TR-O-0116 20

DBFセルフビームステアリングアレーアンテナ信号処理部の開発

田中　豊久

1996. 3.15

ATR光電波通信研究所

目次

第１章　はじめに

昨今、世の中は移動通信ばやりである。電車の中、会議中、レストランで携帯端末の呼出音が鳴る。浸透の速度は目覚ましいものがある。人々は一度移動通信の便利さに気が付けばもう、元に戻れないかもしれない。この勢いで市場が需要が伸びれば将来的に周波数が欠乏状態になることは簡単に予測できる。この問題を解決するためには、新たな周波数の開拓、周波数の繰り返し利用、多重化技術の利用による回線の確保などが検討、研究されている。また、将来的には音声通信にとどまらず、画像通信、B-ISDNに代表される広帯域データ通信などの要求が高まってくるであろう。

一方、電子機器分野では、20年程度前からディジタル化の波が押し寄せて来ている。通信も例外ではなく、既にディジタル通信はサービスが開始されている。今後もディジタル化された通信は益々発展して行くものと予想される。

ATRでは設立当時から将来の高機能移動通信用アンテナとして、ディジタル信号処理を利用したDigital Beam Forming Antennaを提案、研究して来た。このアンテナはアンテナ部に円環パッチアンテナとリングアンテナを2層構造にして組み合わせたセルフダイプレクシングアンテナを用い、さらにそれをアクティブアレーアンテナ構成にしている。アナログ部も集積化が期待できるMMICによって構成も可能である。最後に複雑な信号処理を引き受けるディジタル部に並列処理ディジタル信号処理装置(DSP)で構成される、各種の技術の組み合わさった統合化アンテナである。当初ディジタル信号処理部は汎用DSPチップの搭載したボードを複数枚使用して構築されたDSPシステムを用いて研究が行われていた。汎用ボードを組み合わせたシステムでは、複雑な並列処理と相互のデータのやり取りを制御する事や動作速度の限界があり、また汎用システムであるがゆえに筐体も非常に大型にならざるを得なかった。ここで、より小型化、複雑な処理の実現を目指したDSP部のASIC化に白羽の矢が立った。最終的な目標はすべてのDBFアンテナの処理が1チップに集積化されたASICの完成であるが、1チップASICは作成にコストと汎用性を犠牲にする事を要求する。そこで設計がASICと同じ手法が用いられ、設計自体もASICに転用可能なFPGA(Field Programmable Gate Array)を用いたDSPを構築する事になった。FPGAを用いる利点として、ユーザサイトで設計開発が可能な事、ASIC化へのハード規模などの情報が得られる事が挙げられる。逆に、欠点としてはそのプログラミングが可能なアーキテクチャにより動作速度がASICと比べて劣っている。しかしながら実験システムではデータレートを低く抑える事により、その欠点をカバーすることができる。このようなATRでの歴史的背景のなか、FPGAを用いたDSPボードにより、DBFアンテナの実現性を検証する事を目的としたDSP開発が行われた。本研究レポートでは学術論文等では、取り扱わなかった、設計のノウハウを含めてDBFアンテナが作れるようにまた、既存のシステムが理解できるように記述した。DBF技術を用いたFFTマルチビーム生成部、ビームセレクタ部、位相補正部までを参考文献[1-1]に、CMA(Constant modulus Algorithm)を指導原理としたアダプティブプロセッサについてを参考文献[1-2に]、ビームスペースでアレーアンテナで最大比合成を行なうDBFセルフビームステアリング用プロセッサについては本テクニカルレポートで述べる。

第２章　DBF Self-Beam-Steering Array Antenna

移動体通信においては、近年の加入者の急激な増加に加え、データ通信や画像伝送など通信の高品位化、広帯域化の要求により将来の周波数資源の欠乏が予想されている。このような状況の中、将来の周波数有効利用の期待に応えられる高機能アンテナとしてディジタルビームフォーミングアンテナ(DBFアンテナ)が注目されている[2-1]。DBFアンテナはアクティブアレーアンテナを用い、素子アンテナの受信信号、それぞれをアナログ部において合成などの処理を行わず、A/Dコンバータを用いて、ディジタルデータに変換し、柔軟性に富んだディジタル信号処理技術によってマルチビーム生成、到来方向推定、干渉除去[2-2]-[2-8]などの複雑な機能が実現可能なアンテナである。DBFアンテナはレーダ技術より発展し、近年アダプティブアレーを始めとして通信への応用がさかんに研究されている。これまで通信用として実用に至っていない理由として、ディジタルデバイスの性能が十分にその機能を生かすには不十分であったことが挙げられる。しかしながら、今後、デバイス技術の飛躍的な発展とともに実用化の期待がもたれている。
DBFアンテナで実現されるアルゴリズムの１つとして、到来波へ高速に追従し、またアンテナの受信利得を最大限利用する最大比合成をおこなうセルフビームステアリングアレーが提案されている[2-9,10]。セルフビームステアリングアレーは素子ごとのベースバンド信号または、マルチビームを入力信号として、各入力の電力に応じた比率によって合成される。アルゴリズムはフィードバックを用いていないので安定であり、簡単な演算のみで実現でき、ハードウェアの構築に向いている。
本報告では、FFTによるマルチビームフォーマをプリプロセッサとしたセルフビームステアリングアレーを対象とし、移動体通信用に適応可能なハードウェアにFPGA(Field Programmable Gate Array)をASICとして用いての試作について述べる。

第１節　セルフビームステアリングアレーアンテナシステムの概要

アンテナ部は図2-1.に示すように、素子アンテナは4×4の四角形に配置し、16素子の2層構造のマイクロストリップセルフダイプレクシングアンテナを用いている。FFT演算を採用する為、隣接素子アンテナは$\lambda/2$(λ:搬送波の波長)の等間隔に並べられている。搬送波周波数はL帯(1.545GHz)である。またデータレートは16kbpsである。図2.に受信部セルフビームステアリングアレーアンテナの構成図を示す。試作システムはアンテナ部、周波数変換部、フィルター部、A/Dコンバータ部、ディジタル信号処理部からなっている。次に簡単に各ブロックの動作について説明する。各アンテナ素子ごとに受信された信号は周波数変換部(D/C)により、1.545GHzから32kHzのIF信号に変換される。続くフィルタ部にて高調波が除去され、次に32kHzのIF信号はサンプリング周波数128kHzでA/Dコンバータで8bitのディジタル信号に変換される。次に16素子からの受信データはディジタル信号処理部に渡される。ディジタル信号処理部では、まず16本の固定マルチビームを生成し、設定されたレベル以上のビームを選択し後段へ渡す。試作システムではビーム間の落ち込みを改善するために最大4本までのビームまで選択する。つづく、セルフビームステアリング部では選択されたビームに対し最大比合成を行う構成となっている。ビームステアリングに先立って固定ビーム形成を行うと、アレーの効果により、SNRを改善した信号をセルフビームステアリング部への入力として用いる事ができる。特に衛星電波の受信のようなC/Noが低い信号を取り扱う場合有利である。また同相合成を行う場合に基準となる入力のSNRが高められている事も動作の安定性からも利点となる。

表2-1. 開発システム概要

項目	仕様
アンテナ素子数	16 素子
RF周波数	1.545 GHz
IF周波数	32 kHz
サンプリング周波数	128 kHz
マルチビーム生成数	16 本
ビーム選択数	最大 4本
A/Dコンバータの分解能	8bit
DSP部の演算精度	固定小数点 8bit

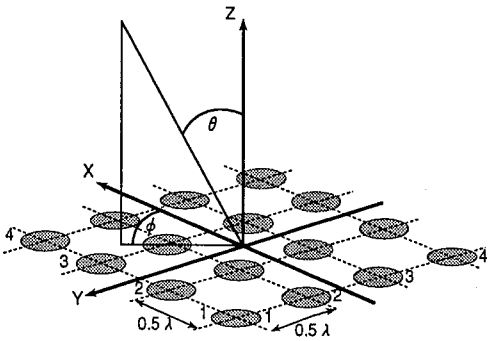


図2-1. アレーアンテナの配置図と座標系

第２節　アナログ部概要

図2-2.にDBFアンテナのブロック図を示す。アナログ部はアンテナ部、周波数変換部、フィルター部、A/Dコンバータ部からなる。各アンテナ素子ごとに受信された信号は周波数変換部(D/C)で、1.545GHzから32kHzのIF信号に変換され、続くフィルタ部で高調波が除去される。次にIF信号はサンプリング周波数128kHzでA/Dコンバータにより8bitのディジタル信号に変換された後、ディジタル信号処理部に渡される。ディジタル信号処理部に関する詳細は以降の章で機能毎に述べる事とする。

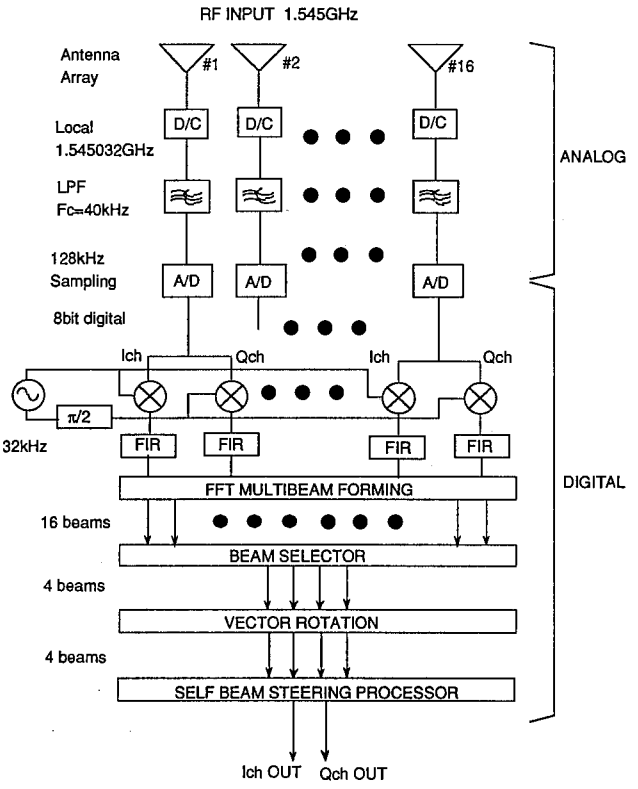


図2-2. DBFセルフビームステアリングアレーアンテナ構成図

第３章　DBF Self-Beam-Steering ASIC Implementation

第１節　DBFセルフビームステアリングアレーアンテナの動作概要 [2-9]

ディジタル信号処理部では、図3-1.に示すように3段階のパイプライン処理によって各サンプル毎にリアルタイムにデータを出力する。3段階の処理はそれぞれ、マルチビームを合成するステージ、ビームを選択し移相を行うステージ、そしてセルフビームステアリングステージである。マルチビームステージでは次の手順で信号処理される。まず素子アンテナの受信信号S_{xy}は、

$$S_{xy}(t_l)=\cos(\omega_0 t_l+\phi_m+\theta_{xy}) \tag{3-1}$$

x=1,2,3,4 y=1,2,3,4

と表わされる。ここでx,yは素子アンテナの座標を表わし(図2-1.参照)、ω_0はIF信号の角周波数、t_lはサンプル時間を表わす。ϕ_mはディジタルQPSKによる変調位相である。θ_{xy}は各アンテナ素子における受信位相の共通のローカル信号からのずれを表わす。まず入力データ式(3-1)は同相成分と直交成分に分ける為、32kHzの固定の位相のローカル信号を掛けられ、式2で表わされる準同期検波される。ローカル信号は4サンプルで1周期する信号で、0、$\pi/2$、π、$3\pi/2$のくり返しである。

$$\begin{bmatrix} i(x,y) \\ q(x,y) \end{bmatrix}=\begin{bmatrix} \cos(n) & 0 \\ 0 & -\sin(n) \end{bmatrix}\begin{bmatrix} S_{xy}(t_l) \\ S_{xy}(t_l) \end{bmatrix} \tag{3-2}$$

x=1,2,3,4　y=1,2,3,4　n=0, $\pi/2$, π, $3\pi/2$.....

次にIch、Qchのデータは式(3-3)に示す10タップのFIRディジタルフィルタを通り、準同期検波時に発生した高調波が取り除かれる。この演算は現在と過去9個の入力I,Qデータに係数を掛ける積和演算である。

$$\begin{bmatrix} i_f(x,y) \\ q_f(x,y) \end{bmatrix}=\sum_{n=0}^{9} h(n)\begin{bmatrix} i_{k-n}(x,y) \\ q_{k-n}(x,y) \end{bmatrix} \tag{3-3}$$

x=1,2,3,4　y=1,2,3,4

i_f及びq_fはそれぞれフィルタリングされたベースバンド信号である。$i_k(x,y)$、$q_k(x,y)$はそれぞれ現在のサンプル時のデータを表わす。次に各ベースバンド信号はビーム合成を行う為、FFT部に渡される。ここでFFTは2次元的に行われる。まず図1.のアンテナの配置と同様に考えれば、式(3-4)で示される様にX軸方向に沿って4素子ずつFFT演算される。

$$\begin{bmatrix} i'(x,y) \\ q'(x,y) \end{bmatrix}=\sum_{n=0}^{3}\begin{bmatrix} \cos(-nx\frac{\pi}{2}) & -\sin(-nx\frac{\pi}{2}) \\ \sin(-nx\frac{\pi}{2}) & \cos(-nx\frac{\pi}{2}) \end{bmatrix}\begin{bmatrix} i_f(n,y) \\ q_f(n,y) \end{bmatrix} \tag{3-4}$$

x=1,2,3,4　y=1,2,3,4

i'とq'はそれぞれ2次元FFTの中間結果を表わす。次にX軸方向に計算されたデータを用いて、式(3-5)に表わされるY軸方向に沿った２次元目のFFT演算が行われる。

$$\begin{bmatrix} I(X,Y) \\ Q(X,Y) \end{bmatrix}=\sum_{n=0}^{3}\begin{bmatrix} \cos(-ny\frac{\pi}{2}) & -\sin(-ny\frac{\pi}{2}) \\ \sin(-ny\frac{\pi}{2}) & \cos(-ny\frac{\pi}{2}) \end{bmatrix}\begin{bmatrix} i'(x,n) \\ q'(x,n) \end{bmatrix} \tag{3-5}$$

x=1,2,3,4　y=1,2,3,4　X=1,2,3,4　Y=1,2,3,4

この結果、同時に図3-2に示す16本のビームのデータが得られる。$I(X,Y)$および$Q(X,Y)$はそれぞれ、合成されたビームを表わし、X、Yはビームの方向に対応している。合成されたビームは、次のサンプルの時に、ビーム選択とアンテナの中心部への移相のステージに渡される。このステージでは15個の振幅比較器を用い、最大4本のビームを選択する。ここでの選択方法は、まずトーナメント式に最も振幅の大きいビームを選びだし、次に選択されたビームを除外して再度比較を行い２番目のビームを選択し、同様に3番目、4番目のビームを選択する方法を用いている。この時、ある基準レベル以下のビームは選択しないようになっている。これは例えば図1で$\theta=0^\circ$、$\phi=0^\circ$の時は、0°ビームのみで受かっており、その他のビームは直交しているのでノイズレベルにあり、入力ビームとしてセルフビームステアリング部での使用に適さないからである。ビームが選択されると式3-6に示されるビーム毎の移相が行われる。

$$\begin{bmatrix} I'_{xy} \\ Q'_{xy} \end{bmatrix} = \begin{bmatrix} \cos(\phi_{xy}) & -\sin(\phi_{xy}) \\ \sin(\phi_{xy}) & \cos(\phi_{xy}) \end{bmatrix} \begin{bmatrix} I_{xy} \\ Q_{xy} \end{bmatrix} \tag{3-6}$$

$$\phi_{xy} = \exp(-j(\frac{3\pi(x-1)}{4}))\exp(-j(\frac{3\pi(y-1)}{4})),$$

x'=0,1,2,3 y'=0,1,2,3

ここでx'=0は図1のアンテナ配置のx=4であり、y'=0 は同様に y=4である。この意味は、x=4およびy=4の時 30° ビームであり、ビームとしてはx=1, y=1の時と対称の方向からであるので、x=4、y=4ではなくx=0、y=0として考慮しなければならないからである。

続いて、セルフビームステアリング部の動作について述べる[2-9,10]。各素子への入力は固定ビームあるいは素子からの準同期検波された復素ベースバンド信号が用いられる。試作システムの場合、受信電力の最も大きなビームを基準入力として用いる。基準入力をX_0、その他の入力をX_nとすると、

$$X_0 = a_0 \exp j(\phi_m+\theta_0) \tag{3-7}$$

$$X_n = a_n \exp j(\phi_m+\theta_0+\delta_n) \tag{3-8}$$

ここで、a_0、a_nは入力ベクトルの振幅、ϕ_mは変調位相、θ_0は共通ローカル信号との位相差、δ_nは基準入力ベクトルとの位相差である。次にX_0の共役復素数とX_nの積により基準入力との位相差が求まる。

$$X_0^* \cdot X_n = a_0 a_n \exp j\delta_n \tag{3-9}$$

さらに式(2)の共役復素数と各入力との積により、各ビームの電力に応じた、かつ基準入力と同相化されたベクトルが求まる。

$$(X_0^* \cdot X_n)^* \cdot X_n = a_0 a_n^2 \exp j(\phi_m+\theta_0) \tag{3-10}$$

さらに、受信機のノイズやバンドパスフィルタなどによる入力の振幅の変動を抑えるため、式(4)にローパスフィルタリングを施している。基準入力ベクトルに同相化された出力Y_n は、

$$Y_n = \sum_{n=0}^{m} W_n \cdot X_n \tag{3-11}$$

$$\cong \tilde{a}_0 \sum_{n=0}^{m} \left[\tilde{a}_n a_n \mathrm{F}\left[\exp j\left(\phi_m+\theta_0\right) \right] \right] \tag{3-12}$$

$$W_n = \mathrm{F}\left[(X_0^* \cdot X_n)^* \right] \tag{3-13}$$

ここで、F[·]はローパスフィルタリングを表わし、また$\tilde{a}_0$、$\tilde{a}_n$はフィルタリングされたベクトルの振幅値である。最終的な出力はY_nを全体の電力で規格化され、その係数αは、

$$\alpha = \sqrt{\sum_{n=0}^{m} |W_n|^2} \tag{3-14}$$

$$\cong \tilde{a}_0 \sqrt{\sum_{n=0}^{m} \tilde{a}_n^2} \tag{3-15}$$

よってセルフビームステアリング部の出力Y'_nは、

$$Y'_n = \frac{Y_n}{\alpha} \tag{3-16}$$

$$= \frac{\sum_{n=0}^{m} W_n \cdot X_n}{\sqrt{\sum_{n=0}^{m} |W_n|^2}} \tag{3-17}$$

$$\cong \frac{\sum_{n=0}^{m}\left[\tilde{a}_n a_n \mathrm{F}\left[\exp j\left(\phi_m+\theta_0\right)\right]\right]}{\sqrt{\sum_{n=0}^{m}\tilde{a}_n^2}} \tag{3-18}$$

ディジタルビームステアリング部のブロック図を図3-3に示す。

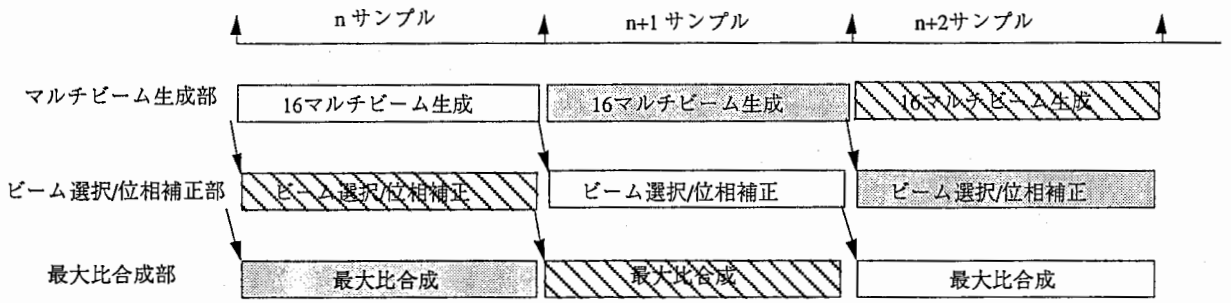


図3-1 ディジタル信号処理部でのパイプライン処理

表3-1 FFT演算のxy軸と各ビームの関係

Y\X	1	2	3	4
1	beam0	beam4	beam8	beam12
2	beam1	beam5	beam9	beam13
3	beam2	beam6	beam10	beam14
4	beam3	beam7	beam11	beam15

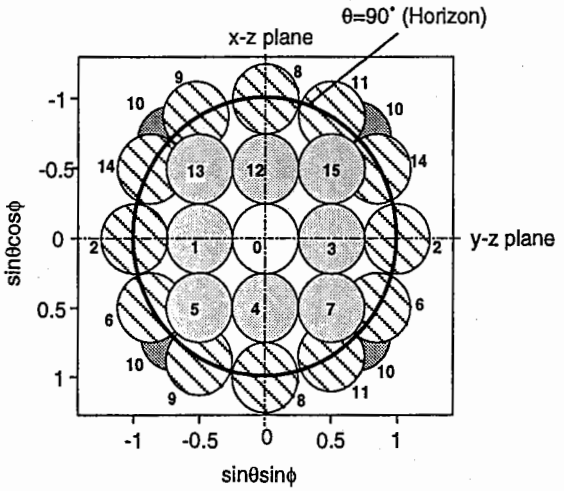


図3-2. ビームの方向とビーム番号

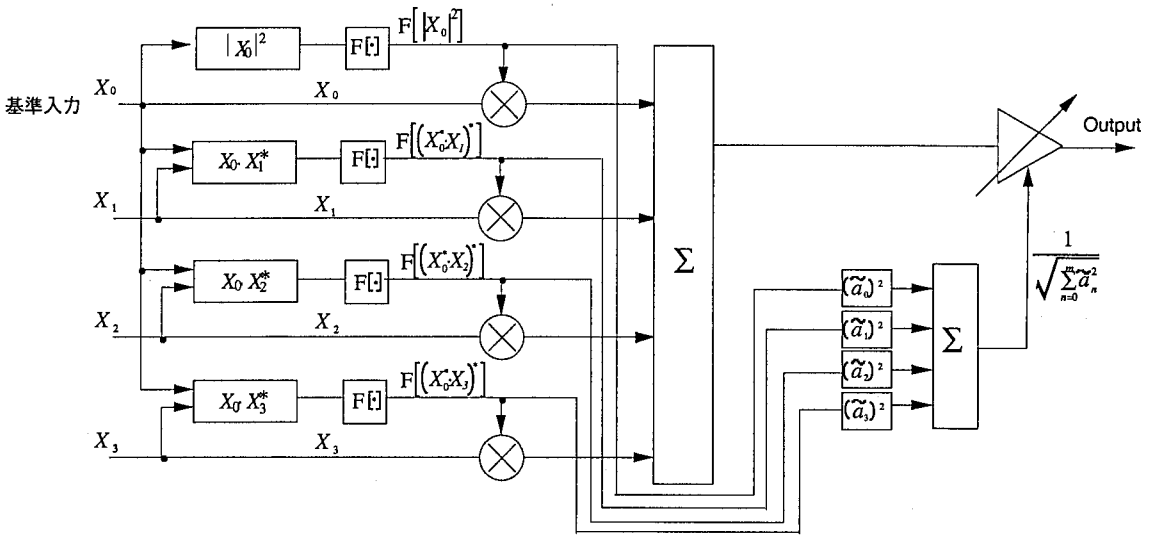


図3-3. セルフビームステアリング部の構成

第2節　DBFセルフビームステアリングアレーアンテナディジタル部概要

試作システムのDSP部はFPGAとよばれる、ユーザサイトで開発および書換え可能な汎用のデバイスを用いている。このFPGAを用いたシステムの利点として、検証された機能をそのままマスクタイプのゲートアレーなどの専用LSIに設計が利用できることが挙げられる。使用したFPGAは、公称25,000ゲート相当のデバイスで、プリント基板 (23.3cm×34.0cm)上に10個搭載されており、25万ゲート相当のASICとして構築可能である。図2-2に示すように試作システムのディジタル部(DSP部)はおもにマルチビーム生成部、ビーム選択部、位相シフト部およびセルフビームステアリング部からなっている[3-1]-[3-4]。マルチビーム生成部では各素子ごとの準同期検波、ナイキストフィルタリング、FFTによる固定マルチビーム生成が行われる。このマルチビーム生成部には8個のFPGAが用いられ、1個あたり2素子分の処理を行う。ビーム選択、位相シフト部、セルフビームステアリング部は残りの2個のFPGA内に構築されている。試作システムではマルチビームを前段で生成し、その中でビーム数を制限するのでセルフビームステアリング部のハードウェア的な負担が軽く、非常にシンプルな構成で実現できた。表3-2に必要となった各部のゲート数を参考値として掲載する。A/Dコンバータに与えられるサンプリング周波数は128kHzであり、1シンボルあたり4サンプルの場合では変調方式をQPSKとした場合、64 kbpsのデータレートに対応できる。A/Dコンバータの分解能は8 bit精度であり、各DSP内のブロックにおける精度も固定小数点8 bit にて演算を行なっている。DSP部の動作クロックは7.04MHzで1積和演算器で1サンプルあたり55回の積和演算が可能である。

表3-2. 構築に必要となったハードウェア規模

機能	必要となったゲート数
マルチビーム生成部	*106,125*
ビーム選択部	*2,889*
位相補正部	*6,107*
最大比合成部	*9,449*
その他 (CPU I/F, etc.)	*3,605*
合計	*128,175*

第3節　セルフビームステアリング処理部の構成

第1項　Components

1.　8bit 固定小数点プロセッサ

セルフビームステアリング部での演算は8-bitの固定小数点精度で行われている。実現できるハードウェア規模に余裕がある場合、スケーリングなどに注意すれば、精度を上げる事も当然できる。

1-1　マルチプライヤ部(Multiplier部)

Multiplier部は2つの8-bitデータを入力とする1の演算を行なう部分と16bitのfull adderから構成されている。DSP内では、特に断らない限り2の補数形式のデータを扱うものとする。

1. A × B = Carry値と演算値
2. Carry値＋演算値

このマルチプライヤ部の1は完全な組み合わせ回路で実現されている。FPGAの開発ツールにはこの部分のlibraryがないので、VHDLで記述しそれを論理合成ツールを用いて回路に変換し用いている。また、特に本FPGAの設計のプロセッサ部の動作速度はこの部分で決まっていると言っても良いと考えられる。これは、Accumulatorのように完全に最適化されたライブラリとして配置配線できないためである。2の部分はLibraryとして供給されている16bit Adderを用いた。

Scaling

DSPの固定小数点演算にはスケーリングは付き物であり、通常、積算の後は結果の上位ビットを用いる。この時に、データの精度を確保するため最適なスケーリングを行なう。例えば、8bit×8bitの演算の結果は16bitデータが得られる、この時に上位の8bit目から15bit目までを結果とするか、7bit目から14bit目までを結果とするかを決定する事である。この部分をアダプティブに制御する方法もあるが簡単のため試作DSPでは固定のスケーリングファクタにした。

Overflow clip

入力信号が大い場合などに、ダイナミックレンジを越えるような計算結果が得られたり、その精度で表現できる数よりも大きくなってしまった場合、オーバーフローになる。オーバーフローになれば符号が反転しその数値も反転してしまう。これを避けるための方法として上位ビットと符号ビットをモニタしそれを越えた場合、最上値あるいは最下値にすればよいが、試作DSPでは簡単なゲートを組み合わせてオーバーフローのモニタを行なっている。

Rounding error

丸め誤差はスケーリングを行なう時に生じる。スケーリング時にただ上位ビットを次段に送った場合、2の補数形式ではマイナスにオフセットがかかってしまう。これを避けるために切り捨てるビットの最上位で0捨1入の処理を行なう。これを実現するテクニックとしては、

1. 切り捨て前のコンポーネントの出力に必要なビット数+1のアダーを配置し、最下位ビットに1を足す。
2. AccumulatorのLoad機能を利用する。

1-2　16 bit Accumulator部

積和演算の和の部分を行なうComponentである。XilinxのlibraryのI6bit Accumulatorをそのまま使用している。各演算グループの開始直前には必ずResetあるいは丸め誤差改善用のデータをAccumulatorにロードしておき使用する。ResetあるいはLoad信号はsequencerから制御される。

1-3　Pipelining

プロセッサ部は1 MCLKですべての処理がなされるのではなく、中間レジスタを用いて3段のパイプライン構成で処理される。パイプライン処理のタイミングを図3-4に示す。試作DSPでのシーケンスについて説明する。システム動作速度を上げる必要があれば、さらに中間レジスタを増やす方法もある。しかし、その分必要となるハードウェア規模も大きくなる。

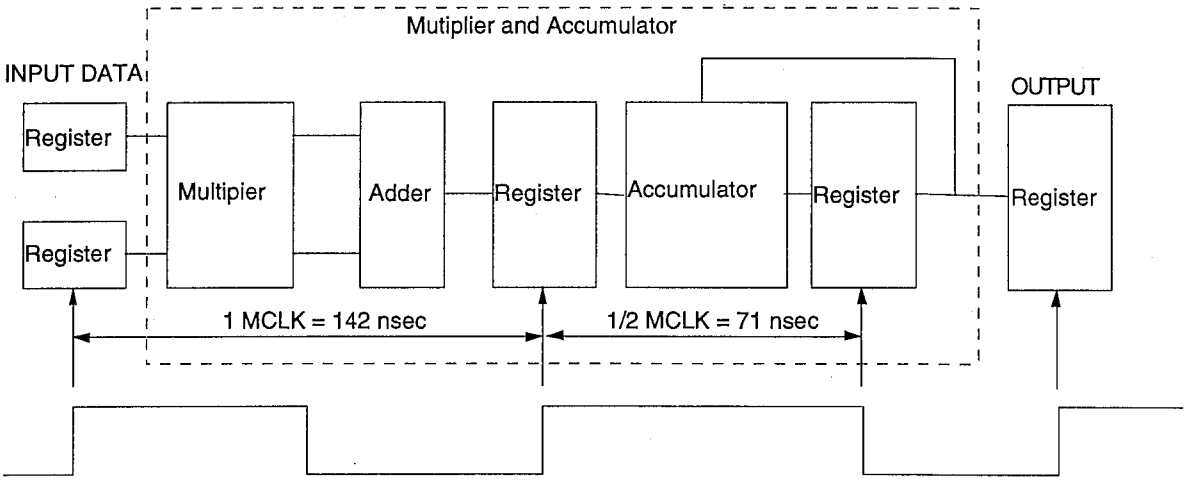


図3-4. プロセッサ部のパイプライン処理

1. MCLKの立上がりエッジでプロセッサ直前の入力レジスタに2つのデータがラッチされる。
2. 次のMCLKの立上がりまでの間に積算を完了させる。
3. 積算結果はMCLKの立上がりエッジで、中間レジスタにラッチされる。
4. その結果はAccumulatorの入力に渡され、Accumulator内のレジスタの数値と和減算が行なわれる。
5. 4の積和結果はMCLKの立ち下がりエッジでAccumulator内のレジスタにラッチされる。

