlw D'4'          ;option 'j'
        movwf   CONT2
l2      movlw   D'250'      ;option 'h'
        movwf   CONT1
l1      bcf   STATUS,RP0    ;bank 0
        bcf   PIR1,TMR1IF   ;clear overflow flag
        movlw   0XEE     ;238
        movwf   TMR1H ;load in TMR1H register
        movlw   0X12     ;12
        movwf   TMR1L ;load in TMR1L register
        bsf     T1CON,TMR1ON    ;start timer1
        btfss   PIR1,TMR1IF     ;TMR1 in register PIR, timer1 overflow?
        goto    $-1
        decfsz  CONT1,f
        goto    l1
        decfsz  CONT2,f
        goto    l2
        return

ms_250:  movlw   D'250'                 ;option 'h'
        movwf   CONT1
loop250 bcf     STATUS,RP0
        bcf     PIR1,TMR1IF
        movlw   0XEE
        movwf   TMR1H
        movlw   0X12
        movwf   TMR1L
```

```
        bsf     T1CON,TMR1ON    ;start to count
        btfss   PIR1,TMR1IF
        goto    $-1
        decfsz  CONT1,f
        goto    loop250
        return

ms_50:movlw     D'50'       ;option 'h'
        movwf CONT1
loop1 bcf   STATUS,RP0      ;bank 0
        bcf   PIR1,TMR1IF   ;clear overflow flag
        movlw   0XEE        ;238
        movwf   TMR1H   ;load in TMR1H register
        movlw   0X12        ;12
        movwf   TMR1L   ;load in TMR1L register
        bsf     T1CON,TMR1ON        ;start timer1
        btfss   PIR1,TMR1IF         ;TMR1 in register PIR, timer1 overflow?
        goto    $-1
        decfsz CONT1,f
        goto    loop1
        return

ms_1:bcf STATUS,RP0
        bcf     PIR1,TMR1IF
        movlw   0XFF
        movwf   TMR1H
        movlw   0X5E
        movwf   TMR1L
        bsf     T1CON,TMR1ON    ;start to count
        btfss   PIR1,TMR1IF
        goto    $-1
        return
END
```

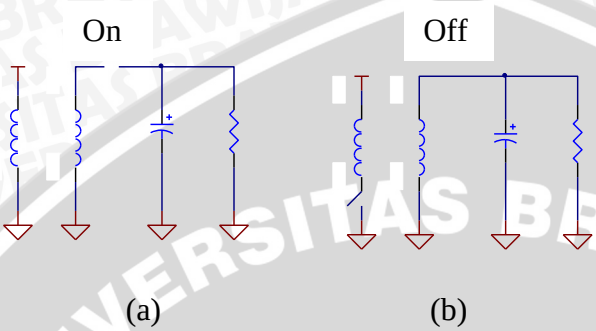

J1
PWR2.5
U2
IN
OUT
GND
LM7805
C8
22uF
VCC
Reset
dip switch
R6
330
U1
XC9536
TCXO
XTAL OSCILLATOR
5V
48.5 MHz
Pin 4
ACK
request
Data Duty
Header 10
Sinyal PWM
Header 2
Konektor 1
Header 4
Konektor 2
Header 4
Data
Header 10
R1
3K
MCLR
S1
Push on
C1
20pF
C2
20pF
XTAL
20MHz
OSC1
OSC2
R2
Resistor
LED0
U4
PIC16F877A-E/P
D1
LED0
R3
Resistor
P3
ICSP
PGD
PGC
C6
1uF
C7
1uF
max232
C9
1uF
5V
C10
1uF
MAX232ACPE
D Connector 9
Title
Size
A4
Number
Revision
Date:
13/01/2014
Sheet of
File:
C:\Users\..\Sheet1.SchDoc
Drawn By:

Qty	Part Num	Description
1	XC9572 -5PC44	*Xilinx CPLD*
1	PIC16f877A	*PIC microcontroller*
1	MAX 232	*RS 232 level*
1	IRFP450	*Power MOSFET*
1	74HC04	*hex inverter*
1	TCXO 48 Mhz	*Crystal Osilator*
1	B66375G0000X127	*Ferit Core*
1	1Amp-18V Universal	*Step down Transfomer*