以上のシーケンスで、1回の積和演算が1.5MCLKで完了する。パイプライン処理は演算の速度を向上させる事ができるが、中間レジスタによって回路規模の増大になり、また直前の演算結果を用いて次の演算を行なう処理などは待ち時間が生じてしまう欠点がある。逆に言えば、FIRフィルタのように連続して積和演算を行なうようなアルゴリズムは効率がよい。最終的な結果は5の演算完了後のMCLKの立上がりエッジで出力用のレジスタにラッチする。

2.　Output & intermidiate register

これは、Accumulatorからの出力を1時的に保持するためのレジスタである。出力レジスタにはセルフビームステアリング部出力のIch、Qch用のレジスタ(in_ofc)と中間レジスタの役割をするレジスタ(mrc_reg8)があり、規格係数を決定するウェイト電力積算値を保持するレジスタ(reg8ofc)がある。これらにはオーバーフロー防止用の回路が付加されている。ラッチのトリガ信号は各sequencerから送り出される。トリガ信号の遅延量の影響を避けるため、レジスタのトリガイネーブル端子にトリガ信号は与えられ、トリガイネーブル状態でレジスタのクロック入力端子に繋がれたClock信号のエッジでラッチは行われる。

3. Multiplexer

入力切り替えのためのコンポーネントであり、制御信号は各sequencerで発生される。またMultiplexerの替わりにトライステートのバッファとバスを用いて入力切り替えはできる。試作DSPでは使用できるBUSの本数に制限があったので、BUSの必要な回路部分(CPU I/Fのデータバスなど)を優先して設計したため、プロセッサへの入力はMultiplexerを用いている。

4. Input register

プロセッサ直前のレジスタや、外部からのデータを保持するレジスタの事を指す。

5. Channel Register/Threshold Register

ビーム選択シーケンスにおいて、電力値の比較が行われるが、この時、ビーム番号と設定スレッシュレベルより大きいかという情報も同時に流される。これらを保持するためのレジスタである。

7. Wn RAM

セルフビームステアリング処理のウェイトを保持するために全チャンネル用にRAMを用いた。4ビーム分だけでは、ビーム選択の順序が入れ代わるとウェイトとビームの関係も入れ代わってしまい不具合を生じる。これをビームとウェイト値を対応させるためRAM方式を用いた。

6. Combination Register (comb_reg)

ウェイト保持のための全チャンネル用にRAMを用いており、選択されないとウェイと値は前回のものが残ってしまう。前回選択されていないビームが選択された時、ウェイト値は0でなければならない。このリセット機能を実現するためのレジスタである。前回選択ビームを記憶し、今回選択のビーム番号と比較する。もし、前回選択されていないビームの場合、リセット用のステータスを出力する。

8. Alpha ROM

ウェイトの電力積算値をアドレスとして、その値の平方根分の1の値を保持しているRead Only Memoryである。ハードウェア量を抑えるため試作回路ではアドレスの深さ32のROMを使用した。つまり複数の入力アドレスに対し同じデータを出力する。平方根分の1の値は電力値が大きくなるとほぼ一定になっており、ウェイト電力値が小さくなるほど差が大きくなる。したがってレベルが十分高ければ同じ係数を用いる事になる。

9. Slave sequencer

セルフビームステアリング部のコントロールを行なうのはss_u29p2というシーケンサである。このシーケンサで取り扱う制御信号は、

1.セルフビームステアリング部の各種マルチプレクサの切り替え信号
2. 各レジスタへのラッチトリガ信号
3. ウェイト用RAMのライトイネーブル信号
4. プロセッサ部へのコントロール信号(加減算コントロール、リセットコントロール)

などである。具体的な信号名とコントロールタイミングについては、後述するTiming Chartの項で述べる。

第２項 Self-Beam-Steeringのためのハードウェア構成

図3-5にセルフビームステアリング部のハードウェア構成を示す。8bit 積和演算器1個とウェイトのフィルタリングのための16bit 積算器1個、データセレクタ、レジスタ、ウェイトを保持するためのRAM、コントローラなどで構成されている。式(3-14)、(3-15)で表わされた規格化定数の演算には、平方根の中、すなわち各ウェイトの二乗の積算値に応じた値をROMに書かれたルックアップテーブルを参照する方式を用いている。各ウェイト値をローパスフィルタリングを行なう部分は一次のIIRフィルタで構築した。ウェイトは各ビームごとにRAMに書き込まれ、選択されたビームが替った場合、新しく選択されたビームのウェイトは0に初期化されて用いられる。基準となるビームが替った場合、急激なウェイトの変動を避け、位相飛びが起きないようにそのまま前回のウェイトを用い、フィルタリングにより滑らかに値が更新される。図6にセルフビームステアリング部における処理のフローを示す。演算の手順は、おおまかに3つに分けられる。

1. 入力ベクトルとウェイトの積和演算と選択ビームの全電力での規格化で得られる出力の計算。
2. ウェイトベクトルの更新の演算。
3. 出力規格化のため、フィルタリングされたウェイトの二乗を積算し、その値をアドレスとして平方根分の1の値をルックアップテーブルROMから読み出す。

以上を1サンプル時間に行うことにより、リアルタイム処理が可能となる。

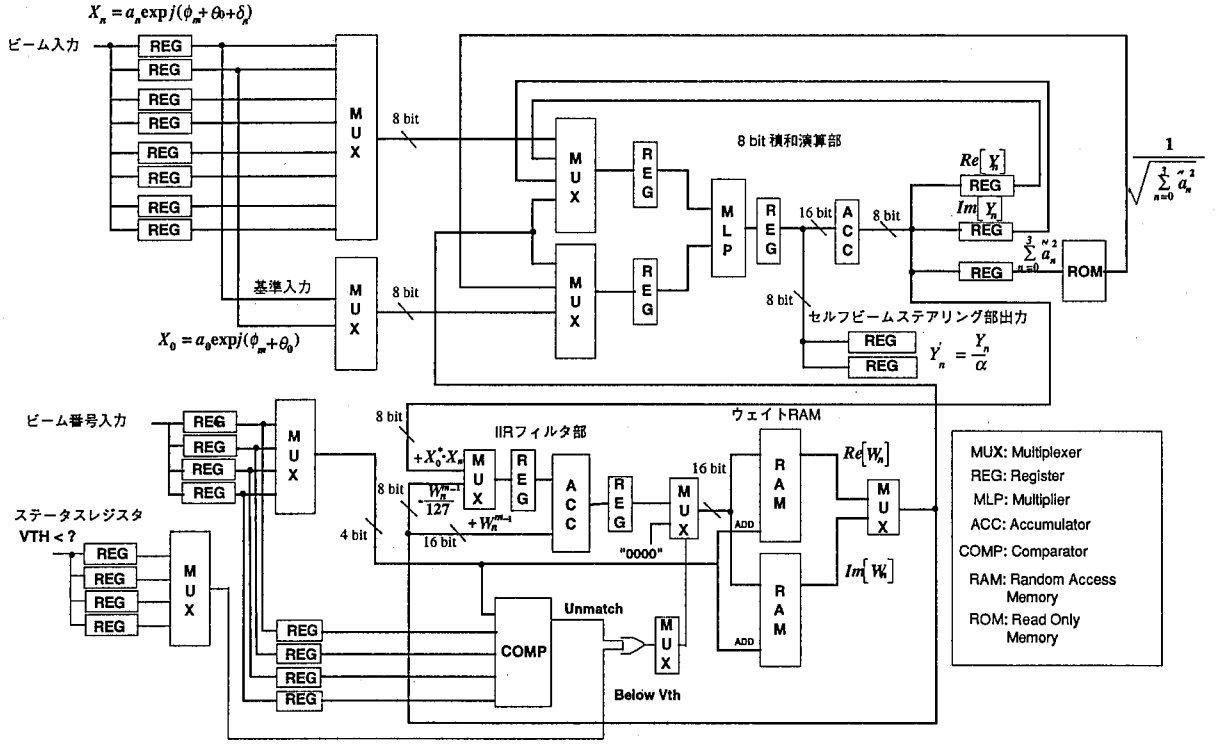


図3-5. セルフビームステアリング部の構成図

入力ベクトルが設定敷居値以下の場合の動作は、図3-6の回路で実現される。電力値比較のステージにおいて同時に設定敷居値との比較も行なっていると述べたが、その情報は4bitレジスタに保持される。次のウェイト更新時に設定敷居値以下のビームの場合、MUXで更新されたビームのウェイトの替わりに"00H"の方がWn RAMに対して出力されるようになる。

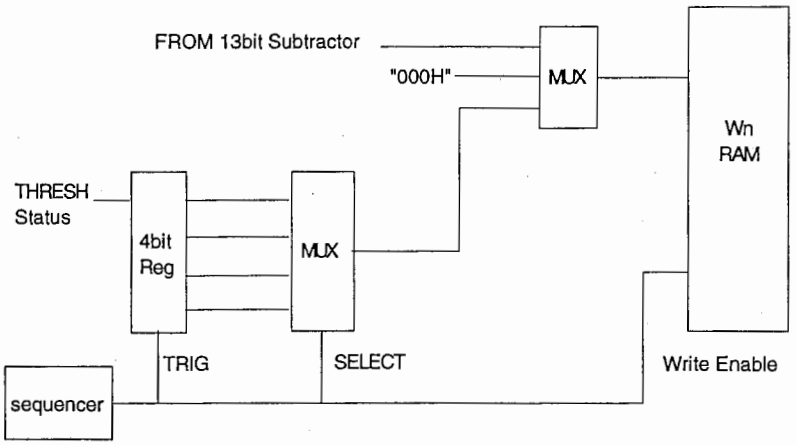


図3-6. Thresh levelの取り扱い

第３項　セルフビームステアリングアルゴリズム概要とTiming Chart

セルフビームステアリングの演算のフローを図3-7に示す。まず、選択されたビームベクターとウェイトベクターを積算しセルフビームステアリングの出力を求める。実数部、虚数部の順序で演算を行なう。さらに前回のウェイト電力値の平方根分の一のデータをROMから引出し、その値を実部、虚部それぞれに掛ける。これが、最終出力となる。この計算はマルチプライヤのみで行われ、Accumulatorは用いられないので、演算の区切りのAccumulatorリセットの待ち時間を利用して行なっている。
続いて、ウェイトベクターの更新の演算に移る。ここでは、リアルタイム処理を実現するため、並列して1次のIIRフィルタリングを行なっている。具体的にはマルチプライヤ-アキュムレータでリファレンスベクトルと入力ベクトルの復素掛け算を行なう、その時にアキュムレータから出力されるデータをウェイトの更新値としてIIRフィルタを構築している16bitアキュムレータに渡す。1次のIIRフィルタリングの演算は前回のウェイト値に1以下の係数を掛け、今回演算された更新分と足し合わせるだけである。1以下の係数を前ウェイトに掛ける場合、マルチプライヤが必要であるが、16bitアキュムレータだけで行なう方法として、

$$W_n^m = X_0^* \cdot X_n + \beta W_n^{m-1} \tag{3-19}$$

$$= X_0^* \cdot X_n + (1-\delta) W_n^{m-1} \tag{3-20}$$

のように、β を (1-α) に置き換えて、αを引くという演算にしている。ここでαの値として、ウェイトRAMの上位bitをシフトして用いている。7bitシフトして127分の1の値を用いるとカットオフ周波数は約200Hzでになる。各ウェイトについて更新を終了すれば、規格化係数を求めるためのウェイト値の電力積算を行なう。この結果はレジスタに保持され、係数ROMのアドレス入力値となる。

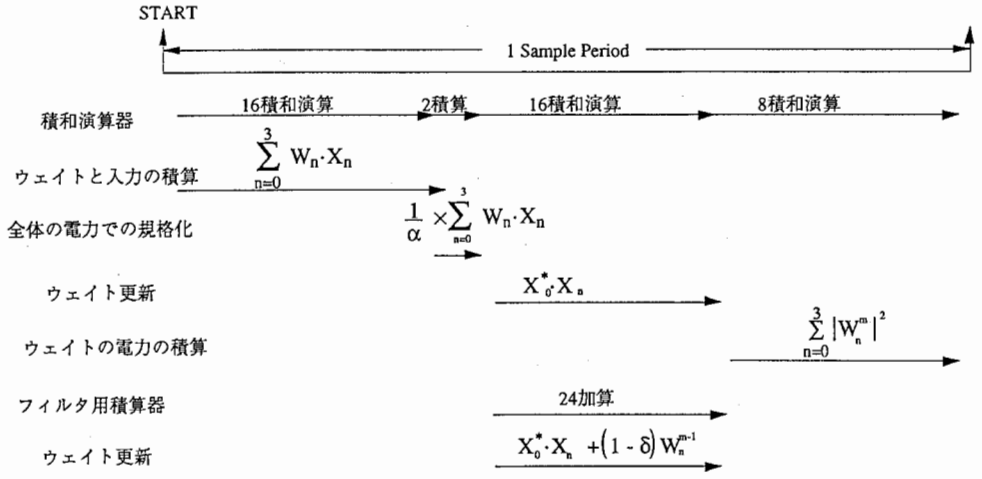


図3-7. セルフビームステアリング部の演算フロー

セルフビームステアリング処理部のTiming Chartを付録に添付する。Timing Chartでは各sequencerからの制御信号のTimingと各サンプリング周期の間に実行する演算の制御の為のフローを示している。Timing Chartに示されている制御信号について説明する。

u9のss_u29p3によるコントロール

RAW_MUX　入力ベクトルデータの出力切り替えコントロール
WNR_MUX　ビーム番号を保持しているレジスタで、実数部の演算時の出力切り替えのコントロール用
WNI_MUX　ビーム番号を保持しているレジスタで、虚数部の演算時の出力切り替えのコントロール用
A_MUX　プロセッサへの入力切り替えコントロール
B_MUX　プロセッサへの入力切り替えコントロール
WN_RAM_WE　ウェイトRAMの書き込みイネーブル信号
YN_REG_TRIG_EN　y_nを保持するレジスタへのトリガ信号
RAW_TRIG　入力レジスタに保持するためのトリガ信号
CHREG_TRIG　ビーム番号保持用レジスタへのトリガ信号
OUT_TRIG_EN　中間レジスタに保持するためのトリガ信号
ACC_LOAD　Accumulatorへの丸め誤差補正用データのロードを行なう。(Accumulatorの初期化)
AS_CONT　Accumulatorの演算で加算か減算を制御するための信号
ACC_EN　Accumulatorの初期化を行なう
COMP_TRIG　ビーム番号を保持するレジスタへのトリガ信号
XR_IQ_SEL　リファレンスビームのIch、Qchの出力切り替えのコントロール用
WN_IQ_SEL　ウェイト実数部用RAMか虚数部用RAMの出力切り替えコントロール
IIR_ACC_LOAD　IIRフィルタ用のAccumulatorに前ウェイト値を読み込むための信号
IIR_AS_CONT　IIRフィルタ用のAccumulatorの演算で加算か減算を制御するための信号
IIR_ACC_EN　IIRフィルタ用のAccumulatorの初期化を行なう
IIR_SEL　IIRフィルタ用のAccumulatorへの入力切り替えコントロール
IIR_ACC_REG_TRIG　IIRフィルタ用のAccumulatorで求められたウェイト更新値を保持するレジスタへのトリガ信号
WN_MON_TRIG　CPU I/F経由でウェイト値を読みだすためのデータ保持用レジスタへのトリガ信号

第４章　むすび

ASICを用いたSelf-Beam-Steering処理部の設計・試作について述べた。試作は8bit固定小数点精度で構築したが、さらに高精度化をはかれば安定な動作が可能であると思われる。またカスタムLSIを適応すればさらに大規模な構成にでき、また微細化技術による高速化も可能である。

第５章　謝辞

日頃より御指導頂く、ATR光電波通信研究所 猪股英行社長に深く感謝いたします。また本研究を進めるにあたり正しい方向へお導き頂いた、唐沢好男室長、千葉　勇元主任研究員、三浦　龍主任研究員に深く感謝いたします。また、無線通信第一研究室の皆様に感謝いたします。

第６章　参考文献

[1-1] 田中豊久, "DBFマルチビームアンテナディジタル信号処理部の開発," ATRテクニカルレポート, TR-O-0114 (TR-O-0117、設計データソフト付き), Mar. 1996.

[1-2] 田中豊久, "BSCMAアダプティブアレーアンテナディジタル信号処理部の開発," ATRテクニカルレポート, TR-O-0115 (TR-O-0118、設計データソフト付き), Mar. 1996.

[2-1] H. Steyskal, "Digital Beamforming Antennas," Microwave Journal, Jan. 1987, pp107-114.

[2-2] K. Takao and K. Uchida, "Beamspace Partially adaptive antenna," IEE Proc., Pt. H, 6, pp.439-444 Dec. 1989.

[2-3] 藤元美俊、菊間信良、稲垣直樹、"マルカート法を用いたCMAアダプティブアレーの多重波抑制特性," 信学論(B-II)、Vol.J74-B-II, No.11, pp.599-607, Nov. (1991).

[2-4] T. Ohgane, T. Shimura, N. Matsuzawa and H. Sasaoka, "An implementation of a CMA adaptive array for high speed GMSK transmission in mobile communications," IEEE Trans. Veh. Technol., vol. 42, pp. 282-288, Aug. 1993.

[2-5] Y. Ogawa, Y. Nagashima and K. Itoh, "An Adaptive Antenna System for High-Speed Digital Mobile Communications," IEICE Trans. Commun., Vol. E75-B, No.5 May 1992.

[2-6]黒岩　登、河野隆二, "アダプティブアレーアンテナによる指向性ダイバーシチ受信の構成法," 信学論(B-II)、Vol.J73-B-II No.11, pp755-763, Nov. (1990).

[2-7] T. Ohgane, "Spectral Efficiency Improvement by Base Station Antenna Pattern Control for Land Mobile Cellular Systems," IEICE Trans. Commun., VOL E77-B, No. 5 May (1994).

[2-8] 千葉　勇、中條　渉、藤瀬　雅行: "ビームスペースCMAアダプティブアレーアンテナ,"信学論(B-II) Vol J77-B-II, No.3, pp.130-138, Mar. 1994.

[2-9] 三浦　龍, 田中豊久, 唐沢好男, "マルチパスのダイバーシチ合成を行なうディジタルセルフビームステアリングアレー," 信学技報 AP95-18, May. 1995

[2-10] 堀江　章夫, 三浦　龍, 唐沢好男, "ビームスペースで最大比合成受信を行なうディジタルセルフビームフォーミングアンテナの追尾特性," 信学技報 AP94-44, Aug. 1995

[3-1] 大滝幸夫, "移動体衛星通信用DBFアンテナ信号処理部の構成とその特性," ATRテクニカルレポート, TR-O-0046, Jun. 1992.

[3-2] 田中豊久, 三浦　龍, 千葉　勇, 唐沢好男、"ASICを用いたDBFマルチビームアンテナの開発," 信学論(B-II), Vol.J78-B-II, no.9, pp602-610, Sep. 1995.

[3-3] T. Tanaka, R. Miura, I. Chiba, and Y. Karasawa, "An ASIC Implementation Scheme to Realize a Beam Space CMA Adaptive Array Antenna," IEICE Trans. commun., vol. E78-B, no.11, pp. 1467-1473, Nov. 1995.

[3-4] 田中豊久, 三浦　龍, 唐沢好男、"DBFセルフビームステアリングアレーアンテナの信号処理部の開発," 信学技報, RCS95-112(1996-01)

第７章　付録

第１節　DSP ASIC Bd 回路図

第１項　u9

sheet1　Beam Selector

sheet2　Self-Beam-Steering processor MAIN Block

sheet3　FPGA I/O ビーム電力データ(u1、u2)

sheet4 FPGA I/O ビーム電力データ(u3、u4)
sheet5 FPGA I/O ビーム電力データ(u5、u6)
sheet6 FPGA I/O ビーム電力データ(u7、u8)
sheet7 FPGA I/O MRC出力
sheet8 FPGA I/O ビーム電力出力Control信号
sheet9 FPGA I/O MRC出力モニタ用
sheet10 FPGA I/O Control信号、Beam最下位ビットアドレス、選択ビームアドレス
sheet11 FPGA I/O CPU I/F データバス
sheet12 CPU I/F(アドレスデコーダ、R/W、ライトパルス、リードイネーブル信号発生)
sheet13 CPU I/F(ライトレジスタ)
sheet14 CPU I/F(リードレジスタ)
sheet15 FPGA I/O MRC出力モニタ用
sheet16 FPGA I/O MRC出力モニタ用
sheet17 FPGA I/O MRC出力モニタ用
sheet18 CPU I/F(リードレジスタ)

第 2 項 Components
mux82_1
mux84_1
ss_comp4
mux88_1
reg8
row_reg83
reg44
mux44_1
thr_reg
comb_reg
shifter83
mux3_2_1
alpha_rom
ss_u29p2
mrc_reg8
FDRS
reg8ofc
ofc2
in_ofc
x74_374

第 2 節 Misc Information

第 1 項 Timing Chart

第 2 項　ALPHA ROM Data

Address	DATA\BIT	7	6	5	4	3	2	1	0
0	127	0	1	1	1	1	1	1	1
1	64	0	1	0	0	0	0	0	0
2	45	0	0	1	0	1	1	0	1
3	37	0	0	1	0	0	1	0	1
4	32	0	0	1	0	0	0	0	0
5	28	0	0	0	1	1	1	0	0
6	26	0	0	0	1	1	0	1	0
7	24	0	0	0	1	1	0	0	0
8	22	0	0	0	1	0	1	1	0
9	21	0	0	0	1	0	1	0	1
10	20	0	0	0	1	0	1	0	0
11	19	0	0	0	1	0	0	1	1
12	18	0	0	0	1	0	0	1	0
13	18	0	0	0	1	0	0	1	0
14	17	0	0	0	1	0	0	0	1
15	16	0	0	0	1	0	0	0	0
16	16	0	0	0	1	0	0	0	0
17	15	0	0	0	0	1	1	1	1
18	15	0	0	0	0	1	1	1	1
19	15	0	0	0	0	1	1	1	1
20	14	0	0	0	0	1	1	1	0
21	14	0	0	0	0	1	1	1	0
22	14	0	0	0	0	1	1	1	0
23	13	0	0	0	0	1	1	0	1
24	13	0	0	0	0	1	1	0	1
25	13	0	0	0	0	1	1	0	1
26	12	0	0	0	0	1	1	0	0
27	12	0	0	0	0	1	1	0	0
28	12	0	0	0	0	1	1	0	0
29	12	0	0	0	0	1	1	0	0
30	12	0	0	0	0	1	1	0	0
31	11	0	0	0	0	1	0	1	1
ROM CODE		0	3	1D	1FFE1	FFFE00E5	7FFE072D	807E3941	838E4A0D

第 3 節　VHDL Listing

第 1 項　ss_u29p2 VHDL Listing

```
-- Copyright(c) 1995 ATR Optical and Radio Communications Research Laboratories
-- All rights reserved
-- Written by Toyohisa Tanaka @Radio Communications Department
-- Rev.A Thu, October, 5, 1995
--
-- Slave Sequencer for MRC
--
-- Funtion
--
-- 1. ROW MUX(2:0) Control
-- 2. ROW_TRIG(3:0) Control
-- 3. XR_IQ_MUX Control
-- 4. WN ADD R (1:0)Control
-- 5. WN ADD I (1:0)Control
-- 6. WE(1:0) for Wn RAM
```

```
-- 7. A_MUX(1:0) Control
-- 8. B_MUX(1:0) Control
-- 9. OUT_TRIG(2:0) Control
-- 10. YN_TRIG(1:0) Control
-- 11. ACC_EN, ACC_EN, AS_CONT
-- 12. IIR_ACC_EN, IIR_ACC_EN, IIR_AS_CONT
-- 13. IIR_MUX Control
-- 14. Wn Trig for monitor

-- b. IN Trig EN for Channel registers
-- c. Wn MUX for I, Q

-- MRST and START are active low signal in this module.
-- MCLK2 has been eliminated due to minimize clock skew.
-- OUT_EN and FIR_SEL are proceeded 1/2 MCLK, because of timing optimization.

library mgc_portable;
use mgc_portable.qsim_logic.all;

-- Slave Sequencer Entity Description

entity ss_u29p2 is
  port(
     MCLK, START, MRST: in qsim_state;
       RAW_MUX: out qsim_state_vector(2 downto 0);
       WNR_MUX, WNI_MUX, A_MUX, B_MUX, WN_RAM_WE, YN_REG_TRIG_EN: out
qsim_state_vector(1 downto 0);
       RAW_TRIG, CHREG_TRIG: out qsim_state_vector(3 downto 0);
       OUT_TRIG_EN: out qsim_state_vector(2 downto 0);
       ACC_LOAD, AS_CONT, ACC_EN, COMP_TRIG, XR_IQ_SEL, WN_IQ_SEL: out qsim_state;
       IIR_ACC_LOAD, IIR_AS_CONT, IIR_ACC_EN, IIR_SEL, IIR_ACC_REG_TRIG: out qsim_state;
  WN_MON_TRIG: out qsim_state_vector(7 downto 0)
    );
end ss_u29p2;

-- sseq_u29 Architecture Description

architecture rtl of ss_u29p2 is

-------------------------------------------------------------------
-- Signal declaration for STATE GENERATOR
-------------------------------------------------------------------
     signal start2: qsim_state := '1';
-------------------------------------------------------------------
-- Signal declaration for MCLK sensitive signal
-------------------------------------------------------------------
-------------------------------------------------------------------
-- Signal declaration for MCLK2 sensitive signal
```

```
-------------------------------------------------------------

        signal count, count2, istate: integer range 0 to 63 :=0;

        signal mux_sel1: qsim_state_vector(2 downto 0);
        signal mux_sela, mux_selb: qsim_state_vector(1 downto 0);
        signal mux_selr, mux_seli: qsim_state_vector(1 downto 0);
        signal mux_sel2, mux_sel3, mux_sel4: qsim_state;

        signal trig1, trig2: qsim_state_vector(3 downto 0);
        signal trig3: qsim_state_vector(2 downto 0);
        signal trig4: qsim_state_vector(1 downto 0);
        signal trig5, trig6: qsim_state;
    signal trig7: qsim_state_vector(7 downto 0);

        signal we1: qsim_state_vector(1 downto 0);
        signal en1, en2, adsu1, adsu2, ld1, ld2: qsim_state;

        type type2_states is (s00, s01);
        type type3_states is (s10, s11, s12);
        type type4_states is (s20, s21, s22, s23);
        type type5_states is (s30, s31, s32, s33, s34);
        type type8_states is (s0, s1, s2, s3, s4, s5, s6, s7);
        type type9_states is (s40, s41, s42, s43, s44, s45, s46, s47, s48);

        signal present_mux1_state  : type8_states := s0;
        signal next_mux1_state     : type8_states := s1;
        signal present_mux2_state  : type2_states := s00;
        signal next_mux2_state     : type2_states := s01;
        signal present_mux3_state  : type2_states := s00;
        signal next_mux3_state     : type2_states := s01;
        signal present_mux4_state  : type2_states := s00;
        signal next_mux4_state     : type2_states := s01;

        signal present_muxa_state  : type4_states := s20;
        signal next_muxa_state     : type4_states := s21;
        signal present_muxb_state  : type4_states := s20;
        signal next_muxb_state     : type4_states := s21;
        signal present_muxr_state  : type4_states := s20;
        signal next_muxr_state     : type4_states := s21;
        signal present_muxi_state  : type4_states := s20;
        signal next_muxi_state     : type4_states := s21;

        signal present_trig1_state : type5_states := s30;
        signal next_trig1_state    : type5_states := s31;
        signal present_trig2_state : type5_states := s30;
        signal next_trig2_state    : type5_states := s31;
        signal present_trig3_state : type4_states := s20;
```

```
        signal next_trig3_state    : type4_states := s21;
        signal present_trig4_state : type3_states := s10;
        signal next_trig4_state    : type3_states := s11;
        signal present_trig5_state : type2_states := s00;
        signal next_trig5_state    : type2_states := s01;
        signal present_trig6_state : type2_states := s00;
        signal next_trig6_state    : type2_states := s01;
        signal present_trig7_state : type9_states := s40;
        signal next_trig7_state    : type9_states := s41;

        signal present_we1_state   : type3_states := s10;
        signal next_we1_state      : type3_states := s11;

        signal present_en1_state   : type2_states := s00;
        signal next_en1_state      : type2_states := s01;
        signal present_en2_state   : type2_states := s00;
        signal next_en2_state      : type2_states := s01;

        signal present_adsu1_state : type2_states := s00;
        signal next_adsu1_state    : type2_states := s01;
        signal present_adsu2_state : type2_states := s00;
        signal next_adsu2_state    : type2_states := s01;
        signal present_ld1_state   : type2_states := s00;
        signal next_ld1_state      : type2_states := s01;
        signal present_ld2_state   : type2_states := s00;
        signal next_ld2_state      : type2_states := s01;

begin

 START2_PROCESS:process(MRST, MCLK)
 begin
  if ( MRST = '1') then
      start2 <= '1';
  elsif ( MCLK'event and MCLK = '0' and MCLK'last_value = '1') then
      start2 <= START;
  end if;
 end process START2_PROCESS;

 INCREMENT_PROCESS:process(istate)
 begin
  count <= istate;
 end process INCREMENT_PROCESS;

 INIT_SEQ:process(START,MCLK) -- incremented by MCLK2 = count
 begin
    if ( START = '0' ) then -- Initializing local phase counters
       istate <= 0;
    elsif (MCLK'event and (MCLK = '0') and (MCLK'last_value = '1')) then
```

```
            istate <= count +1;
    end if;
 end process INIT_SEQ;

 INIT_SEQ2:process(start2,MCLK) -- incremented by MCLK = count2
 begin
    if ( start2 = '0' ) then
          count2 <= 0;
    elsif (MCLK'event and (MCLK = '1') and (MCLK'last_value = '0')) then
          count2 <=  count;
    end if;
 end process INIT_SEQ2;

-------------------------------------------------------------------------
-- MUX CONTROL
-------------------------------------------------------------------------
-------------------------------------------------------------------------
-- ROW_MUX CONTROL
-------------------------------------------------------------------------
 MUX1_STATE_DECODE:process(count)
 begin
 next_mux1_state <= s0;
 case count is
  when 1 | 11 | 19 | 22 => next_mux1_state <= s0;  -- **************************
  when 2 | 10 | 20 | 23 => next_mux1_state <= s1;  -- **************************
  when 3 | 13 | 25 | 28 => next_mux1_state <= s2;  -- **************************
  when 4 | 12 | 26 | 29 => next_mux1_state <= s3;  -- **************************
  when 5 | 15 | 31 | 34 => next_mux1_state <= s4;  -- **************************
  when 6 | 14 | 32 | 35 => next_mux1_state <= s5;  -- **************************
  when 7 | 17 | 37 | 40 => next_mux1_state <= s6;  -- **************************
  when 8 | 16 | 38 | 41 => next_mux1_state <= s7;  -- **************************
  when OTHERS => next_mux1_state <= s0;
 end case;
 end process MUX1_STATE_DECODE;

-------------------------------------------------------------------------
 MUX1_OUTPUT_DECODE:process(present_mux1_state)
 begin
  case present_mux1_state is
   when s0 => mux_sel1 <= "000";
   when s1 => mux_sel1 <= "001";
   when s2 => mux_sel1 <= "010";
   when s3 => mux_sel1 <= "011";
   when s4 => mux_sel1 <= "100";
   when s5 => mux_sel1 <= "101";
   when s6 => mux_sel1 <= "110";
   when s7 => mux_sel1 <= "111";
   when others => mux_sel1 <= "000"; -- Select A
```

```
 end case;
 end process MUX1_OUTPUT_DECODE;

-------------------------------------------------------------------------------
-- WNR_MUX CONTROL
-------------------------------------------------------------------------------
 MUXR_STATE_DECODE:process(count)
 begin
 next_muxr_state <= s20;
 case count is
  when 3 | 4 | 12 | 13 | 25 | 26 | 27 | 28 | 29 | 30 | 44 => next_muxr_state <= s21;  -- ******
  when 5 | 6 | 14 | 15 | 31 | 32 | 33 | 34 | 35 | 36 | 45 => next_muxr_state <= s22;  -- ******
  when 7 | 8 | 16 | 17 | 37 | 38 | 39 | 40 | 41 | 42 | 46 => next_muxr_state <= s23;  -- ******
  when OTHERS => next_muxr_state <= s20;
 end case;
 end process MUXR_STATE_DECODE;

-------------------------------------------------------------------------------
 MUXR_OUTPUT_DECODE:process(present_muxr_state)
 begin
  case present_muxr_state is
   when s21 => mux_selr <= "01";
   when s22 => mux_selr <= "10";
   when s23 => mux_selr <= "11";
   when others => mux_selr <= "00"; -- Select No. 1 channel
  end case;
 end process MUXR_OUTPUT_DECODE;

-------------------------------------------------------------------------------
-- WNI_MUX CONTROL
-------------------------------------------------------------------------------
 MUXI_STATE_DECODE:process(count)
 begin
 next_muxi_state <= s20;
 case count is
  when 3 | 4 | 12 | 13 | 28 | 29 | 30 | 31 | 32 | 33 | 48 => next_muxi_state <= s21;  -- *****
  when 5 | 6 | 14 | 15 | 34 | 35 | 36 | 37 | 38 | 39 | 49 => next_muxi_state <= s22;  -- *****
  when 7 | 8 | 16 | 17 | 40 | 41 | 42 | 43 | 44 | 45 | 50 => next_muxi_state <= s23;  -- *****
  when OTHERS => next_muxi_state <= s20;
 end case;
 end process MUXI_STATE_DECODE;

-------------------------------------------------------------------------------
 MUXI_OUTPUT_DECODE:process(present_muxi_state)
 begin
  case present_muxi_state is
   when s21 => mux_seli <= "01";
   when s22 => mux_seli <= "10";
```

```
   when s23 => mux_seli <= "11";
   when others => mux_seli <= "00"; -- Select No. 1 channel
  end case;
 end process MUXI_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- A_MUX CONTROL
-------------------------------------------------------------------------
 MUXA_STATE_DECODE:process(count)
 begin
 next_muxa_state <= s20;
 case count is
   when 18 => next_muxa_state <= s21;  -- ***********
   when 21 => next_muxa_state <= s22;  -- ***********
   when 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 | 51 | 52 => next_muxa_state <= s23;   -- *****
   when OTHERS => next_muxa_state <= s20;
 end case;
 end process MUXA_STATE_DECODE;

-------------------------------------------------------------------------
 MUXA_OUTPUT_DECODE:process(present_muxa_state)
 begin
  case present_muxa_state is
   when s20 => mux_sela <= "00";
   when s21 => mux_sela <= "01";
   when s22 => mux_sela <= "10";
   when s23 => mux_sela <= "11";
   when others => mux_sela <= "00"; -- Select A
  end case;
 end process MUXA_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- B_MUX CONTROL
-------------------------------------------------------------------------
 MUXB_STATE_DECODE:process(count)
 begin
 next_muxb_state <= s20;
 case count is
   when 18 | 21 => next_muxb_state <= s21;   -- ***
   when 19 | 20 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 =>
next_muxb_state <= s22;   -- ***********
   when 52 => next_muxb_state <= s23;
   when OTHERS => next_muxb_state <= s20;
 end case;
 end process MUXB_STATE_DECODE;

-------------------------------------------------------------------------
 MUXB_OUTPUT_DECODE:process(present_muxb_state)
```

```
 begin
  case present_muxb_state is
   when s20 => mux_selb <= "00";
   when s21 => mux_selb <= "01";
   when s22 => mux_selb <= "10";
   when s23 => mux_selb <= "11";
   when others => mux_selb <= "00"; -- Select A
  end case;
 end process MUXB_OUTPUT_DECODE;

-------------------------------------------------------------------------------
-- XR_IQ_SEL CONTROL
-------------------------------------------------------------------------------
 MUX2_STATE_DECODE:process(count)
 begin
 next_mux2_state <= s00;
 case count is
   when 20|22|26|28|32|34|38|40 => next_mux2_state <= s01;  -- *******
   when OTHERS => next_mux2_state <= s00;
 end case;
 end process MUX2_STATE_DECODE;

-------------------------------------------------------------------------------
 MUX2_OUTPUT_DECODE:process(present_mux2_state)
 begin
  case present_mux2_state is
   when s00 => mux_sel2 <= '0';
   when s01 => mux_sel2 <= '1';
   when others => mux_sel2 <= '0'; -- NOT WE
  end case;
 end process MUX2_OUTPUT_DECODE;

-------------------------------------------------------------------------------
-- WN_IQ_SEL CONTROL
-------------------------------------------------------------------------------
 MUX3_STATE_DECODE:process(count)
 begin
 next_mux3_state <= s00;
 case count is
   when 2|4|6|8|11|13|15|17|23|24|25|29|30|31|35|36|37|41|42|47|48|49|50|51|52 =>
next_mux3_state <= s01;  -- *******
   when OTHERS => next_mux3_state <= s00;
 end case;
 end process MUX3_STATE_DECODE;

-------------------------------------------------------------------------------
 MUX3_OUTPUT_DECODE:process(present_mux3_state)
 begin
```

```
 case present_mux3_state is
  when s00 => mux_sel3 <= '0';
  when s01 => mux_sel3 <= '1';
  when others => mux_sel3 <= '0'; -- Ich
 end case;
end process MUX3_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- IIR_SEL CONTROL
-------------------------------------------------------------------------
 MUX4_STATE_DECODE:process(count)
 begin
 next_mux4_state <= s00;
 case count is
  when 22 | 25 | 28 | 31 | 34 | 37 | 40 | 43 => next_mux4_state <= s01;  -- *******
  when OTHERS => next_mux4_state <= s00;
 end case;
 end process MUX4_STATE_DECODE;

-------------------------------------------------------------------------
 MUX4_OUTPUT_DECODE:process(present_mux4_state)
 begin
  case present_mux4_state is
  when s00 => mux_sel4 <= '1';
  when s01 => mux_sel4 <= '0';   -- Update factor
  when others => mux_sel4 <= '1'; -- WN RAM OUT
 end case;
end process MUX4_OUTPUT_DECODE;


-------------------------------------------------------------------------
-- ROW TRIG
-------------------------------------------------------------------------
 ROW_STATE_DECODE:process(count2)
 begin
 next_trig1_state <= s30;
 case count2 is
  when 47 => next_trig1_state <= s31;  -- **************************
  when 48 => next_trig1_state <= s32;  -- **************************
  when 49 => next_trig1_state <= s33;  -- **************************
  when 50 => next_trig1_state <= s34;  -- **************************
  when OTHERS => next_trig1_state <= s30;
 end case;
 end process ROW_STATE_DECODE;
-------------------------------------------------------------------------
 TRIG1_OUTPUT_DECODE:process(present_trig1_state)
 begin
  case present_trig1_state is
```

```
  when s31 => trig1 <= "0001";
  when s32 => trig1 <= "0010";
  when s33 => trig1 <= "0100";
  when s34 => trig1 <= "1000";
  when others => trig1 <= "0000"; -- Trig nothing
 end case;
end process TRIG1_OUTPUT_DECODE;

-------------------------------------------------------------------------------
-- Wn MONITOR TRIG
-------------------------------------------------------------------------------
 WNMON_STATE_DECODE:process(count2)
 begin
 next_trig7_state <= s40;
 case count2 is
  when 1 => next_trig7_state <= s41;  -- ***************************
  when 2 => next_trig7_state <= s42;  -- ***************************
  when 3 => next_trig7_state <= s43;  -- ***************************
  when 4 => next_trig7_state <= s44;  -- ***************************
  when 5 => next_trig7_state <= s45;  -- ***************************
  when 6 => next_trig7_state <= s46;  -- ***************************
  when 7 => next_trig7_state <= s47;  -- ***************************
  when 8 => next_trig7_state <= s48;  -- ***************************
  when OTHERS => next_trig7_state <= s40;
 end case;
 end process WNMON_STATE_DECODE;
-------------------------------------------------------------------------------
 TRIG7_OUTPUT_DECODE:process(present_trig7_state)
 begin
  case present_trig7_state is
   when s41    => trig7 <= "00000001";
   when s42    => trig7 <= "00000010";
   when s43    => trig7 <= "00000100";
   when s44    => trig7 <= "00001000";
   when s45    => trig7 <= "00010000";
   when s46    => trig7 <= "00100000";
   when s47    => trig7 <= "01000000";
   when s48    => trig7 <= "10000000";
   when others  => trig7 <= "00000000"; -- Trig nothing
  end case;
 end process TRIG7_OUTPUT_DECODE;

-------------------------------------------------------------------------------
-- INPUT CHANNEL REG TRIG
-------------------------------------------------------------------------------
 INTRIG_STATE_DECODE:process(count)
 begin
 next_trig2_state <= s30;
```

```
 case count is
  when 1 | 2 | 3 | 4 => next_trig2_state <= s31;  -- ************************
  when 5 | 6 | 7 | 8 => next_trig2_state <= s32;  -- ************************
  when 9 | 10 | 11 | 12 => next_trig2_state <= s33;  -- **********************
  when 13 | 14 | 15 | 16 => next_trig2_state <= s34;  -- ********************
  when OTHERS => next_trig2_state <= s30;
 end case;
 end process INTRIG_STATE_DECODE;
-------------------------------------------------------------------------------
 TRIG2_OUTPUT_DECODE:process(present_trig2_state)
 begin
  case present_trig2_state is
   when s31 => trig2 <= "0001";
   when s32 => trig2 <= "0010";
   when s33 => trig2 <= "0100";
   when s34 => trig2 <= "1000";
   when others => trig2 <= "0000"; -- Trig nothing
  end case;
 end process TRIG2_OUTPUT_DECODE;
-------------------------------------------------------------------------------
-- OUT TRIG
-------------------------------------------------------------------------------
 OT_STATE_DECODE:process(count)
 begin
 next_trig3_state <= s20;
 case count is
  when 10 | 22 | 25 | 28 | 31 | 34 | 37 | 40 | 43 => next_trig3_state <= s21; -- ****
  when 19 => next_trig3_state <= s22;  -- ***************************
  when 52 => next_trig3_state <= s23;  -- ************************
  when OTHERS => next_trig3_state <= s20;
 end case;
 end process OT_STATE_DECODE;
-------------------------------------------------------------------------------
 TRIG3_OUTPUT_DECODE:process(present_trig3_state)
 begin
  case present_trig3_state is
   when s21 => trig3 <= "001";
   when s22 => trig3 <= "010";
   when s23 => trig3 <= "100";
   when others => trig3 <= "000"; -- Trig nothing
  end case;
 end process TRIG3_OUTPUT_DECODE;

-------------------------------------------------------------------------------
-- YN REG TRIG
-------------------------------------------------------------------------------
 YN_STATE_DECODE:process(count2)
 begin
```

```
 next_trig4_state <= s10;
 case count2 is
  when 19 => next_trig4_state <= s11; -- ********************
  when 22 => next_trig4_state <= s12;  -- ********************
  when OTHERS => next_trig4_state <= s10;
 end case;
 end process YN_STATE_DECODE;
-------------------------------------------------------------------------------
 TRIG4_OUTPUT_DECODE:process(present_trig4_state)
 begin
  case present_trig4_state is
   when s11 => trig4 <= "01";
   when s12 => trig4 <= "10";
   when others => trig4 <= "00"; -- Trig nothing
  end case;
 end process TRIG4_OUTPUT_DECODE;

-------------------------------------------------------------------------------
-- IIR ACC REG TRIG
-------------------------------------------------------------------------------
 IIR_REG_TRIG_STATE_DECODE:process(count)
 begin
 next_trig5_state <= s00;
 case count is
  when 23 | 26 | 29 | 32 | 35 | 38 | 41 | 44 => next_trig5_state <= s01;  -- ***************
  when OTHERS => next_trig5_state <= s00;
 end case;
 end process IIR_REG_TRIG_STATE_DECODE;
-------------------------------------------------------------------------------
 TRIG5_OUTPUT_DECODE:process(present_trig5_state)
 begin
  case present_trig5_state is
   when s01 => trig5 <= '1';
   when others => trig5 <= '0'; -- Trig nothing
  end case;
 end process TRIG5_OUTPUT_DECODE;

-------------------------------------------------------------------------------
-- COMP CH REG TRIG
-------------------------------------------------------------------------------
 COMP_STATE_DECODE:process(count2)
 begin
 next_trig6_state <= s00;
 case count2 is
  when 2 | 6 | 10 | 14 => next_trig6_state <= s01;  -- ***************
  when OTHERS => next_trig6_state <= s00;
 end case;
 end process COMP_STATE_DECODE;
```

```
-------------------------------------------------------------------------
 TRIG6_OUTPUT_DECODE:process(present_trig6_state)
 begin
  case present_trig6_state is
   when s01 => trig6 <= '1';
   when others => trig6 <= '0'; -- Trig nothing
  end case;
 end process TRIG6_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- WN_RAM_WE CONTROL
-------------------------------------------------------------------------
 WE1_STATE_DECODE:process(count2)
 begin
 next_we1_state <= s10;
 case count2 is
   when 23 | 29 | 35 | 41 => next_we1_state <= s11;  -- *******
   when 26 | 32 | 38 | 44 => next_we1_state <= s12;  -- *******
   when OTHERS => next_we1_state <= s10;
 end case;
 end process WE1_STATE_DECODE;

-------------------------------------------------------------------------
 WE1_OUTPUT_DECODE:process(present_we1_state)
 begin
  case present_we1_state is
   when s11 => we1 <= "01";
   when s12 => we1 <= "10";
   when others => we1 <= "00"; -- NOT WE
  end case;
 end process WE1_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- ACC EN
-------------------------------------------------------------------------
 ACC_STATE_DECODE:process(count2)
 begin
 next_en1_state <= s00;              -- Default: '0' = acc enable
 case count2 is
   when 52 | 53 =>
        next_en1_state <= s01;  -- ***** '1' = RESET
   when OTHERS => next_en1_state <= s00;
 end case;
 end process ACC_STATE_DECODE;
-------------------------------------------------------------------------
 ACC_OUTPUT_DECODE:process(present_en1_state)
 begin
  case present_en1_state is
```

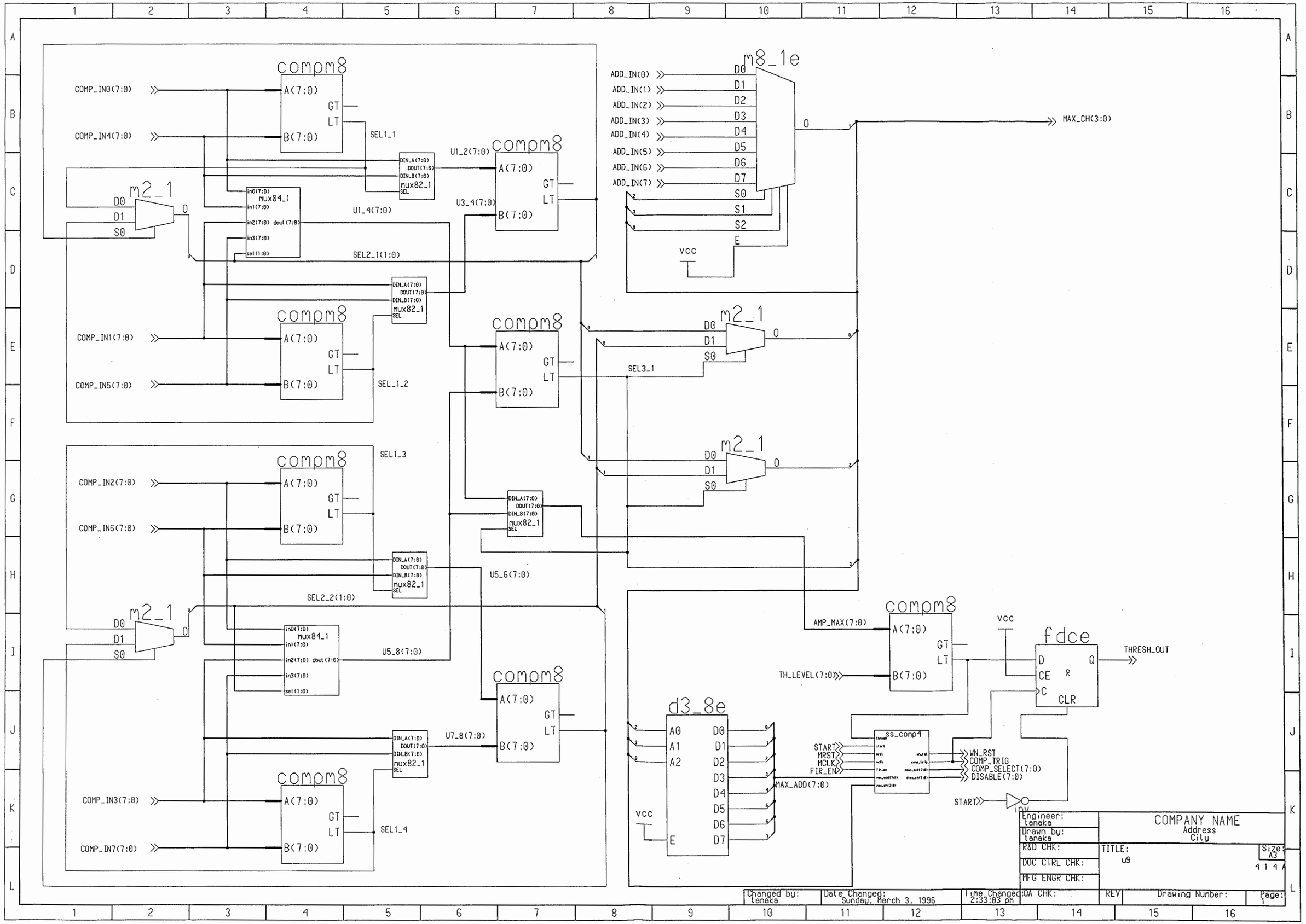
```
  when s00 => en1 <= '0';
  when s01 => en1 <= '1';
  when others => en1 <= '0'; -- ACC Enable
 end case;
end process ACC_OUTPUT_DECODE;

-------------------------------------------------------------------------------
-- IIR ACC EN
-------------------------------------------------------------------------------
 IIR_ACC_STATE_DECODE:process(count2)
 begin
 next_en2_state <= s00;              -- Default: '0' = acc enable
 case count2 is
  when 46 => next_en2_state <= s01;  -- ***** '1' = RESET
  when OTHERS => next_en2_state <= s00;
 end case;
 end process IIR_ACC_STATE_DECODE;
-------------------------------------------------------------------------------
 IIR_ACC_OUTPUT_DECODE:process(present_en2_state)
 begin
  case present_en2_state is
  when s00 => en2 <= '0';
  when s01 => en2 <= '1';
  when others => en2 <= '0'; -- ACC Enable
 end case;
end process IIR_ACC_OUTPUT_DECODE;

-------------------------------------------------------------------------------
--  AS CONT
-------------------------------------------------------------------------------
 AS_STATE_DECODE:process(count2)
 begin
 next_adsu1_state <= s00;            --
 case count2 is
  when 3|5|7|9|24|30|36|42 => next_adsu1_state <= s01; -- '1' = SUBSTRUCT MODE
  when OTHERS => next_adsu1_state <= s00;
 end case;
 end process AS_STATE_DECODE;
-------------------------------------------------------------------------------
 AS_OUTPUT_DECODE:process(present_adsu1_state)
 begin
  case present_adsu1_state is
  when s01 => adsu1 <= '0';
  when others => adsu1 <= '1'; -- Trig nothing
 end case;
end process AS_OUTPUT_DECODE;
-------------------------------------------------------------------------------
-- IIR AS CONT
```

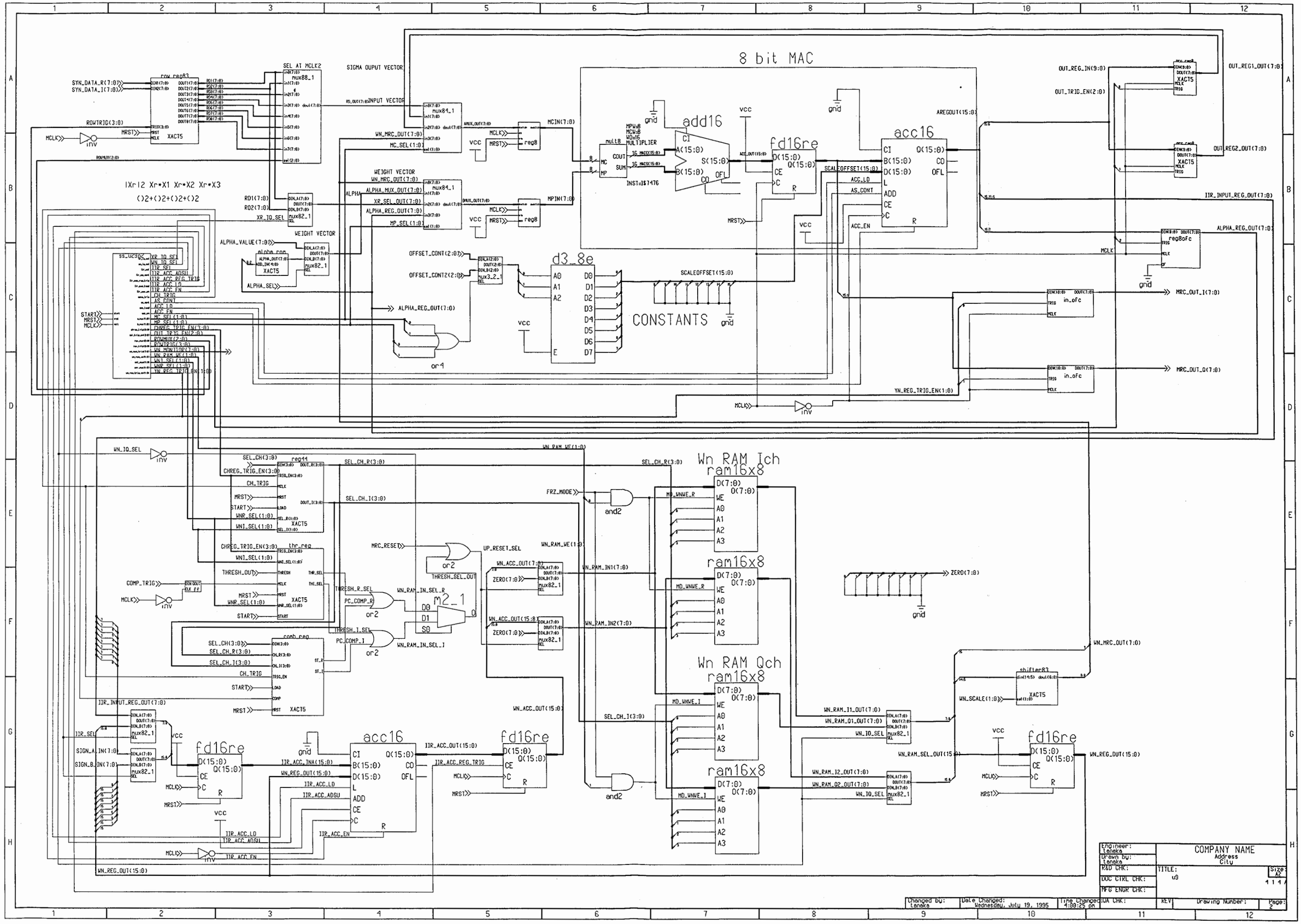
```
-------------------------------------------------------------------------
 IIR_AS_STATE_DECODE:process(count2)
 begin
 next_adsu2_state <= s00;              --
 case count2 is
  when 21 | 24 | 27 | 30 | 33 | 36 | 39 | 42 => next_adsu2_state <= s01; -- '1' = SUBSTRUCT MODE
  when OTHERS => next_adsu2_state <= s00;
 end case;
 end process IIR_AS_STATE_DECODE;
-------------------------------------------------------------------------
 IIR_AS_OUTPUT_DECODE:process(present_adsu2_state)
 begin
  case present_adsu2_state is
   when s01 => adsu2 <= '0';
   when others => adsu2 <= '1'; -- Trig nothing
  end case;
 end process IIR_AS_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- LOAD CONT
-------------------------------------------------------------------------
 LD_STATE_DECODE:process(count2)
 begin
 next_ld1_state <= s00;              --
 case count2 is
  when 1 | 10 | 19 | 22 | 25 | 28 | 31 | 34 | 37 | 40 | 43 => next_ld1_state <= s01; -- '1' = LOAD
  when OTHERS => next_ld1_state <= s00;
 end case;
 end process LD_STATE_DECODE;
-------------------------------------------------------------------------
 LD_OUTPUT_DECODE:process(present_ld1_state)
 begin
  case present_ld1_state is
   when s01 => ld1 <= '1';
   when others => ld1 <= '0'; -- Nothing to load
  end case;
 end process LD_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- IIR LOAD CONT
-------------------------------------------------------------------------
 IIR_LD_STATE_DECODE:process(count2)
 begin
 next_ld2_state <= s00;              --
 case count2 is
  when 20 | 23 | 26 | 29 | 32 | 35 | 38 | 41 => next_ld2_state <= s01; -- '1' = LOAD
  when OTHERS => next_ld2_state <= s00;
 end case;
```