1	LM7805,LM7812	*Voltage regulator*
1	XTAL 20 Mhz	*Crystal Osilator*
1	Db9 female	*RS 232 Conector*
5	1uF, 25 V	*Elco capacitor*
2	1000uF, 50v	*Elco capacitor*
2	20pF	*Disc Capacitor*
2	0.1uF	*Ceramic Capacitor*
1	150nF	*High Voltage Disc Capacitor*
1	320, 1W	*Carbon Film resistor*
4	320	*Carbon Film resistor*
1	3k	*Carbon Film resistor*
2	344-056	*Tactile switch*
1		*PLL Socket 44 pin*
1		*Dip Socket 44 pin*
1		*Dip Socket 16 pin*
1	TDC-002-1-2.5mm	*DC Jack*
2	RC102	*Bridge Rectifier 1 Amp*
2	FR103	*Schottky Diode*
1		*Ferrit beade*
4		*0,4 Watt LED*

Lampiran 4
Data hasil pengukuran tegangan keluaran konverter Flyback

Duty Cycle (%)	Tegangan Keluaran (Volt)
1	1
2	1
3	1
4	15
5	22
6	22
7	57
8	61
9	60
10	70
11	136
12	139
13	146
14	146
15	143
16	208
17	208
18	209
19	209
20	210
21	211
22	298
23	299
24	298
25	300
26	301
27	303
28	304
29	303
30	303
31	307
32	417
33	417
34	417
35	417
36	417
37	416
38	416
39	416
40	418
41	420
42	419
43	424
44	424
45	428
46	428
47	428
48	429
49	428
50	430

Duty Cycle (%)	Tegangan Keluaran (Volt)
6	22
6,1	24
6,2	24
6,3	52
6,4	52
6,5	52
6,6	54
6,7	54
6,8	56
6,9	56
7	57



Gambar 5.1. Skematik rangkaian *Flyback*: (a) Keadaan *On*, dan (b) Keadaan *Off*.

Pada saat keadaan *On* pada rangkaian *Flyback* (Gambar 5.1a), besarnya tegangan yang melewati induktor primer sebanding dengan besarnya tegangan input yang diberikan pada rangkaian atau dapat dituliskan sebagai:

$$V_{Lp} = Vin \qquad (5.1)$$

selama keadaan *On*, besarnya arus yang dihasilkan oleh komponen induktor akibat adanya perubahan tegangan dapat ditentukan melalui persamaan :

$$I_{lp} = (V_{in}/L)x\, t_{on} \qquad (5.2)$$

dimana t_{on} merupakan waktu terjadinya hubungan tertutup (*close circuit*) pada bagian primer dari induktor sehingga mengakibatkan terjadinya beda potensial yang dapat digunakan untuk memodulasi induktor agar menghasilkan arus induksi.

Sedangkan pada keadaan *Off* (Gambar 5.1b) terlihat bahwa pada saat saklar dalam kondisi terbuka maka besarnya V_{Lp} sebanding dengan tegangan keluaran dari rangkaian *Flyback*, serta nilai rasio

lilitan dari transformer tetapi mempunyai polaritas berbeda. Oleh karenanya, besarnya V_{Lp} dapat dituliskan sebagai :

$$V_{Lp} = -Vo/N \qquad (5.3)$$

Sedangkan I_{lp} nilainya sama dengan nol karena tidak ada arus yang mengalirkan pada rangkain dalam kondisi terbuka (*open circuit*).

Tegangan rata-rata yang melintasi induktor dalam keadaan tunak (*steady state*) adalah nol, sehingga untuk mendapatkan fungsi transfer dari rangkaian *Flyback* kita dapat turunkan melalui persamaan 5.1 dan persamaan 5.3.