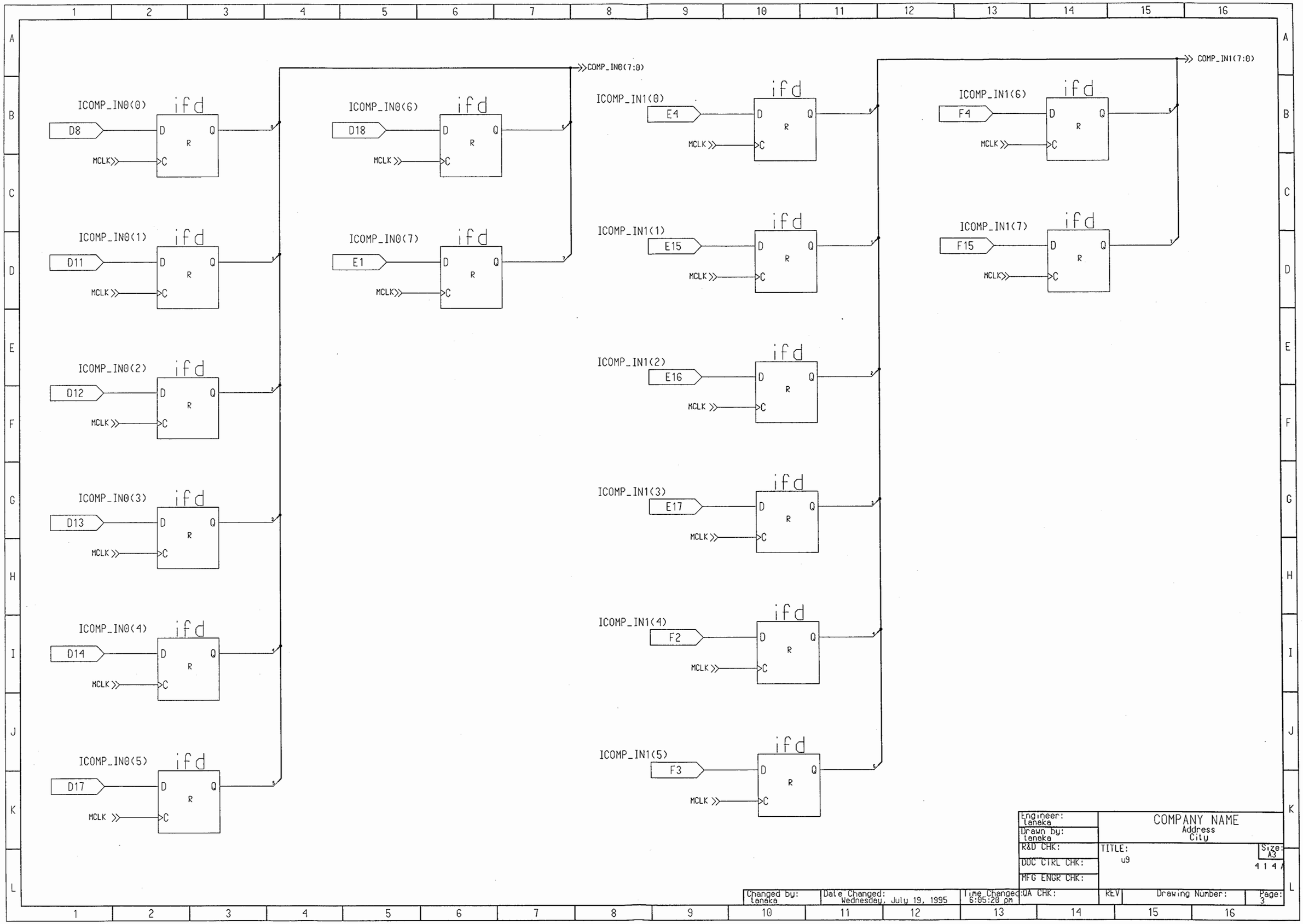
```
 end process IIR_LD_STATE_DECODE;
-------------------------------------------------------------------------
 IIR_LD_OUTPUT_DECODE:process(present_ld2_state)
 begin
  case present_ld2_state is
   when s01 => ld2 <= '1';
   when others => ld2 <= '0'; -- Nothing to load
  end case;
 end process IIR_LD_OUTPUT_DECODE;

-------------------------------------------------------------------------
-- STATE REGISTER PROCESS
-------------------------------------------------------------------------
 MCLK2_STATE_REGISTER:process(mclk, start) -- Triggered by MCLK2
 begin
  if (start = '0') then
      present_trig1_state <= s30;
      present_trig4_state <= s10;
      present_trig6_state <= s00;
      present_trig7_state <= s40;
      present_we1_state <= s10;
      present_en1_state <= s00;
      present_en2_state <= s00;
      present_adsu1_state <= s00;
      present_adsu2_state <= s00;
      present_ld1_state <= s00;
      present_ld2_state <= s00;
  elsif ( MCLK'EVENT AND MCLK = '0' AND MCLK'LAST_VALUE = '1') THEN
      present_trig1_state <= next_trig1_state;
      present_trig4_state <= next_trig4_state;
      present_trig6_state <= next_trig6_state;
      present_trig7_state <= next_trig7_state;
      present_we1_state <= next_we1_state;
      present_en1_state <= next_en1_state;
      present_en2_state <= next_en2_state;
      present_adsu1_state <= next_adsu1_state;
      present_adsu2_state <= next_adsu2_state;
      present_ld1_state <= next_ld1_state;
      present_ld2_state <= next_ld2_state;
  end if;
 end process MCLK2_STATE_REGISTER;
-------------------------------------------------------------------------
 MCLK_STATE_REGISTER:process(mclk, start2) -- Triggered by MCLK
 begin
  if (start2 = '0') then
      present_mux1_state <= s0;
      present_mux2_state <= s00;
      present_mux3_state <= s00;
```

```
        present_mux4_state <= s00;
        present_muxa_state <= s20;
        present_muxb_state <= s20;
        present_muxr_state <= s20;
        present_muxi_state <= s20;
        present_trig2_state <= s30;
        present_trig3_state <= s20;
        present_trig5_state <= s00;
    elsif ( MCLK'EVENT AND MCLK = '1' AND MCLK'LAST_VALUE = '0') THEN
        present_mux1_state <= next_mux1_state;
        present_mux2_state <= next_mux2_state;
        present_mux3_state <= next_mux3_state;
        present_mux4_state <= next_mux4_state;
        present_muxa_state <= next_muxa_state;
        present_muxb_state <= next_muxb_state;
        present_muxr_state <= next_muxr_state;
        present_muxi_state <= next_muxi_state;
        present_trig2_state <= next_trig2_state;
        present_trig3_state <= next_trig3_state;
        present_trig5_state <= next_trig5_state;
    end if;
  end process MCLK_STATE_REGISTER;

-------------------------------------------------------------------------
-- TRIGGERED BY MCLK, MCLK2 PROCESS = FINAL OUTPUT
-------------------------------------------------------------------------

  MCLK2_OUTPUT_REGISTER:process(MCLK, start) -- Triggered by MCLK2
  begin
  if ( start = '0' ) then
    RAW_MUX <= ( OTHERS => '0');
    XR_IQ_SEL <= '0';
    WN_IQ_SEL <= '0';
    IIR_SEL <= '0';
    A_MUX <= ( OTHERS => '0');
    B_MUX <= ( OTHERS => '0');
    WNR_MUX <= ( OTHERS => '0');
    WNI_MUX <= ( OTHERS => '0');
    CHREG_TRIG <= ( OTHERS => '0');
    OUT_TRIG_EN <= ( OTHERS => '0');
    IIR_ACC_REG_TRIG <= '0';
  elsif ( MCLK'EVENT AND MCLK = '0' AND MCLK'LAST_VALUE = '1' ) then
    RAW_MUX <= mux_sel1;
    XR_IQ_SEL <= mux_sel2;
    WN_IQ_SEL <= mux_sel3;
    IIR_SEL <= mux_sel4;
    A_MUX <= mux_sela;
    B_MUX <= mux_selb;
```

```
    WNR_MUX <= mux_selr;
    WNI_MUX <= mux_seli;
    CHREG_TRIG <= trig2;
    OUT_TRIG_EN <= trig3;
    IIR_ACC_REG_TRIG <= trig5;
  end if;
  end process MCLK2_OUTPUT_REGISTER;

-------------------------------------------------------------------------------

  MCLK_OUTPUT_REGISTER:process(MCLK, START2) -- Triggered by MCLK
  begin
  if ( START2 = '0' ) then
    RAW_TRIG <= ( OTHERS => '0');
    YN_REG_TRIG_EN <= ( OTHERS => '0');
    COMP_TRIG <= '0';
    WN_RAM_WE <= ( OTHERS => '0');
    ACC_EN <= '0';
    IIR_ACC_EN <= '0';
    AS_CONT <= '0';
    IIR_AS_CONT <= '0';
    ACC_LOAD <= '0';
    IIR_ACC_LOAD <= '0';
    WN_MON_TRIG <= ( OTHERS => '0');
  elsif ( MCLK'EVENT AND MCLK = '1' AND MCLK'LAST_VALUE = '0' ) then
    RAW_TRIG <= trig1;
    YN_REG_TRIG_EN <= trig4;
    COMP_TRIG <= trig6;
    WN_RAM_WE <= we1;
    ACC_EN <= en1;
    IIR_ACC_EN <= en2;
    AS_CONT <= adsu1;
    IIR_AS_CONT <= adsu2;
    ACC_LOAD <= ld1;
    IIR_ACC_LOAD <= ld2;
    WN_MON_TRIG <= trig7;
  end if;
  end process MCLK_OUTPUT_REGISTER;

-------------------------------------------------------------------------------
-- End of rtl.
-------------------------------------------------------------------------------
end rtl;
```

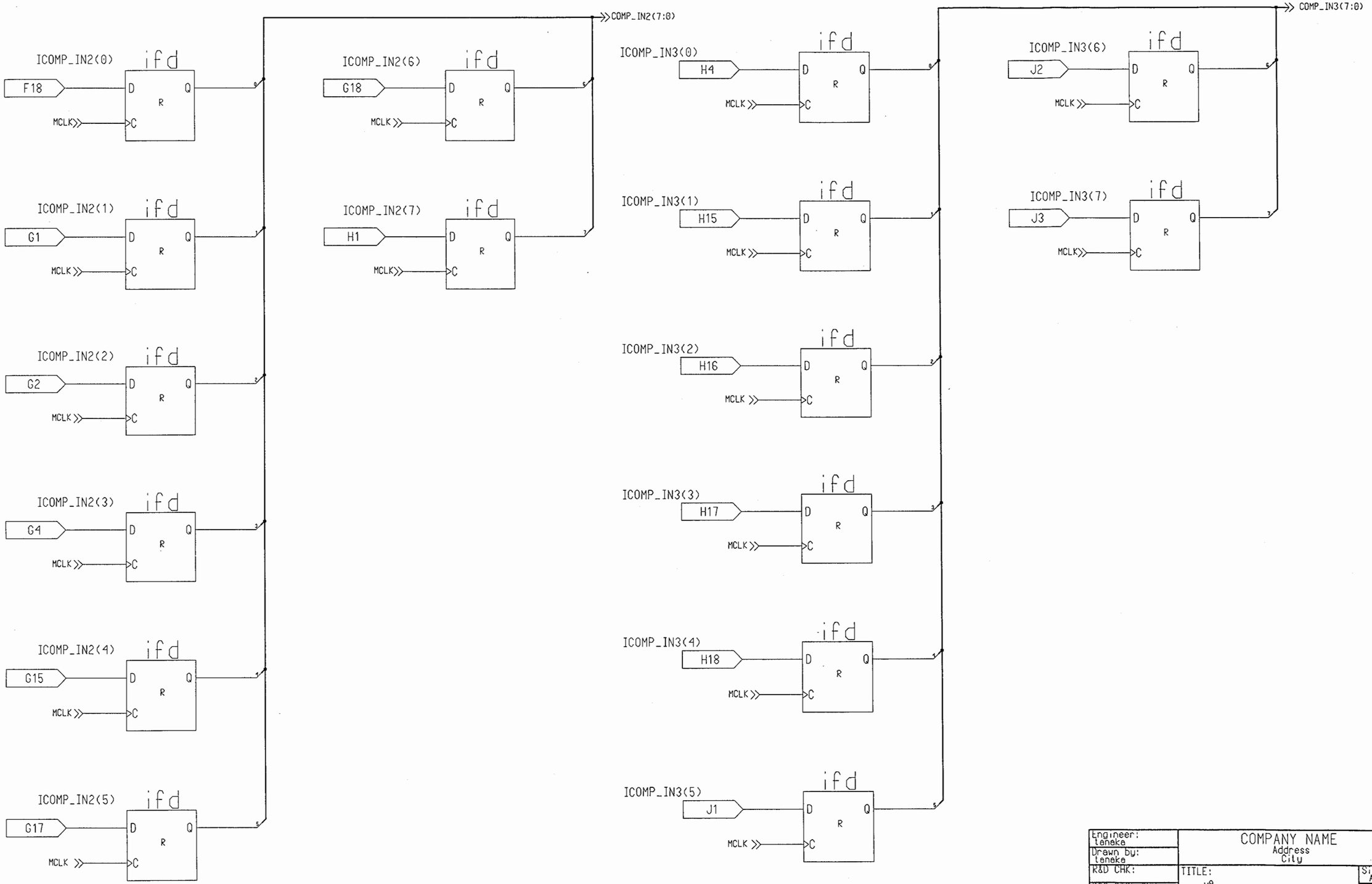

compm8
COMP_IN0(7:0)
COMP_IN4(7:0)
A(7:0)
B(7:0)
GT
LT
SEL1_1
m2_1
D0
D1
S0
O
mux84_1
in0(7:0)
in1(7:0)
in2(7:0)
in3(7:0)
sel(1:0)
dout(7:0)
mux82_1
DIN_A(7:0)
DOUT(7:0)
DIN_B(7:0)
SEL
U1_2(7:0)
U1_4(7:0)
U3_4(7:0)
SEL2_1(1:0)
COMP_IN1(7:0)
COMP_IN5(7:0)
SEL_1_2
SEL1_3
COMP_IN2(7:0)
COMP_IN6(7:0)
U5_6(7:0)
SEL2_2(1:0)
U5_8(7:0)
U7_8(7:0)
COMP_IN3(7:0)
COMP_IN7(7:0)
SEL1_4
m8_1e
ADD_IN(0)
ADD_IN(1)
ADD_IN(2)
ADD_IN(3)
ADD_IN(4)
ADD_IN(5)
ADD_IN(6)
ADD_IN(7)
D0
D1
D2
D3
D4
D5
D6
D7
S0
S1
S2
E
VCC
MAX_CH(3:0)
SEL3_1
d3_8e
A0
A1
A2
D0
D1
D2
D3
D4
D5
D6
D7
E
MAX_ADD(7:0)
AMP_MAX(7:0)
TH_LEVEL(7:0)
fdce
D
Q
CE
R
C
CLR
THRESH_OUT
SS_comp4
START
MRST
MCLK
FIR_EN
WN_RST
COMP_TRIG
COMP_SELECT(7:0)
DISABLE(7:0)
INV
Engineer: teneka
Drawn by: teneka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
u9
Size: A3
Changed by: teneka
Date Changed: Sunday, March 3, 1996
Time Changed: 2:33:03 pm
QA CHK:
REV
Drawing Number:
Page: 1


8 bit MAC
add16
fd16re
acc16
d3_8e
CONSTANTS
SEL AT MCLK2
SIGMA OUPUT VECTOR
INPUT VECTOR
WEIGHT VECTOR
IXr12 Xr*X1 Xr*X2 Xr*X3
()2+()2+()2+()2
OUT_REG1_OUT(7:0)
OUT_REG2_OUT(7:0)
IIR_INPUT_REG_OUT(7:0)
ALPHA_REG_OUT(7:0)
MRC_OUT_I(7:0)
MRC_OUT_Q(7:0)
Wn RAM Ich
Wn RAM Qch
ram16x8
WN_MRC_OUT(7:0)
WN_REG_OUT(15:0)
COMPANY NAME
Address
City

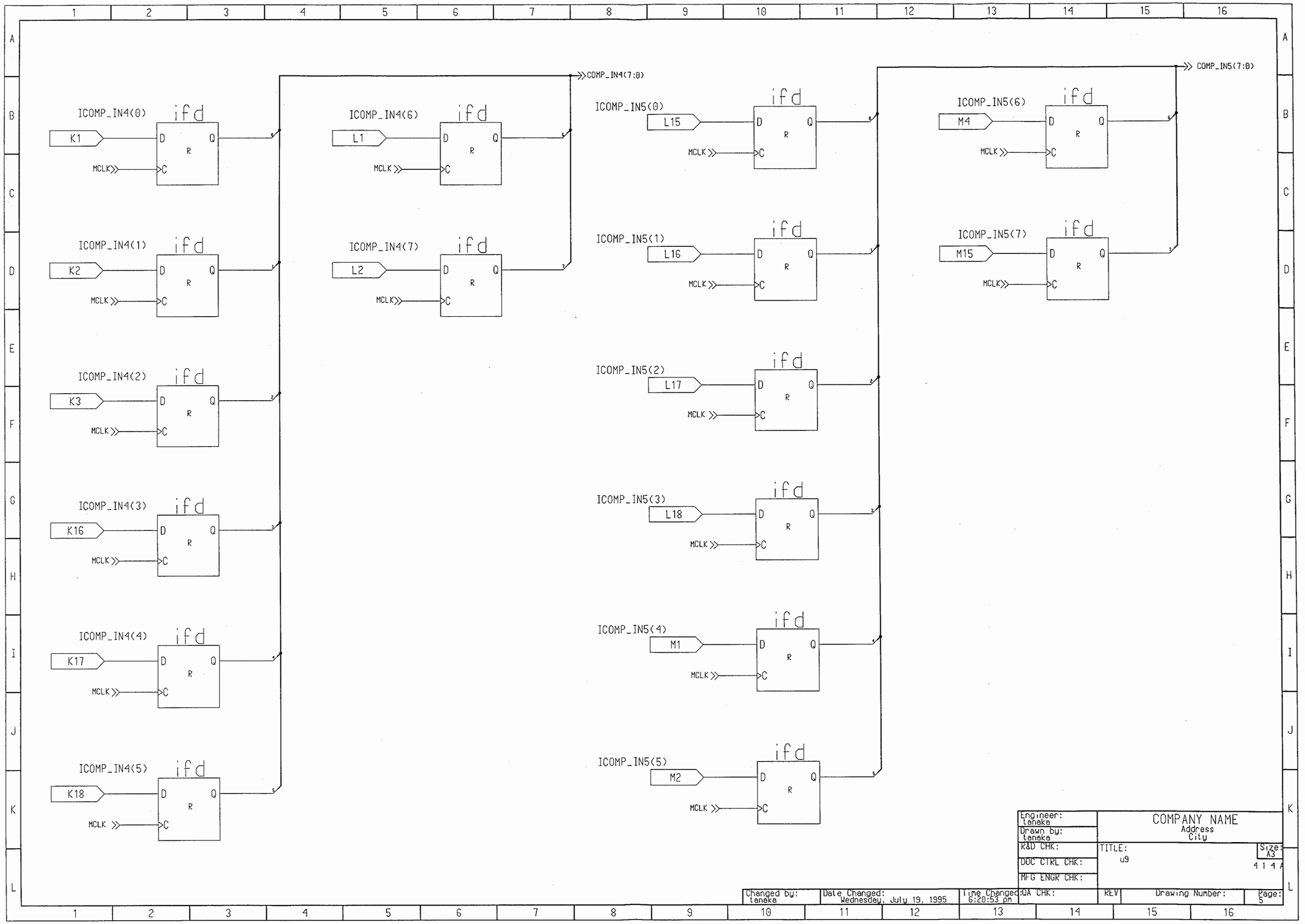


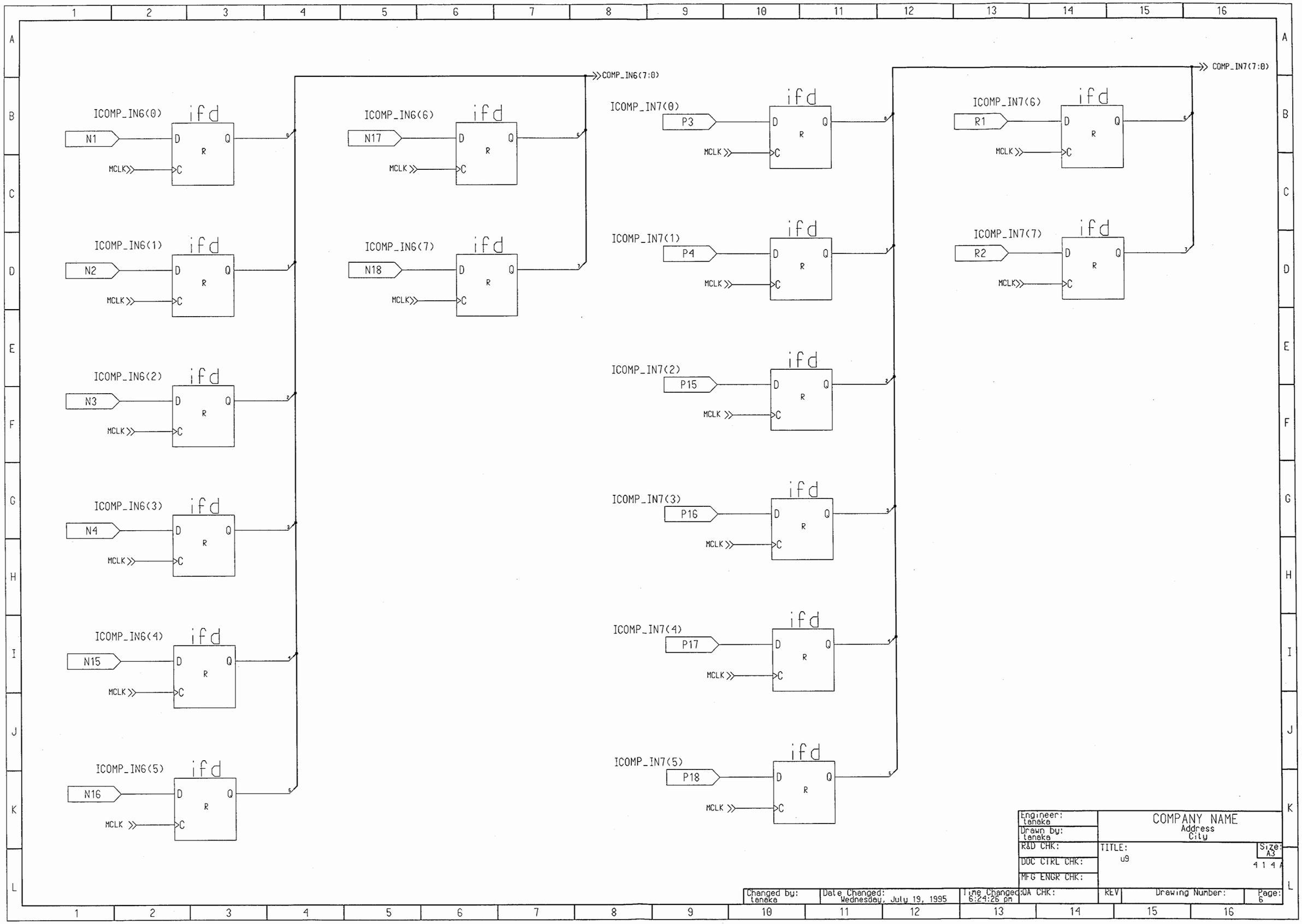
Changed by: taneka	Date Changed: Wednesday, July 19, 1995	Time Changed: 6:05:20 pm	QA CHK:	REV	Drawing Number:	Page: 3

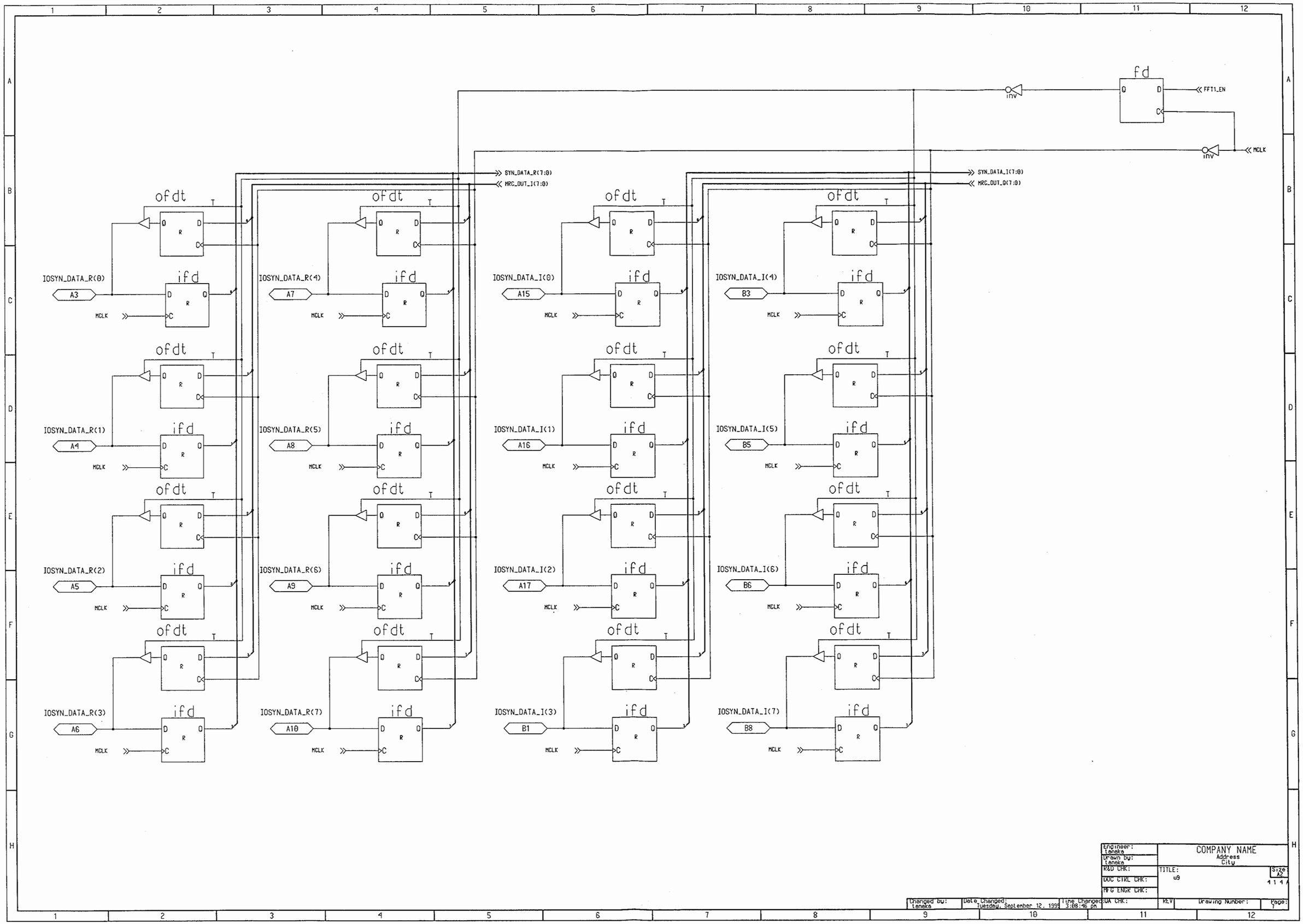
Engineer: taneka	COMPANY NAME Address City
Drawn by: taneka	
R&D CHK:	TITLE: u9
DOC CTRL CHK:	Size: A3
MFG ENGR CHK:	4 1 4 A

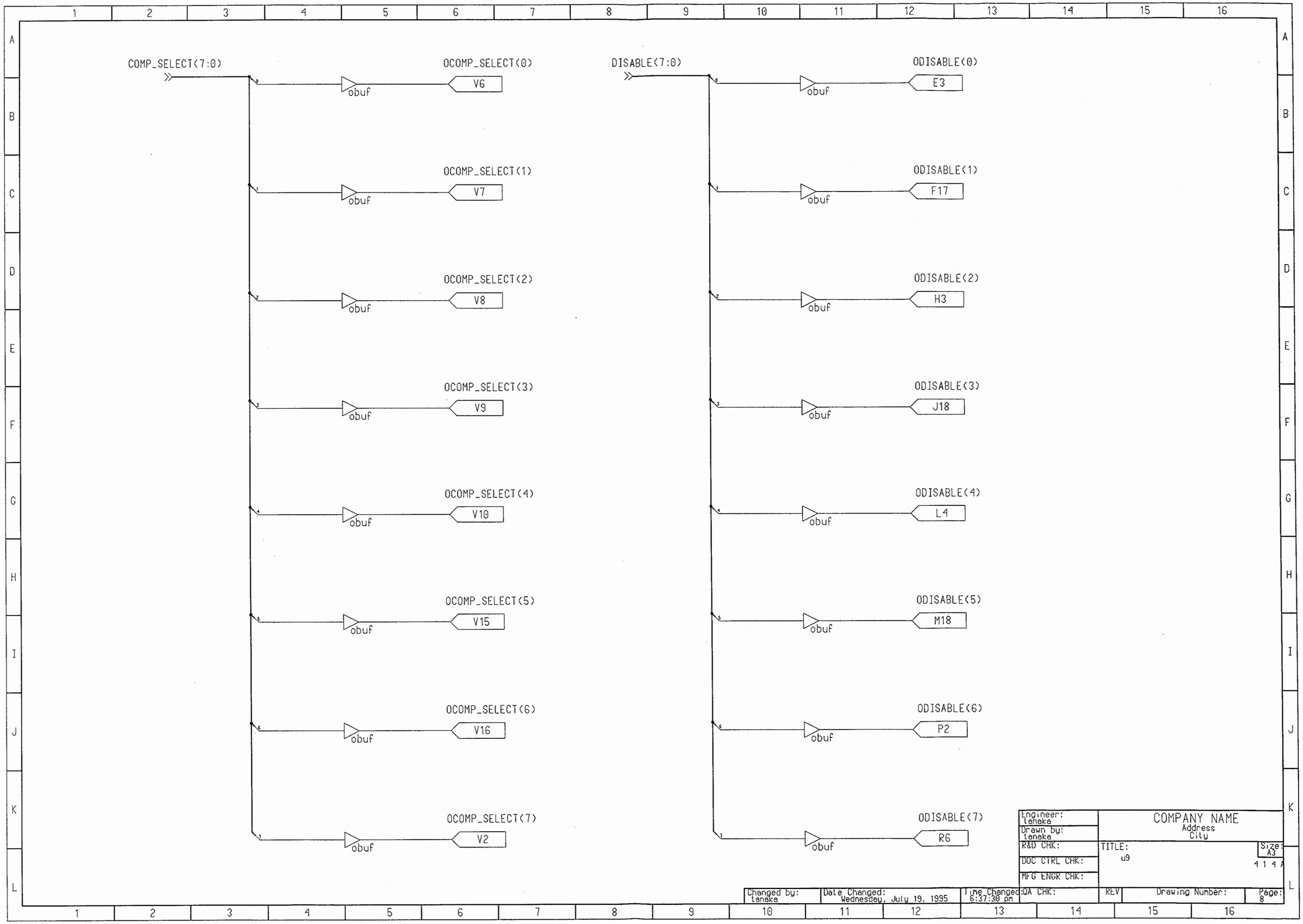


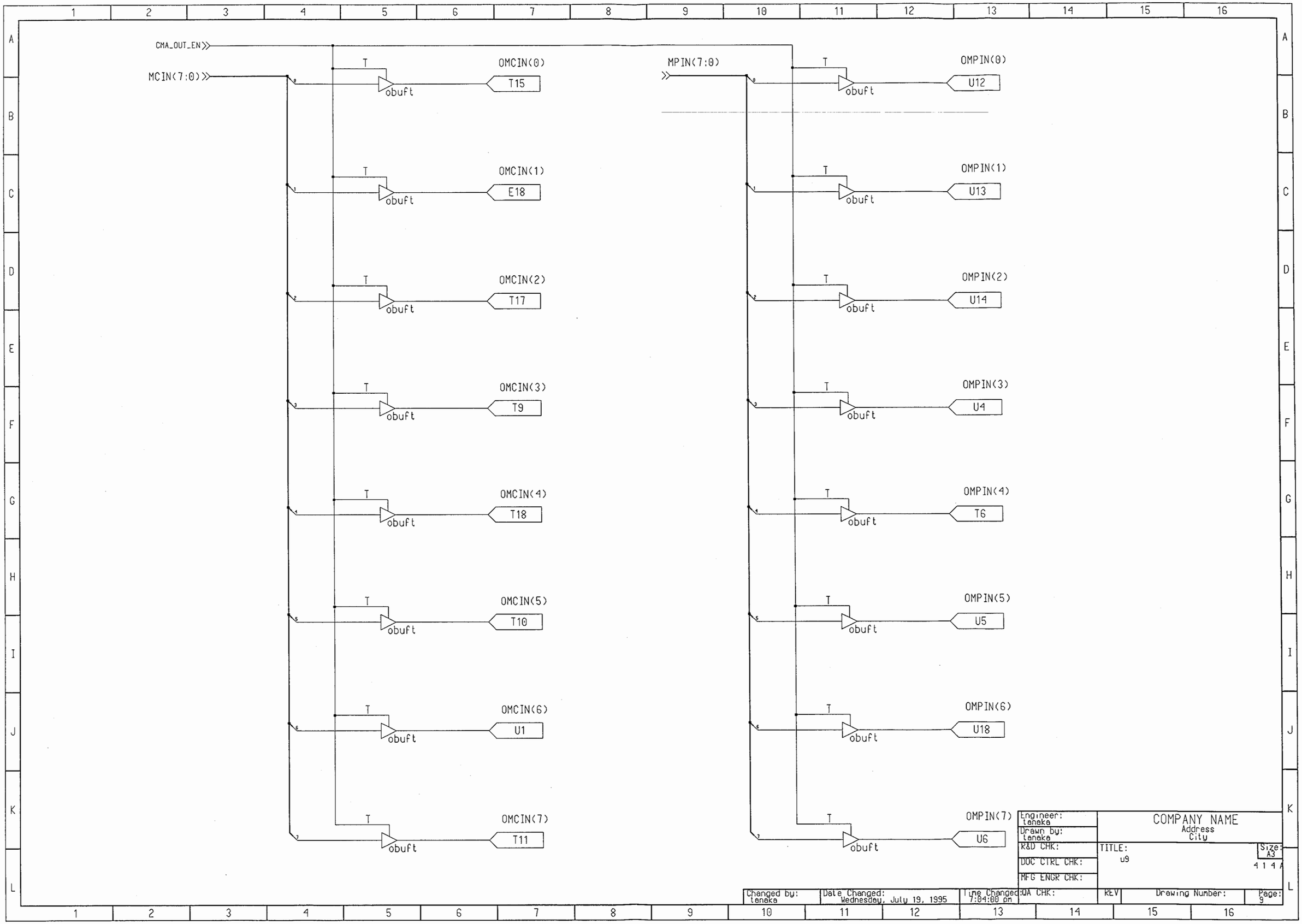
Changed by: teneke	Date Changed: Wednesday, July 19, 1995	Time Changed: 6:17:19 pm	Engineer: teneke	COMPANY NAME Address City		
			Drawn by: teneke			
			R&D CHK:	TITLE: u9	Size: A3	
			DOC CTRL CHK:			4 1 4 A
			MFG ENGR CHK:			
			QA CHK:	REV	Drawing Number:	Page: 4

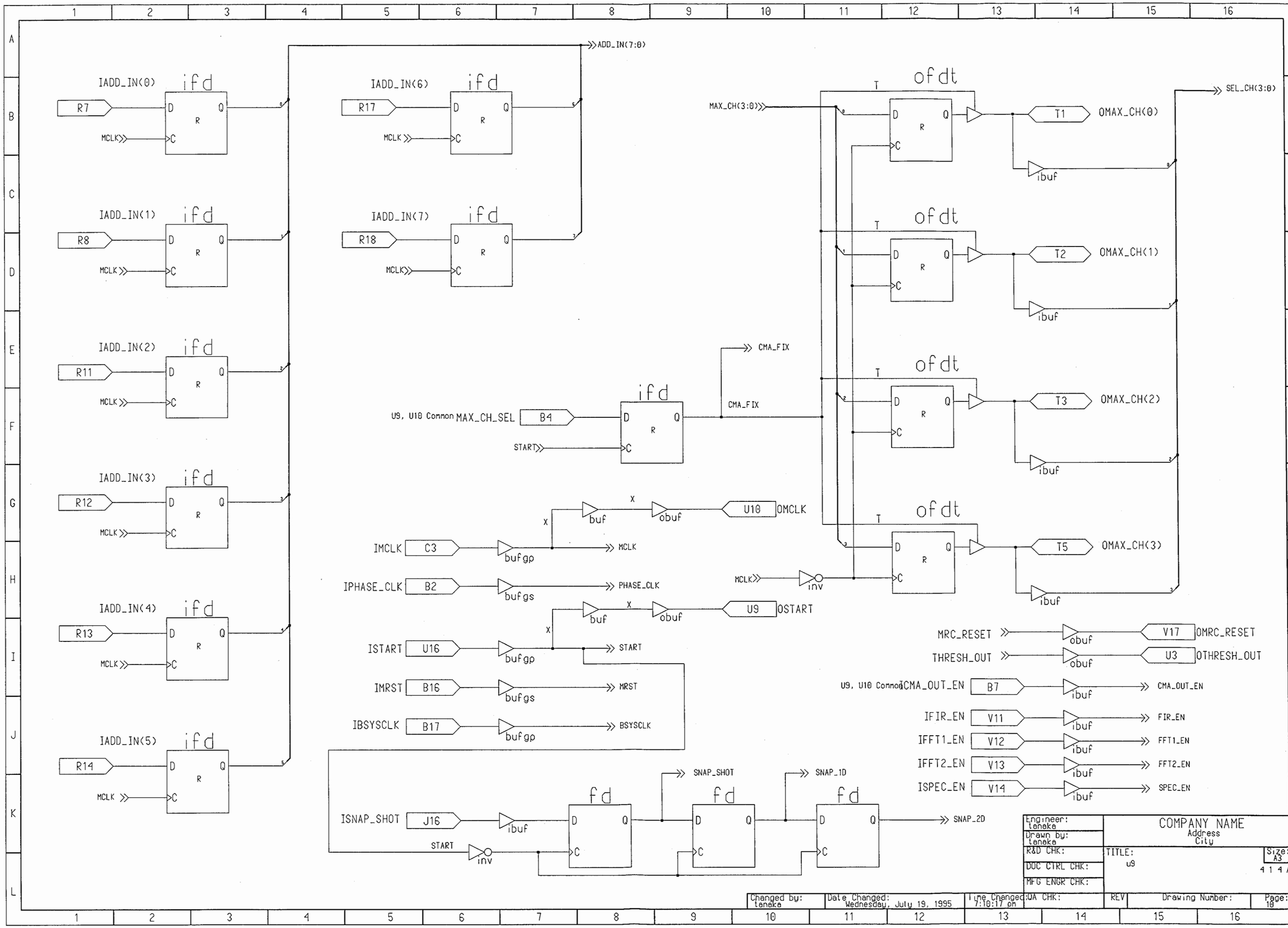

ICOMP_IN4(0)
K1
ICOMP_IN4(1)
K2
ICOMP_IN4(2)
K3
ICOMP_IN4(3)
K16
ICOMP_IN4(4)
K17
ICOMP_IN4(5)
K18
ICOMP_IN4(6)
L1
ICOMP_IN4(7)
L2
COMP_IN4(7:0)
ICOMP_IN5(0)
L15
ICOMP_IN5(1)
L16
ICOMP_IN5(2)
L17
ICOMP_IN5(3)
L18
ICOMP_IN5(4)
M1
ICOMP_IN5(5)
M2
ICOMP_IN5(6)
M4
ICOMP_IN5(7)
M15
COMP_IN5(7:0)
ifd
D
Q
R
C
MCLK
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
u9
Size: A3
4 1 4 A
Changed by: tanaka
Date Changed: Wednesday, July 19, 1995
Time Changed: 6:20:53 pm
QA CHK:
REV
Drawing Number:
Page: 5

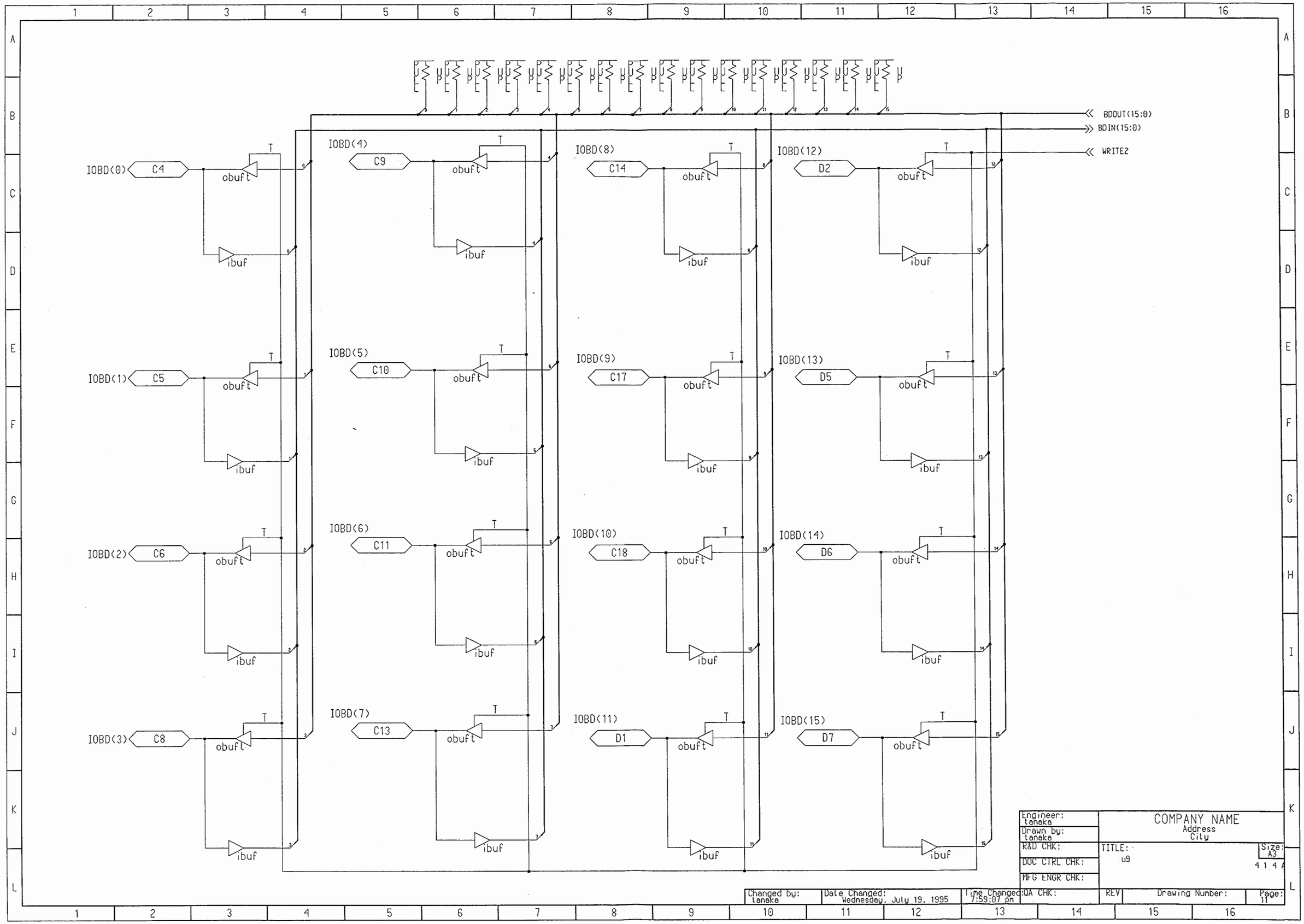

ICOMP_IN6(0)
N1
MCLK
ICOMP_IN6(1)
N2
MCLK
ICOMP_IN6(2)
N3
MCLK
ICOMP_IN6(3)
N4
MCLK
ICOMP_IN6(4)
N15
MCLK
ICOMP_IN6(5)
N16
MCLK
ICOMP_IN6(6)
N17
MCLK
ICOMP_IN6(7)
N18
MCLK
COMP_IN6(7:0)
ICOMP_IN7(0)
P3
MCLK
ICOMP_IN7(1)
P4
MCLK
ICOMP_IN7(2)
P15
MCLK
ICOMP_IN7(3)
P16
MCLK
ICOMP_IN7(4)
P17
MCLK
ICOMP_IN7(5)
P18
MCLK
ICOMP_IN7(6)
R1
MCLK
ICOMP_IN7(7)
R2
MCLK
COMP_IN7(7:0)
ifd
D
Q
R
C
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
u9
Size: A3
4 1 4 A
Changed by: tanaka
Date Changed: Wednesday, July 19, 1995
Time Changed: 6:24:26 pm
QA CHK:
REV
Drawing Number:
Page: 6

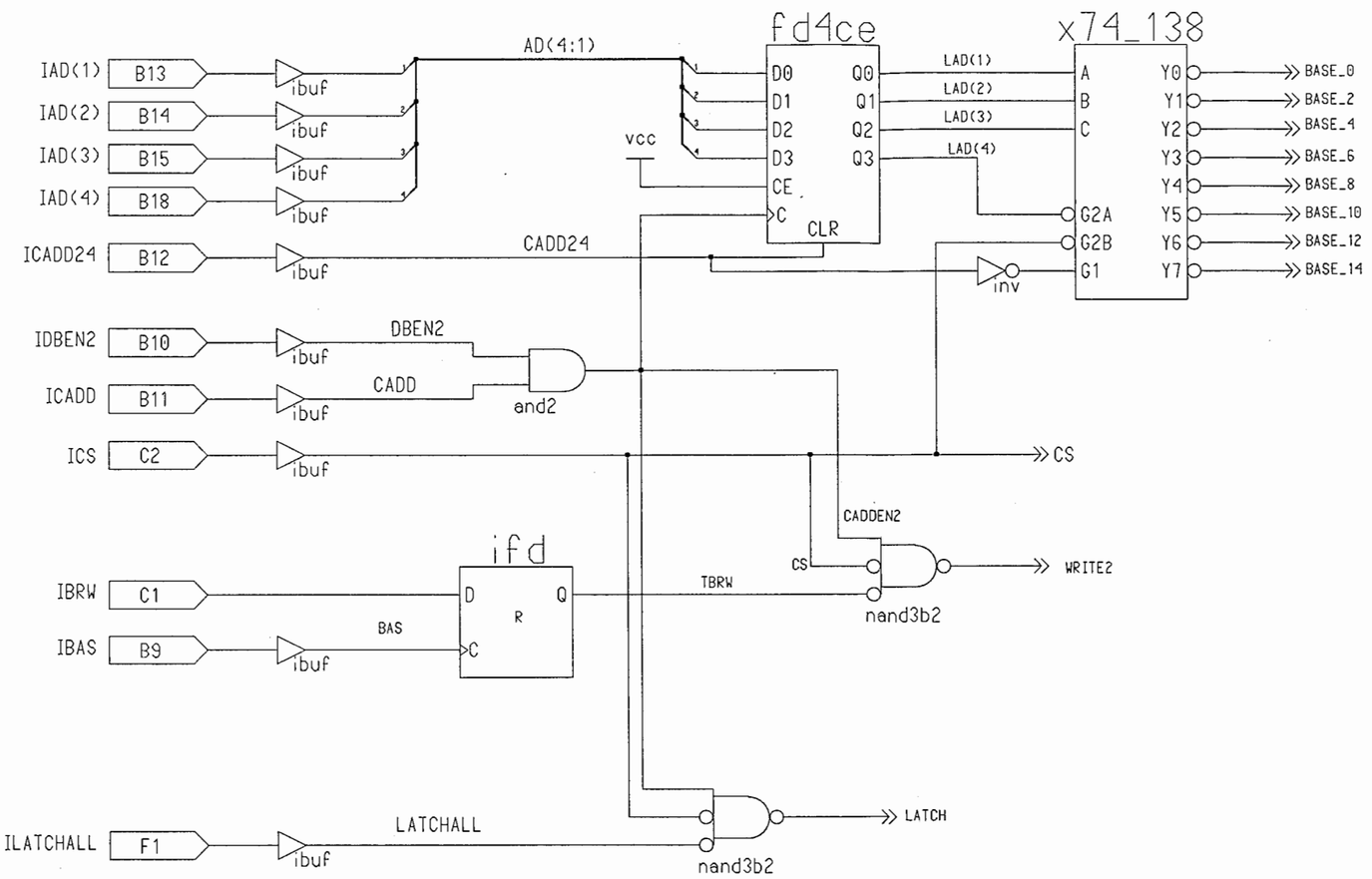

fd
FFT1_EN
MCLK
SYN_DATA_R(7:0)
MRC_OUT_I(7:0)
SYN_DATA_I(7:0)
MRC_OUT_Q(7:0)
IOSYN_DATA_R(0)
A3
IOSYN_DATA_R(1)
A4
IOSYN_DATA_R(2)
A5
IOSYN_DATA_R(3)
A6
IOSYN_DATA_R(4)
A7
IOSYN_DATA_R(5)
A8
IOSYN_DATA_R(6)
A9
IOSYN_DATA_R(7)
A10
IOSYN_DATA_I(0)
A15
IOSYN_DATA_I(1)
A16
IOSYN_DATA_I(2)
A17
IOSYN_DATA_I(3)
B1
IOSYN_DATA_I(4)
B3
IOSYN_DATA_I(5)
B5
IOSYN_DATA_I(6)
B6
IOSYN_DATA_I(7)
B8
ofdt
ifd
Engineer: Tanaka
Drawn by: Tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE: u9
Size: A2
Changed by: Tanaka
Date Changed: Tuesday, September 12, 1995
Time Changed: 3:08:46 pm
QA CHK:
REV
Drawing Number:
Page: 1


COMP_SELECT(7:0)
OCOMP_SELECT(0)
V6
OCOMP_SELECT(1)
V7
OCOMP_SELECT(2)
V8
OCOMP_SELECT(3)
V9
OCOMP_SELECT(4)
V10
OCOMP_SELECT(5)
V15
OCOMP_SELECT(6)
V16
OCOMP_SELECT(7)
V2
DISABLE(7:0)
ODISABLE(0)
E3
ODISABLE(1)
F17
ODISABLE(2)
H3
ODISABLE(3)
J18
ODISABLE(4)
L4
ODISABLE(5)
M18
ODISABLE(6)
P2
ODISABLE(7)
R6
obuf
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
QA CHK:
COMPANY NAME
Address
City
TITLE:
u9
Size: A3
4 1 4 A
Changed by: tanaka
Date Changed: Wednesday, July 19, 1995
Time Changed: 6:37:30 pm
REV
Drawing Number:
Page: 8


CMA_OUT_EN
MCIN(7:0)
MPIN(7:0)
T
obuft
OMCIN(0)
T15
OMCIN(1)
E18
OMCIN(2)
T17
OMCIN(3)
T9
OMCIN(4)
T18
OMCIN(5)
T10
OMCIN(6)
U1
OMCIN(7)
T11
OMPIN(0)
U12
OMPIN(1)
U13
OMPIN(2)
U14
OMPIN(3)
U4
OMPIN(4)
T6
OMPIN(5)
U5
OMPIN(6)
U18
OMPIN(7)
U6
Engineer: tenaka
Drawn by: tenaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
u9
Size: A3
4 1 4 A
Changed by: tenaka
Date Changed: Wednesday, July 19, 1995
Time Changed: 7:04:00 pm
QA CHK:
REV
Drawing Number:
Page: 9


IADD_IN(0)
R7
MCLK
IADD_IN(1)
R8
MCLK
IADD_IN(2)
R11
MCLK
IADD_IN(3)
R12
MCLK
IADD_IN(4)
R13
MCLK
IADD_IN(5)
R14
MCLK
IADD_IN(6)
R17
MCLK
IADD_IN(7)
R18
MCLK
ADD_IN(7:0)
ifd
D
Q
R
C
U9, U10 Common MAX_CH_SEL
B4
START
CMA_FIX
IMCLK
C3
bufgp
MCLK
IPHASE_CLK
B2
bufgs
PHASE_CLK
ISTART
U16
START
IMRST
B16
MRST
IBSYSCLK
B17
BSYSCLK
buf
obuf
X
U10
OMCLK
U9
OSTART
ISNAP_SHOT
J16
ibuf
fd
SNAP_SHOT
SNAP_1D
SNAP_2D
inv
MAX_CH(3:0)
ofdt
T
T1
OMAX_CH(0)
T2
OMAX_CH(1)
T3
OMAX_CH(2)
T5
OMAX_CH(3)
SEL_CH(3:0)
MRC_RESET
V17
OMRC_RESET
THRESH_OUT
U3
OTHRESH_OUT
CMA_OUT_EN
B7
CMA_OUT_EN
IFIR_EN
V11
FIR_EN
IFFT1_EN
V12
FFT1_EN
IFFT2_EN
V13
FFT2_EN
ISPEC_EN
V14
SPEC_EN
Engineer: taneka
Drawn by: taneka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
QA CHK:
COMPANY NAME
Address
City
TITLE:
u9
Size: A3
4 1 4 A
Changed by: taneka
Date Changed: Wednesday, July 19, 1995
Time Changed: 7:10:17 pm
REV
Drawing Number:
Page: 10


BDOUT(15:0)
BDIN(15:0)
WRITE2
IOBD(0)
C4
IOBD(1)
C5
IOBD(2)
C6
IOBD(3)
C8
IOBD(4)
C9
IOBD(5)
C10
IOBD(6)
C11
IOBD(7)
C13
IOBD(8)
C14
IOBD(9)
C17
IOBD(10)
C18
IOBD(11)
D1
IOBD(12)
D2
IOBD(13)
D5
IOBD(14)
D6
IOBD(15)
D7
T
obuft
ibuf
Engineer: taneka
Drawn by: taneka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
u9
Size: A3
4 1 4 A
QA CHK:
REV
Drawing Number:
Page: 11
Changed by: taneka
Date Changed: Wednesday, July 19, 1995
Time Changed: 7:59:07 pm

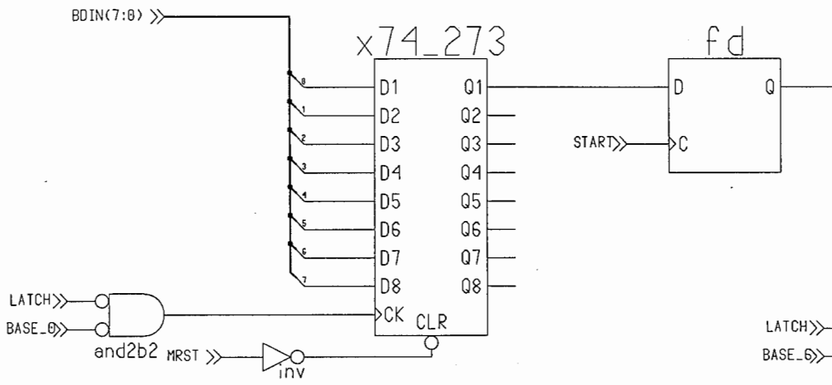


Changed by: taneka	Date Changed: Wednesday, July 19, 1995	Time Changed: 8:02:27 pm			

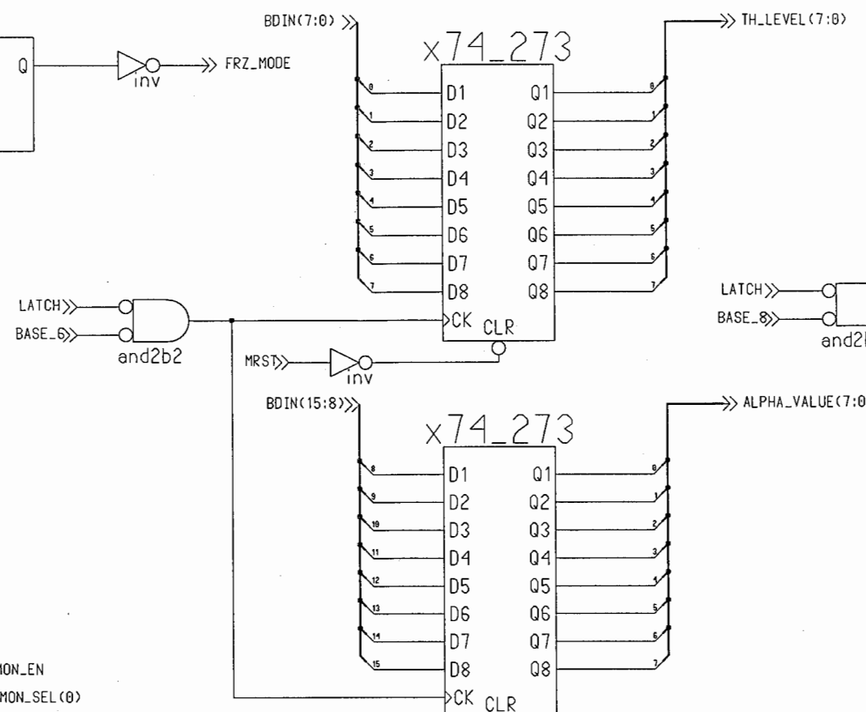
Engineer: taneka	COMPANY NAME		
Drawn by: taneka	Address		
R&D CHK:	City		
DOC CTRL CHK:	TITLE: u9	Size: A3	
MFG ENGR CHK:		4 1 4 A	
QA CHK:	REV	Drawing Number:	Page: 12

WRITE REGISTER

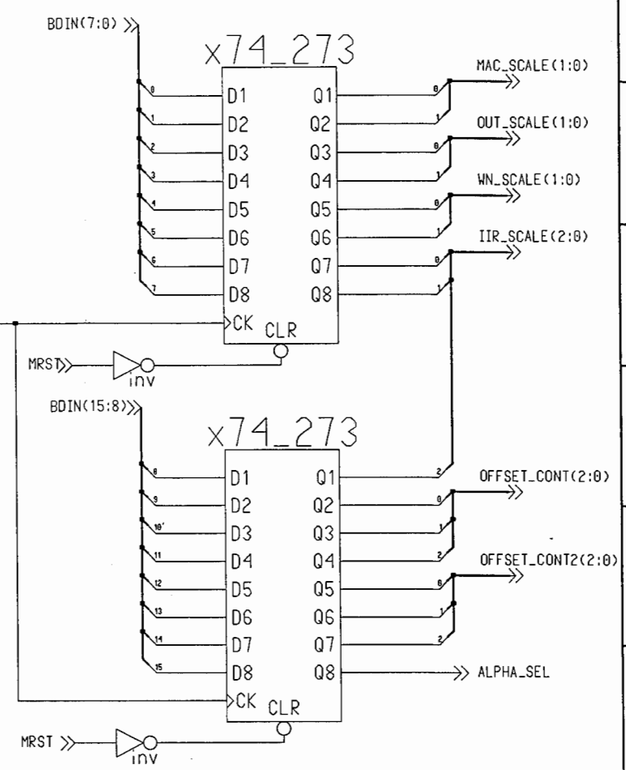
BASE_0



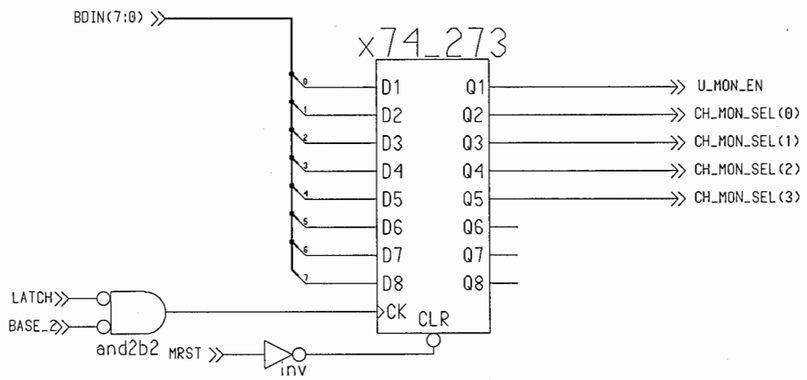
BASE_6



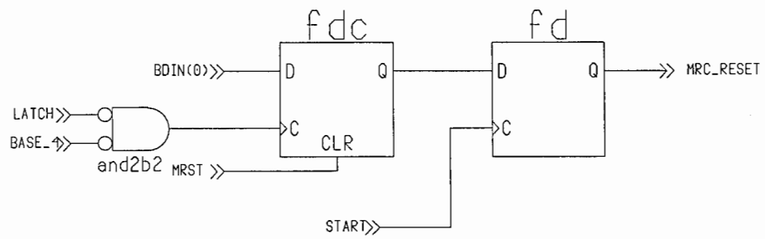
BASE_8



BASE_2



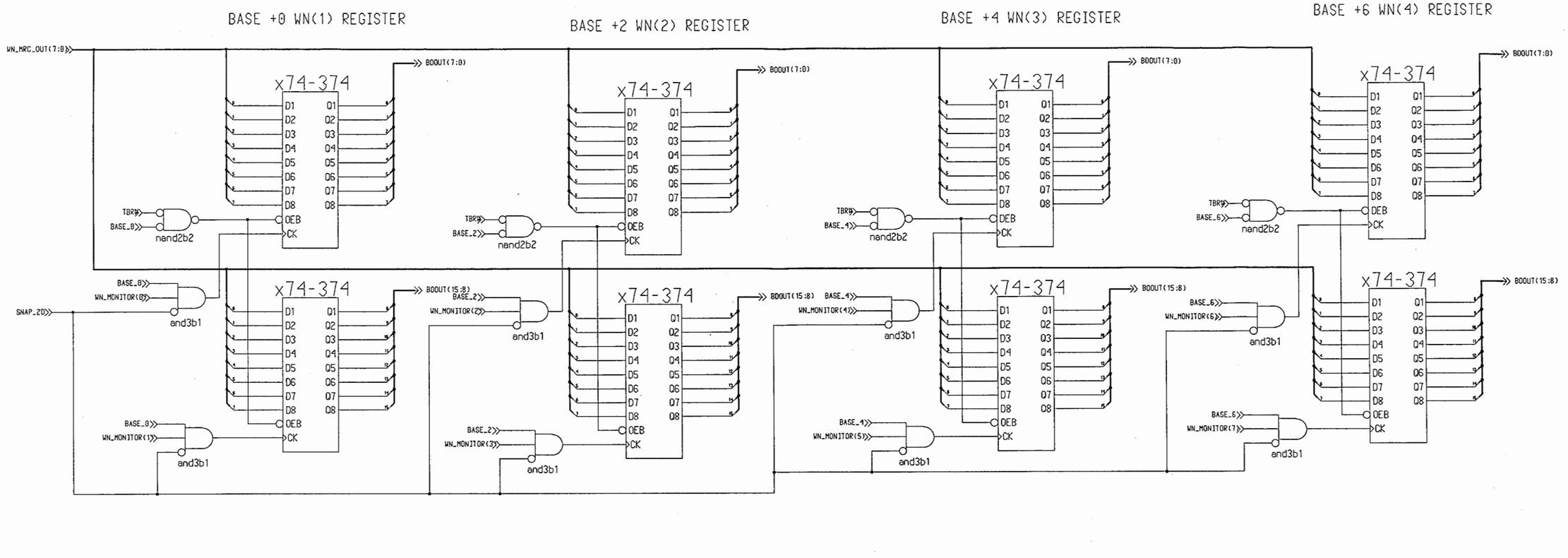
BASE_4



Changed by: taneka	Date Changed: Wednesday, July 19, 1995	Time Changed: 8:25:49 pm	QA CHK:	REV	Drawing Number:	Page: 13	