$$V_{ave} = \frac{1}{Tp}\int_0^{Tp} V(t)dt = 0 \qquad (5.4)$$

berdasarkan kurva dari V_{Lp} (Grafik 5.1a), jumlah luasan V_{Lp} dapat kita peroleh sebagai:

$$V_{ave} = \frac{1}{Tp}\int_0^{ton=DTp}(Vin)dt - \frac{1}{Tp}\int_{ton}^{Tp}(Vo/N)dt = 0$$

$$= \frac{1}{Tp}\left\lfloor Vin\ DT_p - \left(\left\lceil\frac{Vo}{N}T_p\right\rceil - \left\lceil\frac{Vo}{N}DT_p\right\rceil\right)\right\rfloor = 0$$

$$= \frac{1}{Tp}\left\lfloor Vin\ DT_p - \left(\left\lceil\frac{Vo}{N}(T_p - DT_p)\right\rceil\right)\right\rfloor = 0$$

variabel Tp dapat kita hilangkan karena seluruh persamaan mengandung Tp
sehingga

$$V_{ave} = \left\lfloor Vin\ D - \left(\left\lceil\frac{Vo}{N}(1-D)\right\rceil\right)\right\rfloor = 0$$

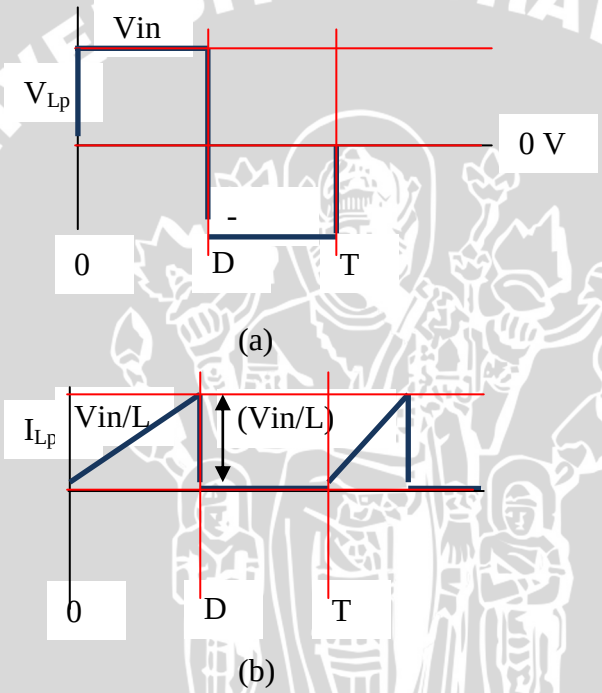
maka

$$Vin\ D = \frac{Vo}{N}(1-D)$$

$$Vout = \frac{ND}{(1-D)}Vin \qquad (5.5)$$

Subtitusikan N sebagai rasio $\frac{Ns}{Np}$, $t_{on} = DT_p$ dan $t_{off} = (1-D)T_p$ maka persamaan 5.5 dapat ditulis sebagai :

$$Vout = \frac{Ns}{Np}\frac{t_{on}}{t_{off}} Vin \qquad (5.6)$$



Grafik 5.1. Grafik tegangan dan arus dari induktor primer *Flyback*:
(a) Tegangan V_{Lp}, (b) Arus I_{Lp}

Perubahan tegangan pada induktor akan menghasilkan arus induksi maka nilai induktansi minimum yang dibutuhkan sebanding dengan arus induksi yang diperlukan, dapat ditulis sebagai:

$$L_{min} = V \frac{\Delta t}{\Delta I} \qquad (5.7)$$

Pada Grafik arus I_{Ls} diatas, diketahui bahwasanya besarnya I_{Ls} sebanding dengan tinggi segitiga dimana:

$$I_{Ls} = \left(\frac{Vin}{L}\right) x\ t_{on}$$

sehingga

$$L = \left(\frac{Vin}{I_{ls}}\right) x\ t_{on} \qquad (5.8)$$

(Anonimous[5])