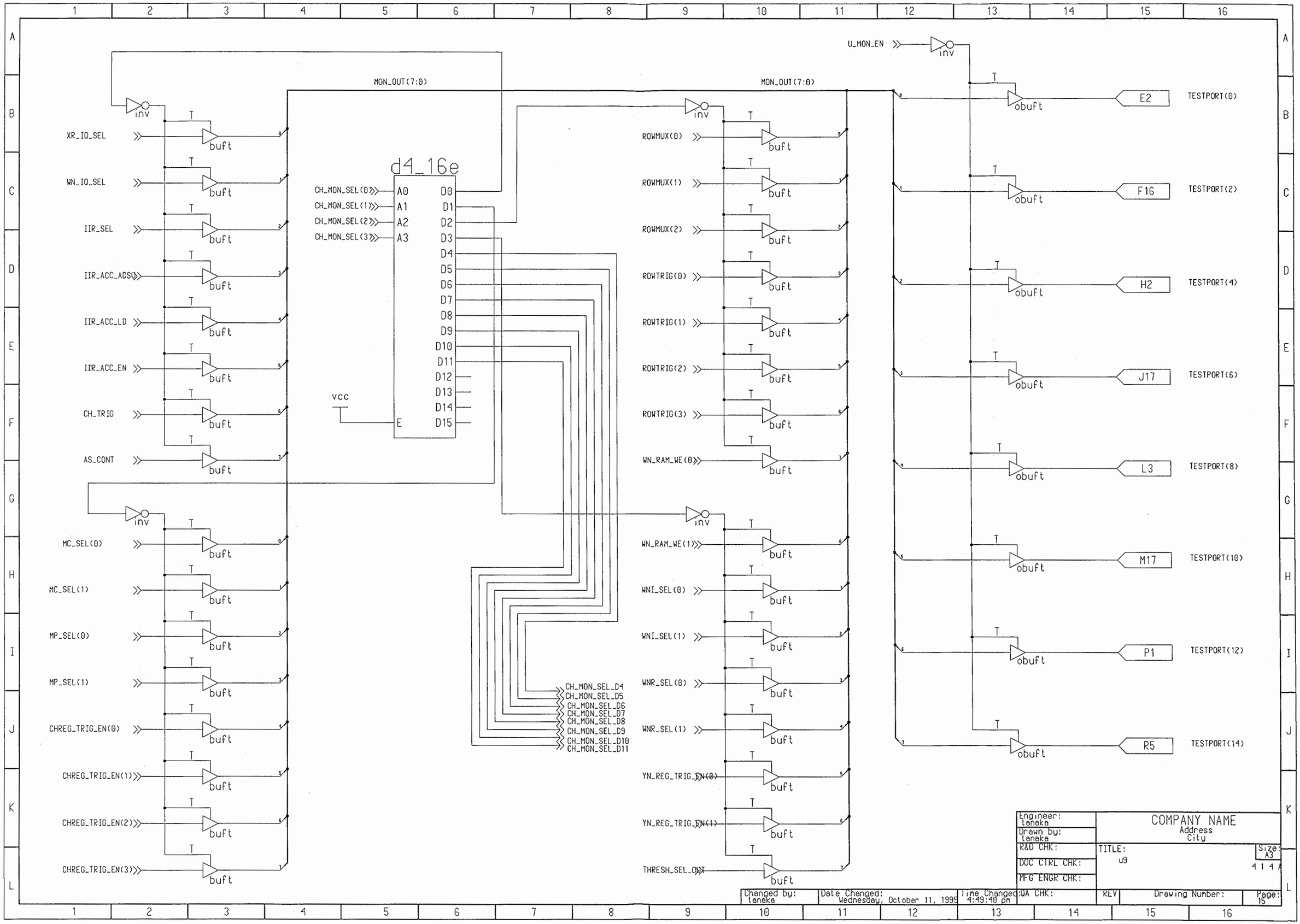
Engineer: taneka	COMPANY NAME	
Drawn by: taneka	Address	
R&D CHK:	City	
DOC CTRL CHK:	TITLE: u9	Size: A3
MFG ENGR CHK:		4 1 4 A

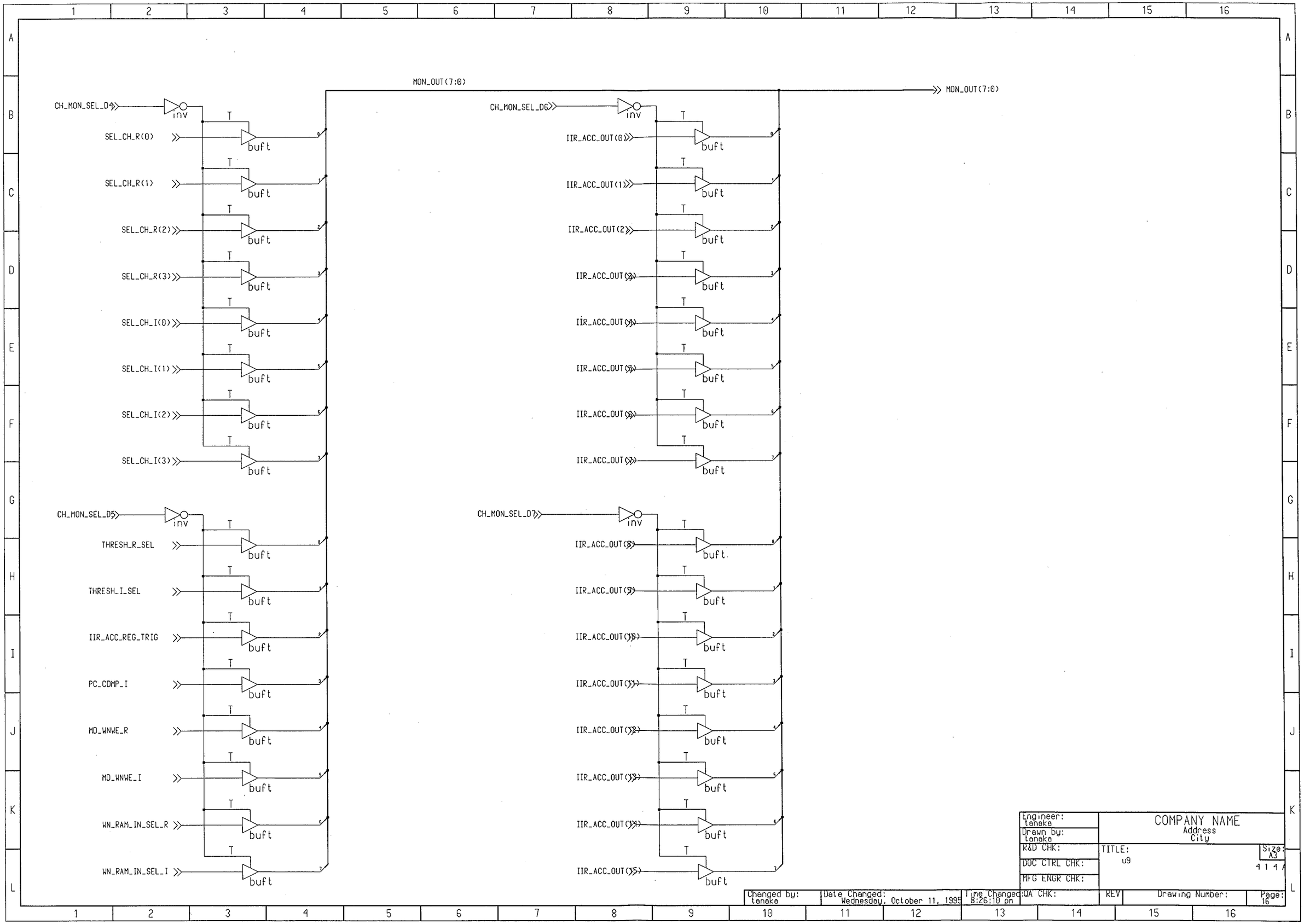
READ REGISTER

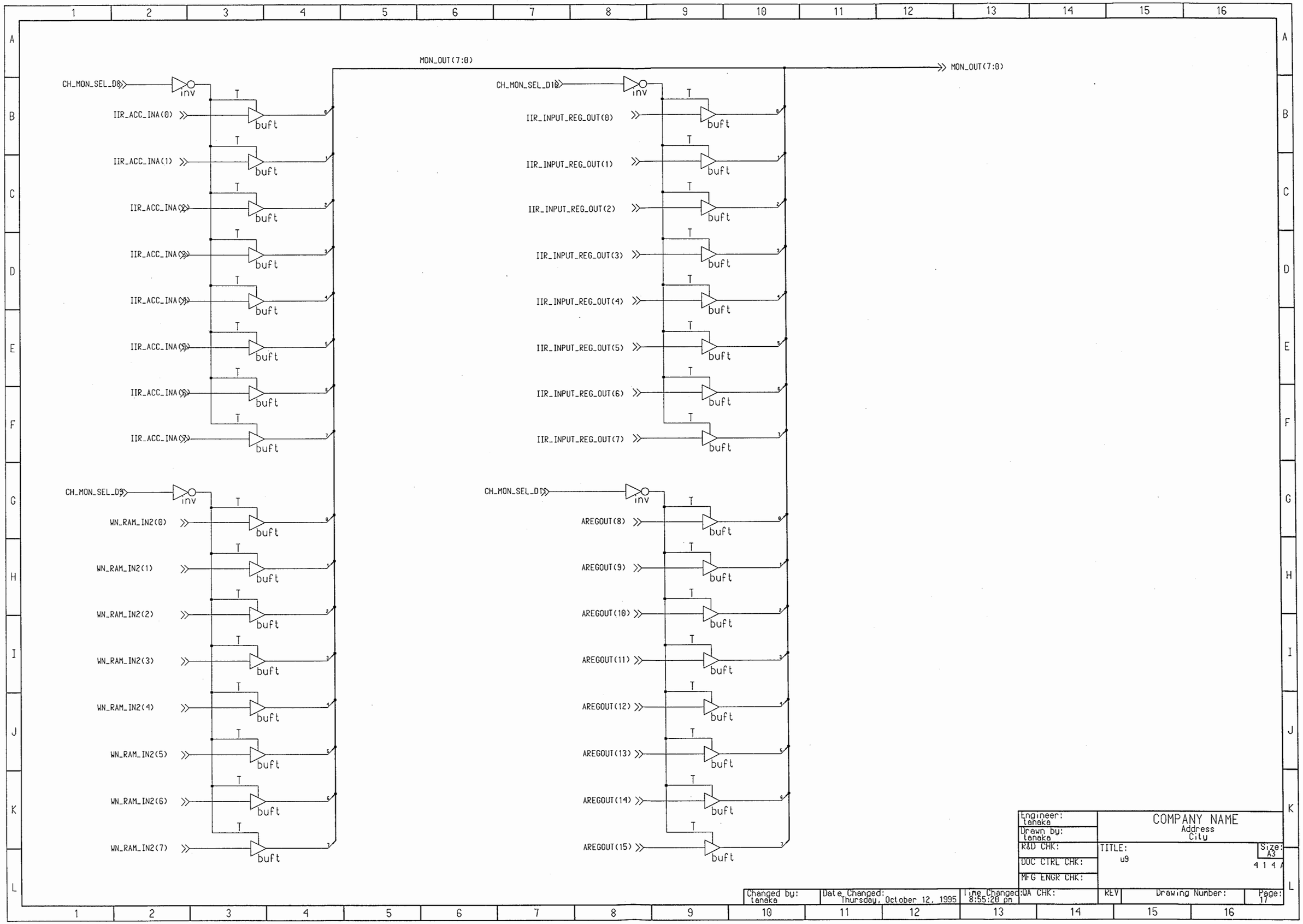


Changed by: taneke	Date Changed: Tuesday, September 12, 1995	Time Changed: 1:47:49 pm	QA CHK:	REV	Drawing Number:

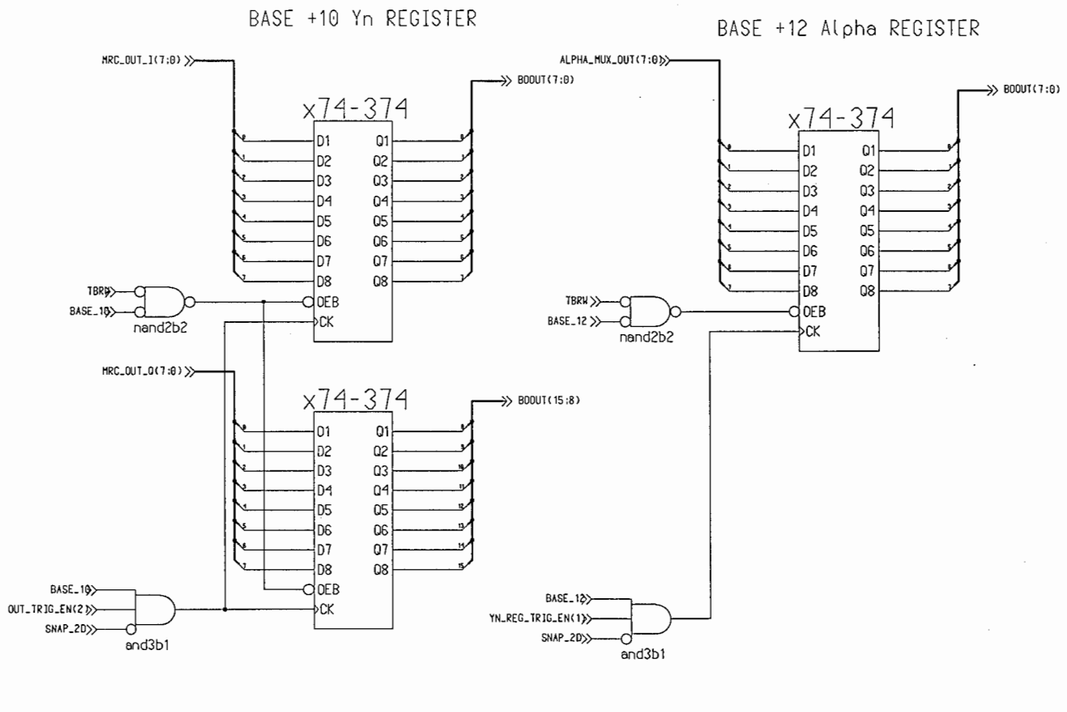
Engineer: taneke	COMPANY NAME		
Drawn by: taneke	Address		
R&D CHK:	City		
DOC CTRL CHK:	TITLE: u9	Size: A2	Page: 14
MFG ENGR CHK:			


MON_OUT(7:0)
MON_OUT(7:0)
U_MON_EN
inv
XR_IO_SEL
WN_IO_SEL
IIR_SEL
IIR_ACC_ADSU
IIR_ACC_LD
IIR_ACC_EN
CH_TRIG
AS_CONT
buft
d4_16e
CH_MON_SEL(0)
CH_MON_SEL(1)
CH_MON_SEL(2)
CH_MON_SEL(3)
A0
A1
A2
A3
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
vcc
E
ROWMUX(0)
ROWMUX(1)
ROWMUX(2)
ROWTRIG(0)
ROWTRIG(1)
ROWTRIG(2)
ROWTRIG(3)
WN_RAM_WE(0)
obuft
E2
F16
H2
J17
L3
M17
P1
R5
TESTPORT(0)
TESTPORT(2)
TESTPORT(4)
TESTPORT(6)
TESTPORT(8)
TESTPORT(10)
TESTPORT(12)
TESTPORT(14)
MC_SEL(0)
MC_SEL(1)
MP_SEL(0)
MP_SEL(1)
CHREG_TRIG_EN(0)
CHREG_TRIG_EN(1)
CHREG_TRIG_EN(2)
CHREG_TRIG_EN(3)
CH_MON_SEL_D4
CH_MON_SEL_D5
CH_MON_SEL_D6
CH_MON_SEL_D7
CH_MON_SEL_D8
CH_MON_SEL_D9
CH_MON_SEL_D10
CH_MON_SEL_D11
WN_RAM_WE(1)
WNI_SEL(0)
WNI_SEL(1)
WNR_SEL(0)
WNR_SEL(1)
YN_REG_TRIG_EN(0)
YN_REG_TRIG_EN(1)
THRESH_SEL_OUT
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
u9
Size: A3
4 1 4 A
Changed by: tanaka
Date Changed: Wednesday, October 11, 1995
Time Changed: 4:49:10 pm
QA CHK:
REV
Drawing Number:
Page: 15

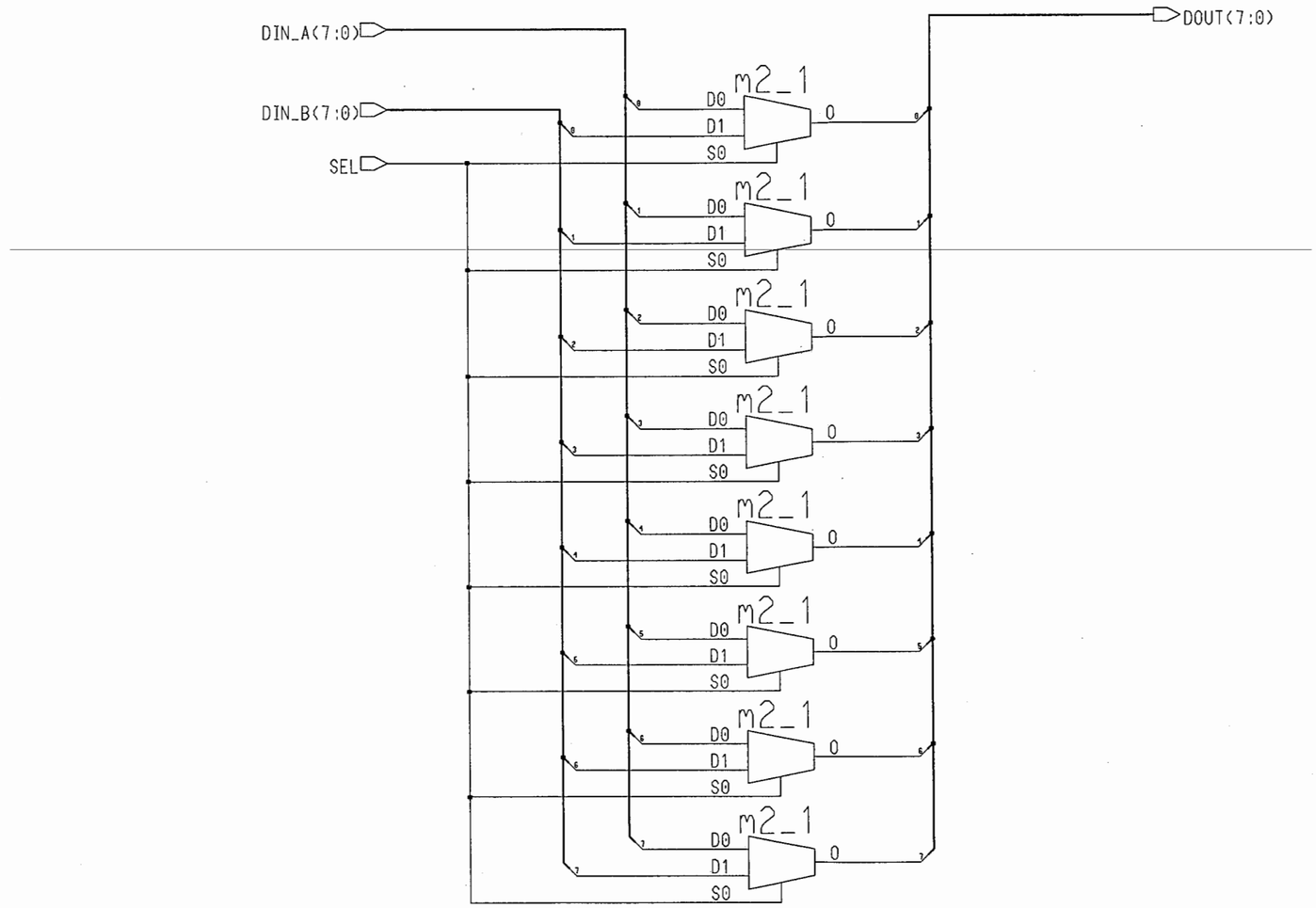

MON_OUT(7:0)
MON_OUT(7:0)
CH_MON_SEL_D4
inv
T
SEL_CH_R(0)
buft
SEL_CH_R(1)
SEL_CH_R(2)
SEL_CH_R(3)
SEL_CH_I(0)
SEL_CH_I(1)
SEL_CH_I(2)
SEL_CH_I(3)
CH_MON_SEL_D5
THRESH_R_SEL
THRESH_I_SEL
IIR_ACC_REG_TRIG
PC_COMP_I
MD_WNWE_R
MD_WNWE_I
WN_RAM_IN_SEL_R
WN_RAM_IN_SEL_I
CH_MON_SEL_D6
IIR_ACC_OUT(0)
IIR_ACC_OUT(1)
IIR_ACC_OUT(2)
IIR_ACC_OUT(3)
IIR_ACC_OUT(4)
IIR_ACC_OUT(5)
IIR_ACC_OUT(6)
IIR_ACC_OUT(7)
CH_MON_SEL_D7
IIR_ACC_OUT(8)
IIR_ACC_OUT(9)
IIR_ACC_OUT(10)
IIR_ACC_OUT(11)
IIR_ACC_OUT(12)
IIR_ACC_OUT(13)
IIR_ACC_OUT(14)
IIR_ACC_OUT(15)
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
u9
Size: A3
4 1 4 A
Changed by: tanaka
Date Changed: Wednesday, October 11, 1995
Time Changed: 8:26:10 pm
QA CHK:
REV
Drawing Number:
Page: 16


MON_OUT(7:0)
MON_OUT(7:0)
CH_MON_SEL_D8
inv
IIR_ACC_INA(0)
IIR_ACC_INA(1)
IIR_ACC_INA(2)
IIR_ACC_INA(3)
IIR_ACC_INA(4)
IIR_ACC_INA(5)
IIR_ACC_INA(6)
IIR_ACC_INA(7)
T
buft
CH_MON_SEL_D10
IIR_INPUT_REG_OUT(0)
IIR_INPUT_REG_OUT(1)
IIR_INPUT_REG_OUT(2)
IIR_INPUT_REG_OUT(3)
IIR_INPUT_REG_OUT(4)
IIR_INPUT_REG_OUT(5)
IIR_INPUT_REG_OUT(6)
IIR_INPUT_REG_OUT(7)
CH_MON_SEL_D9
WN_RAM_IN2(0)
WN_RAM_IN2(1)
WN_RAM_IN2(2)
WN_RAM_IN2(3)
WN_RAM_IN2(4)
WN_RAM_IN2(5)
WN_RAM_IN2(6)
WN_RAM_IN2(7)
CH_MON_SEL_D11
AREGOUT(8)
AREGOUT(9)
AREGOUT(10)
AREGOUT(11)
AREGOUT(12)
AREGOUT(13)
AREGOUT(14)
AREGOUT(15)
Engineer: taneka
Drawn by: taneka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE: u9
Size: A3
4 1 4 A
Changed by: taneka
Date Changed: Thursday, October 12, 1995
Time Changed: 8:55:20 pm
QA CHK:
REV
Drawing Number:
Page: 17

READ REGISTER

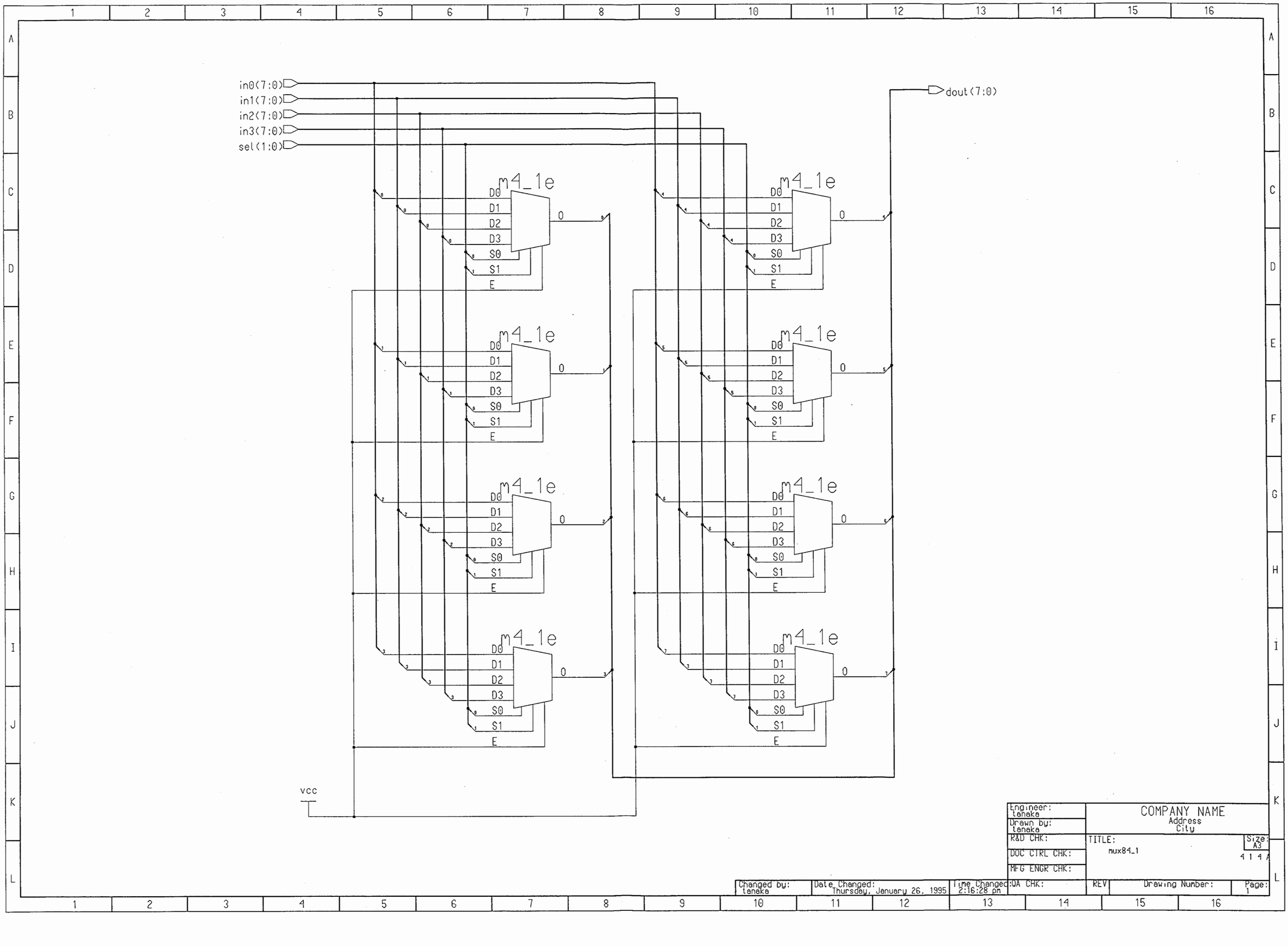


Engineer: tanaka	COMPANY NAME
Drawn by: tanaka	Address
R&D CHK:	City
DOC CTRL CHK:	TITLE: u9
MFG ENGR CHK:	Size: A2
Changed by: tanaka	Date Changed: Wednesday, November 8, 1995
Time Changed: 5:24:28 pm	QA CHK:
REV	Drawing Number:
Page: 18	



Engineer: tanaka	COMPANY NAME
Drawn by: tanaka	Address
R&D CHK:	City
DOC CTRL CHK:	TITLE: mux82_1
MFG ENGR CHK:	Size: A3
QA CHK:	REV / Drawing Number: / Page: 1

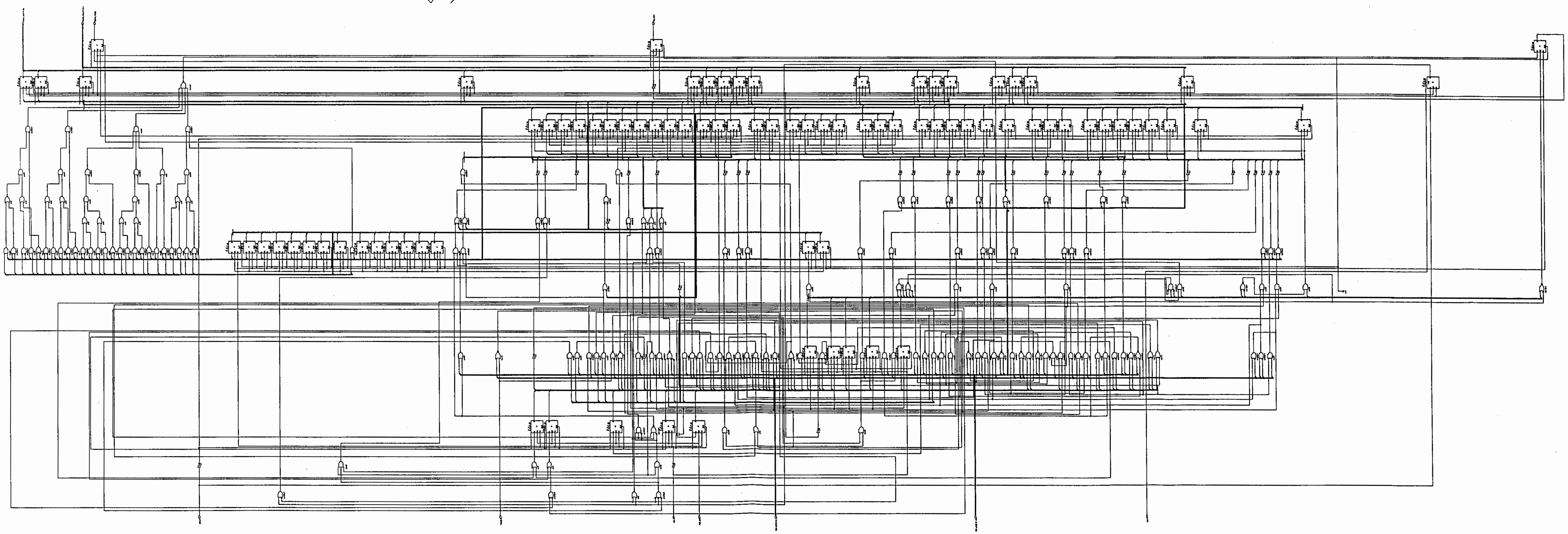
Changed by: tanaka | Date Changed: Wednesday, January 25, 1995 | Time Changed: 7:15:15 pm

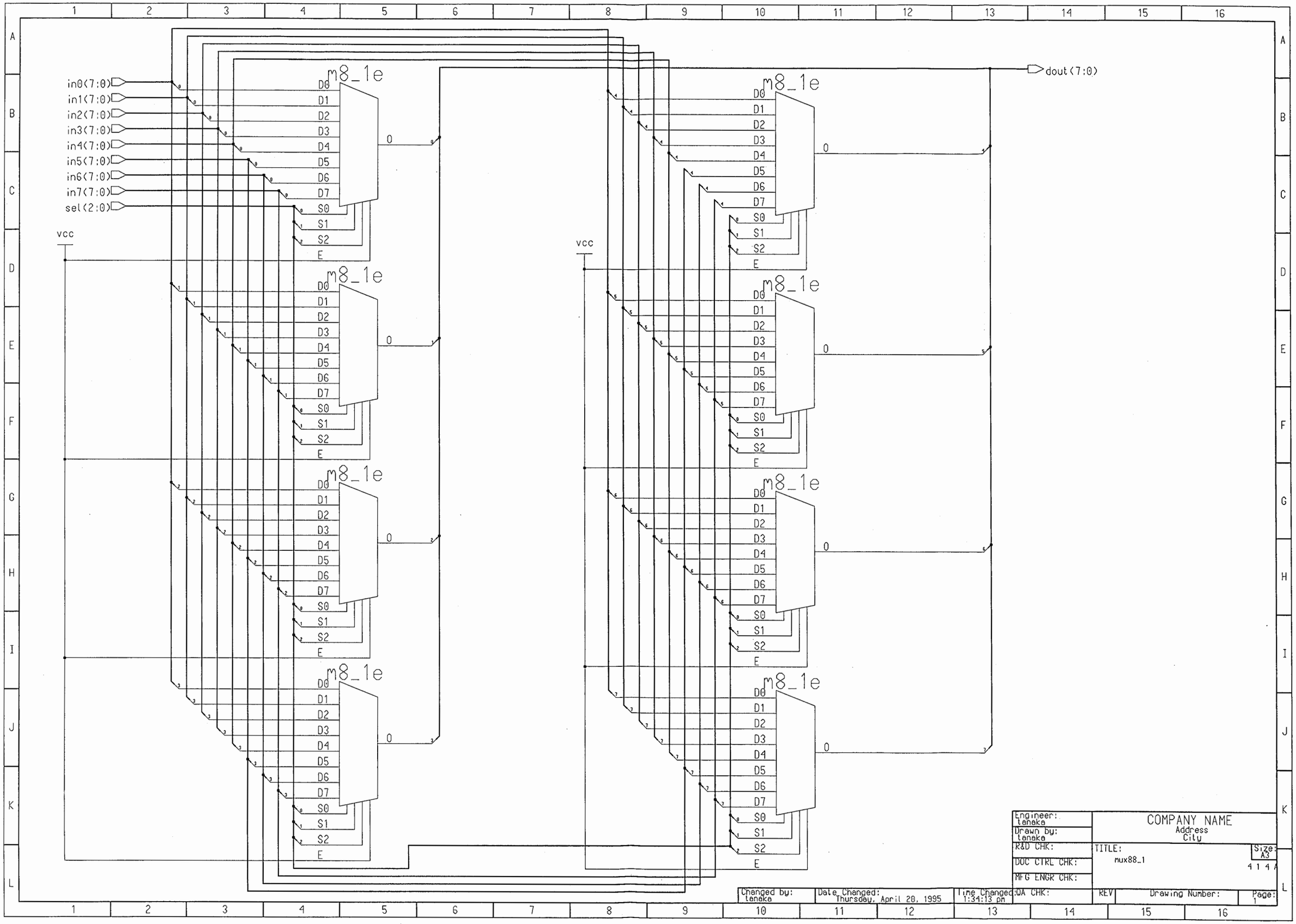


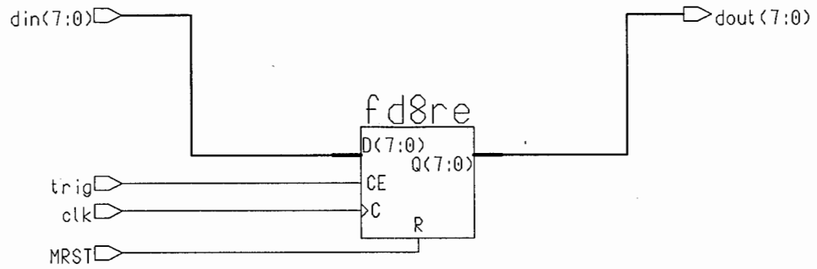
Changed by: tanaka	Date Changed: Thursday, January 26, 1995	Time Changed: 2:16:28 pm	QA CHK:	REV	Drawing Number:	Page: 1
---	---	---	---	---	---	---

Engineer: tanaka	COMPANY NAME Address City	
Drawn by: tanaka		
R&D CHK:	TITLE: mux84_1	Size: A3
DOC CTRL CHK:		4 1 4 A
MFG ENGR CHK:		

SS_comp4

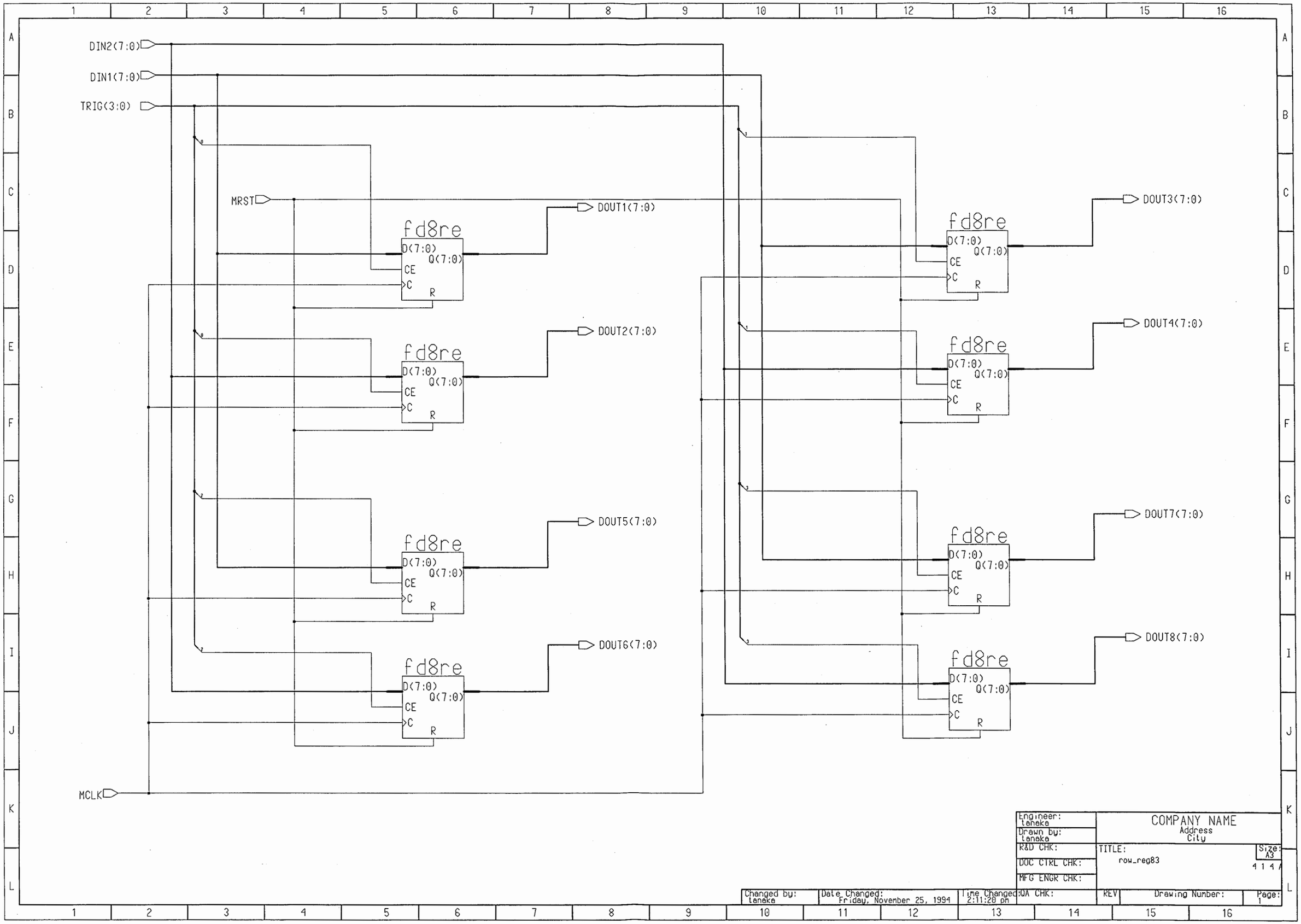


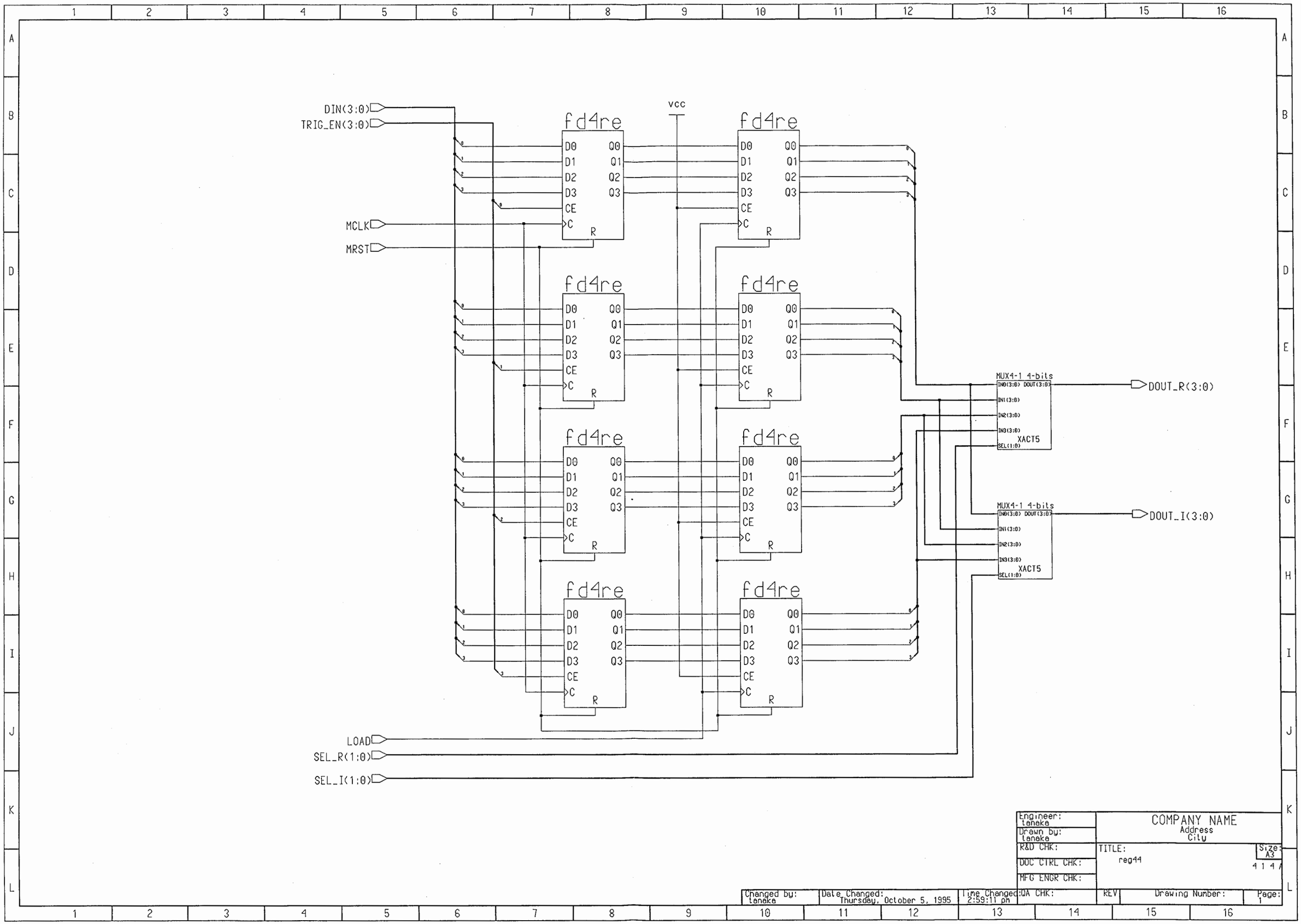

in0(7:0)
in1(7:0)
in2(7:0)
in3(7:0)
in4(7:0)
in5(7:0)
in6(7:0)
in7(7:0)
sel(2:0)
vcc
m8_1e
D0
D1
D2
D3
D4
D5
D6
D7
S0
S1
S2
E
0
dout(7:0)
Engineer: tenaka
Drawn by: tenaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
mux88_1
Size: A3
4 1 4 A
Changed by: tenaka
Date Changed: Thursday, April 20, 1995
Time Changed: 1:34:13 pm
QA CHK:
REV
Drawing Number:
Page: 1

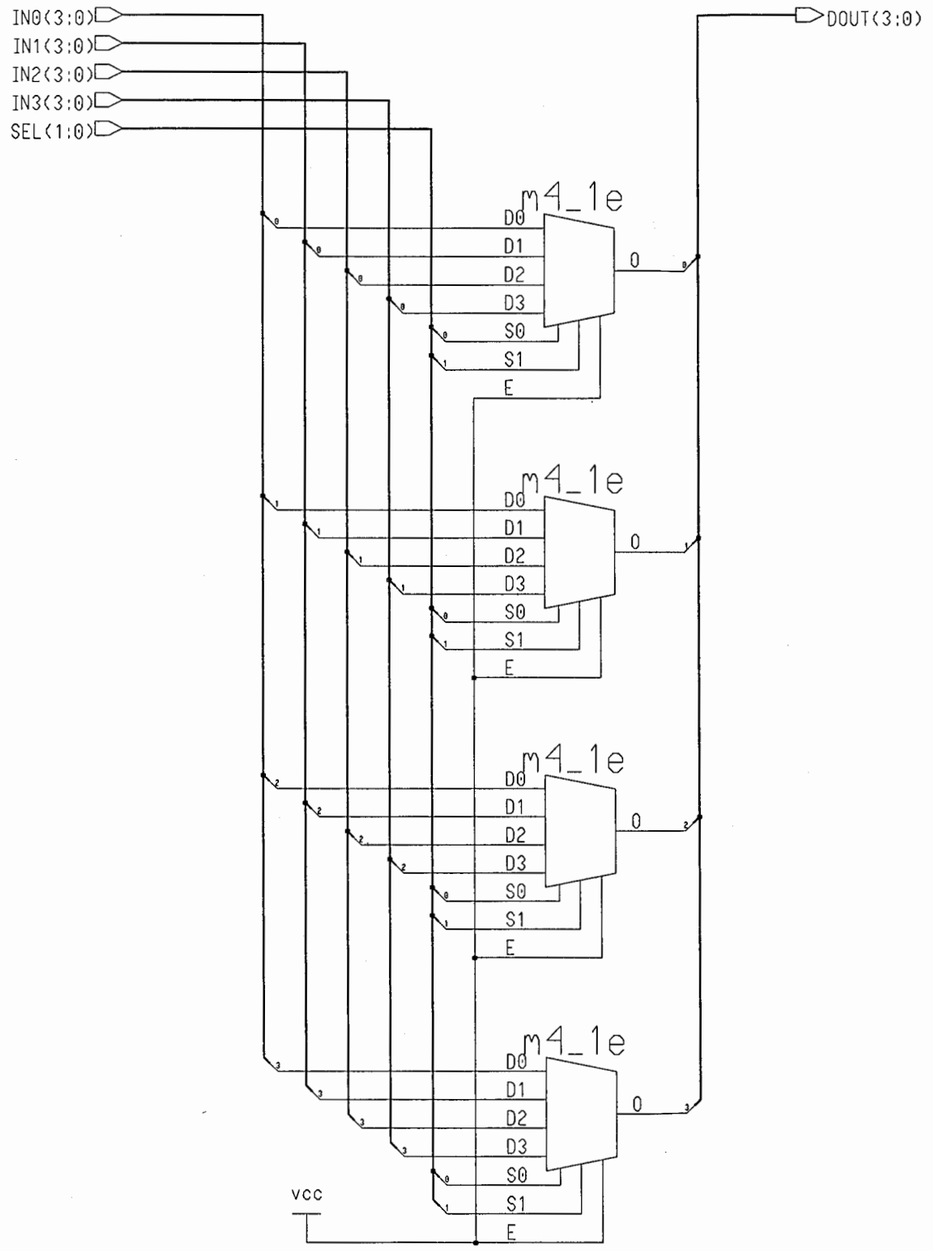


Changed by: tenaka	Date Changed: Wednesday, June 7, 1995			Time Changed: 6:18:07 pm												

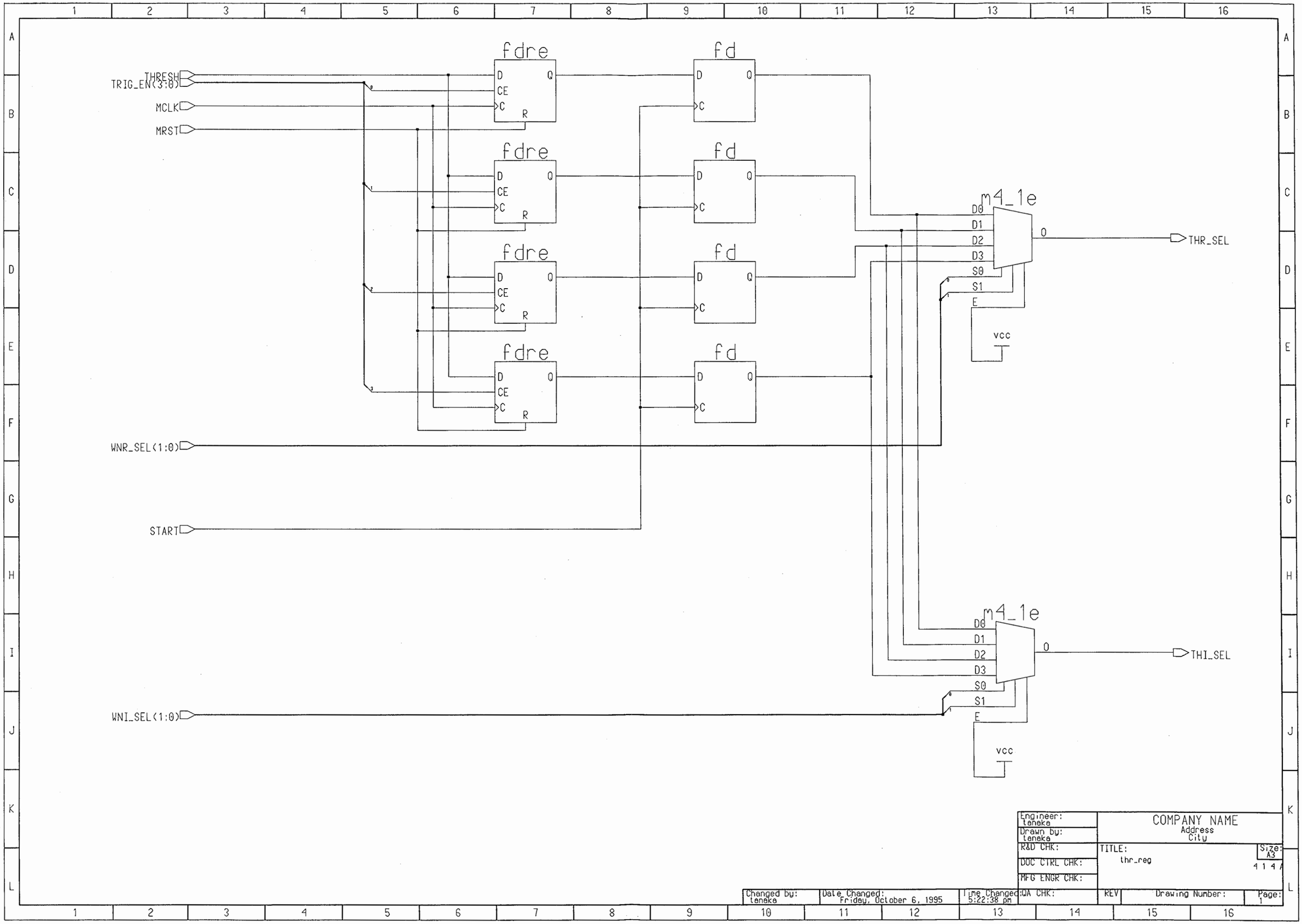
Engineer: tenaka	COMPANY NAME Address City		
Drawn by: tenaka			
R&D CHK:	TITLE: reg8	Size: A3	
DOC CTRL CHK:			4 1 4 A
MFG ENGR CHK:			
QA CHK:	REV	Drawing Number:	Page: 1


DIN2(7:0)
DIN1(7:0)
TRIG(3:0)
MRST
MCLK
fd8re
D(7:0)
Q(7:0)
CE
C
R
DOUT1(7:0)
DOUT2(7:0)
DOUT3(7:0)
DOUT4(7:0)
DOUT5(7:0)
DOUT6(7:0)
DOUT7(7:0)
DOUT8(7:0)
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
row_reg83
Size: A3
Changed by: tanaka
Date Changed: Friday, November 25, 1994
Time Changed: 2:11:20 pm
QA CHK:
REV
Drawing Number:
Page: 1


DIN(3:0)
TRIG_EN(3:0)
MCLK
MRST
LOAD
SEL_R(1:0)
SEL_I(1:0)
VCC
fd4re
D0
D1
D2
D3
CE
C
R
Q0
Q1
Q2
Q3
MUX4-1 4-bits
IN0(3:0)
IN1(3:0)
IN2(3:0)
IN3(3:0)
SEL(1:0)
DOUT(3:0)
XACT5
DOUT_R(3:0)
DOUT_I(3:0)
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
reg44
Size: A3
Changed by: tanaka
Date Changed: Thursday, October 5, 1995
Time Changed: 2:59:11 pm
QA CHK:
REV
Drawing Number:
Page: 1



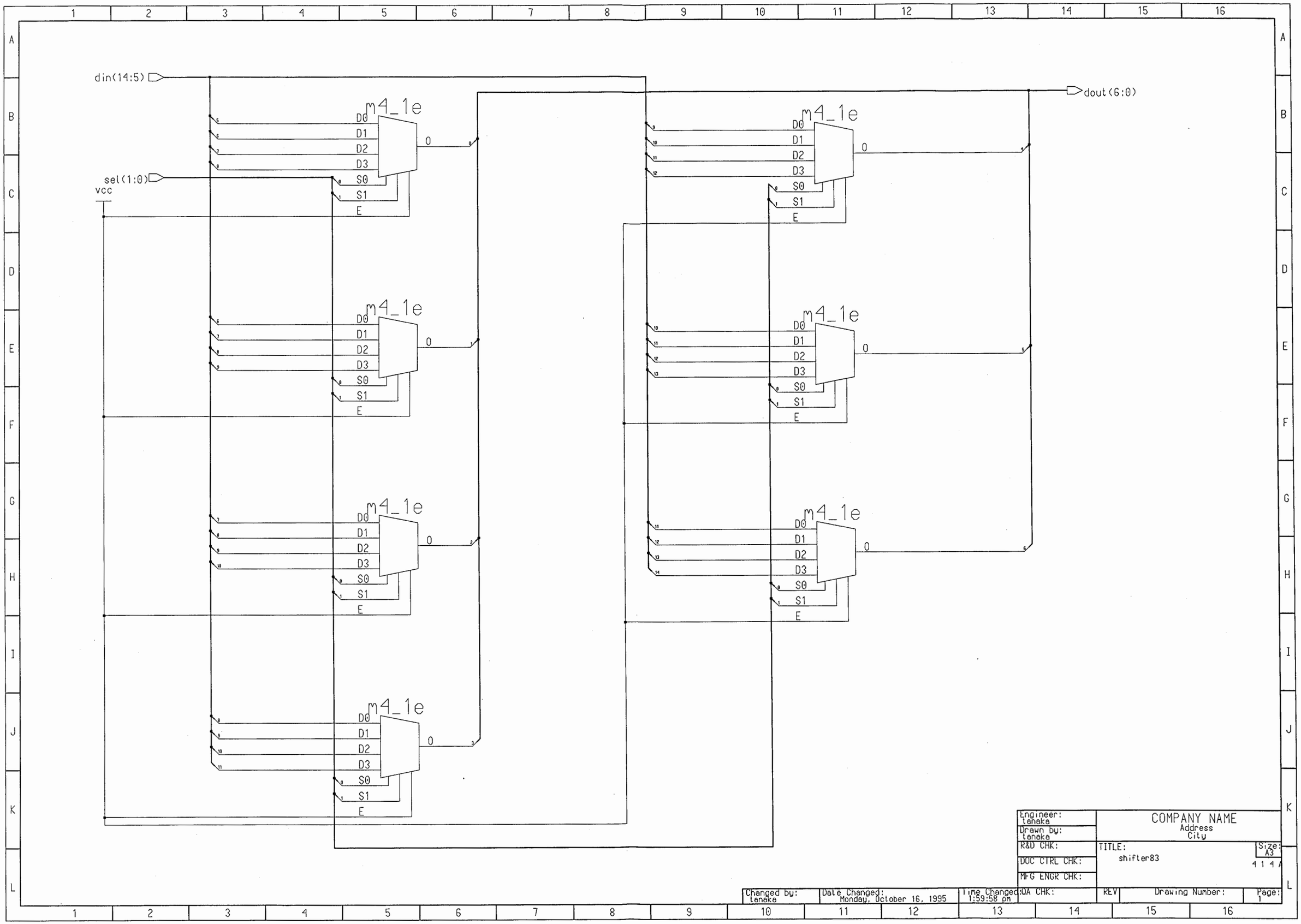
Engineer: tanaka	COMPANY NAME
Drawn by: tanaka	Address
R&D CHK:	City
DOC CTRL CHK:	TITLE: mux44_1
MFG ENGR CHK:	Size: A3
QA CHK:	4 1 4 A
Changed by: tanaka	Date Changed: Monday, February 20, 1995
Time Changed: 7:34:56 pm	REV / Drawing Number: / Page: 1

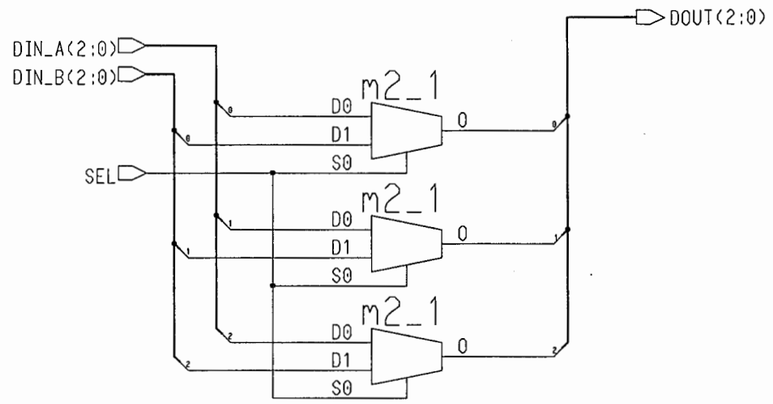

fdre
fd
THRESH
TRIG_EN(3:0)
MCLK
MRST
D
Q
CE
C
R
m4_1e
D0
D1
D2
D3
S0
S1
E
O
vcc
THR_SEL
WNR_SEL(1:0)
START
THI_SEL
WNI_SEL(1:0)
Engineer: taneka
Drawn by: taneka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
thr_reg
Size: A3
4 1 4 A
Changed by: taneka
Date Changed: Friday, October 6, 1995
Time Changed: 5:22:38 pm
QA CHK:
REV
Drawing Number:
Page: 1

LOAD
COMP
FST_SEL_CH(15:0)
CURRENT_CH(15:0)
DIN(3:0)
d4_16e
A0
A1
A2
A3
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
E
vcc
fdre
D
CE
C
R
Q
fd16re
D(15:0)
Q(15:0)
MRST
PREV_CH(15:0)
xor2
COMP_R(15:0)
FF_COMP_R(15:0)
m16_1e
S0
S1
S2
S3
O
ST_R
CH_R(3:0)
CH_I(3:0)
ST_I
TRIG_EN
MRST
MRST

Engineer: tanaka	COMPANY NAME Address City		
Drawn by: tanaka			
R&D CHK:	TITLE: comb_reg	Size: A3	
DOC CTRL CHK:			4 1 4 A
MFG ENGR CHK:			

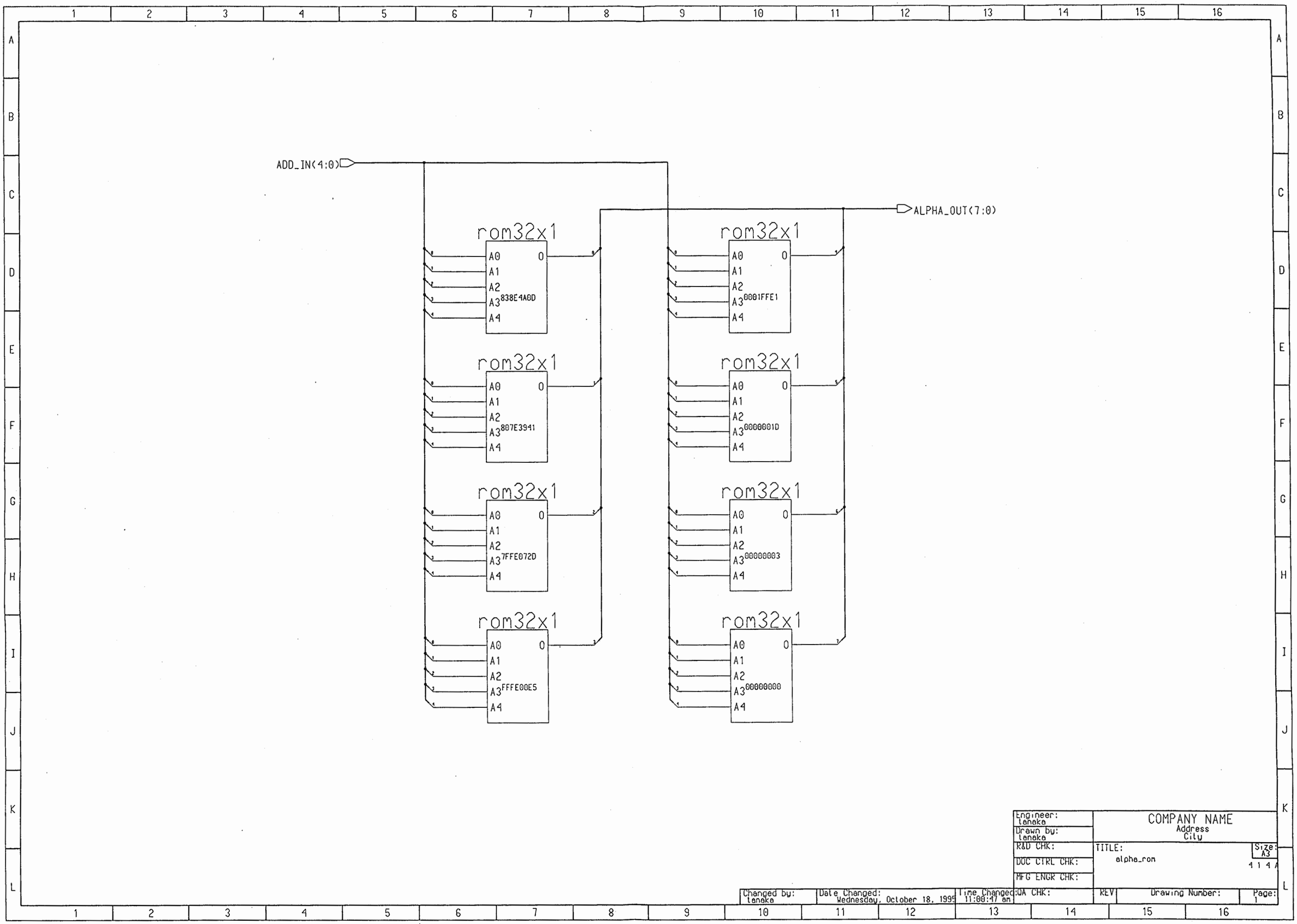
Changed by:	Date Changed:	Time Changed:	QA CHK:	REV	Drawing Number:	Page:
tanaka	Wednesday, October 11, 1995	6:52:45 pm				1


din(14:5)
sel(1:0)
vcc
dout(6:0)
m4_1e
D0
D1
D2
D3
S0
S1
E
O
Engineer: tenaka
Drawn by: tenaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
shifter83
Size: A3
Changed by: tenaka
Date Changed: Monday, October 16, 1995
Time Changed: 1:59:58 pm
QA CHK:
REV
Drawing Number:
Page: 1

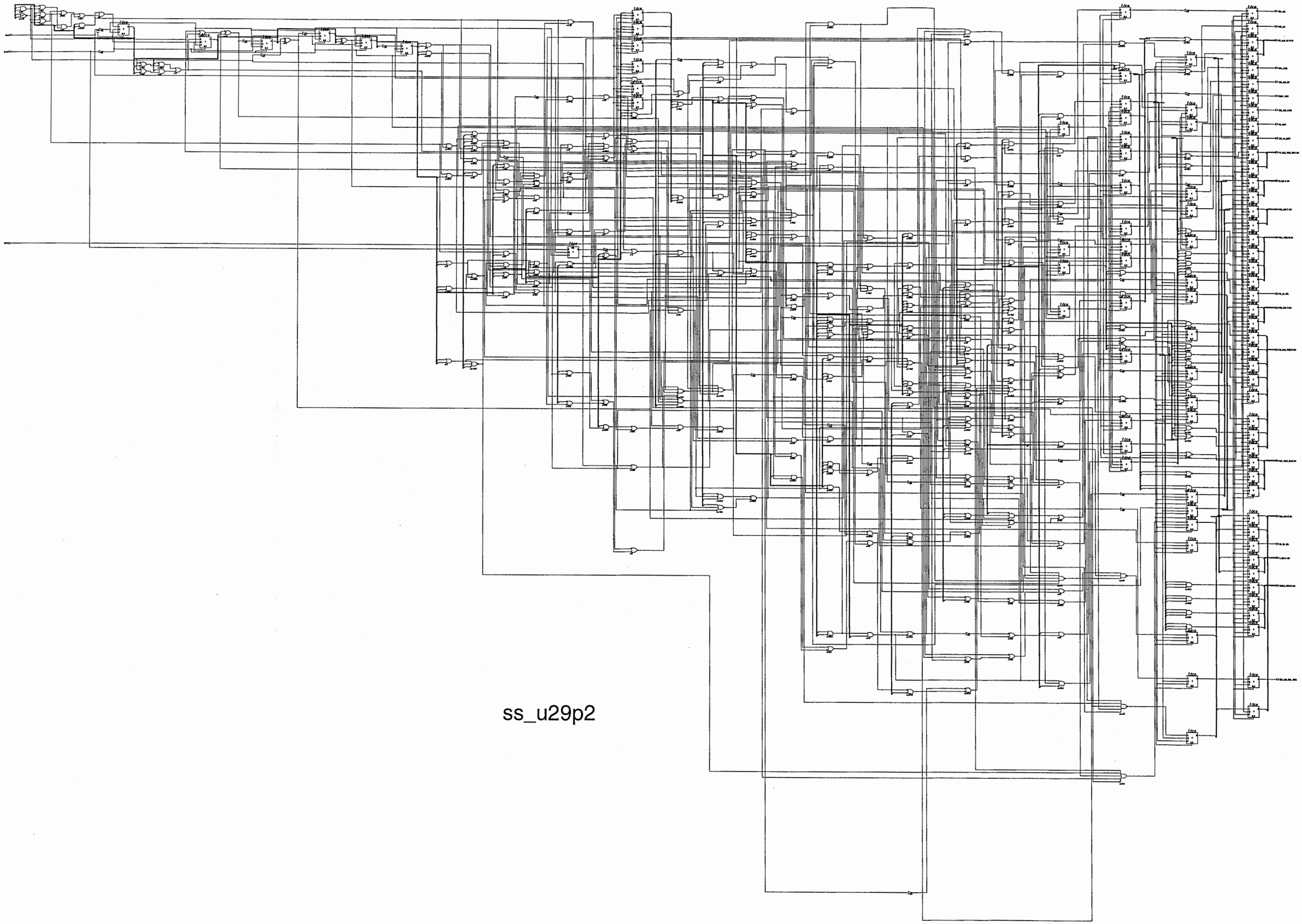


Changed by: tenaka	Date Changed: Friday, July 14, 1995	Time Changed: 11:03:02 am	QA CHK:	

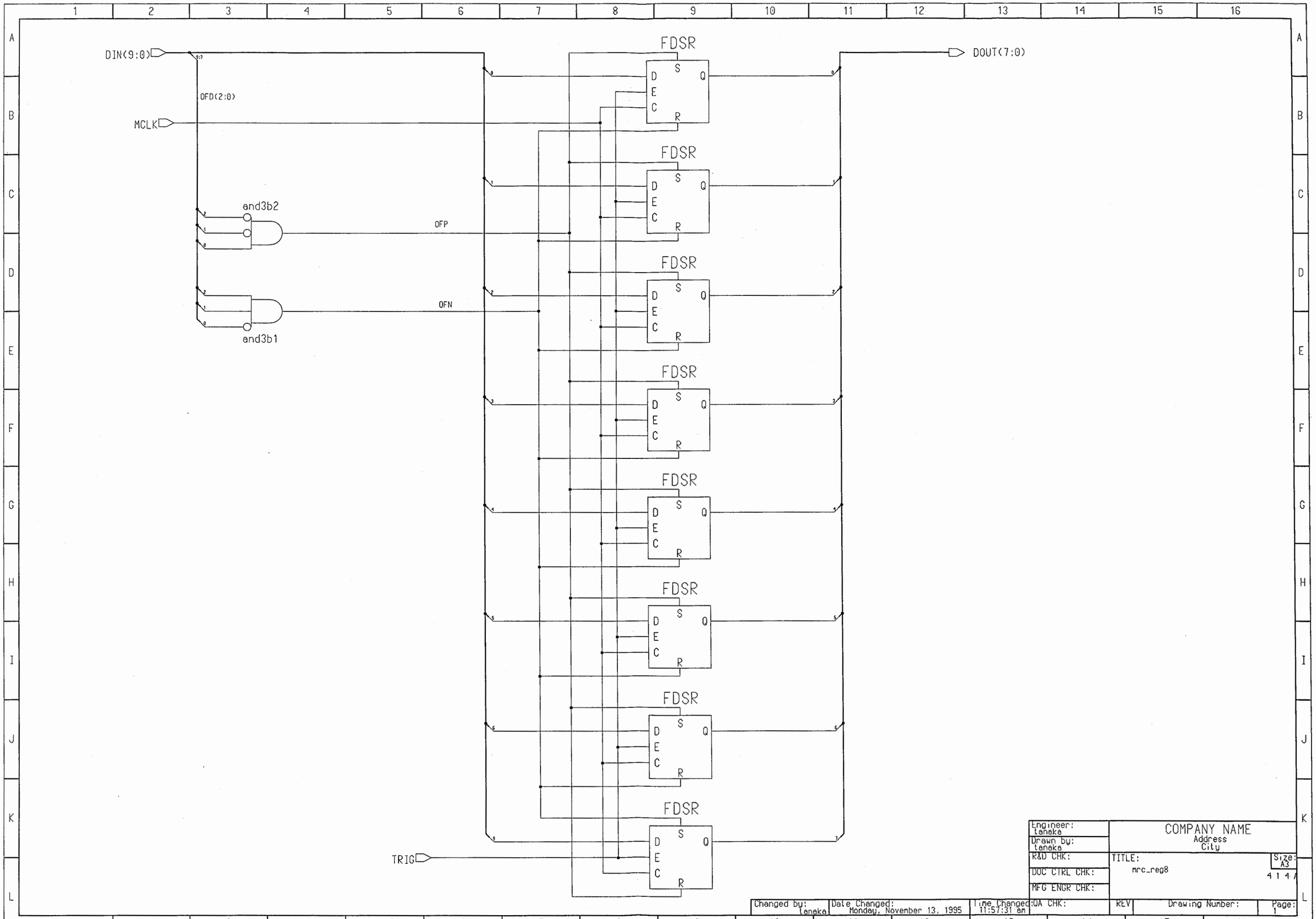
Engineer: tenaka	COMPANY NAME Address City		
Drawn By: tenaka			
R&D CHK:	TITLE: mux3_2_1	Size: A3	
DOC CTRL CHK:			4 1 4 A
MFG ENGR CHK:			
	REV	Drawing Number:	Page: 1

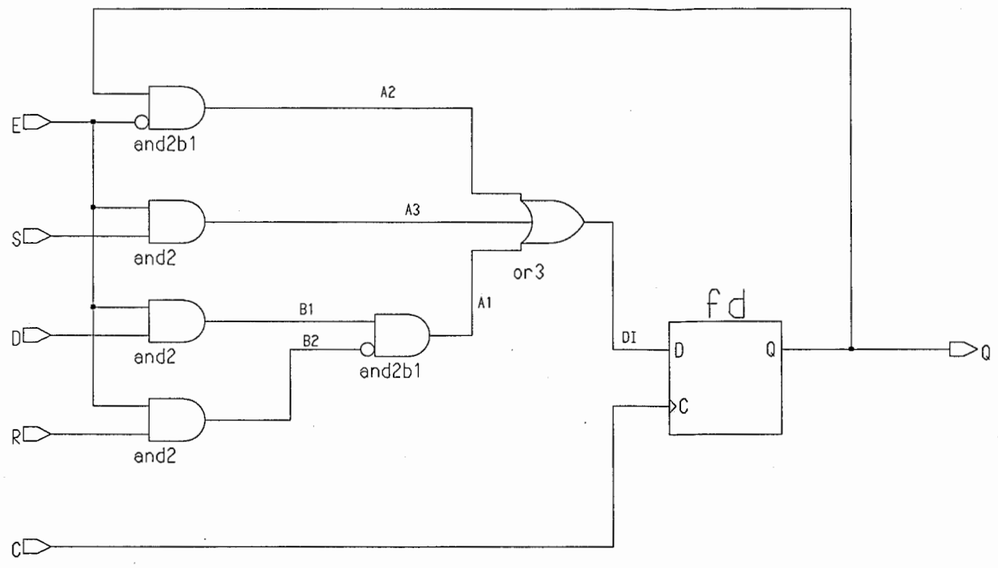


Engineer: tanaka	COMPANY NAME Address City																
Drawn by: tanaka																	
R&D CHK:	TITLE: alpha_rom																Size: A3
DOC CTRL CHK:																	4 1 4 A
MFG ENGR CHK:																	
Changed by: tanaka	Date Changed: Wednesday, October 18, 1995											Time Changed: 11:00:47 am		QA CHK:	REV	Drawing Number:		Page: 1



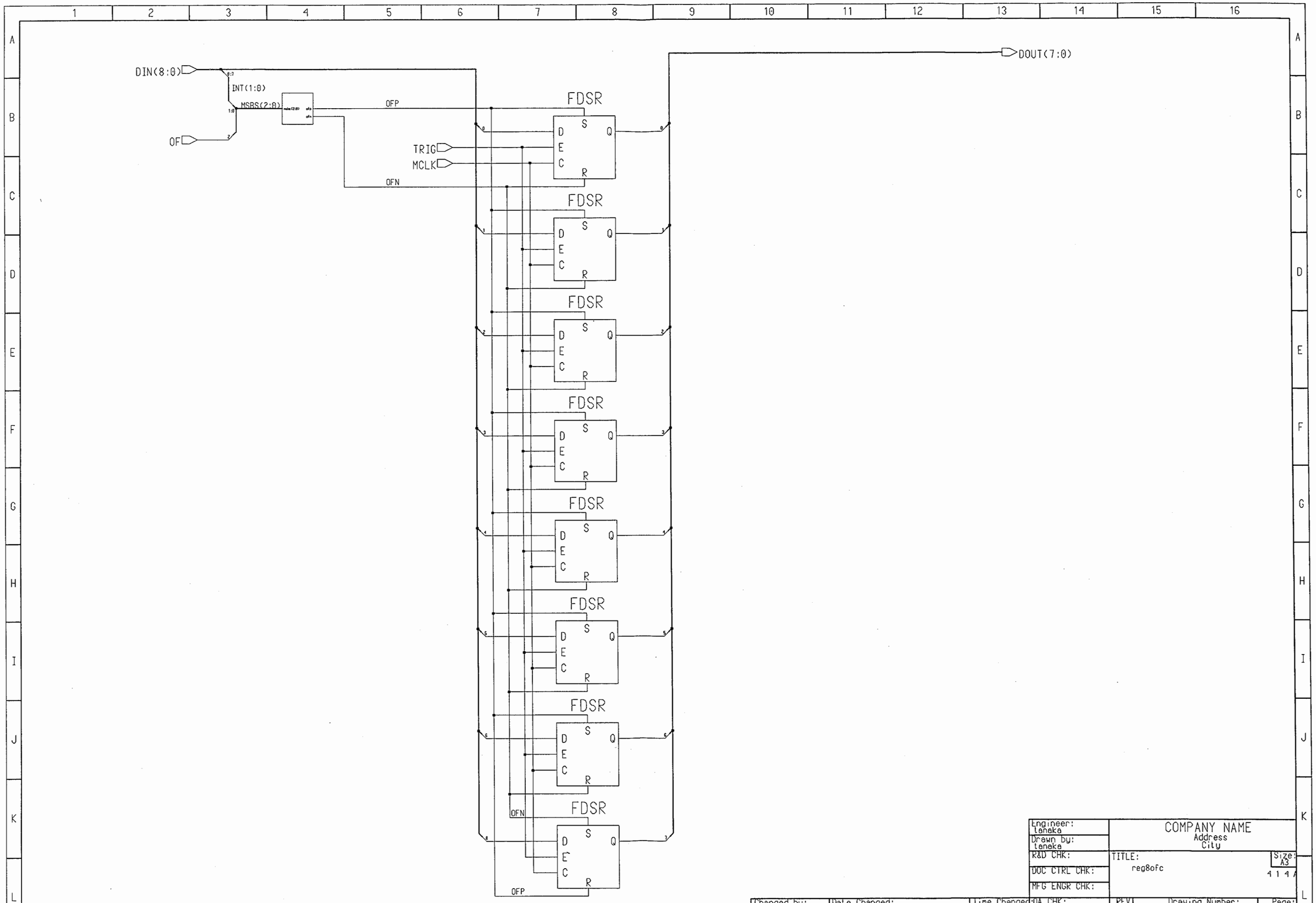
ss_u29p2


DIN(9:0)
9:7
OFD(2:0)
MCLK
and3b2
and3b1
OFP
OFN
FDSR
D
E
C
S
R
Q
DOUT(7:0)
TRIG
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
mrc_reg8
Size: A3
Changed by: tanaka
Date Changed: Monday, November 13, 1995
Time Changed: 11:57:31 am
QA CHK:
REV
Drawing Number:
Page: 1



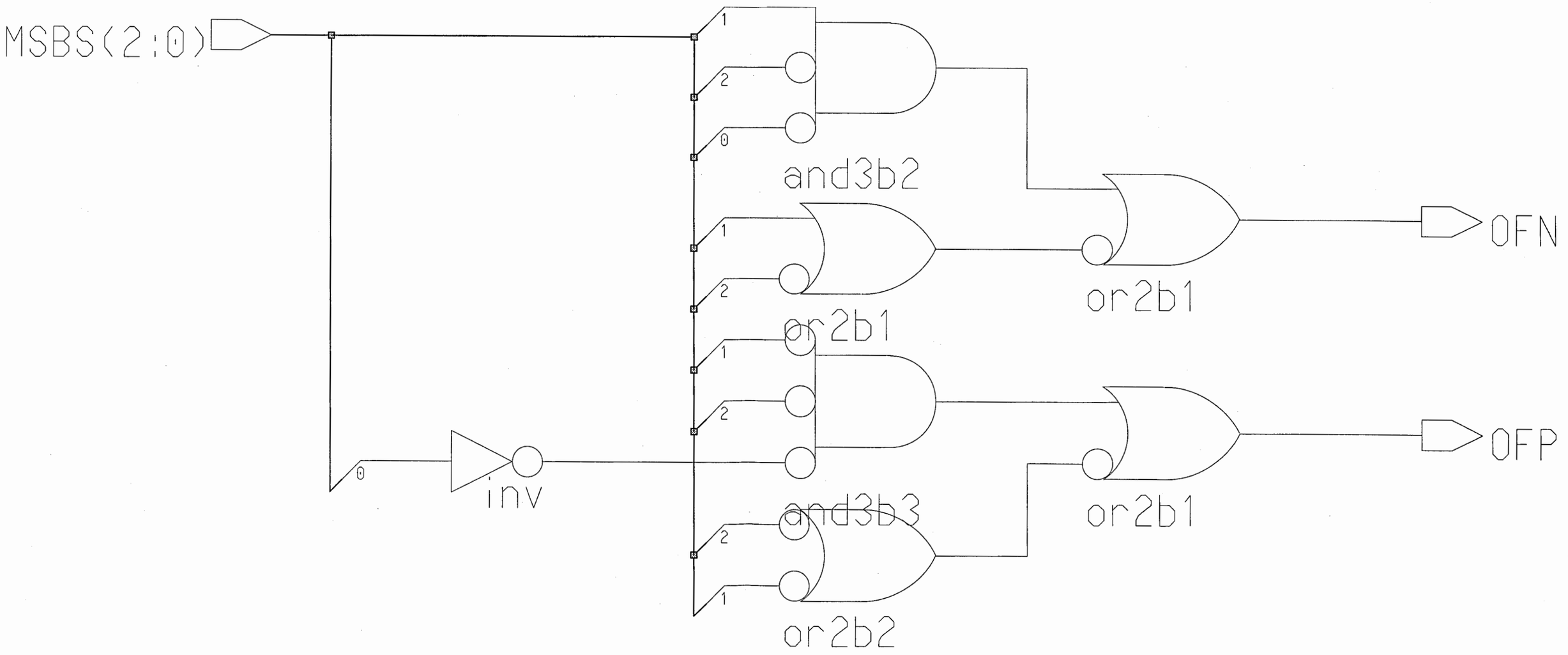
Changed by: tanaka	Date Changed: Thursday, April 27, 1995	Time Changed: 1:42:49 pm	QA CHK:	REV	Drawing Number:	Page: 1

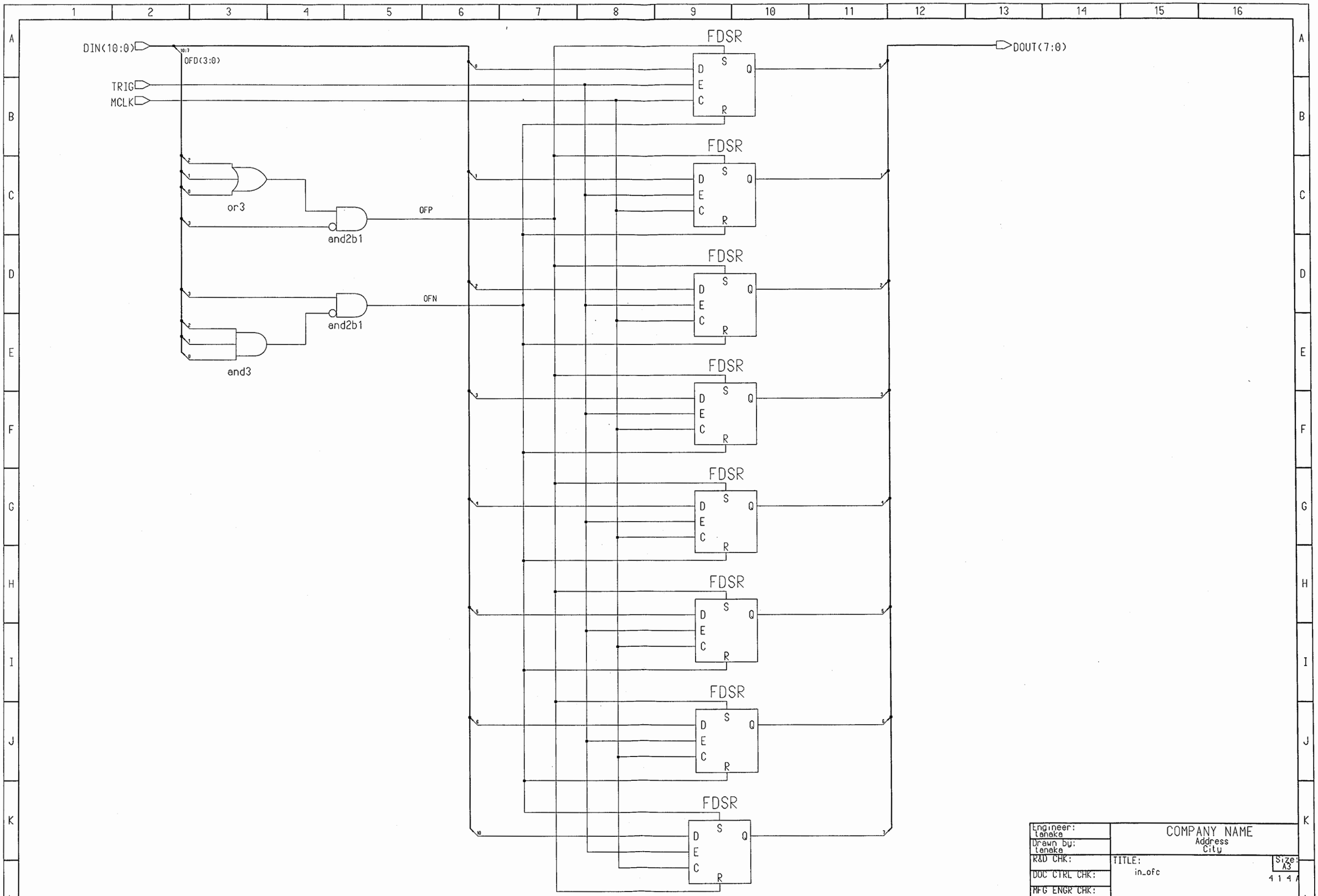
Engineer: tanaka	COMPANY NAME Address City
Drawn by: tanaka	
R&D CHK:	TITLE: FDRS
DOC CTRL CHK:	Size: A3
MFG ENGR CHK:	4 1 4 A

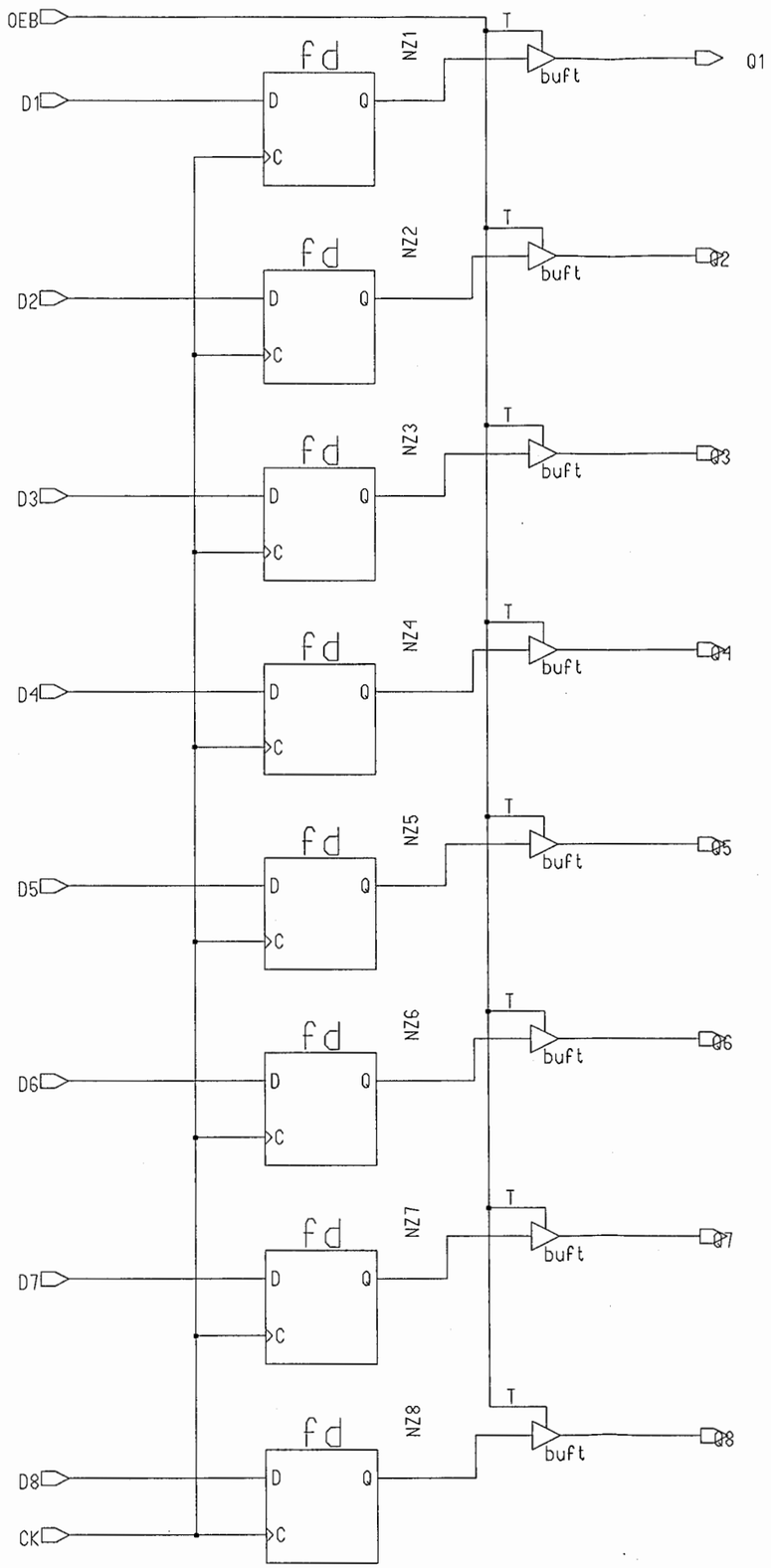


Engineer: tanaka	COMPANY NAME
Drawn by: tanaka	Address
R&D CHK:	City
DOC CTRL CHK:	TITLE: reg8ofc
MFG ENGR CHK:	Size: A3
QA CHK:	REV
Changed by: tanaka	Drawing Number: 4 1 4 A
Date Changed: Tuesday, May 2, 1995	Page: 1
Time Changed: 4:22:53 pm	

ofc2


MSBS(2:0)
1
2
0
and3b2
1
2
or2b1
or2b1
1
2
0
inv
and3b3
2
1
or2b2
or2b1
or2b1
OFN
OFP


DIN(10:0)
OFD(3:0)
TRIG
MCLK
or3
and2b1
OFP
and2b1
OFN
and3
FDSR
D
E
C
S
R
Q
DOUT(7:0)
Engineer: tanaka
Drawn by: tanaka
R&D CHK:
DOC CTRL CHK:
MFG ENGR CHK:
COMPANY NAME
Address
City
TITLE:
in_ofc
Size: A3
Changed by: tanaka
Date Changed: Thursday, May 11, 1995
Time Changed: 7:28:08 pm
QA CHK:
REV
Drawing Number:
Page: 1



Engineer: taneka	COMPANY NAME
Drawn by: taneka	Address
R&D CHK:	City
DOC CTRL CHK:	TITLE: x74_374
MFG ENGR CHK:	Size: A3
QA CHK:	REV: 4 1 4 A
Changed by: taneka	Drawing Number:
Date Changed: Friday, May 19, 1995	Page: 1
Time Changed: 2:29:03 pm